# A Safety Cage for Reliable GNSS and AI-based PNT: an Experience Report*

Guillermo Gomez[1], Alberto Griggio[1][0000−0002−3311−0893]✉, Luca Morelli[1][0000−0001−7180−2279], Theodore Russell[2], Stefano Tonetta[1][0000−0001−9091−7899], and Paweł Trybała[1][0000−0002−6486−1147]

[1] Fondazione Bruno Kessler, Italy
{ggomezarnedo,griggio,lmorelli,tonettas,ptrybala}@fbk.eu
[2] GMV, UK trussell@gmv.com

**Abstract.** Reliable Positioning, Navigation, and Timing (PNT) is a fundamental requirement for the safe operation of autonomous vehicles. While traditional GNSS-based PNT is typically very accurate in open sky conditions, its performance can degrade significantly in environments where satellite visibility is limited or signals are obstructed. To ensure continuity, it is beneficial to complement GNSS with other methods, such as AI-enhanced Visual SLAM, which remain robust under challenging environmental conditions. However, AI-based components can produce untrustworthy results and necessitate a safety cage to monitor and filter their output. This experience report presents the VAIPOSA architecture, which introduces a safety cage to perform runtime verification and fusion of hybrid GNSS and AI-based PNT sources. The safety cage monitors a variety of parameters, including GNSS signal characteristics, AI-based precision metrics, image quality, and vehicle state information, to determine the trustworthiness of each source. We describe the integration of these monitors within a supervisor that fuses data into a reliable pose estimate and discuss its validation in the CARLA simulator using automated test case generation and fault injection scenarios.

**Keywords:** Safety Cage · AI-based PNT · Autonomous Driving · Runtime Monitoring · Formal Methods.

## 1 Introduction

Reliable Positioning, Navigation, and Timing (PNT) is a fundamental requirement for the safe operation of autonomous vehicles. Traditionally, Global Navigation Satellite Systems (GNSS) have served as the primary technology for this task. GNSS is a mature solution capable of computing the absolute position of a system with high precision and accuracy based on satellite signals. However, in safety-critical domains, the positioning solution must be resilient to environmental challenges where satellite signals may be obstructed or distorted by

* The activity was fully funded by the European Space Agency NAVISP Element 1, whose objective is to innovate PNT technologies (EL1-087 bis).

multi-path effects, particularly in tunnels or dense urban "canyons" [29]. In these scenarios, it is critical to augment GNSS receivers with complementary sensors.

Visual Simultaneous Localization and Mapping (SLAM)[12, 14] utilizes images from stereo cameras to compute vehicle poses and is a potential method to address these limitations. While these algorithms range from standard computer vision approaches to state-of-the-art AI-based methods, their data-driven nature poses well known safety risks. Employing such black-box components in safety-critical applications necessitates a safety layer that can monitor and filter their outputs using traditional, rule-based approaches [1]. Building on them, more complex systems propose combining different sensing modalities, such as GNSS, cameras, and inertial sensors, to improve the robustness and accuracy of PNT solutions in challenging environments. These approaches can offer either early-stage fusion, where raw sensor data is combined before processing [11], or late-stage fusion, where independent PNT estimates from each modality are computed and combined at a higher system level [3].

Many works in the literature have proposed various safety architectures where efficient but potentially less reliable solutions are encapsulated by a safety layer that monitors and corrects their output. These safety design patterns have been proposed with different names such as safety monitors, safety cages, and safety shields, and are in essence aligned with the concept of fault detection, isolation and recovery. The advent of AI made the topic even more significant and safety cages have been applied to control functions using reinforcement learning as in [9] or to perception functions using machine learning as in [24].

In this paper, we report on the design and application of a *safety cage* for a verifiable, hybrid GNSS and AI-based PNT engine, developed in the VAIPOSA project and funded by the European Space Agency. Our solution performs runtime verification and fusion of the non-AI and AI PNT sources. The safety cage monitors a diverse set of parameters, including GNSS signal characteristics (e.g., satellite count and confidence statistics), AI-based precision metrics, image quality, and physical vehicle state information (e.g., velocity and "point-on-road" checks). These monitors determine the trustworthiness of each engine, which is then translated into a dynamic weighting logic used to drive an Extended Kalman Filter (EKF) for final pose estimation.

We detail our methodology, which begins with an analysis of the potential faults for each engine and the definition of available information interfaces, from which monitoring conditions are derived. Furthermore, we discuss the validation of this architecture within the CARLA simulator [5]. Using the VIVAS framework [6], we performed automated test-case generation and fault injection to verify the resilience of the system in various unsafe scenarios, such as tunnels.

## 2 Related Work

The challenge of ensuring safety in AI-enabled autonomous systems has led to the adoption of architectural patterns that decouple performance from safety.

