RWU Hochschule Ravensburg-Weingarten
University of Applied Sciences

Scientific Project Report

Sensor Fusion-Based Automatic Emergency Braking for ADAS implemented in CARLA

Guided by:
Prof. Dr. Stefan Elser

Submitted by:
Samanth Krishna - 16944012
Master - Mechatronics

Meghana Ratnam Gudimetla - 18643950
Master - Mechatronics

8th November, 2025

Contents

1 Introduction

This work involves the design of an automatic emergency braking (AEB) module based on sensor fusion with the CARLA simulator (Version 0.10.0) on the Windows 11 platform. The sensor-fusion design involves the combination of image detections from a lightweight YOLOv8 model with geometric features from a 64-channel ray-cast LiDAR (ranged at 60 m with 60,000 points/sec) in such a way that points in the LiDAR signal are matched with the images from the cameras to yield detection confidence and time-to-collision (TTC) measurements. These measurements, in turn, operate on a risk evaluator module, which transitions the finite state machine (FSM) sequence (Inactive -¿ Warning -¿ Braking) based on TTC and stopping distances calculated from physics models of reaction time and maximum deceleration, with majority voting hysteresis, followed by progressive braking based on brake torque, ending with the interruption of braking when the threat is cleared, based on distance/velocity criteria.

To achieve determinism and replayability, CARLA simulates in synchronous mode with a fixed clock delta of

$$\Delta t = 0.05s(20Hz) \tag{1}$$

The minimal OpenDRIVE straight road environment (300 m, with 0.7 road friction coefficient) introduces the ego car and an imaginary static obstacle 50 m down the road to serve as the baseline environment, with weather and physics parameters fixed to allow direct comparisons of perception and decisioning. The fusion module adaptively combines both the LiDAR and RGB confidence based on their respective range (giving priority to the former near-range) and also offers an isolated LiDAR fallback mode whereby braking can be achieved even in the near-range when there's no detection made by the RGB. Spatial filtering involves the front corridor, side boundaries, and height zone, while overlays from Pygame display detection, closest detection, density in the LiDAR points, along with the time-to-collision.

The code base is robust across CARLA 0.10.0 with protective blueprinting and an efficient map generator, further ensuring the parameterization of sensor timelines to synchronize all sensors with the 20 Hz cycle. Even though the proposed approach was shown with the example of a stop situation involving only a straight road, the design is amenable to extensions related to cut-in cars, crossing pedestrians, and multi-object conflicts to systematically explore sensor parameterization, weight settings, and braking strategy in the interpretable assessment of automatic emergency braking.

2 Getting to Know CARLA

2.1 What is CARLA?

CARLA is an open-source simulator designed specifically for the development, training, and validation of autonomous driving systems. They have a high-fidelity environment on Unreal Engine, with a Python/C++ API, OpenDRIVE road format, and a server-client architectural pattern in which the server handles physics, rendering, and actors, and the clients implement the logic. The UE5 generation (CARLA 0.10.0) introduces Lumen, Nanite, and Chaos, along with ROS 2 support, while retaining the Python API.0

2.2 Why CARLA for sensor-fusion AEB?

- **Authentic, high-quality sensors:** On-board RGB/Depth/Semantic cameras, LiDAR (ray-cast & solid-state hybrids), RADAR, GNSS, and more, with varied parameters such as FOV, frames per second, levels of noise, and motion.
- **Deterministic testing:** Synchronous mode with fixed time step. Allows perception/control loop closure, which is important for AEB thresholds/hysteresis tuning.
- **Scenario tooling & standards:** Support of OpenSCENARIO & ScenarioRunner enables standardized, formalized AEB trials (cut-in, crossing VRU, stopped vehicle, etc
- Ecosystem & realism upgrades in 0.10.0. Migration to UE5 adds Lumen/-Nanite rendering, Chaos physics, upgraded town & vehicle models, Python 3.8 to 3.12 wheels, and ROS 2 server support.
- **Open, extensible, well-documented:** "First steps" explanations, foundations, sensors, API references help to facilitate the reproduction and further development of experiments.

2.3 Installation of CARLA

2.3.1 System prerequisites

Requirement	Specification
OS	Windows 11 (minimum for UE5 build).
GPU & driver	NVIDIA RTX-class GPU ($\geq$16 GB VRAM recommended); Windows driver $\geq$560. Note: GPUs $<$ 12 GB VRAM may fail to load Town10.
Disk	$\sim$130 GB free (sim + assets).
Python	3.8–3.12 supported in 0.10.0 (choose one and keep it fixed). pip $\geq$20.3.
Ports	TCP 2000–2001 open (client–server).

Table 1: System requirements for running the UE5 simulation

2.3.2 Obtain the packaged simulator

- Navigate to Downloads $\rightarrow$ CARLA 0.10.0 and fetch the Windows packaged release from the linked GitHub page [1].
- Extract to a short, space-free path, e.g., C: CARLA_0_10_0. The package contains the server (CarlaUnreal.exe), examples, and Python wheels.

2.3.3 Create an isolated Python environment

- ```
  py -3.10 -m venv C:\venvs\carla-aeb
  C:\venvs\carla-aeb\Scripts\activate
  python -V
  pip -V
  ```
  ```

- Install minimal client-side deps (as recommended):

```
pip install --upgrade pip
pip install --user pygame numpy
```

2.3.4 Install the CARLA Python client (wheel from the package)

From the package's wheel directory, install the wheel matching the Python version (look for cp310 for Python 3.10, etc.):

```
cd C:\CARLA_0_10_0\PythonAPI\dist
pip install carla-0.10.0-*-win_amd64.whl
```

Using the bundled wheel ensures client–server version parity.

2.3.5 Launch the server

