Safe and Explainable
Critical Embedded Systems based on AI

D5.1 Case study stubbing and early assessment of case study porting

Version 1.0

Documentation Information

Contract Number	101069595
Project Website	www.safexplain.eu
Contractual Deadline	31.03.2024
Dissemination Level	PU
Nature	R
Author	Mikel Aldalur (IKR), Gabriele Giordana (AIKO), Shruthi Gowda (NAV)
Contributors	Joanes Plazaola (IKR), Francesco Rossi (AIKO)
Reviewer	Francisco Cazorla (BSC)
Keywords	Artificial Intelligence, space, automotive, railway, stubbing

This project has received funding from the European Union's Horizon Europe programme under grant agreement number 101069595.

Change Log

Version	Description Change
V0.1	First draft
V0.2	Reviewed version
V1.0	Final version

Table of Contents

1. Introduction 4
2. Space Case Study 5
2.1 Datasets collection and generation 5
2.2 Architecture 9
2.3 Training 11
2.4 Local testing 11
2.5 Platform testing and early porting....................12
2.6 Scenario Generation 13
2.7 Next steps 14
3. Automotive Case Study....................16
3.1 Datasets collection and generation 16
3.2 Architecture 20
3.3 Training 21
3.4 Local testing 22
3.5 Platform testing and early porting....................23
3.6 Scenarios generation 24
3.7 Next steps 27
4. Railway Case Study 28
4.1 Datasets collection and generation 28
4.2 Architecture 29
4.3 Training 30
4.4 Local testing 31
4.5 Platform testing and early porting....................32
4.6 Next steps 34
5. Toy model 36
5.1 Dataset Generation....................36
5.2 Model Architecture....................36
5.3 Training 37
6. Conclusion....................38

Executive Summary

This document focuses on state-of-the-art and representative mixed-criticality case studies from the automotive, railway and space domains, and their preparation for the project activities. It describes the specifications and stubbing activities performed to deliver the case studies, including designing the models, collecting datasets for training and testing the algorithms, and implementing the features needed in the scope of the project. All these tasks were carried out with continual interaction with other work packages in order to assess the impact on the overall work and provide a common direction.

An accurate implementation of the case studies is paramount, since SAFEXPLAIN outcomes will be demonstrated by integrating its solutions in a commercial toolset for system testing and applying its principles to the case studies. The project wishes to capture real challenges that can emerge from the application of the techniques in industrial-size projects.

In all case studies, Deep Learning is the building block of Critical Autonomous AI-based Systems (CAIS) safety-related functions. All three case studies will follow the same procedure: stubbing of the inputs and the operational environment, porting and integration into the project selected platform, evaluation and assessment within SAFEXPLAIN software environment.

1. Introduction

This project seeks to close the gap between Functional Safety Requirements and the nature of Deep Learning (DL) solutions. Functional Safety systems need deterministic, verifiable and pass/fail test-based software solutions, however, DL-based solutions currently lack explainability, traceability, robustness and security.

AI explainability may help during the process of safety system certification by proposing evidence that helps show certification authorities that AI-based sensors are properly trained, and work as expected. AI explainability may also help by adding diagnosis capabilities to the safety system to implement redundant systems with diagnosis capabilities during safety system operation.

To test the functional safety requirements, the case study solutions will be applied as first test benchmarks. These solutions are specific to each use case and will be described in detail in the following sections, including the operational scenarios, the envisioned applications and their practical implementations. Secondly, each case study section will describe the activities performed so far to prepare the algorithms: the gathering and/or generation of datasets, an essential element for DL applications, the stubbing of inputs from the real operational environment, the training and testing of the AI models and the porting to the platform.

Finally, an early need arose in the first phase of the project: bridging the gap between work packages in order to provide cohesion in the parallel tasks of implementation that were starting. Since the case studies are complex models that need careful design, implementation and training, it was not possible to provide them as an early testbed; nonetheless, the first prototypes of the SAFEXPLAIN software stack (WP4), explainability techniques (WP3) and safety architectures (WP2) needed a DL model to early test their assumptions and identify potential integration issues. To allow this, WP5 devoted effort to provide a DL model fast enough to be designed and trained, without confidentiality issues to be shared openly, and which exercises the core functionalities of the case studies. This model, referred to as the *Toy Model* and described in detail in section 5, was prepared and handed over to the other work packages.

2. Space Case Study

The space case study is composed of a DL algorithm included in a safety critical autonomous system, in the operational scenario of spacecrafts autonomous navigation and docking.

An autonomous GNC (Guidance, Navigation and Control) system is designated to navigate a spacecraft acquiring information from the asset sensors (cameras, star trackers, inertia measurement units and more), assessing the position and attitude of the spacecraft and performing the adequate manoeuvres to specific ends, being motion, orbital station-keeping or approaching to targets.

In this scenario, a crewed spacecraft is envisioned to perform a docking manoeuvre to an uncooperative target (a space station or another spacecraft) on a specific docking site. The system must be able to acquire the pose estimation of the docking target and of the spacecraft itself, to compute a trajectory towards the target and to send commands to the actuators to perform the docking manoeuvre.

The safety goal is to dock with adequate precision and avoid crashing or damaging the assets; since both the spacecraft and the target have a crew, minor damages to the vehicles may put at risk the crew safety. Moreover, approaching the target in an orthogonal way with respect to the docking site is required for a safe manoeuvre.

In order to focus the effort of the project, this case study takes into account a specific module of the envisioned GNC system, the pose estimation module. This component takes as input images from a monocular grayscale camera of the target, and must compute its pose, namely the relative position and rotation (three dimensions each) between the target and the chaser. This is a fundamental step to being able, later in the GNC pipeline, to plan a trajectory and compute actuators commands to follow it.

In particular, the pose estimation functions under evaluation are:

1. Detection of the docking target and its orientation
2. Computation of the relative position and attitude between spacecraft and target
3. Computation of orthogonality to the docking site

2.1 Datasets collection and generation

2.1.1 Real vs Simulated Data

In the context of SAFEXPLAIN, the availability and utility of real-world data often pose significant limitations, particularly when it comes to comprehensive metadata or ground truth information. This scarcity or complete lack of adequate real data necessitates an alternative approach to dataset generation and collection, prompting the utilization of simulated data. Such an approach is not unique to our endeavour; the European Space Agency, among others, has historically relied on simulated datasets to circumvent these limitations [1], [2]. Over the years, various versions of simulated datasets have been progressively opened to the scientific community, marking a significant trend towards the adoption of synthetic data. This trend is evident across numerous institutions, academia, and private entities, underscoring the strategic advantages of synthetic data development.

Concerning the topic of Satellite Pose Estimation, the most relevant one is SPEED+ [3], made by the Space Rendezvous Laboratory (SLAB) of Stanford University.

The creation of simulated data offers unparalleled freedom in generating not only the data itself but also the associated metadata. This flexibility extends to the simulation of extreme or catastrophic conditions, which are crucial for the balanced training of artificial intelligence (AI) models. Ensuring a comprehensive representation of all possible scenarios, especially critical cases, is fundamental for the models' inferencing capabilities and the realism targeted. Our approach draws direct inspiration from the SPEED+, yet we have endeavoured to enhance the realism, diversity, and quality of the associated metadata. This initiative has allowed us to achieve a higher level of complexity, in line with the state-of-the-art in estimating the pose of cooperative and non-cooperative satellites. By adopting a data-centric approach, we have incorporated a wide range of simulated conditions, from standard operational ranges to extreme and catastrophic scenarios.

This methodology enables us to iteratively improve the accuracy, precision, and reliability of our AI model's output. By expanding the scope and depth of our simulated data, we position our project at the forefront of technological advancements, ensuring our AI models are well-equipped to handle a diverse array of challenges with enhanced efficacy.

The generation of our dataset is a meticulously designed process that integrates both real-world constraints and the expansive potential of simulated environments. This dual approach ensures that our AI models are not only trained on a broad spectrum of data but are also capable of generalizing across a variety of unseen, real-world situations. The dataset generation process involves several key steps:

- Synthesis of Simulated Data: Leveraging advanced simulation technologies, we generate a wide array of synthetic data that mirrors real-world phenomena under various conditions, including those that are rare or have not been observed. This process allows for the inclusion of comprehensive metadata, enhancing the dataset's utility for training AI models.
- Enhancement of Realism: Building on the foundation laid by predecessors such as SPEED+, we incorporate advanced algorithms to increase the realism of our simulated data. This includes refining physical models, improving environmental variables, and ensuring that simulated entities behave in ways that are indistinguishable from their real-world counterparts.
- Metadata Enrichment: Each piece of data in our dataset is accompanied by detailed metadata, providing essential context that is crucial for effective model training. This metadata includes information about the conditions under which the data was generated, parameters that influence the simulation, and annotations that facilitate the accurate interpretation of data by AI models.
- Scenario Coverage: Our dataset encompasses a comprehensive range of scenarios, from routine operational conditions to extreme and unlikely events. This ensures that the AI models trained on our dataset are robust and can perform reliably under a wide variety of circumstances.
- Iterative Refinement: The dataset generation process is iterative, allowing for continuous refinement based on feedback from ongoing model training and validation efforts. This cyclical process ensures that the dataset evolves in line with

emerging requirements and technological advancements, maintaining its relevance and effectiveness.

Through this structured and strategic approach to dataset generation, we ensure that our AI models are trained on high-quality, diverse, and realistic data. This foundation empowers the models to achieve high levels of accuracy and reliability, crucial for the success of our project and the advancement of AI applications in our field.

2.1.2 Dataset generation pipeline

Our dataset generation pipeline is a sophisticated system designed to create a comprehensive and detailed dataset for satellite imagery, leveraging state-of-the-art technological components. This pipeline is critical for the development, training, and validation of our artificial intelligence (AI) models, aimed at enhancing satellite pose estimation and other space-related applications. Below is a detailed overview of the pipeline's components and processes:

- Satellite 3D CAD Model Enhancements: The pipeline initiates with an enhanced 3D CAD model of the target satellite. Significant improvements have been made to this model, including the addition of detailed material properties. These enhancements are crucial for achieving realistic simulation outcomes, as they directly affect the model's interaction with simulated environmental conditions.
- Integration into the Unity Simulation Environment: The 3D model is then integrated into the Unity simulation environment. Within Unity, we have meticulously set up various parameters to simulate realistic conditions accurately. These include:
    - Lighting Conditions: Customizable to replicate positions of the sun and lighting scenarios.
    - Environmental Conditions: Incorporates dynamic backgrounds, including a high-resolution Earth asset, which can be adjusted for various viewing angles and positions.
    - Camera Properties: The simulation includes detailed settings for the camera used to capture images of the satellite model. These settings encompass sensor type, optics characteristics, field of view, and additional features such as noise addition and lens distortion. Images are captured both randomly and along predefined trajectories, at set frequencies, to generate a diverse dataset.
- Image Acquisition: the pipeline can generate images in both colour and grayscale formats. While grayscale images are preferred for their efficiency in processing time for space applications (an example in Figure 1), colour images are also produced to future-proof the dataset for potential advancements in perception capabilities.
- Metadata, Labels, and Ground Truth Association: beyond mere image capture, our pipeline excels in its ability to associate each image with a rich set of metadata, labels, and ground truths, as can be seen in Figure 2. These include:
    - Relative 6D Pose Estimation: Data regarding the relative positioning and orientation between the satellite and its target, including Rotation (3D) and Translation (3D) vectors.