In multi-modal PNT systems, the monitoring strategy and supported fault models depend strongly on the system architecture. Early-stage fusion focuses on the integrity of raw sensor data in tightly coupled systems. Alternatively, we adopted late-stage fusion, where independent PNT estimates from different modalities are computed and combined at a higher system level, the monitoring can additionally focus on the consistency and reliability of the individual PNT estimates. In the context of GNSS, this allows us to use a simplified representation of faults (see Section 5.2) such as multipath [16], spoofing and jamming [18], avoiding computationally expensive physical simulations, prohibitive given the large-scale scenarios and temporal horizons considered in this work. This approach preserves the impact of GNSS degradation on the overall PNT solution while enabling the study to concentrate on monitoring the AI-based component.

A well known concept in runtime assurance is the *Simplex architecture* [22], which utilizes a high-performance but potentially untrustworthy controller alongside a high-integrity, formally verified fallback. Safety monitors have been studied by Machin et al. [13], who proposed the Safety Monitoring Framework (*SMOF*) that focuses on synthesizing monitors from safety requirements, starting from a hazard analysis and using formal verification techniques to synthesize the rules. A similar notion, the *Safety Shield*, has been proposed by Könighofer et al. [9]. The related TEMPEST tool [17] allows for the synthesis of shields from LTL specifications to ensure that an AI-based agent (often a reinforcement learning policy) never enters an unsafe state. These shields are meant to be integrated into the system to override unsafe commands pre-emptively, ensuring "correct-by-construction" behavior even when the underlying AI component is a black box. The practical application of these monitors in the automotive domain has been explored in projects such as *SMILE* (and its framework *SMIRK*). These works address the challenge of creating a complete safety case for Machine Learning in safety-critical applications, such as pedestrian emergency braking [24]. They demonstrate how runtime monitors can act as an independent safety mechanism to mitigate the uncertainties of deep neural networks.

Our safety cage implementation is a direct evolution of the above works, with the addition of a logic that balances the AI and non-AI-based PNT sources, based on qualitative and quantitative monitoring properties.

While traditional PNT safety relies on statistical bounds like the Horizontal Protection Level (HPL), modern hybrid systems require more adaptive monitoring. Recent research explores using deep reinforcement learning to optimize sensor fusion and enhance GNSS resilience against interference [27], or substituting part of Kalman filters with recurrent neural networks assuming unknown system dynamic models and noise statistics [19]. Our work contributes to this niche by using a multi-parametric Safety Cage to manage the hand-off between traditional GNSS and AI-based SLAM in urban environments.

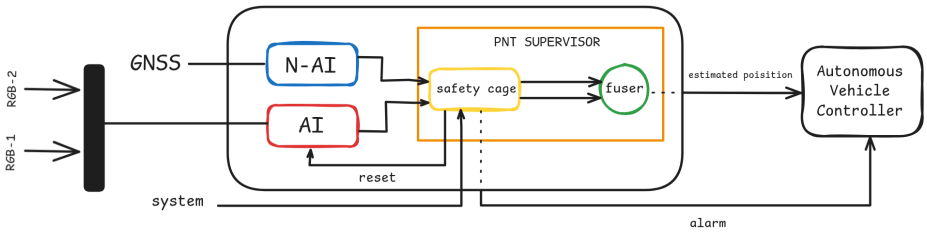


**Fig. 1.** The VAIPOSA safety architecture.

## 3 Safety Architecture

The VAIPOSA safety architecture is depicted in Fig. 1. It comprises two PNT engines, one based on GNSS and one based on visual SLAM aided by an AI module for image matching, which provide independent and redundant estimation of the pose of the vehicle, and a specialized **PNT Supervisor**, which manages the integrity and fusion of the positioning data and is composed of a Safety Cage and a Fuser component.

**GNSS Engine.** MSP3, a precise positioning solver, [8], provides a three-dimensional absolute position in the ECEF reference frame at approximately 1 Hz, which is then projected to the local reference frame. It processes multi-constellation satellite signals and provides essential integrity metrics, such as the Horizontal Protection Level, which statistically bounds the potential position error. A heading pseudo-measurement is additionally derived from the displacement between consecutive GNSS positions when the displacement between them exceeds the minimal threshold. The heading covariance is tied to the estimated yaw rate to account for turn-induced heading estimation uncertainty.

**AI-based Engine.** A positioning component that uses COLMAP-SLAM [14, 15], a real-time Visual SLAM framework built on the COLMAP engine [20]. Roughly speaking, the algorithm works in 3 main steps: (i) feature extraction from the input image pair (coming from two RGB cameras), using a 1st NN (SuperPoint [4] or ALIKED [28]); (ii) feature matching on the stereo pair and between consecutive pairs in time, using a 2nd NN (LightGlue [10]), and successive triangulation of the features; (iii) 3D scene reconstruction and computation of the relative pose (6-vector of position and angles in 3D space) change. The inclusion of the AI components ([4], [28]) significantly enhances the robustness of pose estimation in challenging environments. (In the following, we refer to this as "AI-based" or "SLAM-based" engine.)

**Safety Cage.** The Safety Cage serves as the monitoring layer of the architecture. It continuously analyzes the outputs of both PNT engines alongside system-level metadata, such as camera visibility and image quality. By performing a variety of atomic and temporal checks, the Safety Cage determines the trustworthiness of each engine. These checks are translated into a weight for

each source. Temporal properties that require tracking over time are monitored using NuRV [2], which evaluates formal specifications expressed in LTL.