```
cd C:\CARLA_0_10_0
CarlaUnreal.exe
```

A spectator view of Town10 should open; the server now listens on ports 2000–2001. Allow Windows Firewall if prompted. .

Figure 1: CARLA Town 10 map

3 System Configuration and Implementation

3.1 Scope and design goals

Based on the rationale introduced above, we apply a deterministic, fusion-based AEB system in CARLA v0.10.0 with primary consideration given to:

- (i) repeatability (fixed time stepping & synchronized sensors),
- (ii) interpretable physics (separate model describing the stopping distance),
- and (iii) interpretable fusion (metric mapping of LiDAR points to bounding boxes in the images).

This approach is ideal for Windows 11 environments that require a balance between real-time performance & moderate GPU utilization.

3.2 Deterministic runtime

To ensure consistent interactions between perception and control feedbacks, the simulator is set to operate in *sync mode* and uses a constant time step $\Delta t = 0.05\,\mathrm{s}$ (or $20\,\mathrm{Hz}$). To maintain proper synchronization between them, each sensor is forced to operate at the same rate with the line `sensor_tick = 0.05` setting. Thus, all physics-based processes (such as simulations of camera images and LiDAR scanning) take place at one-to-one intervals [3].

3.3 Minimal scenario for controlled evaluation

A **100 m** straight one-lane road in the OpenDRIVE model is created programmatically (road friction 0.7, level grade) to focus perception and decision-making [4, 5]. The ego vehicle is placed at the origin, with a **stationary obstacle** at **50 m** infront in-lane. Weather is set to **ClearNoon**. This baseline setup is meant to create a simple test bed for merging geometry and semantics before progressing to cut-in or cross-VRU cases.

3.4 Version-robust actor provisioning

The script tests server abilities and employs a tolerant blueprint lookup (safe_find) to account for name differences in CARLA 0.10.0 (dots vs. underscores and year suffixes). It will check for preferred 4-wheel sedans/coupes if they exist; if not, it uses pattern match or fallbacks to any 4-wheel vehicle. This ensures the experiment is transportable with the content packs and still compares similar dynamics [6].

3.5 Sensor Suite and Justification

3.5.1 Forward RGB camera (semantic perception)

Configuration:

- **Resolution**: 960×540 (W×H)
- **Field of View**: 90°

- **Pose**: Front-of-vehicle mount at $x = \texttt{bb.extent.x} + 1.5\,\text{m}$, $z = 1.8\,\text{m}$, pitch $-3°$ (yaw/roll 0) [8].
- **Timing**: sensor_tick = 0.05 s (20 Hz, synchronous)
- **Processing**: YOLOv8n (Ultralytics), confidence threshold 0.25 [16]
- **AEB-relevant classes**: person (0), bicycle (1), car (2), motorbike (3), bus (5), truck (7)

Rationale:

- **Vertical structure at close range.** 64-channel vertical detection provides adequate vertical detail to confidently detect the closest forward surface (bumpers, legs & torso, truck faces). This is especially important in validating time-to-collision and stopping distance analyses.
- **Look-ahead and density.** At 60 k pts/s with 20 Hz rev rates, there are approximately 3 000 points per revolution. In the 90° forward section, there are approximately 750 points per frame; a few hundred usually remain after ego-lane filtering, enough for a reliable nearest-surface calculation without bogging down the CPU or GPU [25].
- **Task-conforming vertical FoV.** The -30°...+10° range highlights the road and bottom front geometric features, thus raising the probability of near-range ground-hit detection and reducing clutter above the target.
- **Tightly synchronized sweep.** Rotation_frequency = 20 Hz and sensor_tick = 0.05 s ensures one 360° sweep per simulator tick. This prevents the effect of partially filled frames on the fusion operation.

Projection for fusion:

The code employs a pinhole projection model with [13, 14] $f = \dfrac{\text{width}}{2 \cdot \tan(\text{FOV}/2)}$ to project the LiDAR points transformed to the camera coordinate frame:

$$u = f \cdot \frac{Y}{X} + c_x, \quad v = f \cdot \frac{Z}{X} + c_y \quad \text{(X forward)}$$

Only points within the image window remain. These equations provide a geometrically correct mapping that prevents heuristic-based gating. This results in detection-based metric distances.

3.5.2 Roof-mounted 64-channel LiDAR (geometric perception)

Configuration:

- **Channels:** 64
- **Range:** 60 m
- **Points per second:** 60 000
- **Rotation frequency:** 20 Hz (set as int(1/FIXED_DT))

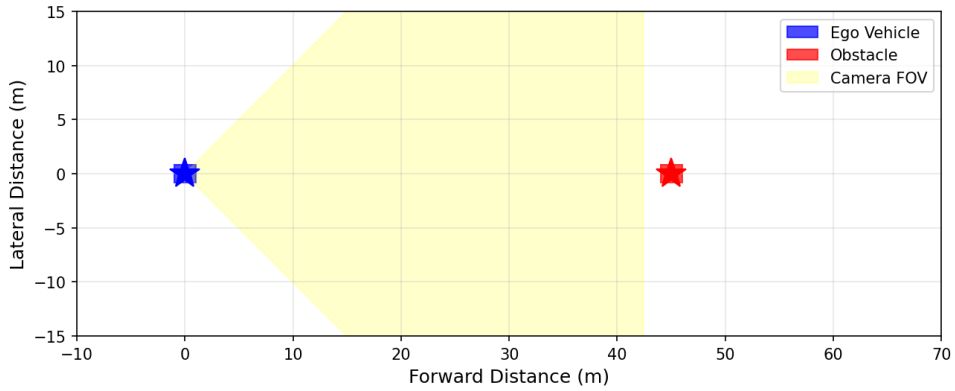

Figure 2: Camera View

- **Vertical FOV:** $+10°$ upper, $-30°$ lower
- **Pose:** roof mount at z = 2.5 m
- **Timing:** sensor_tick = 0.05 s (frame-synchronous with physics & camera) [10, 9]

Rationale:

- **Vertical structure at close range.** 64 channels allow for enough vertical detail to successfully detect the closest forward surface (bumpers, legs or torso, truck faces)–a crucial aspect of calculating time to contact and stopping distance [11].
- **Look-ahead and density.** At 60 k pts/s with 20 Hz, there is approximately 3 000 points per revolution. This results in approximately 750 points per frame in the 90° forward sector; a few hundred remain after ego-lane filtering (see below) and are more than sufficient for a reliable nearest-surface computation [25].
- **Task-adaptive vertical FoV.** The $-30° \ldots +10°$ sector concentrates on the road and bottom front section, providing a high likelihood of near-ground detection and suppressing clutter in the upper regions.
- T**Light synchronization of sweep.** Rotation_frequency = 20 Hz and sensor_tick = 0.05 s guarantees one 360-degree sweep every simulator tick. Also prevents skewing due to partial frame timing during fusion.

Ego-corridor gating (before association):
LiDAR points are filtered in camera coordinate space with physically informed thresholds:

- **Forward Cutoff:** $X > 2.0\,\text{m}$ to eliminate self-collisions with hood, bumper, or roof.
- **Lateral Corridor:** $|Y| \leq 2.5\,\text{m}$ ($\approx 5\,\text{m}$ lane envelope) with emphasis on the envelope directly in front of the collision.
- **Height range:** $-0.5\,\text{m} \leq Z \leq 3.0\,\text{m}$. This captures near-ground echoes (curb/underrides) up to tall vehicle faces. This lessens the clutter and ensures a stable nearest-distance signal that operates AEB.

3.6 Camera, LiDAR Calibration and Height Alignment

An accurate AEB test demands that camera pixel spaces, LiDAR points, and vehicle/road shapes coexist in a unified metric space. For simulation purposes, we could leverage the exact ground truth transformation in CARLA simulation software. This would still follow standard principles to ensure transferability to real-world hardware. This section unfolds:

- (i) **camera intrinsic parameters,**
- (ii) **parameters of the LiDAR model,**
- (iii) **extrinsic transformation** between the LiDAR and camera coordinate frames,
- (iv) **alignment** between height/pitch references on the ground plane, and
- (v) **error metrics.**

All notation used henceforth adheres to the script notations: camera frame with X pointing forward, Y to the left, Z up; image coordinate (u,v) in pixels.

3.6.1 Camera intrinsics

The RGB camera is set to **960×540** px with a FOV = 90°. For the pinhole camera model (no distortion), the intrinsic $\boldsymbol{K}$ [13, 14]

$$\begin{aligned} K &= \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}, \\ f &= \frac{W}{2\tan\left(\frac{\mathrm{FOV}}{2}\right)}, \\ c_x &= \frac{W}{2}, \quad c_y = \frac{H}{2}. \end{aligned} \tag{2}$$

With $W = 960,\ \mathrm{FOV} = 90^\circ \Rightarrow \tan(45^\circ) = 1,$ we obtain the numerical intrinsics:

$$\begin{aligned} &f = 480 \text{ px}, \quad c_x = 480 \text{ px}, \quad c_y = 270 \text{ px}, \\ &K = \begin{bmatrix} 480 & 0 & 480 \\ 0 & 480 & 270 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{3}$$

The standard camera in CARLA is distortion-free. Hence, radial or tangent terms in K_1, K_2, p_1, p_2 can be assumed to be zero. This is exactly what is done in the projection function of the **project_to_image()** function in the script.

3.6.2 LiDAR model (channel elevations and sweep timing)

The 64-channel ray-cast LiDAR model is characterized by:

- **Vertical FoV:** $[\alpha_{\min}, \alpha_{\max}] = [-30^\circ, +10^\circ]$; channel elevation angles α_i are *linearly spaced* in this interval.
- **Azimuthal sampling:** a full 360° sweep at 20 Hz (matching the fixed simulation step), yielding one complete revolution per frame.

- **Range:** 0–60 m; returns beyond this are discarded.

For a channel elevation α, azimuth θ, and range r, the local LiDAR point is [10]

$$\mathbf{p}^{\text{lidar}}(r,\theta,\alpha) = \begin{bmatrix} r\cos\alpha\cos\theta \\ r\cos\alpha\sin\theta \\ r\sin\alpha \end{bmatrix}.$$

These points are generated with high precision as a 3D cloud in the LiDAR coordinate frame (the script reads the XYZ data directly out of the raw data buffer). Key to this is the temporal sampling rate of rotation = 20 Hz with sensor_tick = 0.05 s. This ensures one full cloud per world tick [9].

3.6.3 Extrinsics: LiDAR–camera hand–eye from CARLA transforms

Let $T_{\text{cam}}^{\text{world}}$ and $T_{\text{lidar}}^{\text{world}}$ be the 4×4 homogeneous transforms (rotation $\mathbf{R}$, translation $\mathbf{t}$) mapping each sensor's local frame to *world*; CARLA exposes these via `get_transform().get_matrix()`. The camera-to-world inverse $T_{\text{world}}^{\text{cam}} = (T_{\text{cam}}^{\text{world}})^{-1}$ is used in the code. The LiDAR→camera extrinsic is then

$$T_{\text{lidar}}^{\text{cam}} = T_{\text{world}}^{\text{cam}}\, T_{\text{lidar}}^{\text{world}}.$$

Given a LiDAR point $\tilde{\mathbf{p}}^{\text{lidar}} = [x,\, y,\, z,\, 1]^\top$, the camera-frame point is

$$\tilde{\mathbf{p}}^{\text{cam}} = T_{\text{lidar}}^{\text{cam}}\,\tilde{\mathbf{p}}^{\text{lidar}}, \qquad \mathbf{p}^{\text{cam}} = \begin{bmatrix} X/W \\ Y/W \\ Z/W \end{bmatrix}^\top,$$

followed by pinhole projection:

$$u = f\frac{Y}{X} + c_x, \qquad v = f\frac{Z}{X} + c_y.$$

This is exactly what the said pipeline implements in `transform_points_to_camera(...)` and `project_to_image(...)` (with careful filtering of front-facing points $X > 2.0$ m to remove self-hits).

Baseline vector. In the ego (vehicle) frame, the camera is mounted at