  - Masking and Depth Estimation: Information on the geometric properties critical for docking and engagement scenarios, such as the level of orthogonality at the docking point.
  - This metadata is essential for accurately training and validating AI models, providing a detailed contextual framework for each image in the dataset.
- GUI Customizable Parameters and Exporting Process: A key feature of our pipeline is the extensive customization it offers, allowing for precise control over various parameters, including:
  - Camera Settings: Adjustments to sensor type, optical view angle, and image resolution
  - Operational Range: Defining the minimum and maximum distances within which image capture is conducted, tailoring the dataset to specific simulation needs for both training and testing purposes.

Eventually, the pipeline culminates in the automatic generation of a zipped folder containing all pertinent images, metadata, and ground truths. This folder serves as the foundation for the AI model's training and validation processes.

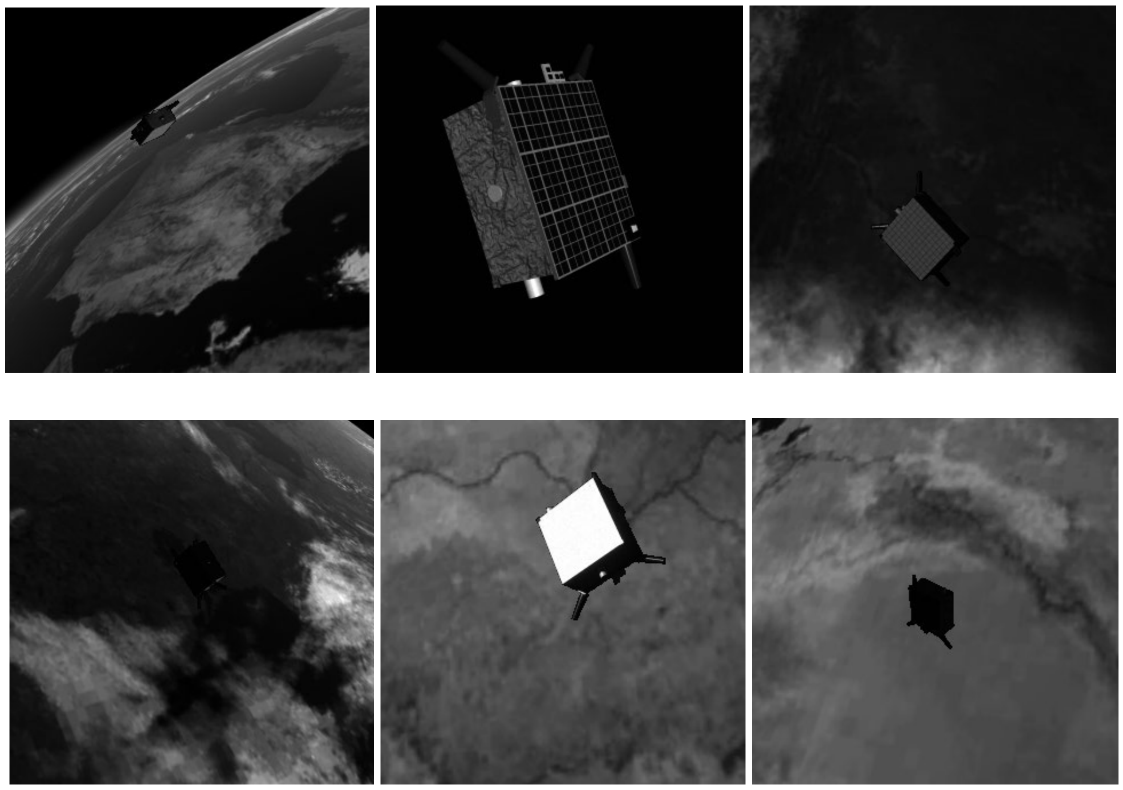


Figure 1. Dataset example images

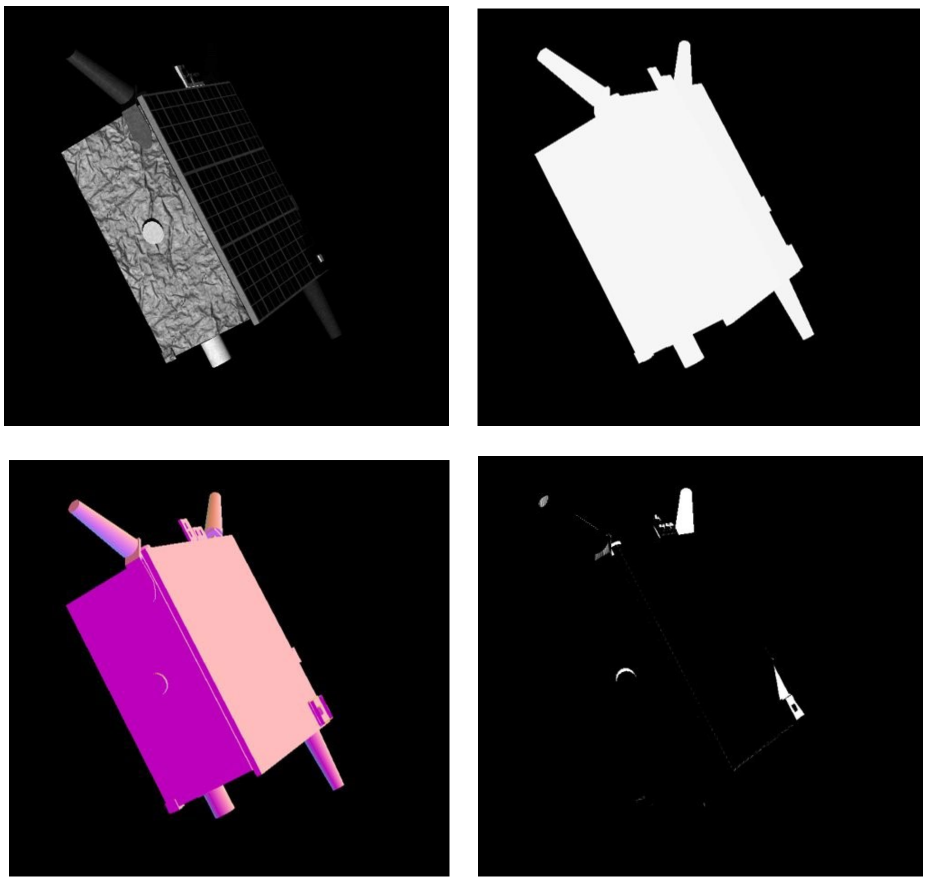


Figure 2. Ground-Truth Example Images, associated to the main camera frame (top left): segmentation mask (top-right), normal map (bottom-left), shadow mask (bottom-right)

2.2 Architecture

This case study employs a multitask deep learning model, specifically utilizing convolutional neural networks (CNNs), to address complex challenges in satellite relative navigation. This model is designed to process a single input — a simulated camera image of a target satellite — and produce multiple outputs, leveraging the strengths of a multitask framework to enhance the efficiency and accuracy of the task at hand. Below, we delineate the architecture and functionalities of our model.

The architecture is structured around a single-input, multiple-output (SIMO) framework, where the input is an image capturing the relative navigation context between the observing satellite and the target satellite. The model is tasked with generating several critical outputs from this input:

- Object Classification: Identifies the type of satellite or object within the image.
- Bounding Box Detection: Locates the object within the image by defining its bounding box.
- Direct Pose Estimation: Provides a 6D pose estimation of the target, combining 3D rotation and 3D translation to offer a comprehensive spatial understanding. Calculates the pose directly from the input image, leveraging the multitask learning framework to integrate seamlessly with other output tasks.

- Segmentation Mask: Outputs a binary mask delineating the silhouette of the satellite, facilitating precise object boundary identification.
- Depth Mask: Like the segmentation mask, this grayscale mask encodes distance information, providing depth perception relative to the camera's position.
- Shadow Mask: A binary mask highlighting the areas directly illuminated by a light source versus those in shadow, aiding in the understanding of lighting conditions.
- Normal Map: An RGB mask that encodes the orientation of the satellite's surface in space, using color coding to represent spatial configurations in a target-centric manner.
- Indirect Pose Estimation: This branch employs a two-step process. First, it detects key points on the target using non-maximum suppression (NMS) algorithms applied to heatmaps of the key points. Then, it utilizes a Perspective-n-Point (PnP) algorithm to reconstruct the object's pose based on these identified key points, offering an alternative pose estimation that complements the direct approach.
- Docking Point Orthogonality Estimation: A dedicated branch within the architecture focuses on estimating the normal relative to the docking site. This component is crucial for understanding the approach vector needed for successful docking operations, providing valuable data for mission planning and execution.

At the heart of our multitask architecture is a shared backbone, specifically employing an Efficient-Net architecture. This choice of backbone is strategic, allowing for "hard parameter sharing" across various tasks. This means the same backbone processes the input image, distributing its learned features to each task-specific head of the network. This approach optimizes the use of computational resources and enables the model to learn more generalized representations.

The whole schema of the implemented AI model architecture is provided in Figure 3.

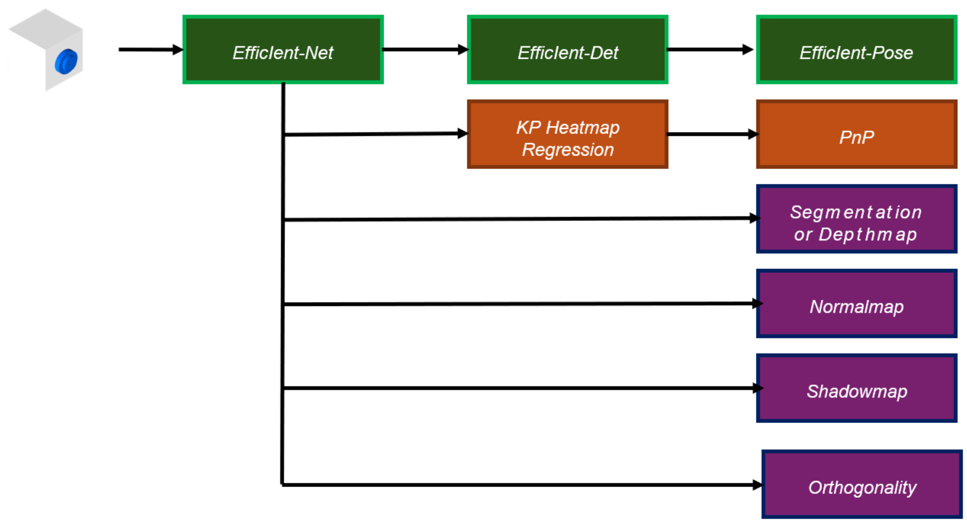


Figure 3. Architecture of the space case study

2.3 Training

The training of the satellite pose estimation multitask deep learning model was meticulously executed on a dataset comprising 60,000 images, adhering to a data split of 70% for training, 20% for validation, and 10% for testing purposes. This section details the training process, the computational infrastructure used, and the techniques applied to optimize the model's performance.