**Fuser.** The Fuser component is responsible for generating the final, unified global pose output. It utilizes an Extended Kalman Filter (EKF) to combine the absolute coordinates from the GNSS engine with the relative pose changes from the AI-based PNT engine. The selection of EKF as the fusion method is motivated by its vast adoption in the industry, ability to combine different measurement modalities (i.e., absolute and relative positioning), as well as its computational efficiency for real-time applications [21, 23]. The filter operates on a 15-dimensional state vector defining position, orientation, linear velocity, angular velocity, and linear acceleration. Notably, velocity and acceleration are *latent* states: no dedicated inertial sensor is fused in the current design. They are propagated by the motion model and implicitly regulated through the Kalman cross-covariance structure. Due to the application of the framework to automotive scenarios, the motion model assumes a constant-acceleration, constant-turn-rate process in the world frame [26]. The discrete process noise covariance is designed to follow a continuous-time kinematic white-noise model for a car-like motion. Finally, all measurement updates use the Joseph form of the EKF covariance update.

## 4 Safety Cage Design and Monitoring Properties

The design of the Safety Cage follows a systematic methodology that begins with an analysis of the PNT potential faults to then derive the monitoring properties.

### 4.1 Fault Analysis of PNT Engines

For the GNSS engine, performance is primarily degraded by physical signal interference, such as:

- **Radio-frequency interference**, where unintentional emitters (industrial equipment, nearby communication systems) saturate the receiver front-end and reduce correlator lock quality.
- **Antenna faults**, including partial occlusion, damaged elements, or degraded ground plane performance.
- **Satellite-related degradations**, such as unhealthy ephemeris data or temporary constellation geometry weakening.

Then, we identified the following potential faults of the AI-based PNT engine:

- **Feature starvation**, where a lack of texture in the environment (e.g., a blank wall, dense fog, smoke, or a dark tunnel) prevents the extraction of reliable keypoints.
- **Degraded luminosity contidions** caused by tunnels with low lighting or zones with less density of traffic lights.
- **Hardware faults**, such as frame drops, stuck images or sudden shut down of cameras.

### 4.2 Engine Interface and Monitoring Data

To detect these faults, we analyze the available information at the interface of each component. The Safety Cage acts as a consumer of both engine outputs and raw system metadata to form a comprehensive view of the system's health.

- **GNSS Interface:** Beyond the absolute position and velocity, this interface provides critical integrity data, including
  - Number of satellites in view of the reciever
  - Horizontal Protection Level (HPL). The HPL represents the radius, expressed in meters, of a circle centered at the estimated horizontal position within which the true position is guaranteed to lie with a predefined integrity risk level (in our case it is $2 \cdot 10^{-9}$).
  - Horizontal Dilution of Precision (HDOP). The HDOP is an indicator of the satellite geometry and this affects the quality of horizontal position estimation. Lower values correspond to better satellite distribution and thus higher positioning accuracy.
- **SLAM Interface:**
  - Number of features extracted per image.
  - Feature extraction time.
  - Covariance matrix of the relative pose change.
- **System Information:** the Safety Cage monitors also external system data, including the status of each sensor, the images from the stereo cameras, the information about motion such as velocity and acceleration, and may have (although it is not necessary) map information about the roads (e.g., number of lanes and their width).

### 4.3 Monitoring Properties and Weighting Logic

The Safety Cage evaluates the reliability of each PNT engine by assigning a dynamic weight $w \in [0, 1]$, which is then used by the Fuser to compute the final pose. This assessment follows a two-tier hierarchical logic:

1. **Qualitative Cutoffs (Hard-Gating):** A set of boolean properties derived from the identified faults. If any of these "cutoff" properties are violated, the engine is considered untrustworthy, an alarm is raised, and the engine output is blocked ($w = 0$).
2. **Quantitative Scoring (Soft-Weighting):** If no qualitative alarms are active, the Safety Cage calculates a fine-grained weight using sigmoid functions. This weight reflects the current "health" of the engine based on rolling window statistics.

To formalize the decision logic and data flow, the Safety Cage outputs a pair of real weights $(w_{\text{GNSS}}, w_{\text{SLAM}}) \in [0, 1]^2$, alongside a set of categorical alarm flags. The output of each engine modified by the corresponding weight is passed to the Fuser, while the alarms are sent to the system controller (see also Fig. 1). The single weight variable, one for each engine, encapsulates both evaluation

tiers: the qualitative assessment acts as a binary hard-gate (forcing $w_i = 0$ upon a critical safety rule violation), while the quantitative assessment computes a continuous score within $[0, 1]$ during nominal or slightly degraded conditions.

When a qualitative cutoff occurs ($w_i = 0, i \in \{\text{GNSS}, \text{SLAM}\}$), the engine output is discarded. In the event of a simultaneous dual failure where both engines fail their qualitative checks ($w_{\text{GNSS}} = w_{\text{SLAM}} = 0$), the PNT Supervisor raises an immediate critical alarm to the vehicle controller, acknowledging that no trustworthy PNT source remains.