$$\mathbf{t}_{\text{cam}}^{\text{veh}} = \begin{bmatrix} L_{\text{front}} + 1.5 \\ 0 \\ 1.8 \end{bmatrix}^\top \text{m},$$

where L_{front} = `vehicle.bounding_box.extent.x` (front half-length), while the LiDAR is at

$$\mathbf{t}_{\text{lidar}}^{\text{veh}} = \begin{bmatrix} 0 \\ 0 \\ 2.5 \end{bmatrix}^\top \text{m}.$$

Hence the LiDAR→camera translation in the vehicle frame is

$$\Delta\mathbf{t} = \mathbf{t}_{\text{cam}}^{\text{veh}} - \mathbf{t}_{\text{lidar}}^{\text{veh}} = \begin{bmatrix} L_{\text{front}} + 1.5 \\ 0 \\ -0.7 \end{bmatrix}^{\top} \text{m}.$$

For instance, if $L_front = 2.3$ m (for typical sedan), then $\Delta t \approx [3.8, 0, -0.7]$: The camera is placed 3.8 m in front and 0.7 m below the LiDAR. Relative orientations include the camera's pitch (-3°) and perhaps the camera's yaw or roll adjustments (as needed); the LiDAR is placed with the default upright orientation [12, 7].

3.6.4 Height and pitch alignment over the ground plane

Even with accurate extrinsics, it is always a good idea to check vertical alignment against the ground plane. Record LiDAR points in the ego-corridor (same X/Y/Z limits as with fusion), and fit a plane

$$\Pi : \ \mathbf{n}^{\top}\mathbf{x} + d = 0$$

(e.g., RANSAC). Transform Π into the camera frame using ${}^{\text{cam}}T_{\text{world}}$ to obtain

$$\Pi_{\text{cam}} : \ \mathbf{n}_{\text{cam}}^{\top}\mathbf{x} + d_{\text{cam}} = 0.$$

Under flat terrain and small pitch θ, the *horizon* in the image is approximately [15] [14]

$$v_{\text{horizon}} \ \approx \ c_y - f \ \tan(\theta_{\text{down}}),$$

with $\theta_{\text{down}} > 0$ for a downward-tilted camera. With $f = 480$ px and $\theta_{\text{down}} = 3°$, $\tan(3°) \approx 0.0524$, so

$$v_{\text{horizon}} \approx 270 - 25.1 \approx 245 \text{ px}.$$

Projecting corridor ground points should produce a tight band whose upper envelope sits *just below* this horizon, confirming both height difference ($\Delta h = 0.7$ m) and pitch.

3.6.5 Practical calibration steps in this setup

- **Step A: Fix the intrinsics.** Compute the pinhole intrinsics K from the chosen image size (W, H) and field of view FOV (in radians):

$$f \ = \ \frac{W}{2 \ \tan(\text{FOV}/2)}, \qquad c_x \ = \ \frac{W}{2}, \qquad c_y \ = \ \frac{H}{2}.$$

$$K \ = \ \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

  For this setup (W=960, H=540, FOV=90°), $\tan(45°)$=1 ⇒ f=480, c_x=480, c_y=270. The CARLA RGB camera is distortion-free by default, so radial/tangential coefficients are taken as zero.

- **Step B: Read extrinsics from the simulator.** At runtime, obtain the homogeneous transforms $T_{\text{cam}}^{\text{world}}$ and $T_{\text{lidar}}^{\text{world}}$ (each 4×4) via the sensor APIs. Form the *hand–eye* transform from LiDAR to camera:

$$T_{\text{lidar}}^{\text{cam}} \ = \ \left(T_{\text{cam}}^{\text{world}}\right)^{-1} T_{\text{lidar}}^{\text{world}}.$$

For any LiDAR point $\tilde{\mathbf{p}}^{\text{lidar}} = [x\ y\ z\ 1]^\top$,

$$\tilde{\mathbf{p}}^{\text{cam}} \;=\; T_{\text{lidar}}^{\text{cam}}\,\tilde{\mathbf{p}}^{\text{lidar}}, \qquad \mathbf{p}^{\text{cam}} \;=\; \begin{bmatrix} \frac{X}{W} & \frac{Y}{W} & \frac{Z}{W} \end{bmatrix}^\top,$$

where (X, Y, Z) are camera-frame coordinates (X forward, Y left, Z up). Project to image pixels using

$$u \;=\; f\,\frac{Y}{X} + c_x, \qquad v \;=\; f\,\frac{Z}{X} + c_y.$$

- **Step C: Check timing alignment.** Use CARLA synchronous mode with a fixed time step Δt and set every sensor's `sensor_tick` to the same value (here $\Delta t = 0.05$ s). Maintain frame timestamps t_k and verify

$$\left| t_k^{\text{cam}} - t_k^{\text{lidar}} \right| \;\approx\; 0,$$

ensuring that each camera image and LiDAR sweep are temporally coherent with the physics tick (one full LiDAR revolution per frame in this setup). [2, 3]

- **Step D: Validate with reprojection error.** Place a static target at known world points $\{\mathbf{X}_j\}$. For LiDAR returns $\tilde{\mathbf{x}}_i$ expressed in LiDAR coordinates, project them into the image via

$$\mathbf{q}_i^{\text{proj}} \;=\; \pi(K\,T_{\text{lidar}}^{\text{cam}}\,\tilde{\mathbf{x}}_i)\,,$$

where $\pi([x\ y\ z]^\top) = [x/z \;\; y/z]^\top$ followed by addition of (c_x, c_y) and multiplication by f as in the equations above. Compute the RMS reprojection error against measured/annotated image points $\mathbf{q}_i$:

$$\varepsilon_{\text{rms}} \;=\; \sqrt{\frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{q}_i^{\text{proj}} - \mathbf{q}_i \right\|_2^2}.$$

In simulation, ε_{rms} should be sub-pixel to a few pixels; larger values indicate a mount or axis-convention mismatch [8, 9].

- **Step E: Height/pitch sanity using the ground.** Fit a local ground plane Π to corridor LiDAR points (e.g., via RANSAC),

$$\Pi: \;\; \mathbf{n}^\top\mathbf{x} + d = 0,$$

transform Π into the camera frame and verify the expected visual horizon for a downward camera pitch θ_{down}:

$$v_{\text{horizon}} \;\approx\; c_y \;-\; f\,\tan(\theta_{\text{down}}).$$

With f=480 and θ_{down}=3°, $\tan(3°) \approx 0.0524 \Rightarrow v_{\text{horizon}} \approx 270 - 25.1 \approx 245$ px. A systematic offset suggests a small pitch or height bias. Two useful sensitivities:

$$\Delta u \;\approx\; f\,\tan(\delta\psi) \quad \text{(yaw error } \delta\psi\text{)}, \qquad \Delta v \;\approx\; f\,\frac{\delta h}{X} \quad \text{(height bias } \delta h \text{ at range } X\text{)}.$$

Example: at X=20 m, δh=0.05 m gives $\Delta v \approx 480 \cdot 0.05/20 \approx 1.2$ px.

- **Step F: Record the final calibration.** For reproducibility, store:
  - (i) the intrinsic matrix K;
  - (ii) the extrinsics $T_{\text{cam}}^{\text{world}}$ and $T_{\text{lidar}}^{\text{world}}$ (or $T_{\text{lidar}}^{\text{cam}}$);
  - (iii) the sensor timing Δt and `sensor_tick`; and (iv) the corridor bounds used in fusion (e.g., X>2.0 m, $|Y| \leq 2.5$ m, $-0.5 \leq Z \leq 3.0$ m).

These constitute the effective calibration and gating parameters of the AEB pipeline.

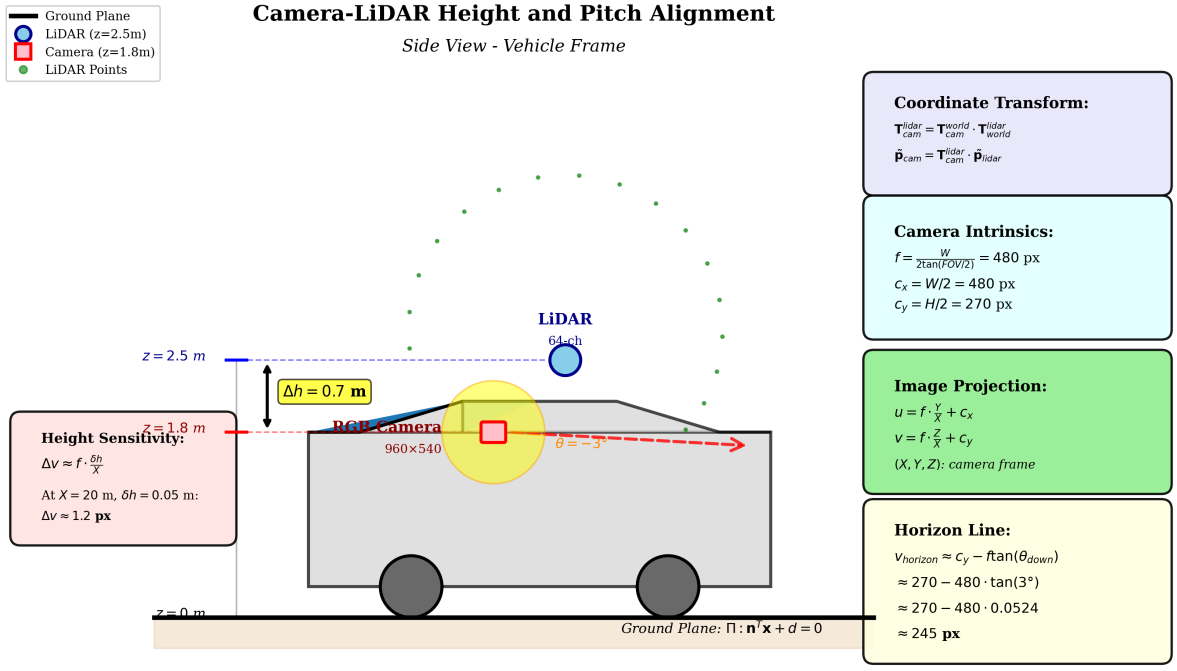

Figure 3: Camera-LiDAR height and pitch alignment geometry showing sensor mounting positions, coordinate frames, and projection equations.

3.6.6 Worked micro-example

Let us consider a LiDAR return that is just ahead of our car, at a distance of 20 metres, and centred after all transforms: $\mathbf{p}^{\text{cam}} = [X, Y, Z]^\top = [20, 0, 0]^\top$ m. With K above,

$$u = 480 \cdot \frac{0}{20} + 480 = 480, \qquad v = 480 \cdot \frac{0}{20} + 270 = 270,$$

i.e., the image center (480, 270) px—exactly where a centered, flat-front obstacle should appear. If the camera is pitched down by 3°, points at finite distance on the ground project to $v > v_{\text{horizon}} \approx 245$ px, forming a band just *below* the horizon, as expected. A residual vertical offset of ∼2 px at 20 m corresponds to

$$\delta h \approx X \frac{\Delta v}{f} \approx 20 \cdot \frac{2}{480} \approx 0.083\,\text{m},$$

a useful rule-of-thumb when diagnosing small mount tweaks.

3.7 Detector Configuration (YOLOv8n)

To link this calibrated sensing stack to its fusion and braking decision logic, this system uses a light and one-stage object detector called **YOLOv8n**, or 'nano,' as its semantic frontend input device. Additionally, its application is limited in that it is expected to deliver **class-filtered** two-dimensional hypotheses (2D bounding boxes, scores, and predicated labels) at a rate of events consistent with its cadence of 20 Hz, allowing LiDAR metric range (nearest surface, forward direction) input into each hypothesis.

Variant selection and latency budget: YOLOv8n is chosen, taking into account the requirement of upholding a maximum frame wall-clock time of 50 ms based on the fixed simulation step ($\Delta t = 0.05\,\mathrm{s}$). Even at input sizes of 960×540, nano stack architectures keep their latency low enough on commodity Windows 11 GPUs, leaving sufficient time for LiDAR processing, association, and valid AEB logic within this tick cycle itself. Under the scenario of AEB, for which penalisations of missed targets are generally more expensive than false positives, nano model design's recall-aiding characteristic, as required at medium-resolution ranges, is helpful, as all additional candidates are removed through **geometry gating** (ego-corridor filtering).

Classes and score thresholds: The script is limited to classes most relevant to emergency brakes:

$$\mathcal{C}_{\mathrm{AEB}} = \{\text{person}(0),\ \text{bicycle}(1),\ \text{car}(2),\ \text{motorbike}(3),\ \text{bus}(5),\ \text{truck}(7)\},$$

implemented via RELEVANT_CLASSES = 0,1,2,3,5,7 and a label map CLASS_NAMES. At inference, the detector is invoked with conf=0.25.

A **lower threshold of confidence** (0.25) is used, which leans towards recall at longer ranges and under marginal visibility conditions (from glare and low contrast), keeping faith that the LiDAR match and risk score will be able to filter out boxes that don't pose actual collision geometry. All this is consistent with the AEB philosophy of preferring early warnings over late warnings and misses.

Non-maximum Suppression (NMS). To remove duplicate boxes, the output of YOLO is then subjected to non-maximum suppression based on the default NMS toolbox provided by the library. Based on the Intersection-over-Union [16, 17, 19]:

$$\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{4}$$

and boxes with large IoU to a high-score box are removed. The script sets the threshold value of IoU unchanged at the ultralytics default, and this is perfect as it neither causes overlapping of distinct entities close-by nor clogs the environment with redundant instances, thus keeping the association phase speedy and clear. In case of future conditions of high density of people, such as platoons, this value can be made a parameter that can balance separation and suppression.

Integration: Ultralytics automatically manages internal resizing and letter boxing as required, and there is no need for pre-resize processing in this script, as this would result in memory copies and is unnecessary and inefficient.

Detection output is retrieved by r.boxes, and its contents are as follows:

- box.xyxy[0] (left–top–right–bottom pixel coordinates),
- box.conf[0] (post-NMS confidence),

- box.cls[0] (class id).

A class gate (if cls_id in RELEVANT_CLASSES) eliminates irrelevant categories up front, minimizing wasted fusion work [18].