The model training was carried out on our corporate server, which is equipped with state-of-the-art hardware to facilitate the intensive computational demands of deep learning tasks. The specifications of our training infrastructure are as follows:

- CPU: AMD Ryzen Threadripper 3970X (32 cores)
- RAM: 128GB
- GPU: Nvidia RTX A6000 (48GB GDDR6)

The training was conducted using Python, as a programming language, and PyTorch, as a deep learning framework. Additionally, OpenCV was utilized for image processing tasks, supporting the implementation of advanced computer vision functionalities.
The training regimen was optimized to achieve the best possible balance among the tasks, with a primary focus on the direct and indirect pose estimation tasks. This optimization process involved careful tuning of the model's parameters to enhance its learning efficiency and accuracy in these critical areas.

To further improve the robustness of the model, online data augmentation techniques were applied during the training phase. These techniques included various transformations such as noise addiction and colour adjustments to the training images. By exposing the model to a broader spectrum of data variations, we aimed to enhance its ability to generalize from the training data to new, unseen images, thereby improving its performance in real-world satellite navigation tasks.

The training was conducted optimally, with the total training time under five days. This efficient timeframe was achieved through the combined use of our powerful computational infrastructure and the targeted optimization of training parameters.

2.4 Local testing

The evaluation of our model's performance has revealed significant insights into its capability in direct pose estimation, particularly in distinguishing between translation and rotation errors. This section presents an analysis of the results obtained, highlighting the precision and accuracy of the multitask deep learning model in satellite relative navigation tasks.

The simulated operation is in the range of 1-20 meters, with a 45° camera FOV, 512x512 pixels input image of grayscale type. The number of tested images is 6000.

METHOD	ERROR	MEAN	STD	MEDIAN	IQR
Direct Pose Estimation	TRANSLATION [m]	0.052	0.075	0.029	0.046
	ROTATION [deg]	2.650	6.958	1.930	1.584

Indirect Pose Estimation	TRANSLATION [m]	0.216	0.476	0.165	0.188
	ROTATION [deg]	3.440	8.569	1.506	1.828

Table 1. Testing results of the space case study

Pose Estimation Accuracy:

- Translation Error: The model demonstrates good performance in estimating the satellite's position, with the mean and standard deviation of translation errors being within a few centimetres. This level of accuracy is notable, especially considering the complexity of satellite relative navigation and the variability of space conditions.
- Rotation Error: For rotation estimation, the model achieves errors within the unit degree range. This precision is exceptional and indicative of the model's robustness in understanding the satellite's orientation in 3D space, even when relying on data from a single frame.

Statistical Performance Indicators used:

- Mean and Standard Deviation: These metrics for both translation and rotation errors show that the model can consistently predict the pose with a high degree of accuracy. The relatively low standard deviation indicates minimal fluctuation in performance across different test cases.
- Median and Interquartile Range: The median and interquartile range further substantiate the model's reliability. These statistics show that most predictions are accurate and clustered around a high accuracy level, with a limited number of outliers.

When comparing the direct and indirect methods of pose estimation, both approaches yield commendable results, with the direct method showing slightly better performance in terms of translation accuracy. This is particularly noteworthy since the direct method operates on raw single-frame data, without the benefit of trajectory synthesis or filtering approaches that might smooth out errors over multiple frames. The achieved results, especially the precision in translation within a few centimetres and rotation within a single degree, are highly satisfactory for the field of satellite navigation. Such accuracy levels highlight the effectiveness of the implemented approach. These outcomes are indicative of the model's potential to significantly improve the accuracy and reliability of satellite pose estimation tasks, providing a solid foundation for further improvements.

2.5 Platform testing and early porting

The initial phase of the space model optimization involved porting the deep learning model from its original PyTorch implementation to the ONNX (Open Neural Network Exchange) format. This format facilitates model sharing and deployment across different platforms and frameworks, enhancing the versatility and applicability of our AI solution.

The model's conversion from PyTorch to ONNX was executed using the standard PyTorch torch.onnx.export function. This process required careful preparation of the model and its inputs to ensure compatibility with ONNX's requirements, such as fixed input sizes for certain layers and the inclusion of all necessary metadata for a complete representation of the model in the new format.

To verify the successful conversion of the model, a statistical analysis was conducted on a set of test images (i.e. 6000). This involved comparing the outputs of the original PyTorch model with those of the ONNX-converted model, focusing on the consistency and accuracy of predictions across both formats. The comparison was based on a tensor-by-tensor analysis, examining the differences in output values to assess any discrepancies that might indicate issues with the conversion process. Moreover, and more understandable, a performance comparison result is here depicted:

METHOD	ERROR	METRIC DIFFERENCE MEAN	METRIC DIFFERENCE STD
Direct Pose Estimation	Translation [m]	2.4e-06	2.3e-05
	Rotation [deg]	8.9e-04	1.8e-03
Indirect Pose Estimation	Translation [m]	0	0
	Rotation [deg]	0	0

Table 2. Pytorch2ONNX Performance Comparison Results

The indirect method has no difference from Pytorch to ONNX, since the double steps made by key-points detection and PnP algorithm do not introduce any disturbance. On the other hand, the differences for the direct method are slight but not impactful, concerning the previously described metrics, for the scenario into consideration. For this reason, we can attest that there is no model degradation from torch to ONNX format.

After the conversion of the model, inference was executed on the platform, the NVIDIA Jetson Orin, in different tests. The platform was set to minimum power (15W), and the model was ported in a full-branches version (both direct and indirect methods, orthogonality and maps computations). Results showed an acceptable performance:

PORTED MODEL	PERFORMANCE
Full-branches	3.7 fps

Table 3. Porting performance results.

In order to provide robust functioning during operations, embedded performance improvements are desirable, and they can be gained through a careful analysis and tuning of the conversion process, without affecting the model components and thus the integration into the software environment inside the platform.

2.6 Scenario Generation

The verification test catalogue for the space case study is envisioned to be developed following the activities on the automotive and railway ones. This is due to the fact that the automotive domain has a much wider range of already available guidelines and testing scenarios, from which it is easy to build a consistent and complete catalogue, and that the railway operational scenario is easily comparable with the automotive one, allowing to smoothly derive test cases for it. The space domain, being the most complex and the least explored in terms of autonomous systems, will leverage the work done on the previous two.

Nevertheless, the preparation for the activity has already started; the aforementioned dataset generation pipeline was designed with the possibility of producing images based on trajectories and with different customizable parameters, which allows to simulate realistic operational environments. As soon as test cases will be developed, they will be enacted and verified without further effort.

2.7 Next steps

As we move forward with the development of our AI model, some key areas have been identified for further optimization and enhancement. These areas are crucial for improving the model's performance and ensuring its applicability in practical satellite navigation tasks, especially in complex operational scenarios such as docking. Below, we outline the future development directions and the steps we plan to undertake.

- WP5 development:
    - Auto-Weighting Strategy for Loss Functions: given the model's architecture, which accommodates multiple tasks with varying degrees of importance, a critical area for development is the refinement of auto-weighting strategies. These strategies involve dynamically adjusting the weight of each task's loss function during training. This optimization aims to balance the learning process more effectively, ensuring that primary tasks, such as direct and indirect pose estimation, receive appropriate emphasis compared to secondary tasks. Implementing and perfecting auto-weighting strategies will require extensive testing and could significantly enhance the model's overall accuracy and efficiency.
- Integration with WP2 (safety):
    - Trajectory Simulation for Safety Analysis: Another area for development concerns the simulation aspects, particularly the simulation of trajectories under various conditions. By examining limit conditions and specific trajectory types, in conjunction with relative pose conditions and lighting factors, we can gain deeper insights into how these variables impact pose estimation accuracy. This investigation is not only critical for improving the model's performance but also for ensuring the safety and reliability of space docking operations, where precise trajectory prediction and adjustment are paramount. The test cases catalogue that will be developed together with the safety experts will be enacted by actual scenarios of trajectories and environmental conditions to evaluate the model performance and safety level.
- Integration with WP3 (XAI):
    - Application and Testing of Explainability techniques: While WP5 is providing the case studies, WP3 is releasing tools and techniques for explaining and verifying datasets and models. In the next phase of the project, XAI techniques will be applied to the space model, identifying the best approaches for enabling and validating safe executions of the algorithm.
- Integration with WP4 (platform):
    - Complete the Platform Porting: Another step in our development roadmap involves completing the porting of our model to the envisaged platform. The porting process will involve optimizing the model for the target hardware and software environment, ensuring that it can operate efficiently and reliably.
    - Integrate the Case Study into the project software stack: The algorithm will be prepared for integration as a ROS2 application, modelling core

components as nodes communicating with each other; this will allow to integrate the model with the SAFEXPLAIN API and access to the features implemented in WP4 on the Orin platform. This overall effort will ensure that our model can be seamlessly integrated with other work packages work within the current project, facilitating and enabling all the developed platform features.

3. Automotive Case Study

The detection system's primary objective is to prevent collisions with all road users, including vehicles and vulnerable users, by identifying them as relevant to a collision. The automotive use-case is of higher complexity as the system operates within a challenging environment characterized by (1) [4] a large field of view, dynamic surroundings, numerous road users, road markings, and traffic signs. (2) Multiple AI modules and the need for a consistent coordinate system. (2) different domains requiring domain adaptation.

Using a case requires detailed scene understanding, identifying different objects and signs in the scene, finding collision-relevant objects, estimating distances and thresholds, and deciding on the appropriate action. The system architecture comprises a Perception Module responsible for data processing and AI model execution, along with a Plan Module tasked with identifying collision-relevant objects, ensuring uniform coordinate systems, and integrating vehicle dynamics. By effectively executing these modules, the system aims to safeguard vulnerable road users while navigating complex and dynamic traffic environments.

3.1 Datasets collection and generation

Multiple Datasets were considered for automotive use cases as there are different models with different criteria.