### 4.4 Qualitative Cutoff Properties

Qualitative properties are temporally extended properties expressed in LTL over the variables provided by the engines' interface. When the property is violated an alarm is sent to the vehicle controller and some recovery action is taken (either the output of the corresponding PNT engine is not sent to the Fuser or some reconfiguration of the engine is triggered). More specifically, we monitored the following conditions:

- **GNSS Properties**
  - $\mathbf{G}_{[0,3]}\ hpl < th$ , where $th$ is a threshold based on the average road width or, if the information is available, on the width of the current road the vehicle is driving. Its violation will trigger the informative alarm *high_hpl* and block the output of the GNSS-based PNT to the fuser.
- **SLAM Properties**
  - $\mathbf{G}_{[0,3]}\ \neg low\_luminosity$ . Its violation will trigger the $swap\_feature\_extractor$ alarm. that will switch to a feature extractor better suited for low-illumination imagery.
  - $\mathbf{G}_{[0,2]}\ \neg frozen\_image_{\{left,right\}}$ . This requires the image of the left/right camera not to remain frozen for 2 seconds. Its violation will trigger the informative alarm $frozen\_image$ that will block the output of the AI PNT to the fuser.
  - $\neg camera_{\{left,right\}}\_alive\ \mathbf{U}_{[0,10]}\ camera_{\{left,right\}}\_alive$. This requires that the left/right camera cannot remain unresponsive for up to 10 seconds. Its violation will trigger the $camera_{\{left,right\}}\_dead$ alarm and block the output of the AI-based PNT to the fuser.

### 4.5 Quantitative Weighting Properties

For engines that satisfy the qualitative cutoff properties, the Safety Cage computes a dynamic weight $w \in [0, 1]$. This approach allows the fuser to gracefully handle performance degradation without abruptly disconnecting a source.

**Parametric Sigmoid Function.** Each monitoring property utilizes a parametric sigmoid function to map a real-time health metric $x$ to a weight:

$$f(x; c, \alpha) = \frac{1}{1 + e^{-k(x-c)}}$$

where $c$ is the center of the transition and $\alpha$ defines the sensitivity interval $[x_1, x_2]$ (with $x_{1,2} = c \mp \alpha/2$). The slope $k$ is derived to ensure the function reaches near-boundary values at the interval endpoints.

**GNSS Weighting Logic.** The GNSS weight is primarily driven by the stability of the HPL. The system computes a normalized metric $x = -hpl(t)/W_{hpl}$ where $W_{hpl}$ is the mean of $hpl$ in the last 20 steps. By applying the sigmoid function with $c = -1.5$ and $\alpha = 1.3$, the monitor effectively penalizes the GNSS source when the current uncertainty spike exceeds the recent average by a safety-tuned margin.

**SLAM Weighting Logic.** The weight of the AI-based engine is computed with three complementary metrics derived from the SLAM interface:

1. **Feature Drop Ratio:** This metric detects texture-less environments by monitoring the ratio $\frac{nf(t)}{W_{\text{nf}}}$ of the latest feature count to the mean of features in the last 10 steps. By applying the sigmoid function with $c = 0.5$ and $\alpha = 0.5$, the weight drops sharply if the number of detected features falls significantly below the recent trend.
2. **Extraction Time Ratio:** This metric monitors the relative change in feature extraction time (again normalizing with respect the mean of the last 10 steps). In this case, the sigmoid function has been set with parameters $c = -3$ and $\alpha = 1.5$.
3. **Covariance Spike Monitor:** This metric uses the norm of the diagonal sub-matrix related to the position of the SLAM covariance. In this case, the input of the sigmoid function is the deviation of current norm from the average of the last 10 steps, using $c = -3$ and $\alpha = 1.5$.

The final weight $w$ for the AI-based PNT is the minimum value produced by the three metrics. This ensures that a single indicator of failure (e.g., a feature drop) is sufficient to reduce the trust in the engine.

**Parameter tuning.** Several of the formulas above depend on specific parameters, such as $c$ and $\alpha$ for the sigmoid function, the interval durations for bounded temporal operators, and the window lengths for rolling averages. We selected these values empirically by tuning them via simulation on a subset of our test scenarios (see §5) to effectively balance sensitivity and robustness in the ability of the supervisor to react to anomalous situations. A more systematic exploration of these choices, including a formal sensitivity analysis or machine-learning-based optimization, is left for future research.

## 5 Experimental Evaluation

We have implemented our proposed safety cage architecture in a prototype tool written in Python, and integrated it in the CARLA [5] urban driving simulation environment for experimentally evaluating its performance.

We evaluate the effectiveness of our safety cage architecture by integrating our prototype implementation in an autonomous vehicle inside CARLA, and by

executing an extensive set of simulation scenarios, considering multiple environment conditions and different fault situations.