Functional role in fusion stack: As for each box that is retained, there is a metric association step that is performed by the pipeline, and this involves LiDAR points that have been transformed into camera space and projected into pixels that are selected based on inclusion criteria, and these criteria can be based on $(u, v) \in$ box. **Forward Camera Axis distance** $X_{\min}$ among these associated points, one defines the object's nearest collision surface, and this gives time-to-collision per detection TTC $= \dfrac{X_{\min}}{v}$ and supports a distance-aware confidence fusion:

- **LiDAR confidence** based on associated point number (where threshold is adaptively decreasing as a function of range).
- **Camera confidence** from box.conf,
- **Range-weighted blending** that enhances LiDAR dominance for closer ranges and assigns more weight to the camera for longer ranges.

If YOLO gives no box and LiDAR shows that there is a very near fron-tal surface (say, for example $X_{\min}$ small or TTC short), a LiDAR fallback can still activate AEB, ensuring safe behaviour, especially when facing motion blur or partial occlusions, which may occur during low speeds.

3.8 LiDAR-to-Camera Transformation and Image Projection

With a class-filtered 2D hypotheses rate of 20Hz from its detector, it is essential to embed each LiDAR return into the camera frame and project it into the image plane. All of this is done explicitly and frame-synchronously, based on the poses provided by the simulator and its pinhole camera intrinsics, as previously defined.

3.8.1 Coordinate frames and transforms

We use three frames:

- **LiDAR frame** ($\mathcal{L}$): origin at the LiDAR, axes fixed to the sensor.
- **World frame** ($\mathcal{W}$): global map frame.
- **Camera frame** ($\mathcal{C}$): origin at the camera's optical center; X points forward (depth), Y is lateral, Z is vertical.

Per frame, the program queries CARLA for the 4×4 homogeneous transforms

$$T_{\mathcal{L}}^{\mathcal{W}}, \quad T_{\mathcal{C}}^{\mathcal{W}} \in SE(3),$$

and immediately forms the **world-to-camera** inverse

$$T_{\mathcal{W}}^{\mathcal{C}} = \left(T_{\mathcal{C}}^{\mathcal{W}}\right)^{-1}.$$

The **LiDAR-to-camera** extrinsic then follows by composition:

$$T_{\mathcal{L}}^{\mathcal{C}} = T_{\mathcal{W}}^{\mathcal{C}} T_{\mathcal{L}}^{\mathcal{W}}.$$

In code, points are treated as **row vectors** $\tilde{p} = [x\ y\ z\ 1]$ and post-multiplied by the transpose of each 4×4 matrix (equivalent to using column vectors with left multiplication). The sequence is:

1. **LiDAR → world:**

$$\tilde{\mathbf{p}}^{w} = \tilde{\mathbf{p}}^{c}\,(T_{L}^{w})^{\top}.$$

2. **World → camera:**

$$\tilde{\mathbf{p}}^{c} = \tilde{\mathbf{p}}^{w}\,(T_{W}^{c})^{\top}.$$

3. **De-homogenize:**

$$\mathbf{p}^{c} = [X\ Y\ Z] \;=\; \frac{1}{W}\,[\tilde{X}\ \tilde{Y}\ \tilde{Z}], \qquad W = \tilde{W}.$$

Only **front-facing** points are meaningful for projection and distance: the implementation first removes anything with very small or negative forward component, applying

$$X > X_{\min} = \texttt{MIN\_FWD\_X} = 2.0 \text{ m},$$

which reliably eliminates self-hits from bumper/hood and other mount artifacts.

3.8.2 Pinhole projection and intrinsics

The camera is pinhole with **no distortion**. Given image width $W = 960$, height $H = 540$, and FOV $= 90°$, the **focal length** and principal point are

$$f = \frac{W}{2\tan(\mathrm{FOV}/2)} = \frac{960}{2\tan 45°} = 480 \text{ px}, \qquad (c_x, c_y) = \left(\frac{W}{2}, \frac{H}{2}\right) = (480, 270) \text{ px}.$$

The intrinsics are

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

For any camera-frame point $\mathbf{p}^{c} = [X\ Y\ Z]$ with $X > 0$, pixel coordinates are computed exactly as in the implementation:

$$u = f\,\frac{Y}{X} + c_x,$$
$$v = f\,\frac{Z}{X} + c_y.$$

A **depth guard** $X > 0.1$ protects against division by near-zero depth, and a **viewport test** keeps only points with $0 \le u < W$ and $0 \le v < H$. This yields two arrays of equal length: the 2D pixel coordinates and their corresponding 3D camera-frame vectors [13, 12].

3.8.3 Corridor gating in the camera frame

Direct projection is, by itself, not sufficient for range prediction of stable AEB, and we have added task-specific geometric priors on the camera frame before projection into detector boxes:

- **Forward corridor:** $X > 2.0$ m (already applied).
- **Lateral bound:** $|Y| \leq 2.5$ m (one lane plus margin).
- **Height band:** $-0.5 \leq Z \leq 3.0$ m (from near-ground to tall vehicle faces).

Formally, the valid set is

$$\mathcal{S} = \{\mathbf{p}^c : X > 2.0,\ |Y| \leq 2.5,\ -0.5 \leq Z \leq 3.0\}\,.$$

Filtering to $\mathcal{S}$ removes roadside clutter, overhead structures, and residual self-points, **stabilizing the nearest-surface estimate** that ultimately drives TTC and stopping-distance checks [24].

3.8.4 Numerical and implementation:

- **Row vector math:**Because they're batched as **(Nx4)** row vectors, each point is multiplied by the transpose of each transformation. This is algebraically equivalent to a left multiplication by a column vector, but this way we avoid a Python loop for each point and stay within a BLAS call.
- **Homogeneous Normalization:** Following the 4×4 chain, we have a division by the homogeneous part *W*, and this gives us *(X, Y, Z)*, true Euclidean space, because, in CARLA sensor transforms, *W* is always 1, and this is future-proof.
- **Depth and viewport guards:** Both checks, *X>0.1* and the pixel window test, avoid invalid divisions and indexing outside images. They, together with the corridor gate, reduce association noise significantly.
- **Units remain consistent:** All transformation measures and distances are written in meters, and only angles get translated into pixels through f and the center of projection, making it possible for the forward vector X, used in the equation of kinematic brakes, to be employed without having to convert its units.
- **Synchronous timing:** With LiDAR rotation rate set at 20Hz and sensor tick rate at 0.05 s, each RGB frame is combined with a complete sweep of LiDAR scan of 360° from the same simulation tick, essential for condition $X_{\min}$ and stable **TTC**

3.9 Perception–Fusion and Physics-Grounded AEB Actuation

Since each LiDAR return is represented within the camera frame and projected onto the image, we can now combine semantic proposals from the detector with metric ranges from LiDAR into estimates of collision risk and actuate braking responses accordingly. This section combines perception fusion logic and AEB decision/control operation at a rate of 20 Hz.

3.9.1 Association and per-object range/TTC

Let $\mathcal{B} = \{B_k\}$ be the post-NMS detections (class $c_k \in \{0, 1, 2, 3, 5, 7\}$, confidence $s_k \in [0, 1]$, box $B_k = [x_1, y_1, x_2, y_2]$). Let $\mathcal{S} = \{\mathbf{p}_i^{\mathcal{C}}\}$ be the LiDAR points in the camera frame after corridor gating ($X > 2.0$ m, $|Y| \leq 2.5$ m, $-0.5 \leq Z \leq 3.0$ m), each paired with its pixel (u_i, v_i). Associate points to boxes via

$$\mathcal{S}_k = \left\{ \mathbf{p}_i^{\mathcal{C}} \in \mathcal{S} \;\middle|\; (u_i, v_i) \in B_k \right\}.$$

Define the forward distance to the nearest surface:

$$D_k = \min_{\mathbf{p}_i^{\mathcal{C}} \in \mathcal{S}_k} X_i \quad \text{if } \mathcal{S}_k \neq \emptyset.$$

Given ego speed $v = \|\mathbf{v}_{\text{ego}}\|$, the time-to-collision for box k is

$$\text{TTC}_k = \begin{cases} \dfrac{D_k}{v}, & v > 0.1 \text{ m/s}, \\ \infty, & \text{otherwise.} \end{cases}$$

Independently, keep a LiDAR-only nearest forward distance [20]

$$D_\ell = \min_{\mathbf{p}_i^{\mathcal{C}} \in \mathcal{S}} X_i, \qquad \text{TTC}_\ell = \frac{D_\ell}{v} \text{ (if } v > 0.1 \text{ m/s).}$$

3.9.2 Distance-aware confidence fusion

Fuse geometric and semantic evidence as

$$\boxed{\hat{s}_k = w_\ell(D_k)\, s_k^\ell + w_c(D_k)\, s_k^c, \qquad w_\ell(D_k) + w_c(D_k) = 1}$$

with

$$s_k^\ell = \min\left(1,\ \frac{n_k}{\tau(D_k)}\right), \quad n_k = |\mathcal{S}_k|, \quad \tau(D_k) = \max\big(3,\ \lfloor 10 - 0.5\, D_k \rfloor\big),$$

and

$$s_k^c = s_k, \qquad w_c(D_k) = \min\big(0.7,\ \max(0,\ D_k/30)\big), \quad w_\ell(D_k) = 1 - w_c(D_k).$$

(An additional proximity boost clamps $\hat{s}_k$ toward 1 when $D_k < 15$ m.)

3.9.3 Physics-based stopping model

The stopping distance combines reaction and braking components [21, 22]:

$$d_{\text{stop}}(v) = v\, t_r + \frac{v^2}{2\, a_{\max}}, \quad t_r = 0.5 \text{ s}, \quad a_{\max} = 8.0 \text{ m/s}^2.$$

A 15% safety margin is applied internally, i.e. $1.15\, d_{\text{stop}}(v)$.

3.9.4 High-risk decision (per frame)

For any detection k with $\mathcal{S}_k \neq \emptyset$, declare fusion risk if

$$\text{risk}_k \;=\; \Big[\; \text{TTC}_k < 2.0 \text{ s} \;\;\vee\;\; D_k < 1.1\, d_{\text{stop}}(v) \;\Big] \;\wedge\; \Big[\; \hat{s}_k > 0.15 \;\Big] \;\wedge\; \Big[\; n_k \geq 2 \;\Big].$$

Aggregate over boxes as $\text{risk}_{\text{fuse}} = \bigvee_k \text{risk}_k$. A geometry-only safeguard triggers when

$$\text{risk}_\ell \;=\; \Big[D_\ell < 0.8\, d_{\text{stop}}(v) \;\;\vee\;\; \text{TTC}_\ell < 1.5 \text{ s} \Big] \quad \text{(applied only if } D_\ell < 20 \text{ m)}.$$

The per-frame risk flag is

$$\text{risk} = \text{risk}_{\text{fuse}} \;\vee\; \text{risk}_\ell.$$

3.9.5 Temporal smoothing and state machine

To suppress one-frame spikes, use a 3-frame majority: maintain the last three risk flags; set risk_{maj} = true if at least 2 of 3 are true [23, 40]. The finite-state controller advances on risk_{maj} (only when $v > 0.5$ m/s):

- **Inactive** $\rightarrow$ **Warning**: when risk_{maj} becomes true.
- **Warning** $\rightarrow$ **Braking**: if risk_{maj} persists on the next tick.
- **Braking** $\rightarrow$ **Inactive**: once the hazard clears and the vehicle is nearly stopped, i.e.,

$$D_{\min} > D_{\text{release}} = 8.0 \text{ m} \quad \text{and} \quad v < 1.0 \text{ m/s}, \qquad D_{\min} = \min\big(\{D_k\} \cup \{D_\ell\}\big).$$

Figure 4: State Warning

3.9.6 Brake command shaping

During **Braking**, throttle is zeroed, steering is neutralized, and brake is commanded as a monotone function of TTC:

$$\boxed{b = \text{clip}\big(0.3,\; 1.0,\; 2.5 - 1.5\,\text{TTC}\big),}$$

where $\text{TTC} = D_{\min}/v$ for $v > 0.1$ m/s (otherwise treated as large). The lower bound 0.3 guarantees tangible deceleration; the upper bound 1.0 represents full braking authority.