Dataset	Description	Classes	Env
VOC	- 20 Classes 16551 training 4952 test images	person, bird, cat, cow, dog, horse, sheep, airplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, sofa, potted plant, TV	NA
COCO	- 80 classes 118287 training 5000 test images	person, vehicle, animal, indoor, outdoor, appliances, electronics, furniture, food, kitchen, sports, accessory	everyday scenes
BDD	BDD100K	person, rider, car, truck, bus, train, motorcycle, bike, traffic light, traffic sign	day-time, night-time
Mapillary	Large-scale street-level image dataset - 25,000 high-resolution images	annotated into 66/124 object categories	various weather, season and daytime.
CityScapes	- 30 classes	Human, vehicle, road, construction, nature, sky, object, void	Daytime, Several seasons
NuScenes	1000 scenes of 20 secs each,	car, bus, bike, truck, people and others	rain: 19.4%, night: 11.6%
Oxford RobotCar	No detection/ segmentation GT	Pedestrians, riders, vehicles (car, bus)	dusk, night, night+rain,

dataset			overcast(summer+winter), snow, sun
Euro City Person	4 seasons, 12 countries, 31 cities, 47.300 images, 238.200 persons	Pedestrians and Riders (bicycles, moped, bikes)	Day/ Night, all four seasons
KITTI	mobile robotics and autonomous driving - no GT for semantic segmentation	building, sky, road, vegetation, sidewalk, car, pedestrian, cyclist, sign/pole, and fence	
CU Lane	Large-scale challenging dataset for academic research on traffic lane detection - 88880 for training set,	Lanes	mounted cameras on six different vehicles - urban, rural, and highway scenes.

Table 4. List of datasets considered during data stubbing

We explored many datasets during the stubbing process and evaluated them on various criteria: the availability, the different classes, recording conditions.

We also collected data for testing and explored synthetic and simulated data.

- Detection:
    - COCO
    - BDD
- Segmentation
    - Mapillary
    - CU Lane

3.1.1 Simulator

To test the automotive-use case collision avoidance system, we explored the use of simulators. Using simulators for testing collision avoidance systems in automotive applications is a common practice due to its safety, cost-effectiveness, reproducibility, scalability, and rapid prototyping benefits. Simulators provide a controlled environment for engineers to explore various scenarios without risking real vehicles or lives, allowing for precise replication of conditions and rapid iteration of system improvements.CARLA (an open-source simulator for autonomous driving research) simulator [11] has a huge and detailed eco system and multiple components that requires a deep dive understanding. We explored the Documentation, the tutorial videos, the examples to utilize the software effectively. Some simulation visuals relevant to the use case are shown in Figure 4.

Figure 4. Examples of CARLA simulator

3.1.2 Selection of Models

Detection. We explored multiple detection networks (object detectors) and backbone (encoder) networks to choose the most efficient and effective (Table 5).

Detection Head	Backbone	Dataset
ThunderNet	ShuffleNet-v2; EfficientNet-B0	VOC; COCO
YOLO; SSD	MobileNet-v2; DeiT-T	Corrupted COCO
DETR	DarkNet-19; ResNet-18	BDD; Cityscapes
CenterNet; TTFNet; FCOS; NanoDet	Xception; HarDNet-68; VoVNet	Kvasir-SEG

Table 5. All the models and datasets considered

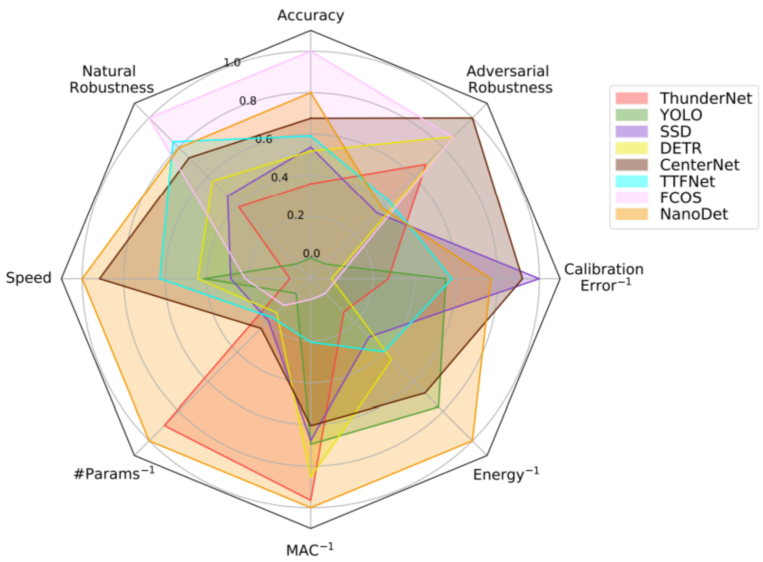


Figure 5. Overall summary of all the detection networks against various metrics

The analyses are summarized in Figure 5, where each vertex corresponds to a metric, and the eight different colors represent different detectors. We report eight metrics, namely accuracy, robustness to natural and adversarial corruptions, speed, number of parameters, MAC (Multiply-Accumulate operations) count, energy consumption, and calibration error (which measures reliability). The plot is represented such that the ideal network should occupy the entire octagon. Such a network would have the highest accuracy, robustness, and speed, the lowest number of parameters and MAC count, while consuming the lowest energy and being the best calibrated.

NanoDet: NanoDet is an innovative lightweight detector that doesn't rely on anchors for detection. It leverages Adaptive Training Sample Selection (ATSS) to automatically pick positive and negative training samples based on object characteristics. Using a Generalized Focal Loss (GFL) for classification and regression, NanoDet optimizes performance by extending Focal Loss into the continuous domain. It also employs Generalized IoU loss (GIoU) to handle non-overlapping cases effectively. With a Feature Pyramid Network (FPN) utilizing three feature maps, NanoDet enhances lower-level features via Path Aggregation Network (PAN) blocks, which include bottom-up path augmentation. These PAN blocks feed into individual detection heads for computing classification labels and bounding boxes, optimizing object detection accuracy.

YOLOX - YOLOX, introduces significant advancements to the YOLO series with features such as an anchor-free approach, decoupled detection head, and advanced label assignment strategy SimOTA. This architecture enhances efficiency by reducing parameters and **GFLOPs**. It is built upon YOLOv3 and has multiple versions – small, medium, large and still has computation demands. Hence, we consider multiple detection networks to test on the hardware.

Segmentation. Segmentation is a computer vision task that involves partitioning an image into multiple segments or regions based on certain characteristics. We explore different segmentors for better scene understanding, road markings and lane detection.

RGPNet is a real-time semantic segmentation network designed for complex scenarios, featuring an asymmetric encoder-decoder structure with an innovative adaptor module. This module captures multiple levels of abstraction to refine segment boundaries and facilitates better gradient flow through short-paths. Results are shown in Figure 6.

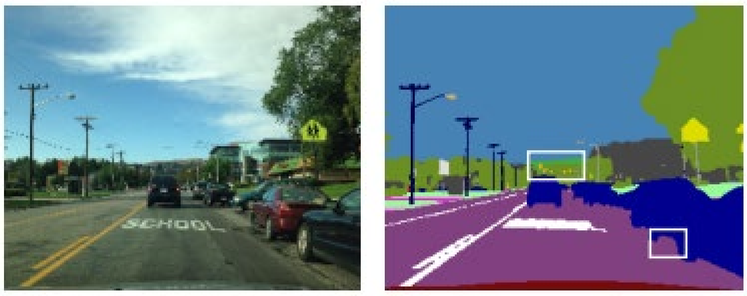


Figure 6. Examples of Semantic Segmentation from RGPNet segmentation network

CLRerNet: CLRerNet is a novel lane detection method that significantly improves upon existing approaches by introducing LaneIoU, a novel intersection-over-union metric that considers local lane angles, enhancing accuracy and reliability in lane detection systems. By leveraging dynamic sample assignment and replacing traditional IoU loss with LaneIoU for regression of horizontal coordinates, CLRerNet ensures effective learning of confidence scores and appropriate penalization of predicted lanes at different tilt angles. Few examples are shown in Figure 7.

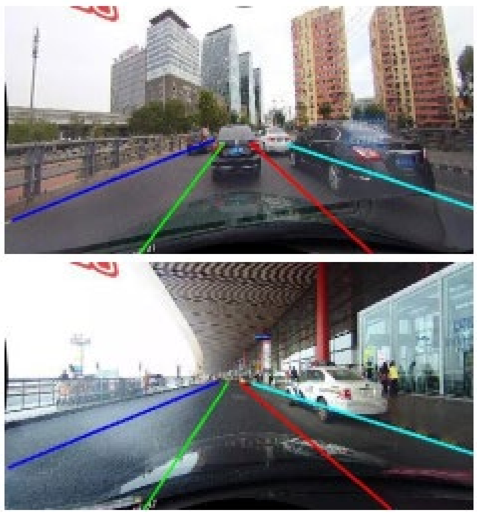


Figure 7. Examples of lane detection using CLReRNet

3.2 Architecture

The SAFEXPLAIN automotive case study architecture (Figure 8) comprises two primary modules: the AI Perception Module and the Plan Module.

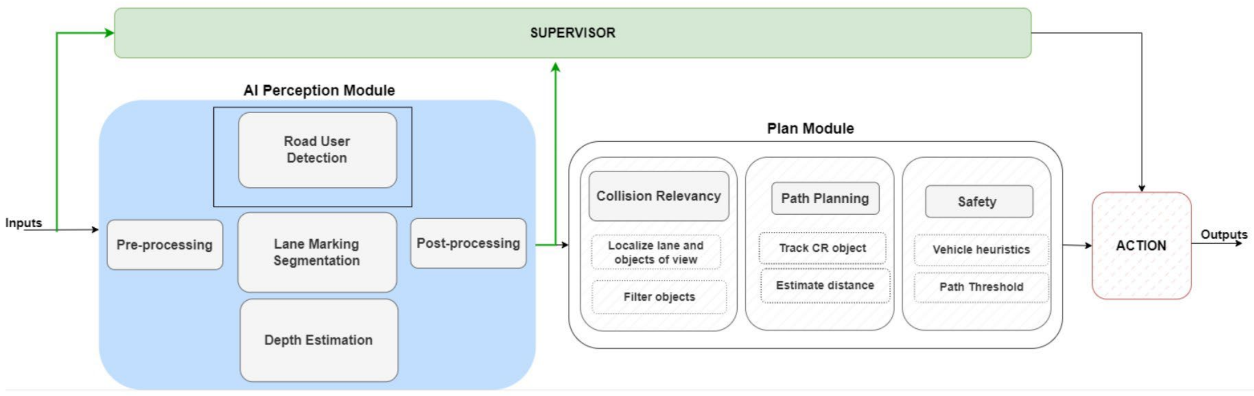


Figure 8. Architecture of Automotive Use case

The Perception Module encompasses various AI models and data processing operations, starting with data processing, pre-processing, and post-processing. Initially, the data undergoes preparation, including transformation and standardization, to ensure compatibility with AI models. A detection model then identifies road users, such as pedestrians, cars, vehicles, and cyclists, followed by the segmentation model, which comprehensively segments elements like roads, lanes, sidewalks, and structures to understand the scene better. Finally, a depth system assesses the distance to objects within the scene for enhanced spatial awareness.