### 5.1 Simulation Setup

We use the standard Python API provided by CARLA to integrate our safety cage in the simulator. More specifically, we develop a custom client for the CARLA server, which instantiates an autonomous vehicle equipped with a virtual GNSS receiver (to provide data to the GNSS-based PNT engine) and two RGB cameras mounted on the front of the vehicle (which provide data to the AI-based PNT engine). The vehicle is given as input a trajectory to follow, and uses the autopilot functionalities provided by CARLA for autonomous driving.[3] For repeatability, we configure CARLA to operate in synchronous mode. At each simulation tick, the client queries the relevant sensors (GNSS receiver and RGB cameras) for data, feeds it to the corresponding PNT engines (if new data is available), computes the weights for their solutions and fuses them to obtain the PNT output. We use a duration of 0.2 seconds for each tick, which corresponds to the operating frequency (5 Hz) of the AI-based PNT engine. The GNSS-based engine, instead, operates with a frequency of 1 Hz. All components of our architecture were able to operate within their specified frequencies. In particular, COLMAP-SLAM took on average 88.9 ms to process each image pair in our experiments; regarding the safety cage component, the computational overhead of the NuRV monitors was almost negligible, thanks to the simplicity of the properties and the lack of assumptions.

As described earlier, we integrated a high-fidelity virtual multi-constellation GNSS receiver into the simulator. This receiver generates simulated GNSS satellite signals, incorporating models of real-world effects such as ionospheric delay, and provides them to the commercial off-the-shelf (COTS) GNSS-based PNT engine, MSP3, [8], for position estimation. A further advantage of this approach is that the GNSS signal simulator supports signal occlusion, allowing the local CARLA environment to influence GNSS signal quality. Consequently, environmental features such as trees, tunnels, buildings, and urban canyons dynamically affect GNSS reception, thereby improving realism. We selected this approach over the built-in GNSS sensor of CARLA because the latter relies on a simplistic model that merely adds noise to the vehicle ground-truth position.

To determine the set of visible satellites, at each simulation tick we compute a signal quality factor for every satellite that is above the horizon. This factor ranges from 0.0, representing complete blockage, to 1.0, representing clear-sky visibility. The value is computed by casting a ray from the ego vehicle to each satellite in the sky. Any geometry in the simulation environment that intersects the ray reduces the signal quality linearly according to a predefined attenuation factor determined by the object type. These signal quality values are then used by the GNSS simulator to model the resulting GNSS measurements more accurately.

[3] Specifically, we use an instance of the `BasicAgent` class from the `carla.agents.navigation` module for controlling the vehicle, providing it the input trajectory as a global plan to follow.

### 5.2 Scenario Generation and Fault Injection

For evaluating our safety cage architecture, we generate a set of simulation scenarios using an approach based on coverage-guided automated test case generation. In particular, we adopt the generic VIVAS methodology [6] that generates simulation scenarios using a formal model of the system under test, its environment (called the abstract system), and scenario exploration through symbolic reasoning techniques. Scenarios are generated by systematically exploring values for predicates expressing constraints on the main parameters affecting the computations of the two PNT solutions along the following dimensions:

**Map of the world.** We consider different built-in maps of CARLA, providing different environmental features (e.g. dense urban aeras, rural/forest areas, highways, narrow roads, tunnels);

**Coordinates of the map on Earth,** affecting the availiability of satellites. We discretize the possible coordinates corresponding to locations of some relevant cities at different latitudes;

**Weather and visibility conditions,** affecting the quality of the images captured by the RGB cameras and provided as input to the AI-based PNT engine. We select among the available weather conditions provided by CARLA (e.g. "clear day at noon", "foggy night", "hard rain at sunset");

**Trajectory to follow.** To represent navigation tasks in a generic way, we introduce the concept of abstract Point Of Interest (POI). Conceptually, a POI consists of a location on a map with a set of semantic tags attached to it. We then define a navigation task as a sequence of POIs (global plan) that the autonomous vehicle must reach in the given order. The intermediate waypoints (local plan) between two POIs are determined by the COTS autonomous driving controller. We define different semantic tags corresponding to different features in the enviornment (e.g. "rural area", "forest area", "high buildings area", "tunnel"), and enumerate trajectories traveling through different such areas;

**Fault conditions.** To evaluate the safety cage performance in non-nominal situations, our simulator supports the injection of faults that can be activated at run time. Faults are defined with a type and an interval (in simulation time) during which they are active. A fault of type "GNSS receiver" affects the GNSS-based PNT engine. For each such fault, we specify (a) which GNSS constellations should be available and (b) an upper bound $N$ for the number of satellites that should be visible the duration of the fault. During the simulation, for all ticks when a GNSS receiver fault is active, we limit the satellites available for the computation of the GNSS measurement by randomly selecting $N$ satellites and blocking all the remaining ones. Faults of type "RGB camera", instead, affect the AI-based PNT engine. For each such fault, we additionally specify the camera affected (left, right, or both) and the kind of defect of either (a) image completely black, or (b) image stuck at a specific frame. We constrain each generated scenario to include one fault of the above kinds.