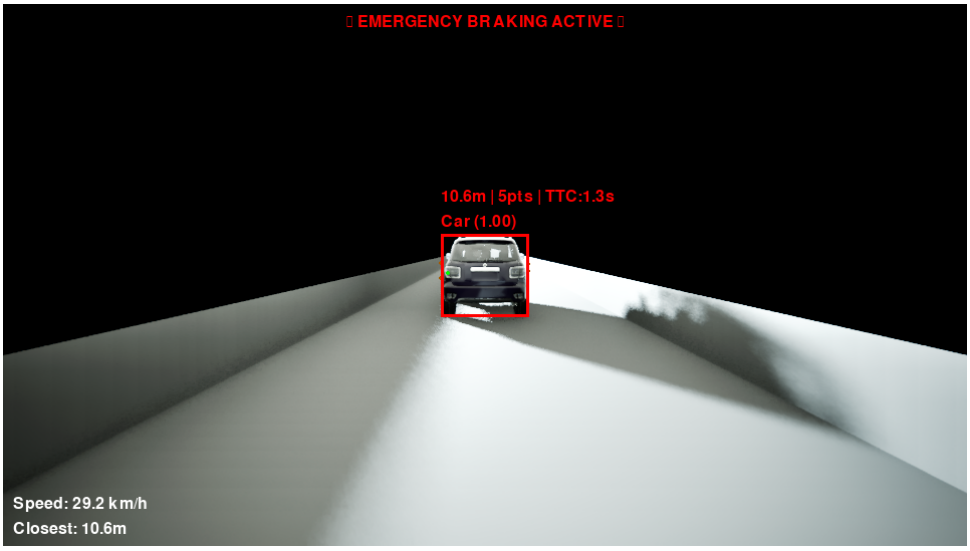

Figure 5: State Breaking

3.9.7 Rationale for thresholds and guards

- TTC < 2 **s or** $D_k < 1.1\, d_{\mathbf{stop}}(v)$**:** couples a time-critical trigger with a physics-critical distance check, covering both high-speed/medium-range and low-speed/short-range threats.
- $\hat{s}_k > 0.15$ **and** $n_k \geq 2$**:** resists single-point noise and stray boxes while preserving recall (geometry-only fallback protects against vision misses).
- **3-frame majority at 20 Hz:** adds only ~ 0.10–0.15 s inertia yet meaningfully reduces chatter from transient association [26, 27].

3.10 End-to-end step-through (per 50 ms tick)

1. **Sensing:** acquire the RGB frame and a full LiDAR revolution (synchronous timing).
2. **Detection:** run YOLOv8n; retain AEB-relevant classes.
3. **Projection:** transform LiDAR to camera, project to pixels, apply corridor gating.
4. **Association:** for each box, collect in-box points; compute D_k, TTC_k, and $\hat{s}_k$.
5. **Risk:** evaluate fusion triggers and LiDAR-only safeguard; apply 3-frame majority.
6. **FSM:** transition Warning/Braking as needed; test release conditions.
7. **Actuation:** apply brake shaping $b = \mathrm{clip}(0.3, 1.0, 2.5 - 1.5\,\mathrm{TTC})$ in Braking; otherwise, pass user controls.
8. **Telemetry:** render state, nearest distance, and projected LiDAR overlays for audit.

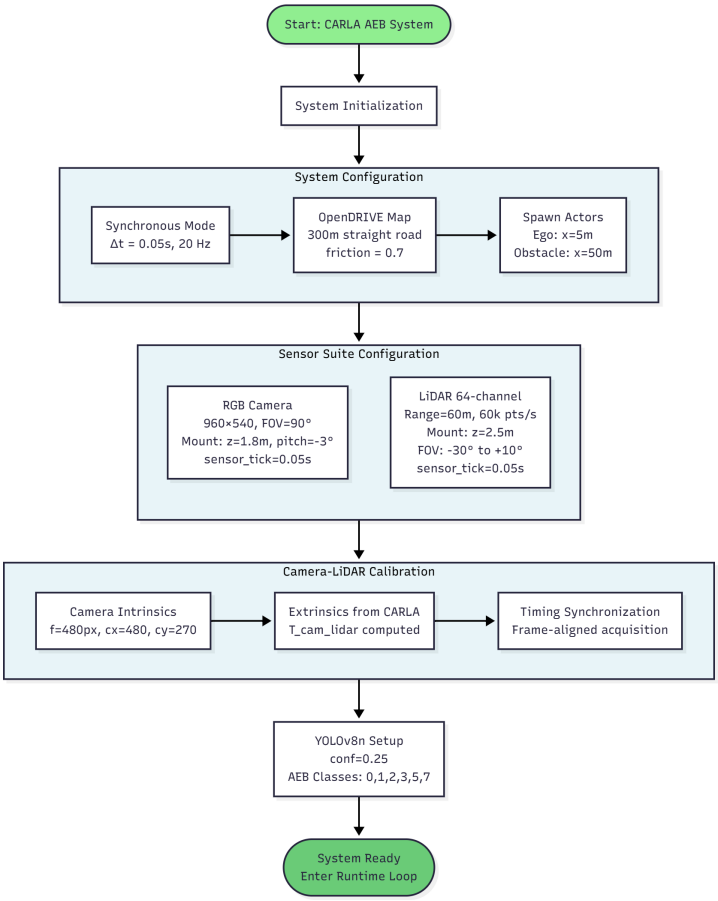

Figure 6: Flowchart 1: System Setup & Initialization

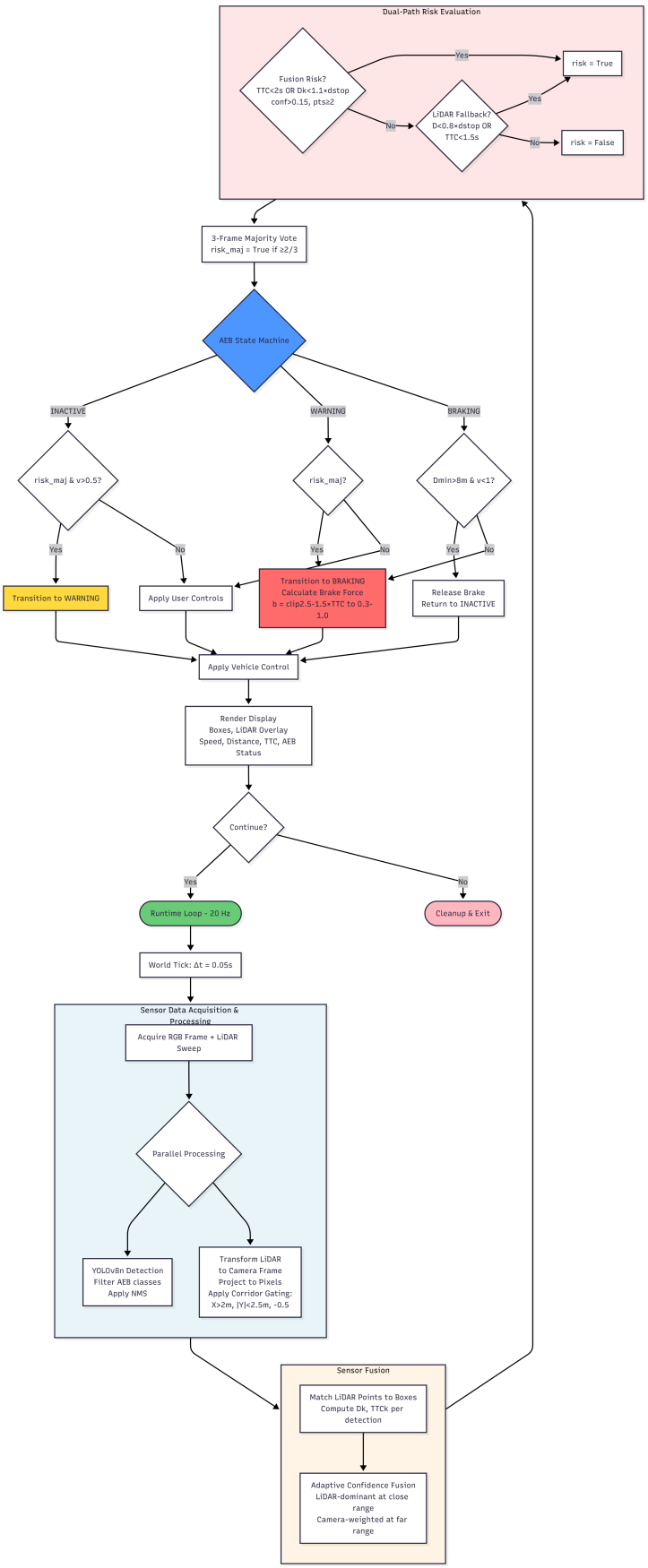

Figure 7: Flowchart 2: Runtime Perception & AEB Control Loop

4 OpenDRIVE Maps: Concept, Usage in CARLA, and Why We Chose Them

To enable reusable AEB tests that don't require the computational resources of city-wide assets, our framework incorporates ASAM OpenDRIVE, a road network description format that compactly and machine-readably represents lanes, road marking, curvature, junction geometry, and other non-moving road properties. ASAM OpenDRIVE is intended for driving simulation and validation of ADS and ADAS systems, encoding road network logic, not photo-realistic scenery, that road simulators, such as CARLA, can render into a simplified 3D geometry for simulation purposes only. By following an open, widely adopted format, this enables maps that last, being portable between tools, and used over longer-term research cycles [28] [29].

In CARLA, a map is notionally divided into two parts:

- (i) the OpenDRIVE road description (roads, lanes, dimensions, lane types, intersections, traffic signals),
- and (ii) the 3D town model that comprises scene objects and resources such as meshes, textures, vegetation, buildings, and more.

CARLA can be used through its Python API, allowing one to work solely with road network defined by OpenDRIVE, without any consideration for heavy environment content, or investigate car and road interactions and longitudinal control, such as applications of Autonomous Emergency Breaking (AEB). In this case, only (i) is required, and (ii) is only optional [4].

CARLA provides a straightforward way for OpenDRIVE ingestion independent of any simulation environment and CARLA applications. You pass through the XODR contents as a string via **'client.generate_opendrive_world(opendrive, parameters)'**, and CARLA creates a light-weight road mesh based on parameters (e.g., point sampling, longest segment, junction smoothing, wall height, shoulders). Even then, this call returns a world that is ready for immediate physics interaction, sensor, and navigation capability, bypassing loading entirely asset-heavy Town maps altogether. And this is precisely what we have used as our testbed for 'straight road' cases, providing only a minimal XODR composing a one-carriageway road of 100 m and appropriate parameters for generating a simple road geometry that is deterministic [30].

Why this is an issue for performance: these Town maps have lots of visually interesting content but are asset-heavy (textures, foliage, large geometries, and shadowing), all of which compete for GPU memory, draws, and CPU streaming, all of which compete for perception processing time on our system. Large town environments, such as Town10, are renowned for pushing mid-range GPUs, and this is one of the pain points of evaluating perception stacks, as reported by users of these stacks. By comparison, our one-road OpenDRIVE environment is extremely sparse, consisting of only the necessary road geometry and some basic default props, enabling us to allocate time from frame time into YOLO processing, LiDAR processing, and fusion runs at a predictable rate of 20 Hz [31].

5 Performance Evaluation of AEB in Euro NCAP Scenarios

The Euro NCAP (European New Car Assessment Programme) is a consumer safety rating agency, which thoroughly tests advanced driver assistance systems, such as Autonomous Emergency Braking (AEB) [32] [33]. It is a part of the Safety Assist rating, in which Euro NCAP performs a series of standardized car-to-car AEB tests. During these tests, several scenarios have been set to check the AEB system on a car-to-car basis. The scenarios include Car-to-Car Rear Stationary, referred to as CCRs, in which the vehicle has to approach a stationary target car; Car-to-Car Rear Moving, referred to as CCRm, in which a slower-moving lead car is ahead; and Car-to-Car Rear Braking, referred to as CCRb, in which a lead car moves first and suddenly brakes. The AEB system's job is to warn the car's driver and apply the brakes to avoid a collision.

5.1 Scenario 1: CCRs (Car-to-Car Rear Stationary)

Scenario: For the CCRs tests, the ego vehicle begins approximately 50 meters behind a stationary target car, a soft crash-able target simulated as a car, on a straight road [33]. The testing mandated by Euro NCAP involves a series of speed activations ranging from 10 to 50 km/h [32]. The user's AEB system was tested over 20 scenarios ranging from approximately 13 km/h to approximately 50 km/h initial speed. The test did not include any braking action from the human driver, as the ego vehicle solely relied on AEB to avoid the stationary object. The results from the test scenarios are presented in Table 2.

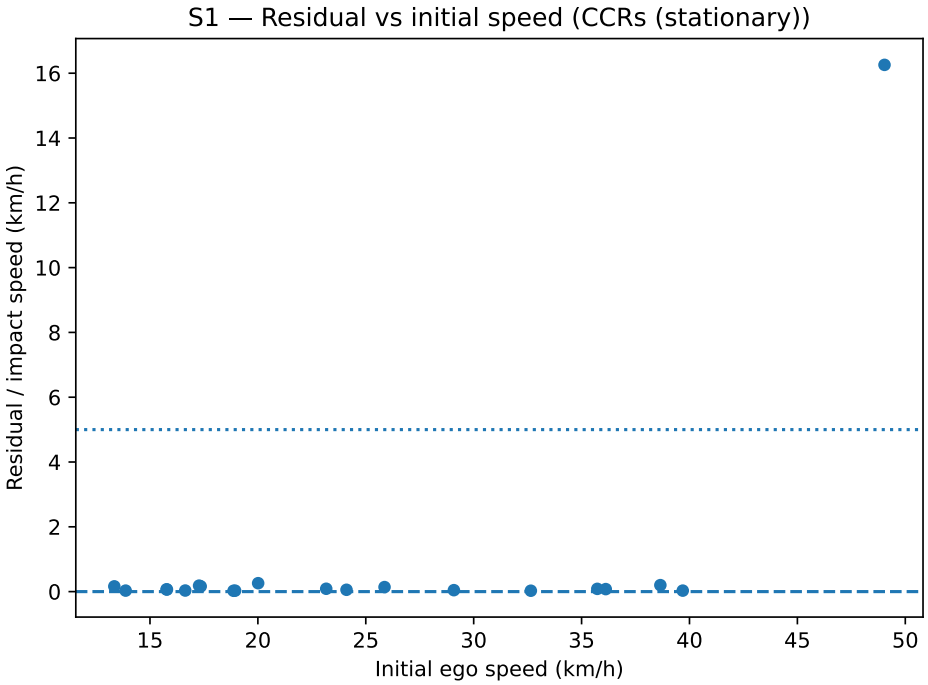

Figure 8

The AEB system could avoid collision successfully in 19 situations from 20, testing up to the initial speed of approximately 39.7 km/h. But at the highest tested speed, approximately 49 km/h, the ego car could not avoid the stationary car, a collision took

place – soft target struck, along with a residual impact speed of approximately 16 km/h. The final impact speed, as depicted in Table 2, remains close to 0 km/h for initial speeds below 40 km/h. But it drastically increases to 16 km/h at 50 km/h. The minimum value of Time-to-Collision per scenario reduced as the initial speed incremented. Even at the lowest initial speed of approximately 13 km/h, TTC at closest approach was about 0.97 seconds, but it reduced as low as 0.14 seconds, which means a collision is imminent, while moving at approximately 50 km/h. For speeds ranging from 10-20 km/h, TTC at closest approach always remained well above 0.7-1.0 seconds. But for speeds higher than 30 km/h, the minimum TTC reduced significantly below 0.5 seconds, implying the AEB system waited till the very end to start braking as the initial speed increased.

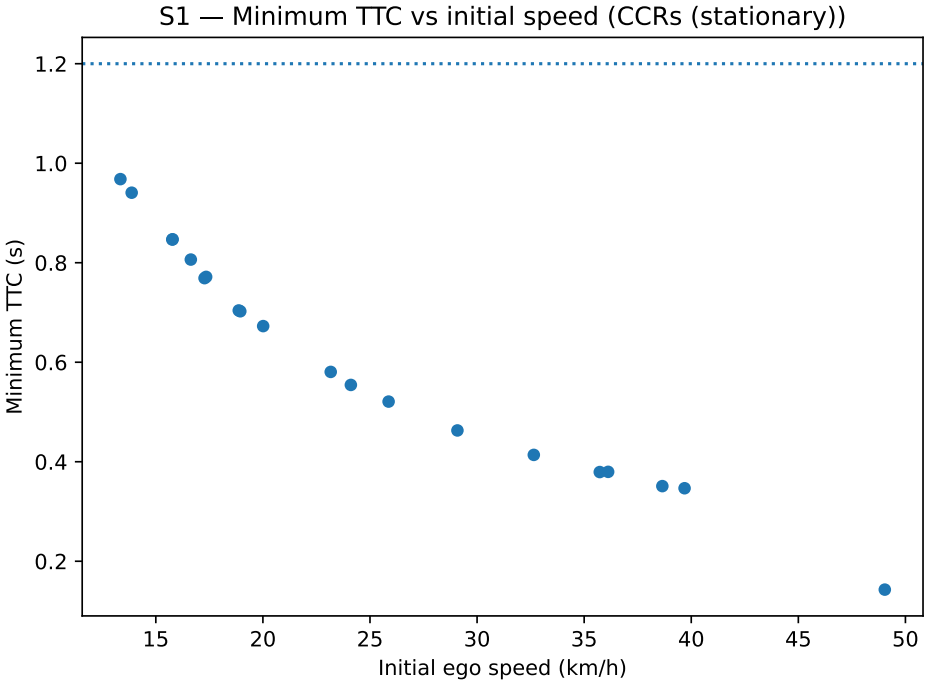

Figure 9

Timing & Activation: It can be observed from the log messages that the AEB activation timing as well as AEB response timing were speed-dependent. In the low-speed scenario (speeds ranging from 13-20 km/h), it could be noticed that the AEB activation was delayed – for example, brakes were not activated till 2.4 seconds into the run at 13 km/h in Run 0. This is presumably because the time-to-collision values were high. In contrast, AEBs were activated very close to start, within 0.06-0.27 seconds from the start, in the moderate-speed range of 17-39 km/h. This contradictory approach, while AEBs were activated very early in the moderate-speed range, as well as delayed in the lowest speed range, might be due to control functionality logic as well as camera sensing capabilities. It is important to note, moreover, that in the highest speed scenario, there was a substantial delay in AEB activation, as the first genuine sensing of the stationary car took place approximately 6.5 seconds into the run, which is very late, as the two cars were approximately ¡15 m apart. The AEB, as a result, got activated only about 0.27 seconds prior to impact, which, in fact, was long past the point of impact. The result is a false negative scenario of camera functionality, as it can be observed from the log messages, the camera never detected the stationary object in this scenario, thereby leaving

the lidar task to activate the AEB at close range. In fact, as can be observed from the log messages, the camera never detected the object in the CCRs scenario. The functionality, as a matter of fact, experiences a known difficulty in sensing nearby stationary objects at long range, as a result of which AEBs are delayed until very close to the object in high closure rates. [36] In our testing scenario, as a matter of fact, it resulted in a failed scenario, namely, 50 km/h collision.

Table 2: AEB CCRs scenario result parameters, stationary target context. The results include the initial speed of ego, collision avoided, the value of the remaining impact speed, minimum time to collision, AEB braking onset time, reaction time from detection to braking, as well as the peak value of the braking jerk.

Run ID	Ego Speed (km/h)	Collision Avoided?	Residual Speed (km/h)	Min TTC (s)	Brake Onset Time (s)	Reaction Time (s)	Peak Jerk (m/s^3)
0	13.3	Yes	0.2	0.97	2.412	2.412	720.9
1	13.9	Yes	0.0	0.94	0.266	0.266	651.0
2	15.8	Yes	0.1	0.85	0.263	0.263	654.9
3	15.8	Yes	0.1	0.85	0.268	0.268	652.1
4	16.6	Yes	0.0	0.81	0.268	0.268	652.1
5	17.3	Yes	0.2	0.77	0.267	0.267	651.9
6	17.3	Yes	0.2	0.77	0.267	0.267	654.9
7	18.9	Yes	0.0	0.70	0.269	0.269	653.9
8	18.9	Yes	0.0	0.70	0.266	0.266	651.1
9	20.0	Yes	0.3	0.67	0.266	0.266	651.0
10	23.2	Yes	0.1	0.58	0.264	0.264	668.8
11	24.1	Yes	0.1	0.55	0.066	0.066	150.1
12	25.9	Yes	0.3	0.48	0.066	0.066	150.1
13	29.1	Yes	0.0	0.43	0.066	0.066	150.1
14	32.6	Yes	0.0	0.38	0.066	0.066	150.1
15	35.7	Yes	0.1	0.38	0.264	0.264	54.1
16	36.1	Yes	0.1	0.36	0.066	0.066	150.1
17	38.7	Yes	0.1	0.33	0.066	0.066	150.1