Conversely, the Plan Module receives outputs from the Perception Module and focuses on identifying collision-relevant objects. It integrates outputs from all AI models, aligning them in a common coordinate space, whether pixel-based or real-world. This involves identifying lanes and the objects within them, tracking their movements, and estimating distances. Additionally, the module incorporates vehicle dynamic heuristics to establish thresholds for distance and time to collision. These thresholds are critical parameters used to trigger actions such as braking or issuing warnings before a potential collision occurs, enhancing overall safety and collision avoidance strategies.

Figure 9 shows the detailed architecture implementation with all the components, including the coordinate systems, the pre and post processing and planning modules.

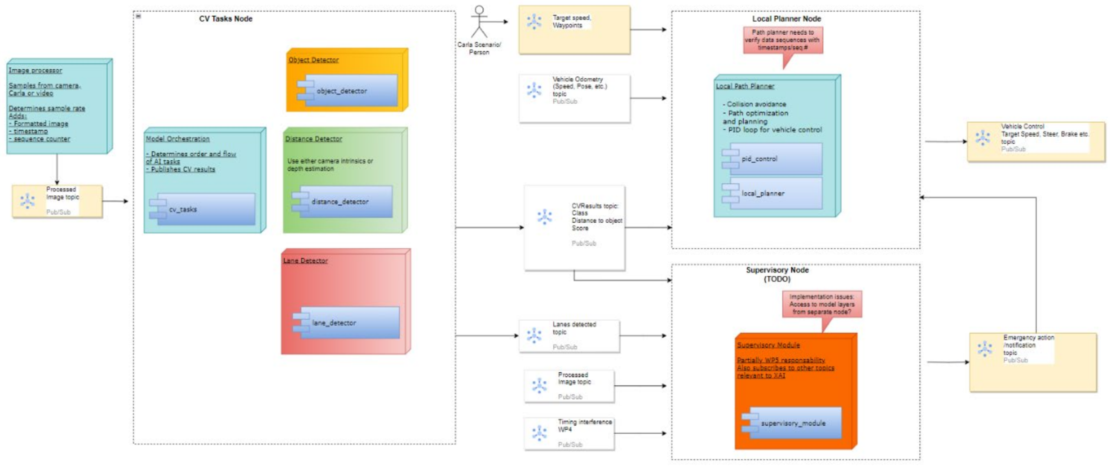


Image processor
Samples from camera, Carla or video
Determines sample rate
Adds:
- Formatted image
- timestamp
- sequence counter
Processed Image topic
Pub/Sub
CV Tasks Node
Object Detector
object_detector
Model Orchestration
- Determines order and flow of AI tasks
- Publishes CV results
cv_tasks
Distance Detector
Use either camera intrinsics or depth estimation
distance_detector
Lane Detector
lane_detector
Carla Scenario/ Person
Target speed, Waypoints
Vehicle Odometry (Speed, Pose, etc.) topic
Pub/Sub
CVResults topic: Class Distance to object Score
Pub/Sub
Lanes detected topic
Pub/Sub
Processed Image topic
Pub/Sub
Timing interference WP4
Pub/Sub
Local Planner Node
Path planner needs to verify data sequences with timestamps/seq #
Local Path Planner
- Collision avoidance
- Path optimization and planning
- PID loop for vehicle control
pid_control
local_planner
Supervisory Node (TODO)
Implementation issues: Access to model layers from separate node?
Supervisory Module
Partially WP5 responsability
Also subscribes to other topics relevant to XAI
supervisory_module
Vehicle Control Target Speed, Steer, Brake etc. topic
Pub/Sub
Emergency action /notification topic
Pub/Sub

Figure 9. Detailed architecture of automotive use case

3.3 Training

Table 6 contains the shortlisted models. In this section, we delve into the training specifics of these models.

- Detection - The Nanodet-Hardnet is trained on BDD training images. The initial learning rate (LR) is decayed by a factor of 0.1 and we use a common batch size of 32 and confidence threshold of 0.01 for all the experiments. Number of iterations is 240K with optimizer.
- Segmentation – CLRerNet is trained on CULane dataset. It is trained with ADAM optimizer with initial learning rate of 0.0006 with cosine decay.
- Depth – Depth model is a transformer network trained in a self-supervised way to perform monocular depth estimation with unknown camera intrinsics. However, please note that this is only used (as a backup) if camera intrinsics are not available. If camera intrinsics are available, we directly use this to project our distance from pixel to real world coordinates.

Final models-datasets		
Detection	NanoDet- [6] YOLO-X [12]	COCO [4] BDD [5]
Segmentation	CLRerNet – DLANet [7]	CULane [8]
Depth	- Transformer [9] - Using Camera Intrinsics if available	Kitti [10]

Table 6. Shortlisted modules with deep neural networks and datasets

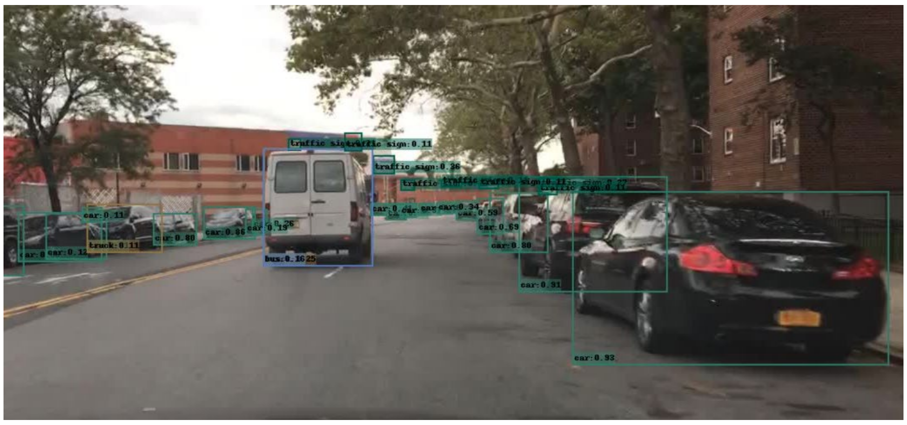
3.4 Local testing

We assess the performance of different models using desktop GPUs and analyse the results through performance evaluation and visualization.

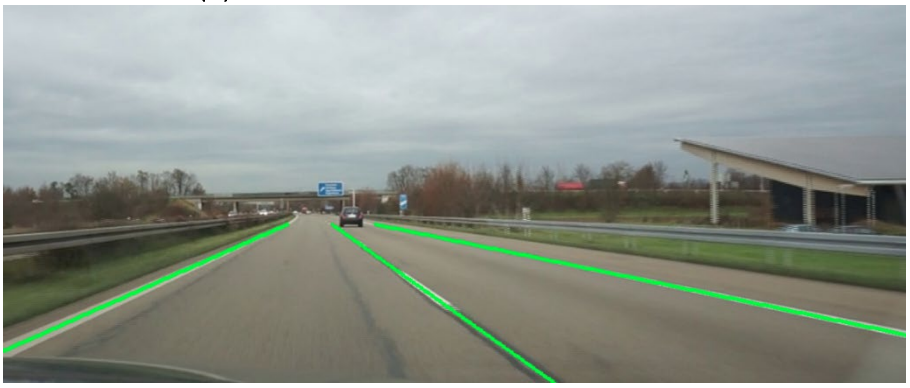
The performance of models in terms of accuracy and speed both on the desktop GPU are presented in Table 7. Further, visualizations of the results on real videos and simulated scenarios and presented in Figure 10.

Task	Accuracy (mAP)	Speed (FPS) on GeForce RTX 3090
Detection	mAP: COCO - 33.61 BDD - 30.09	45
Lane Segmentation	F1 Score: CULane - 81.12	12

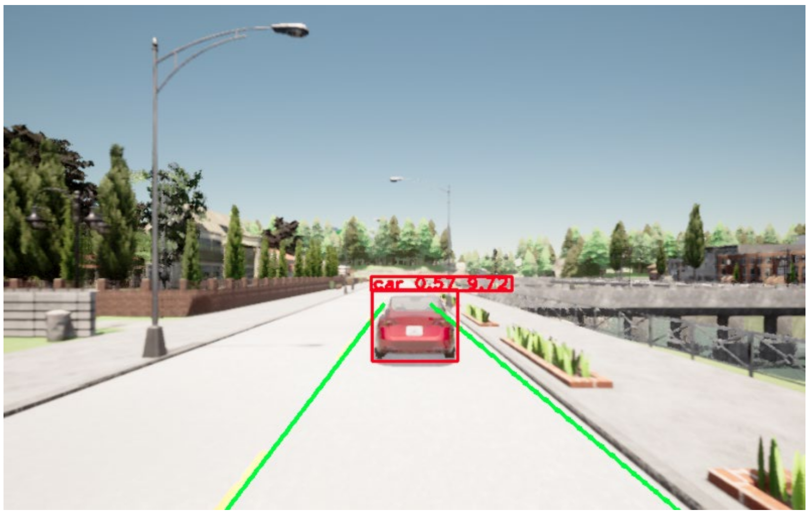
Table 7. Performance on GeForce RTX 3090 desktop GPU



(a) Detection of road users and vehicles



(b) Segmentation of lanes in the road



(c) Detection of objects and segmentation of lanes in synthetic data

Figure 10. Detection (a) and lane segmentation (b) on real images and simulated image (c)

3.5 Platform testing and early porting

Moreover, our deployment involves porting the developed models onto a dedicated hardware platform tailored for the project's requirements. The system architecture, as depicted in Figure 11, integrates components including the CARLA Simulator, ROS2 agent, Safexplain Middleware (SMW), and the automotive use case agent. This setup facilitates interaction with the simulator to facilitate testing across various safety scenarios. The performances of these components on the project hardware platform are meticulously documented in Table 8.

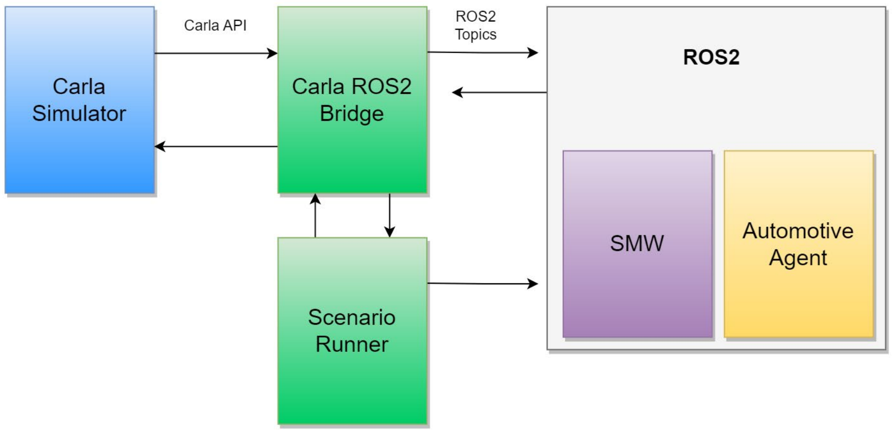


Figure 11. System architecture with CARLA Simulator, ROS agent, Safexplain Middleware (SMW) and the automotive use case agent

The lane segmentation network is slower than the detector, owing to the heavy backbone and the feature pyramid networks. Notably, the deployment utilizes the trained PyTorch models without any optimization. However, our forthcoming strategy entails optimizing the models using ONNX to achieve enhanced performance. This optimization process will be guided by meta-information extracted from the models, which will be provided to the supervisor architecture for seamless integration and performance enhancement.