### 5.3 Results

To evaluate the quality of the PNT solution computed by our engine, we use two common metrics for the evaluation of trajectory errors, namely the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE). The former (ATE) evaluates the global consistency of the estimated trajectory when compared with the ground truth, whereas the latter (RPE) measures the local trajectory accuracy, i.e., incremental changes of subsequent vehicle poses, in the vehicle reference frame. Both are widely used metrics in evaluating positioning systems when ground truth data is available (see e.g. [25]). We use the implementations available in the field, the `evo` Python package, for the evaluation of odometry and SLAM [7].

For each simulation run, we extract both the ground truth trajectory and the trajectory computed by the VAIPOSA PNT from the simulation logs, and compute the ATE and RPE using the `evo` library. We then aggregate the data by computing the median and root mean squared error (RMSE) of the two trajectory errors. We evaluate the performance of the default safety cage configuration by comparing the values for such metrics against those produced by the baseline configurations using only a single engine (either GNSS-based or AI-based). Additionally, we also evaluate a configuration using fixed values for the weights given to the output of the two PNT engines in the fusion algorithms, effectively disabling the safety cage. (In the following, we refer to such configuration as "fixed weights".)

**Table 1.** Summary of per-scenario statistics comparing the default configuration against the baselines (GNSS-only and SLAM-only) and the fixed-weights configurations. All the values are expressed in meters (with the best ones highlighted).

| Metric | Default | GNSS only | SLAM only | Fixed weights |
|---|---|---|---|---|
| **ATE - median** | Max: 74.95 | Max: 291.17 | Max: 912.08 | Max: 22.91 |
| | Med.: 1.51 | Med.: 1.68 | Med.: 7.18 | Med.: 1.57 |
| | Mean: 2.81 | Mean: 3.64 | Mean: 49.31 | Mean: 3.55 |
| | MAD: 1.54 | MAD: 2.26 | MAD: 46.07 | MAD: 2.31 |
| **ATE - RMSE** | Max: 83.77 | Max: 3139.90 | Max: 1094.30 | Max: 660.28 |
| | Med.: 4.86 | Med.: 20.14 | Med.: 16.24 | Med.: 6.10 |
| | Mean: 8.51 | Mean: 62.73 | Mean: 68.32 | Mean: 23.39 |
| | MAD: 6.02 | MAD: 53.07 | MAD: 62.82 | MAD: 19.71 |
| **RPE - median** | Max: 2.01 | Max: 19.94 | Max: 2.08 | Max: 2.02 |
| | Med.: 0.20 | Med.: 0.30 | Med.: 0.11 | Med.: 0.25 |
| | Mean: 0.33 | Mean: 0.44 | Mean: 0.52 | Mean: 0.48 |
| | MAD: 0.15 | MAD: 0.19 | MAD: 0.42 | MAD: 0.30 |
| **RPE - RMSE** | Max: 30.18 | Max: 492.14 | Max: 16.89 | Max: 205.63 |
| | Med.: 0.89 | Med.: 5.40 | Med.: 0.42 | Med.: 1.36 |
| | Mean: 1.74 | Mean: 13.94 | Mean: 1.66 | Mean: 6.34 |
| | MAD: 1.11 | MAD: 11.37 | MAD: 1.45 | MAD: 5.44 |

A summary of the results is reported in Table 1, showing different statistics (max, median, mean, and Median Absolute Deviation) of the metrics above over our data set, which consists of 360 scenarios. From the results, we can see that the default configuration is the best performing one for the majority of the

indicators. This is true especially when considering the ATE errors, indicating its ability to better adapt to changing conditions and promptly react to unusual behavior of one the PNT solutions, compared to the individual engines, in terms of global positioning accuracy. Specifically, greatly lower ATE RMSE values indicate the engine robustness to outliers at different difficulty levels. The lowest MAD of this metric demonstrates also the consistency of the quality of results delivered by the default configuration. Despite the lack of the "global" aspect, in the autonomous vehicle navigation, RPE metrics also play a critical role in assuring the appropriate quality of positioning in the local scope, which directly interferes with the real-time vehicle control and planning. The results show that the VAIPOSA configuration outperforms GNSS-only and fixed weights configurations across almost all statistics, and is very close to the SLAM-only configuration, which directly provides relative pose estimations, while increasing the overall resilience to outliers in the provided PNT results.

In order to gain additional insights, we study the variations in performance in the presence of faults. By construction, each simulation scenario in our data set contains one fault activation (either of the GNSS receiver or of one or both on-board cameras, as described in §5.2 above). To evaluate the impact of such faults, for each simulation run we extract two additional trajectories: (i) a "pre fault" trajectory, consisting of the prefix of the simulation execution up to the time point at which the fault is activated; and (ii) a "during fault" trajectory, consisting of the subsequence of time points during which the fault is active.