18	39.7	Yes	0.1	0.31	0.066	0.066	150.1
19	49.0	No	16.3	0.14	0.267	0.267	53.5

Braking Effectiveness and Jerk: The AEB always succeeded in bringing the vehicle to a complete stop, at least up to 40 km/h, while in the failed scenario at 50 km/h, it significantly reduced the oncoming vehicle's speed from approximately 50 km/h to 16 km/h. Nevertheless, it has been noticed that the braking action became excessively abrupt. The deceleration jerk is particularly high, reaching values as high as 720 m/s^3 during a low-speed test, while it averaged $650m/s^3$ in most tests (Table 2). Jerk values as high as 650 m/s^3 are well beyond the expected comfort level. A value of 0.6 m/s^3 has been cited as being perceptible, while 0.9 m/s^3 can be considered the highest acceptable level for a comfortable jerk in automated braking systems [38]. The high jerk associated with AEB indicates a very abrupt rise in the braking forces, essentially a "hard slam" on the brakes. The system would presumably have waited for the very last second to start braking in some scenarios, starting with a very hard braking of the vehicle. The fact is, in some higher vehicle speeds, like 35-40 km/h, it seemed to record a highest jerk value of 54 m/s^3, suggesting a smooth start of braking perhaps because of prior notice given to the system, along with application of staged braking. In fact, most commercial AEB systems follow a more relaxed staged braking action initiated, for example, approximately at a 2.5s to 3s TTC, further followed by full braking action initiated approximately at a 1.0s to 1.5s TTC [37]. In our experiments, it appears as if it would have completely skipped the initial partial braking action, directly transitioning to a hard braking action as in the scenario at a very later time.

In brief, while the AEB system in the stationary car scenario performs well until modest urban speeds, it has a poor performance at higher inter-urban speeds (above 50 km/h). Improvements in sensor fusion processing, as well as earlier system activation, would be mandatory in order to fulfill the demands of current safety norms at higher

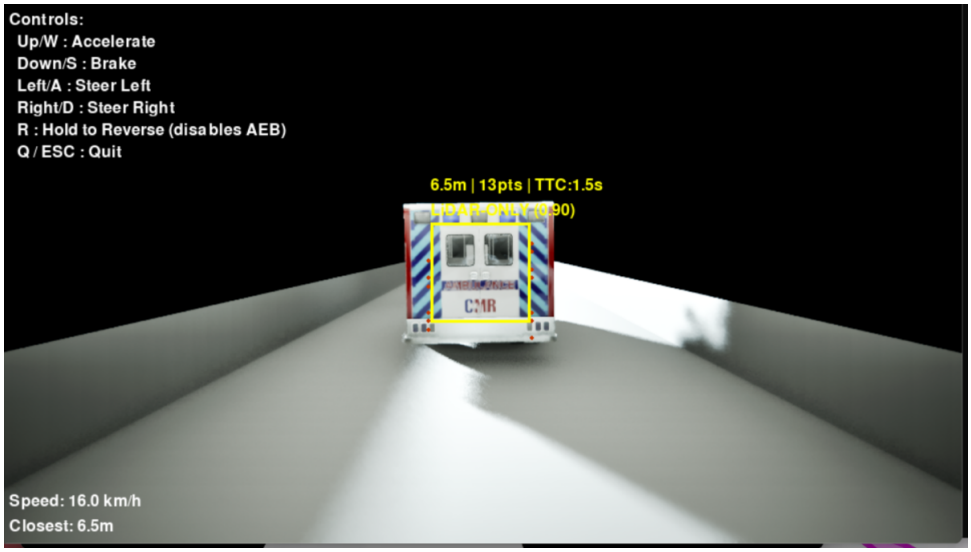

Figure 10: Scenario 1: CCRs

vehicle speeds.

5.2 Scenario 2: CCRm (Car-to-Car Rear Moving)

Scenario: In the CCRm scenarios, a lead car (Global Vehicle Target) moves at a fixed speed of 20 km/h, while the ego vehicle, being faster, pursues from the rear. The initial separating gap was approximately 50 m. The Euro NCAP procedure demands AEB system testing in CCRm scenarios from 30 to 80 km/h ego vehicle speed, fixed at 20 km/h for the lead car [32]. The user's system has been evaluated on 20 scenarios, testing the ego vehicle's initial speed varied from approximately 31 km/h to 59 km/h while maintaining a fixed lead car speed of about 22 km/h. Table 3 below highlights the results.

Table 3: Results for AEB CCRm scenario (moving lead vehicle at ∼20 km/h). Ego speeds 30–60 km/h were tested. All runs avoided collision (no contact). TTC = time-to-collision; Δv = ego–lead speed difference.

Run ID	Ego Speed (km/h)	Lead Speed (km/h)	Collision Avoided?	Residual Speed (km/h)	Min TTC (s)	Brake Onset Time (s)
0	31.2	22.0	Yes	0.08	4.13	0.000
1	32.6	22.0	Yes	0.12	3.51	0.000
2	32.9	22.0	Yes	0.34	3.38	0.000
3	34.1	22.1	Yes	0.02	3.05	0.000
4	35.7	22.0	Yes	0.06	2.68	0.000
5	37.2	22.0	Yes	0.03	2.44	0.000
6	38.9	22.0	Yes	0.06	2.30	0.000
7	39.7	22.0	Yes	0.05	2.11	0.000
8	40.0	22.1	Yes	0.05	2.00	0.000
9	41.0	22.1	Yes	0.05	1.89	0.000
10	43.8	22.0	Yes	0.08	1.63	0.000
11	44.0	22.0	Yes	0.00	1.55	0.000
12	44.6	22.1	Yes	0.00	1.48	0.000
13	44.9	22.0	Yes	0.00	1.37	0.000
14	47.6	22.0	Yes	0.00	1.25	0.000
15	49.0	22.0	Yes	0.00	1.63	1.501
16	56.4	22.0	Yes	0.00	1.22	1.978
17	57.3	22.0	Yes	0.00	1.15	1.982
18	57.6	22.0	Yes	0.00	1.15	1.892
19	59.2	22.0	Yes	0.00	1.08	1.921

Results: The AEB system demonstrated successful collision avoidance in all 20 tests

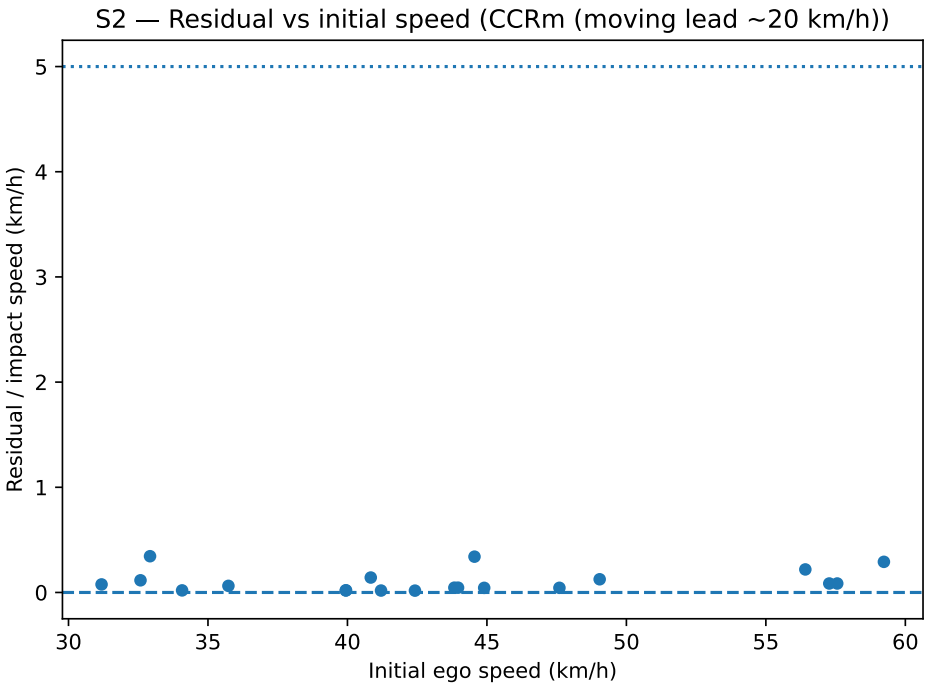

Figure 11

of the moving-target scenario. Even at the highest tested speed ($\sim 59\,\text{km/h}$, a $59 - 22 \approx 37\,\text{km/h}$ relative closing speed), the ego vehicle braked on time without hitting the lead vehicle. The residual speed at the point of closest approach or stop in all tests was practically 0 km/h (actually, under 0.5 km/h; see Table 3), which means the ego vehicle braked on time to stop fully or to match the speed of the lead vehicle. The relationship between initial and residual speed shows both are at, or very close to, 0 km/h as expected, since there was never an impact. The minimum TTCs, as expected, naturally decrease as the initial ego vehicle speed and the closing speed increase. At the lowest tested speed ($\sim 31\,\text{km/h}$), the minimum TTC was approximately 4.1 seconds (very safe), while at the highest tested speed ($\sim 59\,\text{km/h}$), the TTC reached as low as 1.08 seconds at the closest approach. In fact, in both cases the TTC stayed well above 1.0 second, which indicates a (small) safety margin in all cases. Note, however, that in all tests of the CCRm scenario the TTC never dropped below 1 second, as opposed to the CCRs scenario; this can be attributed to continuous tracking of the lead vehicle from the onset in CCRm, unlike CCRs.

Comparative Assessment: The AEB system's performance in CCRm scenarios is good. The system could avoid collisions at closing speeds of up to approximately 37 km/h. The Euro NCAP protocol would have tested up to 80 km/h ego speed (60 km/h), which we did not approach, but it appears the system might be operable at slightly higher ego speeds as well, although the TTC would fall below 1 s at approximately 60 km/h ego speed [32]. Nevertheless, avoiding a collision at approximately 60 km/h relative speed is commensurate with expectations for a high-end AEB system—for example, the upcoming UNECE AEBS regulation (R152) requires testing only up to 60 km/h for moving traffic, which our system complies with in this testing [35]. The upcoming regulation within the United States would require collision avoidance at 62 mph (approximately 100 km/h), though it particularly includes a "slow-moving lead" scenario (as in CCRm testing) at

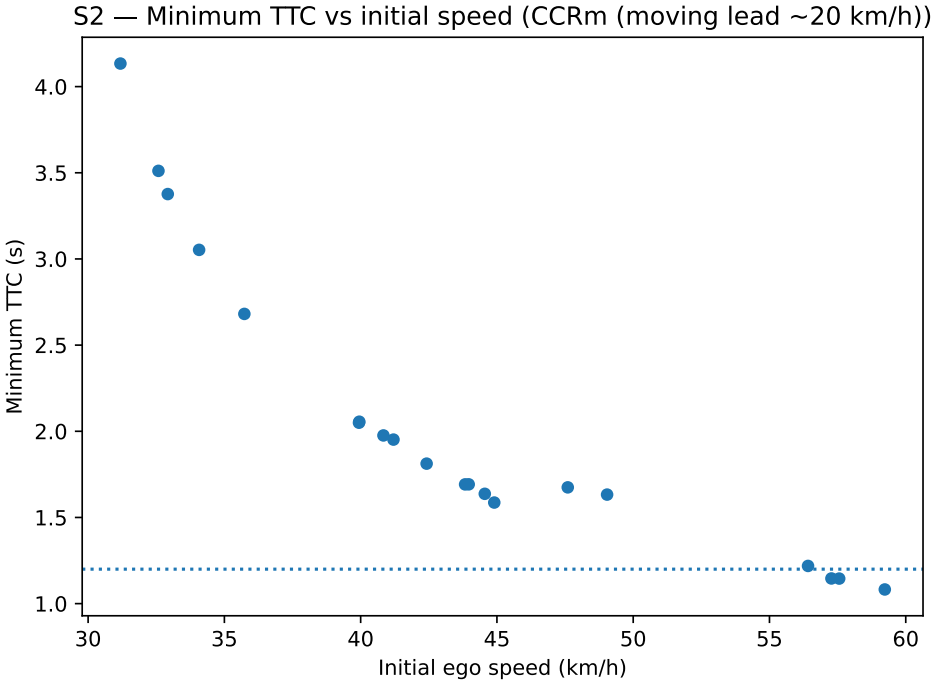

Figure 12

test velocities of 50 and 80 mph (approximately 50 and 80 km/h). Our testing at approximately 60 mph ego ($\sim$ 37 km/h closure degree) indicates it complies at 50, although it would require validation to see how it performs at 80 mph (60 km/h closure degree), though it would very likely hard-brake anyway.

In a real-world effectiveness context, strong CCRm performance is essential, as it is well known that a substantial percentage of highway collisions involve a faster vehicle approaching a slower vehicle. Research indicates that frontal AEB can potentially mitigate approximately 70% of rear-end collisions, especially those at moderate speed differentials, while AEB systems have already reduced rear-end collision rates by approximately 50% today [39]. The fact that the tested system could prevent all such collisions within our 30$-$60 km/h testing range dovetails nicely with these percentages.

It also underlines the advantage of early action, as it did not take too long, maintaining TTC $\geq$ 1.1 s at minimum, thereby precluding last-instant braking. This approach is much safer and more comfortable. If we consider the time margins, AEB triggering time in the 59 km/h vs. 22 km/h scenarios saw AEB activate approximately 1.9 s into the scenario, at a calculated TTC $\approx$ 3.3 s — leaving more than 2+ seconds for braking. The ego vehicle braked with about 1.1 s TTC remaining, suggesting it did not quite test the very edge of the physical system.

This approach is well designed and also implies that the system could be calibrated to make the AEB system activate approximately at a fixed TTC threshold value ($\sim$ 2.0$-$2.5 s), as proposed, for such moving target scenarios, which makes sense within industry expectations [37].

In brief, on CCRm scenarios, the AEB system performs well, as it avoids all collisions while having some margin, as well as decelerating a bit softer compared to the CCRs scenario. The system complies with, or even surpasses, the actual test protocol demands up to the tested speed. It could be expected to achieve full credits in the Euro NCAP's

CCRm table for at least 60 km/h. In order to approach demands for future regulations, namely tests conducted at 80-100 km/h, it would be necessary to start braking earlier, although at 60 km/h, this AEB performs well.

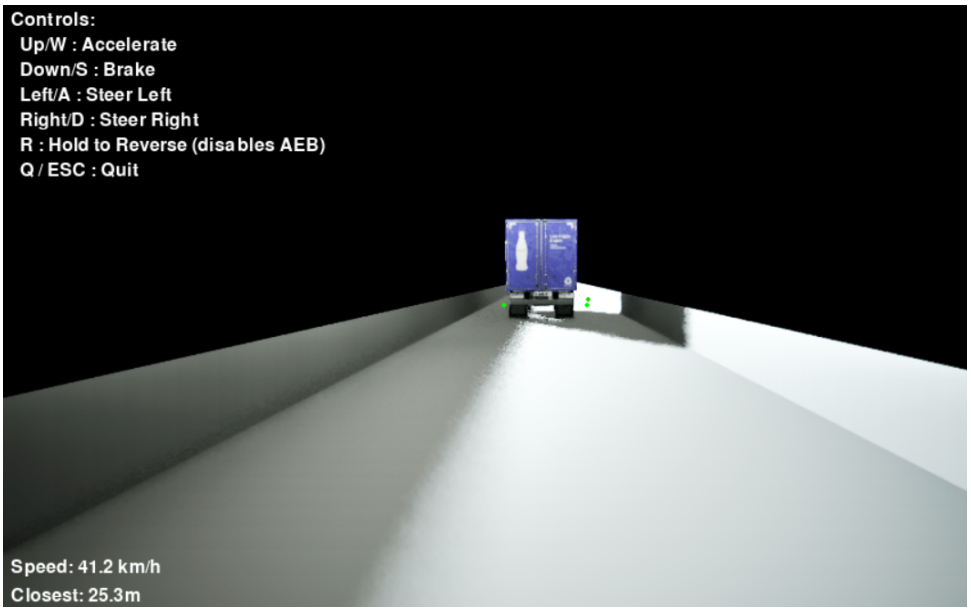

Figure 13: Scenario 2: CCRm

5.3 Scenario 3: CCRb (Car-to-Car Rear Braking)

Scenario Setup: The CCRb scenario has a higher degree of complexity, as the lead vehicle moves forward at a certain speed and eventually brakes suddenly, requiring a response from the ego vehicle. For our testing, the lead vehicle moved forward at 30 km/h, followed by an emergency braking action with a deceleration rate of about $-6\,\mathrm{m/s^2}$, a common value in Euro NCAP regulations. The initial gap, approximately 40 m in our testing, is slightly shorter than those in the other CCRs/CCRm scenarios, presumably to make a braking action necessary. The point at which the lead vehicle initiated braking varied from test to test ($\sim 2.4\,\mathrm{s}$ vs. $\sim 2.9\,\mathrm{s}$ from the start, depending on test conditions). The ego vehicle, in turn, started from a speed ranging from about 31 to 60 km/h. In essence, these scenario parameters resemble those in a highway environment, where unexpected heavy braking from a vehicle in front can be expected. The ego vehicle's AEB system is expected to activate its brakes to avoid the pending collision, thereby, at least, reducing a possible impact speed. Table 4 encompasses the results of testing in the CCRb scenario.

Result: Even in the most difficult setting (Run 19, ego speed approx. 60 km/h vs. lead vehicle 30 km/h, lead vehicle braking hard), the ego vehicle braked in time to avoid a collision with the slowing vehicle. In all 20 runs, the collision avoidance success rate was 100%. In most cases, the final speed is close to 0, meaning the ego vehicle synchronized well with the slowed-down/stopping lead vehicle towards the end. In some cases, a final speed difference of 0.06–0.23 km/h can be observed, but this difference is irrelevant. The table below illustrates the difference in final speed vs. initial speed. The final speed remains at 0 for all test points, indicating an irrelevant effect. Based on Table 4, the effect can be considered negligible.