Task	Speed (FPS) on Nvidia ORIN
Detection	26
Lane Segmentation	6.6

Table 8. Performance on NVIDIA ORIN device

3.6 Scenarios generation

To effectively and efficiently test the automotive case, we rely on a standard catalogue for applying V&V (Verification and Validation) strategies. Using this, we have constructed a catalogue comprising scenarios that encompass the potential situations and events the automated driving system might face.

This is discussed in detail with figures and parameters below.

Goal: The detection system's goal is to avoid collisions with all road users (vehicle and vulnerable users) identified as collision relevant.

Automotive AI module Challenges:

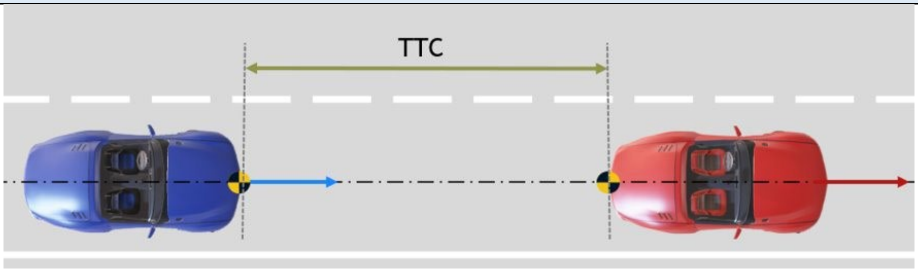
- Large field of view, Dynamic surroundings, more road users, road markings, traffic signs
- Consistent Coordinate systems – Pixel to real-world for all 3 tasks (approximate calibration)
- Domain adaptation

Settings:

- Urban scenarios with moderate traffic conditions + Few cases on highway
- Road user is at least 30% visible, exposure time is more 3s and the size is more than 80cm.
- Assume heuristics (speed, deceleration)

Safety Scenario 1: DS1 from the Catalogue:

Driving following a target vehicle on highway

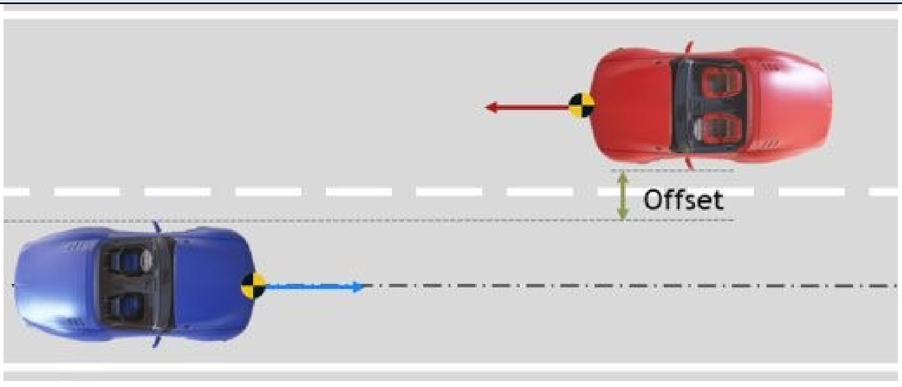


- The Ego vehicle drives with a longitudinal acceleration lower than 2m/s2 towards a moving target vehicle and is at a distance corresponding to a Time To Collision (TTC) of at least 4 s.

- The Ego vehicle speed range is [50 km/h, 130 km/h]
  The target vehicle drive at 80 km/h
  The following environmental conditions shall be present:
  - Dry and daylight with minimum 1000 lux and Sun angle >15° to horizon
  - Dry and night with maximum 10 lux
- Road surface is asphalt or concrete
- The following Pre-conditions shall be respected:
  - both vehicles shall keep steady speed and path and path
  - steering angle shall be lower than the override threshold
  - yaw rate shall be lower than the override threshold

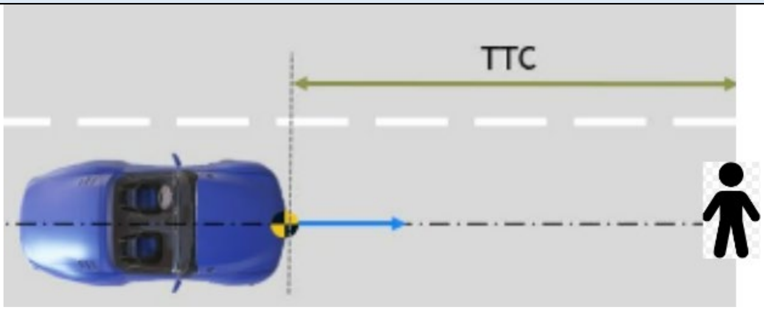
Safety Scenario 2: DS6 from the Catalogue:

Driving with a target vehicle coming from opposite direction



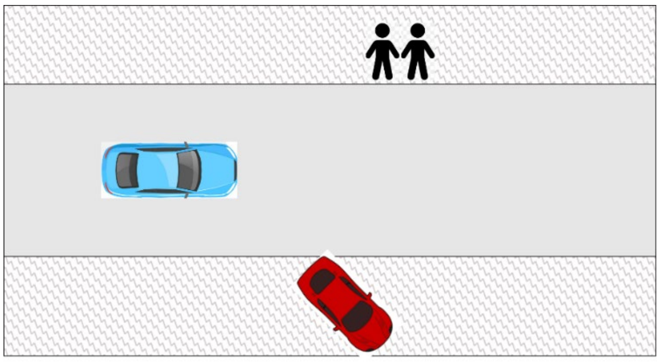
- The Ego vehicle drives with a longitudinal acceleration lower than 2m/s2 towards a moving target vehicle and is at a distance corresponding to a Time To Collision (TTC) of at least 4 s.
- The Ego vehicle speed range is [50 km/h, 130 km/h]
- The target vehicle drive from 10 to 30 km/h
- The offset between the vehicles is 1,5 m
- Road surface is asphalt or concrete
- The following environmental conditions shall be present:
  - Dry and daylight with minimum 1000 lux and Sun angle >15° to horizon
  - Dry and night with maximum 10 lux
- The following Pre-conditions shall be respected:
  - both vehicles shall keep steady speed and path
  - steering angle shall be lower than the override threshold
  - yaw rate shall be lower than the override threshold

Safety Scenario 3: Same as DS1 but with Pedestrian on the road instead of car



- The Ego vehicle drives with a longitudinal acceleration lower than 2m/s2 towards a static pedestrian and is at a distance corresponding to a Time To Collision (TTC) of at least 4 s.
- The Ego vehicle speed range is [50 km/h, 130 km/h]
  Road surface is asphalt or concrete
- The following environmental conditions shall be present:
  - Dry and daylight with minimum 1000 lux and Sun angle >15° to horizon
  - Dry and night with maximum 10 lux

Safety Scenario 4: Same as DS1 but with no one on the road but with parked car and pedestrians on sidewalk (to test the basic case of not doing any action)



- The Ego vehicle drives with a longitudinal acceleration lower than 2m/s2 on a empty road
- The Ego vehicle speed range is [50 km/h, 130 km/h]
  Road surface is asphalt or concrete
- The angular offset of parked car is 45 degrees
- The following environmental conditions shall be present:
  - Dry and daylight with minimum 1000 lux and Sun angle >15° to horizon
  - Dry and night with maximum 10 lux

3.7 Next steps

We enumerate the next steps below:

- WP5 - Enhance Model Performance: Through extensive testing on various scenarios and datasets, optimize the models to improve both accuracy and speed.
- WP5 - Enhance Plan Module: Refine the Plan module for better and accurate post-processing through extensive testing.
- WP2- safety architecture integration is prioritized, emphasizing the incorporation of safety protocols into the process pipeline. This integration guarantees compliance with safety standards and ensures the project's overall security and reliability.
- WP3 - Supervisor Integration from WP3 for both data and model explanations. Integrate supervisor modules tailored for the automotive use case. As there are multiple models at play in the use-case, explainability of these models must be explored and integrated. The supervisor aids the decisions of the deep neural networks by providing additional reliability check.
- WP4 - The integration of the SAFEXPLAIN API involves incorporating the developed API into the project framework. Integrate and test the perception module (deep learning components), the planning module, the ROS agents and the Carla simulator. Finally integrate with hardware components for comprehensive testing and validation.

4. Railway Case Study

This CS is pursuing the development of a safety functions that will minimize the risk associated with Automatic Train Operation. The scenarios developed will offer approaches for minimizing the risk of a train running over or injuring people on the track as well as avoiding damages during opening/closing operations on the platform. Four activities are underway to support these scenarios, training dataset generation, track detection techniques research, obstacles detection techniques research, stereo distance calculation.

4.1 Datasets collection and generation

Dataset collection for Railway is a challenging task that requires a lot of resources and expertise. Unlike other domains, such as automotive or robotics, there is not much data available for railway applications, and the generation of real data is costly and time-consuming. Moreover, most of the existing datasets are proprietary and not accessible to the research community.

Two ways of collecting or creating a dataset. One of the would be to collect or fin real data and the other one would be to create synthetic datasets.

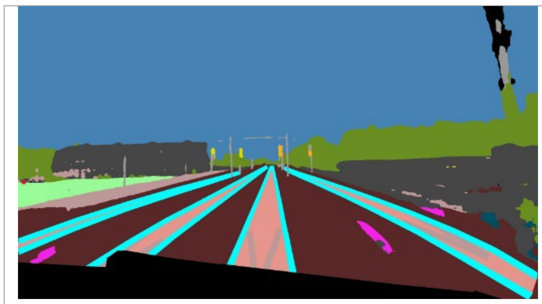
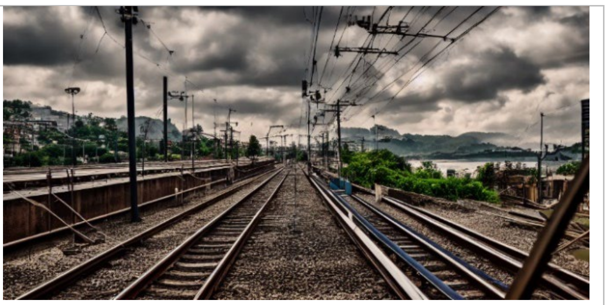
As previously said, creating a dataset from real scenarios would be hard. The process of capturing railway frames is expensive in terms of time and resources, as many parties take part in operations. The track and the train need to be allocated, with the bureaucratic obstacles that this entails. A driver to drive the train and a person to take frames are needed in order to record the runs. Therefore, just for the obtention of the images, train and track allocation time and cost, the time and cost of the driver, and the time and cost of the recorder need to be considered.