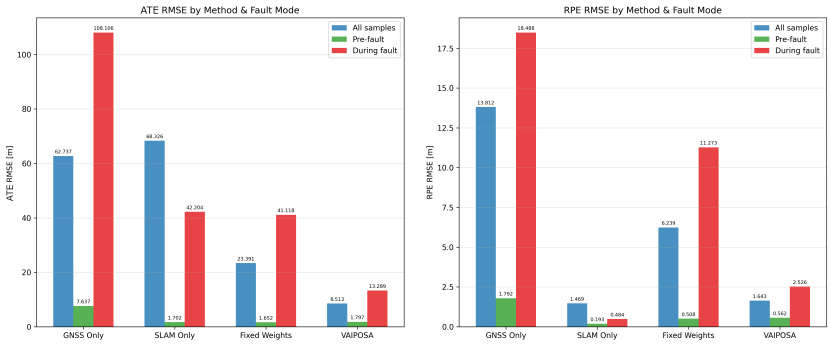


**Fig. 2.** Impact of faults on ATE and RPE RMSE for the different configurations.

An overview of the results is shown in Figure 2, displaying histograms of the RMSE for the ATE and RPE values obtained with the four different configurations over our data set. As expected, from the results we can conclude that the safety cage is especially important during non-nominal situations, substantially reducing the observed trajectory errors (particularly for the ATE case). In particular, the performance of the PNT engine is not degraded during the regular use (ATE, RPE in all/pre-fault conditions), but is significantly improved when

introducing faults to any of the engines. In the current configurations, experiments showed greater and more consistent improvement in global positioning capabilities (ATE during fault), while in terms of relative positioning, the median error does improve over the baselines, but still retains a visible tail of higher errors. We conjecture this could be potentially addressed by further expanding the safety cage capabilities of detecting and acting on SLAM faults. Finally, we wish to comment on the different trend in results obtained by the SLAM configuration compared to the others, where the measures for "all samples" are worse than those for "during fault" situations. This occurs because SLAM systems naturally accumulate error and drift over time. Since the "during fault" trajectories are shorter sub-trajectories of the full "all samples" runs, they only capture a fraction of the total drift. The "all samples" metrics evaluate the entire duration, allowing error to compound continuously and ultimately resulting in the worst overall performance. Additionally, some cases producing a high RMSE value for the SLAM-only configuration are due not to scenarios containing faults, but simply to difficult-to-reconstruct 3D scenes.

## 6 Lessons Learned and Conclusion

In this report, we presented the design and evaluation of a safety cage for hybrid PNT systems. Our results demonstrate that the VAIPOSA architecture successfully maintains positioning integrity by dynamically weighting GNSS and AI-based PNT sources, significantly reducing the impact of engine-specific faults in various scenarios, without degrading performance during nominal conditions. The development and validation of the VAIPOSA Safety Cage revealed several key insights:

- Despite the many works on safety monitors and shields, there is still a lack of methodology in defining the monitoring properties, and experimenting with the system or a simulator is essential to tune the monitoring parameters.
- To ensure generality and robustness, monitors should be stateful. For instance, after experimentation, we realized that the HPL is initially too pessimistic and we considered its trend rather than the absolute value until a stabilization point.
- There is an inherent trade-off between minimizing absolute position error and maintaining trajectory smoothness. While our current filter minimizes error, it can lead to discontinuous trajectories, whereas the AI-based PNT solution offers smoothness but lacks absolute global accuracy.
- Fusing global and local PNT solutions as well as exploiting an extensive interface with the PNT engines were key to enable robust performance in a wide range of conditions, even using simple filter-based fusion algorithms.
- Using a simulator with 3D environment models, jointly with an automatic testing procedures of VIVAS, was essential for debugging and created a realistic and high-performance benchmarking environment that can be used also for evaluating other PNT solutions for autonomous vehicles. However, we explicitly acknowledge that our current evaluation is simulation-only and

based on a limited set of faults, while a more realistic validation setup on physical target platforms is required to fully capture the complexity of physical hardware degradation in real-world deployments.
– The GNSS-based PNT engine used in our prototype already performs internal filtering. We conjecture that having access to "raw" GNSS measurements and performing the fusion only after the safety cage would provide better control and potentially superior results.

Future research will focus on evolving the safety cage into a fully stateful monitor capable of providing formal safety certificates and protection levels for the fused output. Additionally, to address a key limitation of our current configuration, which lacks an Inertial Measurement Unit (IMU), we plan to explore the integration of inertial sensors to further improve resilience during extended periods of GNSS and visual faults.