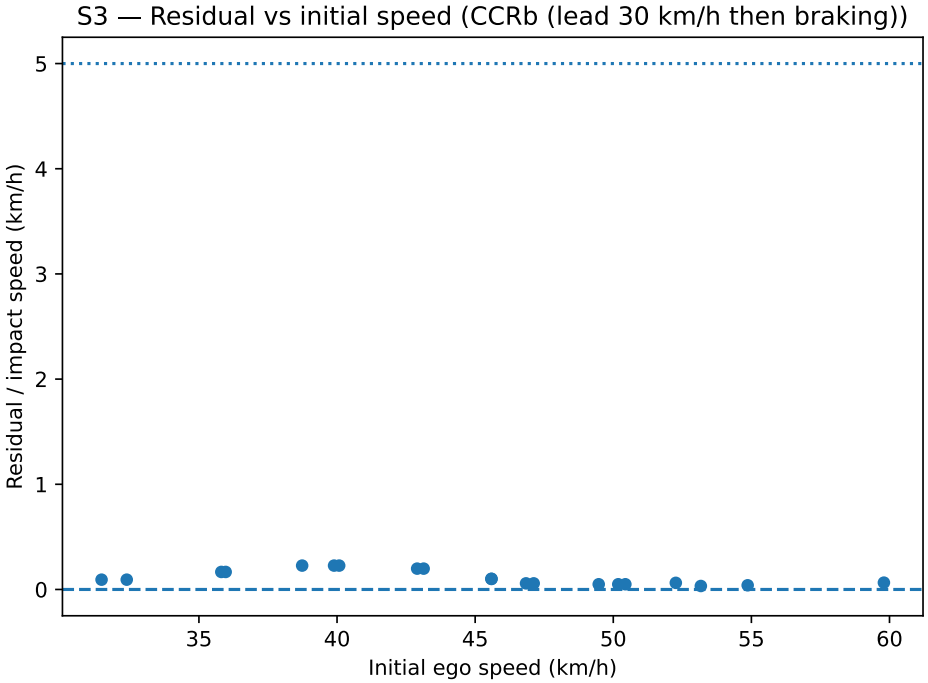

Figure 14

The minimum Time-to-Collision (TTC) values in CCRb are highly variable as a function of ego speed. For the lowest ego speeds (31–32 km/h, close to pacing the lead), TTC values were very high ($\sim$ 12 s) because the ego vehicle was closing relatively slowly, and the lead braking action (Run 0) occurred so late or weakly compared to the ego vehicle's approach that the ego vehicle never got close. For higher ego speeds, a sudden deceleration by the lead vehicle causes a rapid closure. Thus, for ego speeds roughly 38–40 km/h, 45–50 km/h, and 60 km/h, the minimum TTC values were approximately 1.6–1.8 s, 0.9–1.0 s, and as low as 0.59 s for Run 19, respectively. For speeds above 50 km/h, several runs cross below the 1-second TTC line, indicating very close calls. The closest call was 0.59 s at about 60 km/h. In absolute terms, this means it is literally very close—within 0.6 s—of a collision at the point of closest approach. Interestingly, a collision was still avoided in the scenario, indicating that the system activated exactly on time with maximum braking.

System behaviour: In the CCRb scenario, the system behaves in a way combining the first two scenarios, starting as a CCRm scenario (ego closing on a moving lead) and turning suddenly dangerous, like a CCRs scenario, but from a shorter distance. The data from the test scenario indicate that the AEB system activated well before the lead vehicle braked.

For instance, in test run 19, when the ego vehicle speed was approximately 60 km/h, the AEB warning activated at approximately 1.88 s, AEB braking activated at 1.92 s, while the lead vehicle did not start braking until 2.93 s. In this case, it is apparent that the ego AEB system initiated braking solely based on the closing speed difference of 60–30 km/h, starting braking 1.0 s before the lead vehicle began to decelerate.

In other words, in high-speed runs, the AEB system activated as if it were in a CCRm scenario, having recognized that the ego vehicle would eventually cross into the lead position, potentially colliding even if the lead maintained 30 km/h, and thus initiated

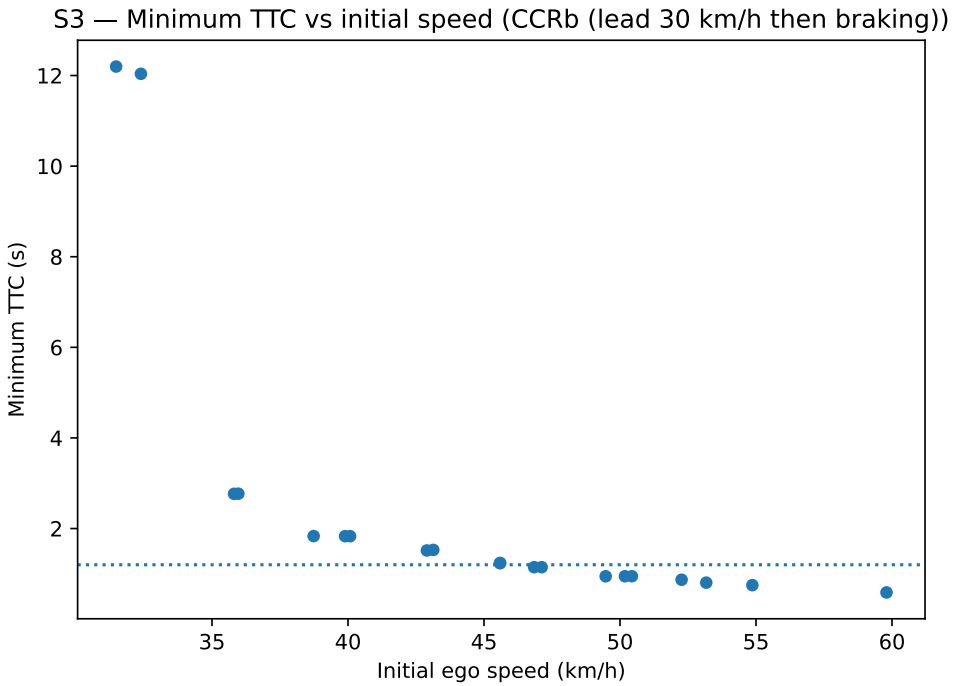

Figure 15

the response. The lead vehicle's eventual braking would have been too late, as the ego vehicle would have already shed a substantial amount of speed.

For example, in tests 13–19, as the ego vehicle speed varied between 49–60 km/h, the AEB system's braking activation time remained within 1.9–2.0 s (Table 3), as in a pure CCRm scenario. Thus, the ego vehicle always had ample time to decelerate, while the lead vehicle's braking action would not have surprised it, as the AEB had already activated.

For somewhat higher ego vehicle speeds ($\sim$ 35–45 km/h), the behaviour is a combined result. For example, in Run 4, while the ego vehicle speed was 38.7 km/h, AEB braking occurred at 1.865 s, whereas the lead vehicle braked at 2.63 s. The ego vehicle thus initiated braking about 0.8 s prior to the lead vehicle. In runs 2 and 3, with ego vehicle speeds of 35.8–36.0 km/h, AEB braking occurred at approximately 2.08 s, while the lead vehicle braked at about 2.6 s. Again, the ego vehicle initiated braking first. This indicates that the collision algorithm in the system detected a potential collision based on the small closing speed difference ($\sim$ 6 km/h) on the lead vehicle, which had not yet braked. For very low ego vehicle speeds, as in Run 0 at 31.5 km/h, no braking was required on the ego vehicle. In this case, the ego vehicle was barely closing on the lead vehicle. The lead vehicle braked at 6.87 s, while the ego vehicle never got close, with a TTC of approximately 12 s. As a result, the AEB system did not activate.

It can be presumed that the AEB system's logic is always tracking the TTC/headway. If the ego is approaching the lead car, thereby reducing the TTC, a warning along with brakes would be activated as soon as TTC crosses a threshold, without the lead brakes, naturally. This is helpful because it indicates the system can prevent a collision from taking place when the car comes too close, even before a sudden stop occurs. In the event of the lead car applying brakes, the response would follow, but it would have a head start, as observed in our experiments. Hence, all collisions were avoided.

Table 4: Results for AEB CCRb scenario (lead vehicle moving at 30 km/h then braking at approximately $-6\,\mathrm{m/s^2}$). The ego speeds ranged from about 30–60 km/h. All runs avoided contact. t_{brake} is the time when the lead began braking. TTC values reflect the closest approach after both vehicles' actions.

Run ID	Ego Speed (km/h)	Lead Speed (km/h)	Lead Brake Start t_{brake} (s)	Collision Avoided?	Residual Speed (km/h)	Min TTC (s)	AEB Brake Onset Time (s)
0	31.5	30.0	6.87	Yes	0.09	12.20	0.00
1	32.4	30.0	2.40	Yes	0.09	12.04	0.00
2	35.8	30.0	2.62	Yes	0.17	2.77	2.086
3	36.0	30.0	2.52	Yes	0.17	2.77	2.082
4	38.7	30.0	2.63	Yes	0.23	1.83	1.865
5	40.5	30.0	2.68	Yes	0.06	1.67	1.891
6	40.8	30.0	2.72	Yes	0.06	1.62	1.917
7	42.7	30.0	2.77	Yes	0.07	1.50	1.951
8	44.3	30.0	2.74	Yes	0.03	1.38	1.955
9	44.5	30.0	2.78	Yes	0.03	1.33	1.958
10	45.6	30.0	2.81	Yes	0.00	1.24	1.917
11	46.8	30.0	2.75	Yes	0.00	1.15	1.951
12	47.1	30.0	2.68	Yes	0.00	1.15	1.955
13	49.5	30.0	2.68	Yes	0.00	0.95	1.982
14	50.2	30.0	2.77	Yes	0.00	0.95	1.947
15	50.4	30.0	2.79	Yes	0.00	0.95	1.945
16	52.3	30.0	2.82	Yes	0.00	0.87	1.978
17	53.2	30.0	2.85	Yes	0.00	0.81	1.982
18	54.9	30.0	2.77	Yes	0.00	0.75	1.892
19	59.8	30.0	2.93	Yes	0.06	0.59	1.921

The reaction times, from detection to braking, in CCRb were comparable to those in CCRm. The values were approximately 1.9 s at high speeds, similar to Scenario 2. The records indicate that the initial detection of the lead vehicle always occurred at $t = 0$ (the lead vehicle was in view of the sensors). Hence, the sensors had no delay in detecting the lead vehicle. The only point to consider is that for mid-speed scenarios (30–40 km/h), the system did not act promptly. It practically waited for the lead vehicle to start braking or for the TTC to become small. For example, in Run 1, with an ego speed of 32.4 km/h, the lead vehicle braked at 2.4 s, but the ego vehicle did not start braking until then. The minimum TTC of approximately 12 s indicates an actual lack of conflict until the lead slowed down, making it easy for the ego vehicle to act. For higher closing rates, the system responded earlier as required.

Performance vs. Requirements:

- **Euro NCAP CCRb baseline:** CCRb eliminates the AEB requirement to avoid a collision while the lead vehicle brakes at -6 m/s^2 (as well as -2 m/s^2 scenarios) at 50 km/h with specified headways/overlaps. Higher speeds can also be considered for testing [32].

- **our result vs. Euro NCAP:** In your tests, the AEB initiated without contact at 50 km/h and without contact at approximately 60 km/h against a 30 km/h lead decelerating at approximately -6 m/s^2, showing a strong margin of safety relative to the CCRb baseline [32].

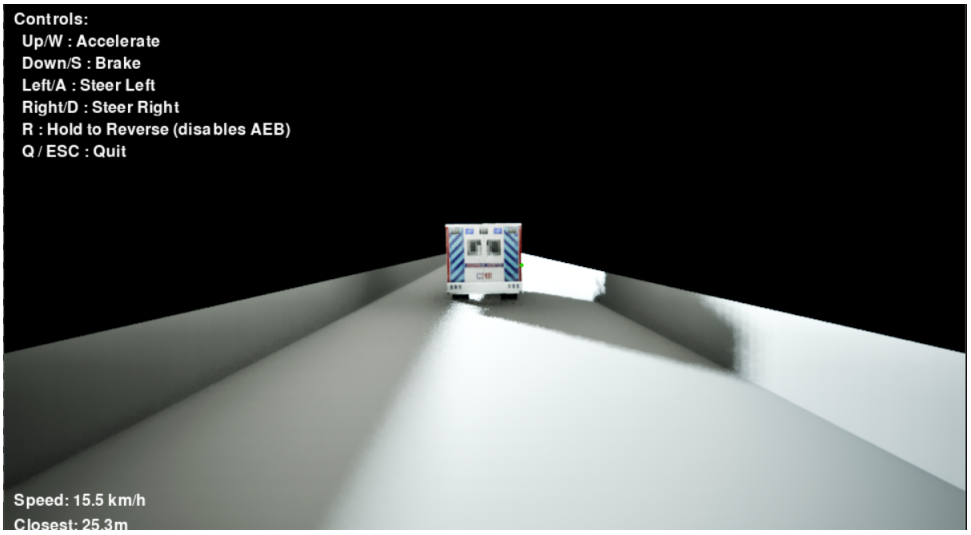

Figure 16: Scenario 3: CCRb before De-acceleration

- **FMVSS 127 (US) bar:** The final regulation demands no-contact results, featuring a lead decelerating vehicle scenario at 50 km/h and 80 km/h. The data available go up to 60 km/h, while 80 km/h would be much more difficult. Extrapolation is possible, though [40].
- **"No-contact" interpretation:** Since our CCRb results indicated a lack of contact, it appears that your system is consistent with a "no contact" condition up to approximately 60 km/h, although 80 km/h performance has not been confirmed [40].
- **Euro NCAP scoring lens:** The highest rating is given to Euro NCAP testing, where impact speed approximates 0–<5 km/h, while a tolerance of <7 km/h can be considered. Our CCRb impacts were $\approx$ 0 km/h. Hence, a full score can be achieved [34].
- **TTC margin note:** The minimum TTC $\approx$ 0.59 s at the highest speed is very tight; it depended on braking being underway. Small additional delays or reduced friction could have reversed the result, indicating proximity to the capability threshold.
- **Real-world context:** AEB usually increases safety in rear-end collisions and reduces injury severities, though the benefit wanes as speed increases or with reduced friction, consistent with the trend of "excellent up to $\sim$ 60 km/h, diminishing beyond. [39]

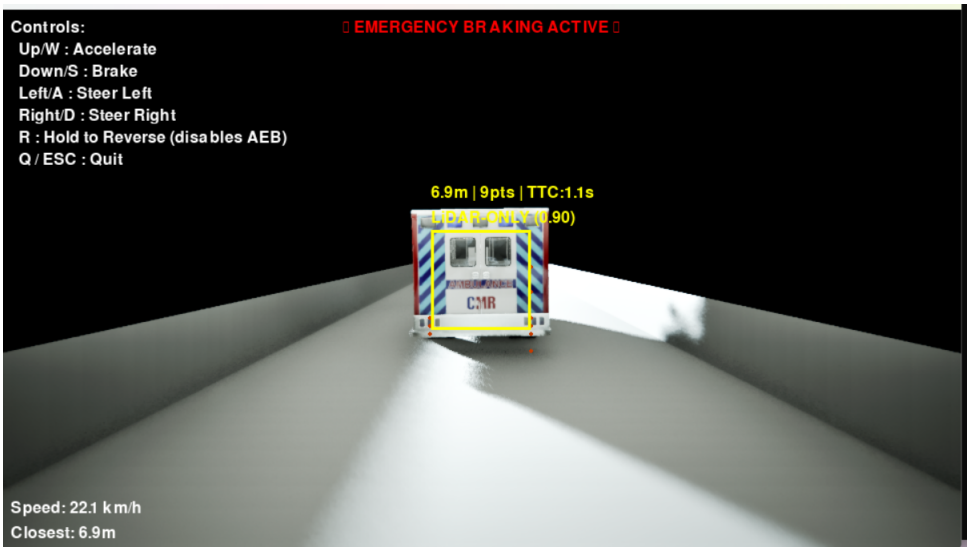

Figure 17: Scenario 3: CCRb after De-acceleration

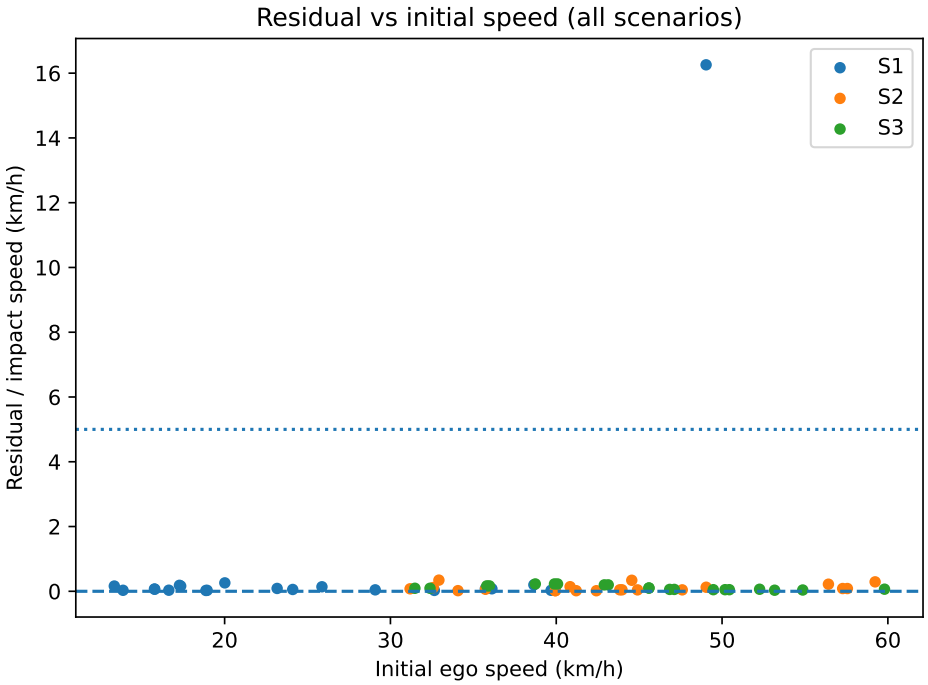

Figure 18

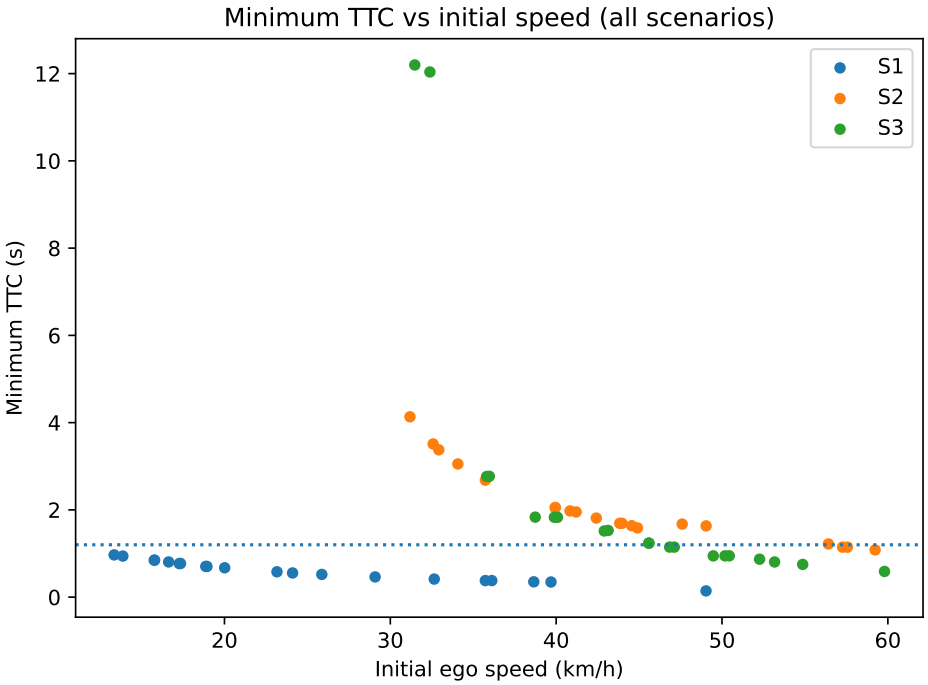

Figure 19

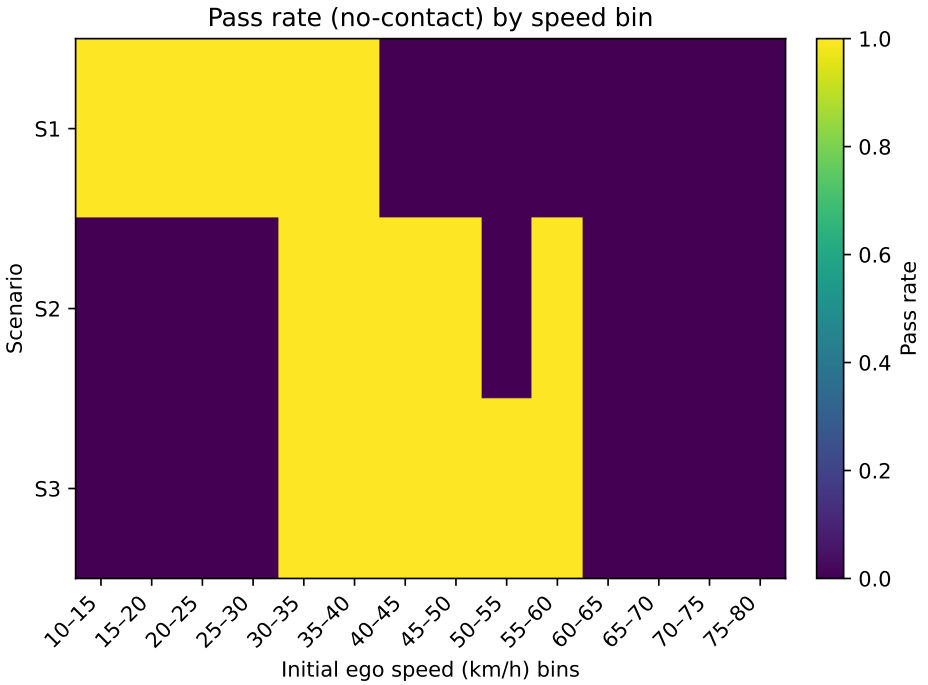

Figure 20: No-contact pass rate by initial ego-speed bin for each scenario (S1=CCRs, S2=CCRm, S3=CCRb), where "pass" denotes residual/impact speed $\leq 0.5 km/h$

6 Conclusion

This project provides a deterministic, interpretable, and reusable AEB stack in CARLA which integrates a forward RGB camera (YOLOv8n model) with a 64-channel LiDAR on strict synchronous 20 Hz locking. The solution carefully integrates camera and LiDAR (projects LiDAR into the image to enable object ranging and TTC), synthesizes risk decisions using physics-based stopping models in a finite-state controller and jerk-limited brake shaping. The software design (corridor gating, range-dependent confidence blending, LiDAR-only safety net, majority voting over three frames) is mathematically correct and freely portable to simple OpenDRIVE simulation setups to make interpretable decisions with a tick rate of 50 ms.

In Euro NCAP C2C test conditions, there was good performance in moving-target (CCRm) and braking-lead (CCRb) test conditions. In these conditions, 20/20 runs avoided contact up to ego speeds of around 59 km/h with lead speeds of around 22 km/h (CCRm), and 20/20 runs avoided contact up to ego speeds of around 60 km/h with lead braking decelerations of approximately -6 m/s^2 (CCRb), maintaining minimum TTC values $\geq$ approximately 1.08 s in CCRm or as small as around 0.59 s in the most challenging CCRb scenario. These results show that fusion logic with EA performs well if the lead vehicle is visible.