On the other side, one of the few exceptions is RailSem19, an open-source dataset of images for railway scenes. However, RailSem19 is limited in its scope and diversity, and does not cover all the aspects of railway analysis. In any case, it is a great opportunity for this CS to make an effort to adapt this dataset and attempt to focus on the options that RailSem19 gives.

It would be also an option to create synthetic dataset. Generating synthetic dataset even with a simulator or an AI model would be good way in terms of training a model for its robustness. This kind of dataset has good and bad sides.

The bad side of using this kind of dataset would be that there could be a gap between the real and the synthetic data. But the great side would be that, for example, there is no limitation in generating any kind situation or that weather and lightning conditions could be changed easily. That gives the opportunity to generate situations that may not happen in real scenarios, for example having a car on a railway or having different kind of snowy days.

In this CS we present two datasets to train semantic segmentation models for rail scenes. The first dataset is generated with a train simulator that renders realistic images of trains, tracks, signals, and other objects in various weather and lighting conditions with an additional post-processing step that applies a stable diffusion algorithm to smooth the boundaries, reduce the noise and give a realistic tone. The second dataset is RailSem19, a real-world dataset that contains 19 classes of rail objects annotated on high-resolution images captured by a camera mounted on a train. See Figure 12 and Figure 13 showing two of the options.

 Figure 12: RailSem19 dataset image	 Figure 13: TrainSimulator + StableDiffusion

4.2 Architecture

The main goal is to create a simple architecture to obtain a good performance, for that it needs to be simple. This means that the design should avoid unnecessary complexity and focus on clarity, cohesion, and modularity. A simple architecture is easier to understand, maintain, and extend, and it reduces the risk of errors and bugs. A simple architecture also enables faster development and testing, as well as better scalability and reliability. Therefore, simplicity is not only a desirable quality, but also a strategic advantage for the CS. In this CS the main architecture is separated in other modules as shown in Figure 14.

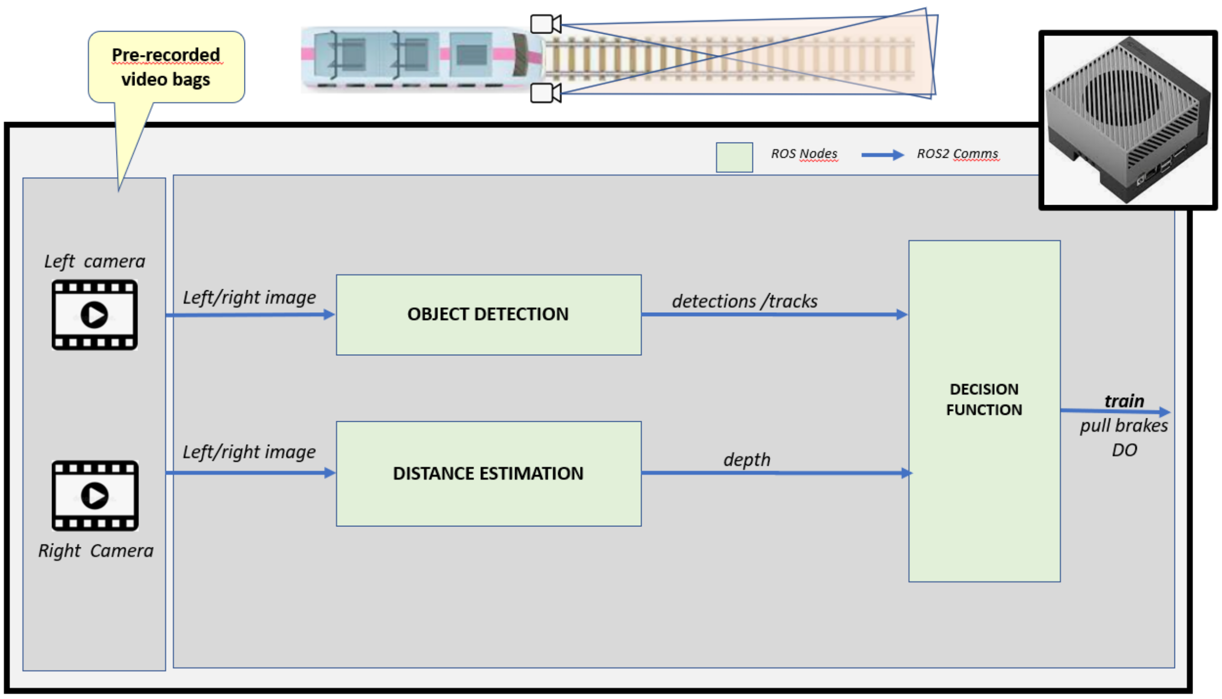


Figure 14: Railway Case Study Architecture

In this CS, as the simulator is not an online simulator, this means that it has not the possibility to send the images automatically, images are stored and used when needed. That is why in the left of the image there is a message of pre-recorded video bags.

In the main functions, we can differentiate three parts that are going to be running in different threads. These are the specifications of each one:

- Distance estimation: It takes the two images given by the cameras, the images of the right and left camera, and returns a depth image containing the distances in each pixel.
- Object detection: It takes the two images given by the cameras, the images of the right and left camera, and returns a list of objects and its bounding boxes.
- Decision function: It takes both distance estimation and object detection outputs and decides whether it is a risk situation or not. Depending on the situation it sends a message or pull the brakes.

4.3 Training

Depending on the techniques used, the training and implementation activities differ from each other. Object detection, Instance segmentation, semantic segmentation and stereo distance estimation are the ones used in this CS.

The models used in this CS have different licenses but every single one is "open source", considering that some of them are not available for commercial use.

Each of the models or techniques is trained and configured differently so each of them will be explained separately.

- Object detection and Instance segmentation: The models analysed for this purpose in the CS are different versions of YOLO. The training has been done with a RailSem19 dataset for object detection mixed with COCO dataset to have more classes of objects that RailSem19 has. See Figure 15.

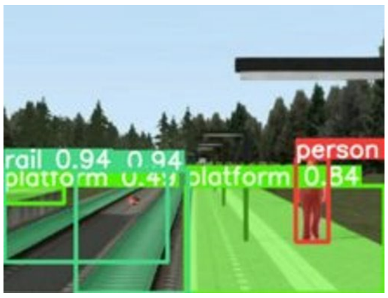


Figure 15: Object detection + instance segmentation

- Semantic segmentation: The models analysed are different versions of Segment Anything and SegFormer. These models were trained with RailSem19 dataset prepared for semantic segmentation. See Figure 16.

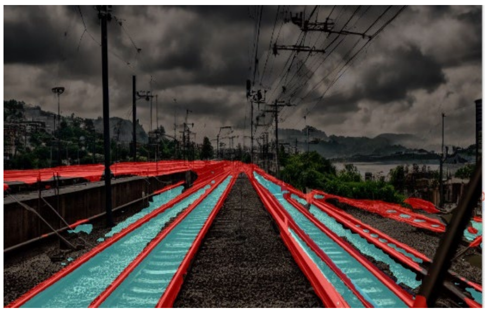


Figure 16: Sematic segmentation for rail detection

- Distance estimation: The distance estimation is implemented using different libraries. There tests had to be done to choose the best parameters to ensure the proper operation. See Figure 17.

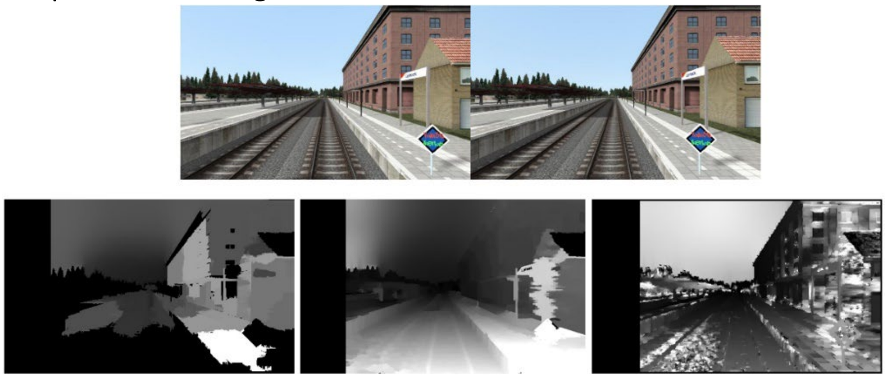


Figure 17: Depth estimation example

In summary, the four distinct computer vision tasks involve various deep learning architectures and datasets designed to address unique challenges related to the CS.

4.4 Local testing

Regarding the testing process for each component of the CS, a workstation is used. This involves all component such as object detection, instance segmentation, semantic segmentation, and stereo distance estimation. Testing results are shown below:

- Object Detection: YOLOv7 and YOLOv8n were trained for RailSem19 and COCO datasets mix. The results obtained were quite good.

	Precision	Recall	mAP0.5	mAP0.95
YOLOv7	0.69201	0.54102	0.60559	0.48221
YOLOv8n	0.69414	0.41693	0.44956	0.31434

Table 94: Comparing YOLOv7 and YOLOv8n

- Semantic Segmentation: These models are used for Dataset Generation with Stable Diffusion and for testing rail segmentation system. The results obtained were quite good, but the models used were big and consume high GPU resources.

	mean_accuracy	IoU	overall_accuracy
Segformer	0.6509	0.50299	0.8566

Table 105: Segformer training results

- Instance Segmentation: YOLOv7 and YOLOv8n were trained for RailSem19 and COCO datasets mix. The results obtained were quite good.

	Precision	Recall	mAP0.5	mAP0.95
YOLOv7-seg	0.38856	0.23832	0.036007	0.0246
YOLOv8n-seg	0.58946	0.40033	0.42723	0.26248

Table 116: Comparing YOLOv7-seg and YOLOv8n-seg

- Distance Estimation Testing: Post parameter optimization during development, comprehensive testing ensued for the stereo distance estimation component. Leveraging suitable test scenarios and reference measurements, gauged precision levels attained by generated depth maps or point clouds. Evaluation criteria comprised disparity error, absolute difference, percentage of correct predictions at varying thresholds, and overall computational efficiency.

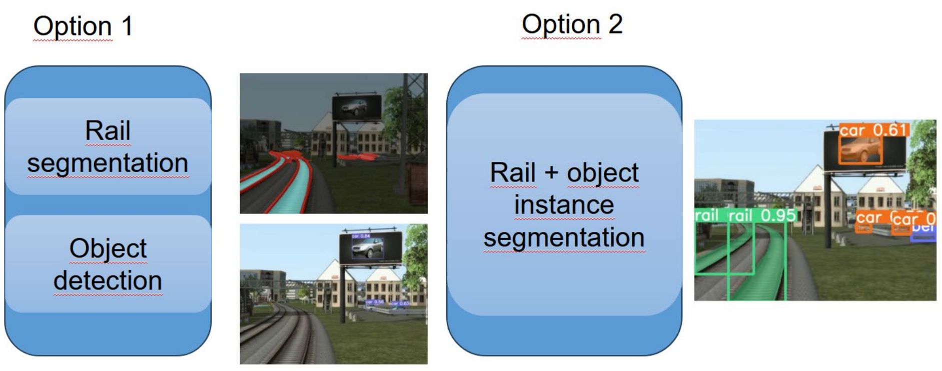


Figure 48: Segmentation or detection options

Overall, after some tests we considered two options to do the rail segmentation and object detection, see Figure 18. Option 1 seems to be more time consuming than the option 2, also it consumes more computing resources.