## References


1. Deliverable of PASSIONS (Software Product Assurance For Autonomous On-Board Software) - Executive Summary Report. https://nebula.esa.int/sites/default/files/neb_tec_studies/3154/public/PASSIONS_Executive_Summary_Report_v1.0.pdf (2023), last accessed May 20, 2026
2. Cimatti, A., Tian, C., Tonetta, S.: NuRV: A nuXmv extension for runtime verification. In: RV. LNCS, vol. 11757, pp. 382–392. Springer (2019)
3. Cremona, J., Civera, J., Kofman, E., Pire, T.: Gnss-stereo-inertial slam for arable farming. Journal of Field Robotics **41**(7), 2215–2225 (2023)
4. DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description. In: CVPR Workshops. pp. 224–236 (2018)
5. Dosovitskiy, A., Ros, G., Codevilla, F., López, A.M., Koltun, V.: CARLA: an open urban driving simulator. In: CoRL. pp. 1–16. Proceedings of Machine Learning Research, PMLR (2017)
6. Goyal, S., Griggio, A., Tonetta, S.: System-level simulation-based verification of Autonomous Driving Systems with the VIVAS framework and CARLA simulator. Sci. Comput. Program. **242**, 103253 (2025)
7. Grupp, M.: evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo (2017), last accessed May 20, 2026
8. Hutchinson, M., Tiwari, S., Zheng, Y., Moradi, R.: Carrier Phase Positioning Techniques for Mass Market GNSS Receivers: Enhancement of MSP3 Precise Point Positioning (PPP) Software. In: 7th International Colloquium on Scientific and Fundamental Aspects of GNSS (2019), conference abstract.
9. Könighofer, B., Bloem, R., Jansen, N., Junges, S., Pranger, S.: Shields for Safe Reinforcement Learning. Commun. ACM **68**(11), 80–90 (2025)
10. Lindenberger, P., Sarlin, P.E., Pollefeys, M.: Lightglue: Local feature matching at light speed. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 17627–17638 (2023)
11. Liu, J., Gao, W., Hu, Z.: Optimization-based visual-inertial slam tightly coupled with raw gnss measurements. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 11612–11618 (2021)
12. Macario Barros, A., Michel, M., Moline, Y., Corre, G., Carrel, F.: A comprehensive survey of visual slam algorithms. Robotics **11**(1), 24 (2022)

13. Machin, M., Guiochet, J., Waeselynck, H., Blanquart, J., Roy, M., Masson, L.: SMOF: A Safety Monitoring Framework for Autonomous Systems. IEEE Trans. Syst. Man Cybern. Syst. **48**(5), 702–715 (2018)
14. Morelli, L., Ioli, F., Beber, R., Menna, F., Remondino, F., Vitti, A.: COLMAP-SLAM: A framework for visual odometry. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **48**, 317–324 (2023)
15. Morelli, L., Trybała, P., Razzino, A.V., Remondino, F.: Deep learning in visual odometry for autonomous driving. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences **48**, 93–99 (2025)
16. Pereira, P.M.C., da Silva, H.D.M., Lima, C.M.G.S.: Advancements in multipath mitigation for gnss receivers: Review of channel estimation techniques. Space: Science & Technology **5**, 0278 (2025)
17. Pranger, S., Könighofer, B., Posch, L., Bloem, R.: TEMPEST - Synthesis Tool for Reactive Systems and Shields in Probabilistic Environments. In: ATVA. Lecture Notes in Computer Science, vol. 12971, pp. 222–228. Springer (2021)
18. Psiaki, M.L., Humphreys, T.E.: Gnss spoofing and detection. Proceedings of the IEEE **104**(6), 1258–1270 (2016)
19. Revach, G., Shlezinger, N., Ni, X., Escoriza, A.L., Van Sloun, R.J., Eldar, Y.C.: Kalmannet: Neural network aided kalman filtering for partially known dynamics. IEEE Transactions on Signal Processing **70**, 1532–1547 (2022)
20. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR Workshops. pp. 4104–4113 (2016)
21. Schreiber, M., Königshof, H., Hellmund, A.M., Stiller, C.: Vehicle localization with tightly coupled gnss and visual odometry. In: 2016 IEEE Intelligent Vehicles Symposium (IV). pp. 858–863. IEEE (2016)
22. Sha, L.: Using simplicity to control complexity. IEEE Softw. **18**(4), 20–28 (2001)
23. Skog, I., Handel, P.: In-car positioning and navigation technologies—a survey. IEEE Transactions on Intelligent transportation systems **10**(1), 4–21 (2009)
24. Socha, K., Borg, M., Henriksson, J.: SMIRK: A machine learning-based pedestrian automatic emergency braking system with a complete safety case. Softw. Impacts **13**, 100352 (2022)
25. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of rgb-d slam systems. In: IROS. pp. 573–580. IEEE (2012)
26. Tao, L., Watanabe, Y., Yamada, S., Takada, H.: Comparative evaluation of kalman filters and motion models in vehicular state estimation and path prediction. The Journal of Navigation **74**(5), 1142–1160 (2021)
27. Wisniewski, M., Chatzithanos, P., Guo, W., Tsourdos, A.: Benchmarking Deep Reinforcement Learning for Navigation in Denied Sensor Environments. J. Intell. Robotic Syst. **111**(3), 103 (2025)
28. Zhao, X., Wu, X., Chen, W., Chen, P.C., Xu, Q., Li, Z.: Aliked: A lighter keypoint and descriptor extraction network via deformable transformation. IEEE Transactions on Instrumentation and Measurement **72**, 1–16 (2023)
29. Zhu, N., Marais, J., Bétaille, D., Berbineau, M.: Gnss position integrity in urban environments: A review of literature. IEEE Transactions on Intelligent Transportation Systems **19**(9), 2762–2778 (2018)