The major constraint exists in the stationary target (CCRs), with 19/20 runs assuming a collision was prevented until approximately 39.7 km/h, but a miss occurred at around 49 km/h with $\sim$16 km/h residual speed. Logs indicate that the camera did not pick up the stationary target in this test case, resorting to a late LiDAR fallback response near impact. The braking response was initiated approximately 0.27 s before impact, which was not sufficient. Secondarily, braking jerk was observed to frequently spike (up to $\sim$650-720 m/s^3 in some lower/moderate speed trials), indicating a sharp response transition. Both point to (i) a perception problem against stationary objects at larger distances or higher closure speeds and (ii) a need for earlier, articulated braking commands to manage jerk while retaining a safety margin.

In conclusion, there is a rigorous fusion-control pipeline with encouraging results for CCRm/CCRb and areas to improve to close the gap for CCRs related to braking comfort.

7 Future Scope

1. **Upgrade perception for stationary hazards (CCRs):**:
    - Enhance small or long-range stationaries using (a) detector retraining with high FOV small object priors, (b) aggregation or tracking to accumulate confidences over time, or (c) fusion rules to boost the contribution of LiDAR to values above 30m without a camera box.
    - Add a radar stream in fusion (range/radial velocity), which can separate stationary world objects from ego-motion and improve activation speed for high closure speeds.
    - Add a "stationary ahead sentinel" function to initiate a gentle pre-brake if TTC to nearest in-lane LiDAR surface crosses an increased threshold before vision confirmation or based on confirmation.

2. **Earlier, staged, and jerk-limited braking:** Transition to a two-stage approach (e.g., advisory/partial braking around 2.5-3.0s TTC, full braking around 1.0-1.5s), in conjunction with specific jerk limits or S-curves defined in the braking signal to lower maximum jerk values without increasing stopping distances.

3. **Risk modeling & fusion robustness:**

   - Hard thresholding replaced with uncertainty-aware fusion (probabilistic occupation or evidential weighting), so that camera dropouts are not too conservative with response times; safety channel based purely on LiDAR data with dynamic corridor size for faster speeds.
   - Add velocity-aware dynamic thresholding. Tighten TTC/distance constraints based on ego speed and reduce dependencies on late-frame matches.

4. **Scenario and ODD expansion:** Extend OpenDRIVE to non-straight roads involving cut-ins, adjacent lane vehicles, overlaps (full or partial), VRU (pedestrians/cyclists) crossing or longitudinal interactions. Add night conditions, glare, rain, fog, wet or low- roads, and grades. Port the same stack to larger Town maps to test budgets in the face of loaded assets.

5. **Verification, scoring, and analytics:**

   - Base assessment against Euro NCAP criteria (impact speed bands, false positives, comfort), and record complete telemetry data (detections, ranges, TTC values, state transitions, and actuations).
   - Complete Monte-Carlo sweeps for speed, gap, friction values, and sensor noise to explore safety margins around which TTC values fall below around 1 s for situations similar to those found in CCRb.

6. **Systems engineering & deployment realism:**

   - Profile end-to-end latency per tick with budget for detector/LiDAR association with headroom set to 20 Hz. Also test 30-40 Hz for faster use cases.
   - Also consider integration with ROS 2 interfaces and modularity for easier transition to hardware-in-loop or real-vehicle testing.

These above-mentioned features should serve to reduce or eliminate the gap between CCRs at ≥ 50 km/h speeds. The system does not lessen the excellent performance that was proven using CCRm/CCRb.

References

[1] CARLA Simulator — Releases (GitHub). Available at: https://github.com/carla-simulator/carla/releases [Accessed: Apr 20, 2025].

[2] Synchrony and time-step — CARLA. Available at: https://carla.readthedocs.io/en/latest/adv_synchrony_timestep/ [Accessed: May 10, 2025].

[3] Synchrony and time-step — fixed_delta_seconds = 0.05 example (v0.9.9). Available at: https://carla.readthedocs.io/en/0.9.9/adv_synchrony_timestep/ [Accessed: May 10, 2025].

[4] ASAM OpenDRIVE standalone mode — CARLA. Available at: https://carla.readthedocs.io/en/latest/adv_opendrive/ [Accessed: May 10, 2025].

[5] client.generate_opendrive_world — CARLA Python API. Available at: https://carla.readthedocs.io/en/latest/python_api/ [Accessed: May 10, 2025].

[6] Python API reference — CARLA. Available at: https://carla.readthedocs.io/en/latest/python_api/ [Accessed: May 10, 2025].

[7] Python API reference (v0.9.12) — get_velocity(), get_transform(). Available at: https://carla.readthedocs.io/en/0.9.12/python_api/ [Accessed: May 10, 2025].

[8] Camera sensor attributes — CARLA (v0.9.5 docs). Available at: https://carla.readthedocs.io/en/0.9.5/cameras_and_sensors/ [Accessed: May 10, 2025].

[9] LiDAR attributes (channels, rotation_frequency, points_per_second) — CARLA (v0.9.5). Available at: https://carla.readthedocs.io/en/0.9.5/cameras_and_sensors/#lidar-sensor [Accessed: May 18, 2025].

[10] Sensors reference — LiDAR — CARLA. Available at: https://carla.readthedocs.io/en/latest/ref_sensors/ [Accessed: May 18, 2025].

[11] Remark on channel distribution across vertical FoV — CARLA Chinese docs mirror. Available at: https://bbs.carla.org.cn/doc-detail/1af970dab09b4c78b5ad35c15b2f8d8b [Accessed: Jun 7, 2025].

[12] Project 3D bounding boxes to camera — CARLA tutorial. Available at: https://carla.readthedocs.io/en/latest/tuto_G_bounding_boxes/ [Accessed: Jun 10, 2025].

[13] OpenCV: Camera Calibration and 3D Reconstruction. Available at: https://docs.opencv.org/master/d9/d0c/group__calib3d.html [Accessed: Jun 10, 2025].

[14] UZH CV lecture note: From FOV to focal length in pixels. Available at: https://rpg.ifi.uzh.ch/docs/teaching/2021/03_pinhole_camera_model_handout.pdf [Accessed: Jun 10, 2025].

[15] Open3D: segment_plane (RANSAC plane fitting). Available at: http://www.open3d.org/docs/latest/python_api/open3d.geometry.PointCloud.html#open3d.geometry.PointCloud.segment_plane [Accessed: Jun 10, 2025].

[16] Ultralytics YOLO — Predict mode. Available at: https://docs.ultralytics.com/modes/predict/ [Accessed: Aug 20, 2025].

[17] Ultralytics YOLO — Python usage. Available at: https://docs.ultralytics.com/usage/python/ [Accessed: Aug 20, 2025].

[18] Ultralytics Results / Boxes API. Available at: https://docs.ultralytics.com/reference/engine/results/ [Accessed: Aug 20, 2025].

[19] Intersection over Union (IoU) in Object Detection. Available at: https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/ [Accessed: Aug 20, 2025].

[20] W. Yang *et al.*, "Research on Longitudinal Active Collision Avoidance of Intelligent Vehicles Based on AEB," *Sensors*, 2019. Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC6864679/.

[21] Braking distance — Wikipedia. Available at: https://en.wikipedia.org/wiki/Braking_distance [Accessed: Aug 20, 2025].

[22] Braking distance — AMSI teacher module. Available at: https://www.amsi.org.au/teacher_modules/pdfs/Maths_delivers/Braking5.pdf [Accessed: Aug 20, 2025].

[23] UN Regulation No. 152 — Automatic/Advanced Emergency Braking for M1/N1. Available at: https://unece.org/sites/default/files/2024-04/R152r2E.pdf [Accessed: Aug 20, 2025].

[24] Hidden Point Removal (HPR) — view-based point cloud visibility. Available at: https://igl.ethz.ch/projects/hidden-point-removal/ [Accessed: Nov 7, 2025].

[25] Velodyne HDL-64E S3 Datasheet. Available at: https://resources.mis.scripts.mit.edu/media/FileVault/RSS2011/misc/Velodyne%2064.pdf [Accessed: Sep 7, 2025].

[26] Z. Wang *et al.*, "Research on Hierarchical Control Strategy of Automatic Emergency Braking System," *Vehicles*, 2023. Available at: https://www.mdpi.com/2032-6653/14/4/97.

[27] F. Lai *et al.*, "An Automatic Emergency Braking Control Method for Improving Ride Comfort," *Vehicles*, 2024. Available at: https://www.mdpi.com/2032-6653/15/6/259.

[28] ASAM OpenDRIVE — Standard overview. Available at: https://www.asam.net/standards/detail/opendrive/ [Accessed: Sep 7, 2025].

[29] ASAM OpenDRIVE Base Standard V1.7.0 (HTML). Available at: https://www.asam.net/fileadmin/Standards/OpenDRIVE/ASAM_OpenDRIVE_BS_V1-7-0.html [Accessed: Oct 15, 2025].

[30] Core Map — CARLA. Available at: https://carla.readthedocs.io/en/latest/core_map/ [Accessed: Nov 7, 2025].

[31] CARLA Simulator — GitHub Issue #6457. Available at: https://github.com/carla-simulator/carla/issues/6457 [Accessed: Oct 15, 2025].

[32] Euro NCAP — AEB Car-to-Car (C2C) Test Protocol v4.3 (PDF). Available at: https://www.euroncap.com/media/79864/euro-ncap-aeb-c2c-test-protocol-v43.pdf [Accessed: Oct 15, 2025].

[33] OxTS — AEB: Car-to-Car Rear Stationary (CCRS). Available at: https://www.oxts.com/aeb-car-to-car-rear-stationary-ccrs/ [Accessed: Oct 15, 2025].

[34] Euro NCAP — Assessment Protocol: Safety Assist (Collision Avoidance) v1.0.4 (PDF). Available at: https://www.euroncap.com/media/79866/euro-ncap-assessment-protocol-sa-collision-avoidance-v104.pdf [Accessed: Oct 15, 2025].

[35] UNECE — UN regulation on advanced emergency braking systems for cars (press release). Available at: https://unece.org/sustainable-development/press/un-regulation-advanced-emergency-braking-systems-cars-significantly [Accessed: Oct 15, 2025].

[36] Allen & Allen Blog — "U.S. will require all new cars to have auto emergency braking systems." Available at: https://www.allenandallen.com/blog/u-s-will-require-all-new-cars-to-have-auto-emergency-braking-systems/ [Accessed: Oct 15, 2025].

[37] JSAEM — Journal article. Available at: https://jsaem.my/index.php/journal/article/view/130 [Accessed: Oct 30, 2025].

[38] MDPI Electronics — Journal article. Available at: https://www.mdpi.com/2079-9292/8/9/943 [Accessed: Oct 30, 2025].

[39] IIHS — "IIHS eyes higher-speed test for automatic emergency braking." Available at: https://www.iihs.org/news/detail/iihs-eyes-higher-speed-test-for-automatic-emergency-braking [Accessed: Oct 30, 2025].

[40] FMVSS No. 127 — Automatic Emergency Braking (Final Rule). Available at: https://www.nhtsa.gov/sites/nhtsa.gov/files/2024-04/final-rule-automatic-emergency-braking-systems-light-vehicles_web-version.pdf [Accessed: Oct 30, 2025].