4.5 Platform testing and early porting

Similar tests were done in the platform. Mostly measuring execution times and CPU and GPU usages.

- Object Detection and Instance Segmentation Testing:

	Detection time	Segmentation time
YOLOv7	40ms	130ms
YOLOv8	25ms	50ms

Table 127: Detection and segmentation inference times

- Distance Estimation Testing:

	Inference time
Distance estimation	10ms

Table 138: Distance estimation inference time

Regarding the porting, there was a need to install all the dependencies. Everything was ported then to ROS2 to have an easier porting to the SAFEXPLAIN API, which makes use of ROS2 in the background. As seen in the architecture of Figure 14, each of the block is a ROS2 node and it receives and sends topics defined as rows.

Scenario generation

We have created a document that describes the Operational Design Domain (ODD), which is the set of conditions and scenarios under which an automated driving system can operate safely and reliably. We have also developed a catalogue of scenarios that cover the possible situations and events that the automated driving system may encounter in the ODD, see Table 14 and Figure 19.

Operational Scenario 2	
With the conditions specified, the following operational scenario is described: A stopped object is detected, classified as a car, which is situated on the side of the track. Another object is detected too. It is classified as a pedestrian and situated on the tracks in front of the train. Train is moving at constant 30 km/h speed. If the detected objects are not positioned on the tracks, will not activate any warning or braking while they are on one side. If the objects are positioned on the tracks, it will be analyzed if they are classified as critical or not. In case they are critical and is detected in the warning zone distance, that is to say, the distance the pedestrian is positioned is less than the warning distance threshold, a warning to the DMI will be displayed (in our case, an activation of an output signal). If the distance is higher, no action will be taken. If the estimated distance is between warning and breaking thresholds, the corresponding digital output signals will be activated. And in case the estimated distance is less than the breaking threshold, the corresponding digital output will be activated.	
Scenario Conditions:	
Scenery	
Maximum Speed Limit	60 km/h
Countryside	Yes
Multiple tracks	Yes
Distance threshold (warning)	[1001, 1500] m
Distance threshold (warning & reduce)	[701, 1000] m

Distance threshold (breaking activation)	700 m
Environmental Conditions	
Cloudy day	Yes
Daylight	[400, 1400] lm
Dynamic elements	
Person	Pedestrian at 5 km/h
Vehicle	Car at 60 km/h

Table 149: Scenario catalogue example

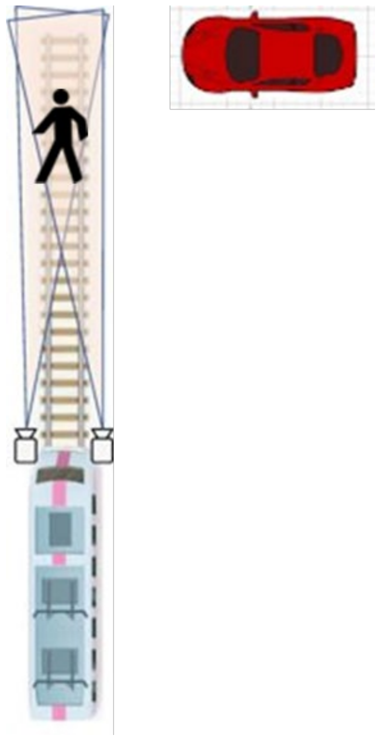


Figure 59: Example scenario

4.6 Next steps

The next steps of the project entail a integration of the previously worked WPs - WP2, WP3 and WP4. These vital steps include the integration of the SAFEXPLAIN API, performance testing and improvement, supervisor integration, and safety architecture integration.

- WP4: SAFEXPLAIN API integration is the process of integrating the API developed in the project. This is designed to leverage the SAFEXPLAIN API's capabilities to augment the safety and explainability of AI-based critical embedded systems.
- WP3: Supervisor integration is the process of integrating a supervisor into the CS. This mechanism will allow the supervisor to monitor the project and ensure that it is running correctly. The supervisor will also be able to evaluate the AI results.

- WP2: Safety architecture integration is the process of integrating the safety architecture into the project. This will ensure that the project meets the safety requirements and is designed to be safe and secure.

These integrations collectively form the essence of D5.2.

Furthermore, moving towards the D5.3, Performance testing and improvement will be addressed. This is the process of testing and improving the performance of the project. This will ensure that the project meets the performance requirements and is optimized for speed and efficiency.

5. Toy model

In the developmental stages of our project, a toy model was created to demonstrate the potential of deep learning for object detection within images. This simplified model, designed for demonstrative purposes rather than operational robustness, was pivotal in validating our approach to AI-based object detection and facilitating early integration tests with the SAFEXPLAIN's platform.

5.1 Dataset Generation

The dataset for this toy model comprised 10,000 images of satellites, each annotated with bounding boxes to identify satellite targets. The generation methodology mirrored that of the space use case (i.e., Satellite Pose Estimation), utilizing the same simulation environment to create varied scenarios, stripped of their most sensitive elements in order to share the asset without IP issues. These scenarios included different satellite distances and configurations and diverse backgrounds featuring Earth and various lighting conditions. This approach ensured that the toy model could be trained on a dataset that, while simplified, still reflected the complexity and variability of a specific environment (i.e., space).

5.2 Model Architecture

The toy model was trained using the SSD_Lite_MobileNetV3_Large model, a variant of the Single Shot MultiBox Detector (SSD) optimized for mobile devices, offering a balance between speed and accuracy. This architecture is particularly suited for edge computing scenarios where computational resources are limited. The choice of MobileNetV3 Large, known for its efficiency and effectiveness in object detection tasks, was strategic, allowing for rapid training and integration within our platform.

SSD Lite MobileNetV3 Large employs lightweight depth-wise separable convolutions and incorporates features like Squeeze-and-Excitation blocks, which enhance model efficiency without compromising performance. This design is optimized for real-time applications, making it an ideal choice for the toy model.

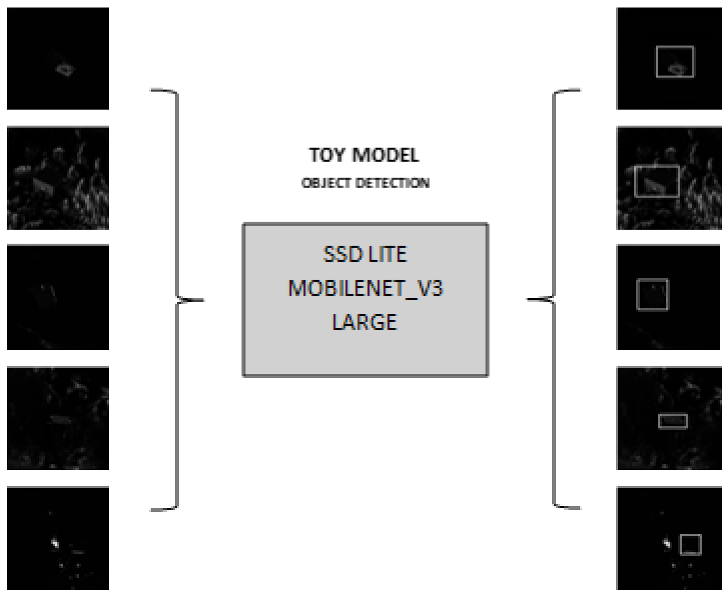


Figure 20. Toy model architecture

5.3 Training

Training was executed in a PyTorch environment, leveraging its dynamic graph computations and extensive library support to streamline the development process. Python served as the programming language, facilitating rapid development cycles and easy integration with the data generation pipeline. The training process was completed within a reasonable timeframe, demonstrating the model's capacity to learn from the generated dataset effectively. Post-training, the toy model was fully integrated into the platform, enabling preliminary testing of AI tasks common to our use cases, such as object detection within images.

It's important to emphasize that the toy model was not intended to meet the operational robustness required for space missions. Its primary purpose was to serve as a proof of concept, illustrating the feasibility of employing deep learning for AI-based object detection tasks within our technological framework. This early demonstration was crucial for aligning technological specifications and ensuring seamless integration with the broader project platform.

6. Conclusion

In this first deliverable from Work Package 5, the activities of the first phase of the project are reported, showing the work done to meet the commitment on Milestone 2 (MS2) – namely, providing case studies ready to be integrated with other work packages outputs.

The industrial partners providing the case studies have developed significant AI models for applications in critical systems employing Artificial Intelligence to reach autonomy, with a good variety of approaches and from different industrial domains, in order to provide a wide benchmark for building and verifying SAFEXPLAIN guidelines and software assets. Datasets have been adequately gathered or synthetized, the essential features for building safety-relevant cases have been implemented, and models have been trained reaching satisfiable and realistic results.

A good effort was spent in synchronizing with the other Work Packages in the project, both in live discussions and with the support of informal documents, to guarantee a common and consistent direction. Safety experts in the consortium have been involved in architecting the safety operational scenario and the safety features to be assessed in the case studies, XAI researchers have been made aware of the case studies techniques and tasks to guide the design and implementation of explainability tools. Platform and software stack engineers have been in constant touch with WP5 for mutually assess requirements and needs, envisioning integration between case studies and SAFEXPLAIN software stack; in order to ease the process, the toy model provided a simple AI example mimicking a case study implementation and functioning without requiring the same effort and was exploited for early requirements verification and integration testing.

The current results of this first phase are in line with the expectations and objectives posed by the consortium at the beginning of the project, and a solid basis on which the following steps will be taken.

Acronyms and Abbreviations

-	ATSS	– Adaptive Training Sample Selection
-	CAD	– Computer-Aided Design
-	CAIS	– Critical Autonomous AI-based Systems
-	CNN	– Convolutional Neural Network
-	CPU	– Central Processing Unit
-	CS	– Case Study
-	D	– Deliverable
-	FPN	– Feature Pyramid Network
-	GFL	– Generalized Focal Loss
-	GIoU	– Generalized Intersection over Union
-	GNC	– Guidance, Navigation and Control
-	GPU	– Graphics Processing Unit
-	GUI	– Graphical User Interface
-	LR	– Learning Rate
-	MS	– Milestone
-	NMS	– Non-Maximum Suppression
-	ODD	– Operational Design Domain
-	ONNX	– Open Neural Network Exchange
-	PAN	– Path Aggregation Network
-	PnP	– Perspective-n-Point
-	SIMO	– Single-Input, Multiple-Output
-	SLAB	– Space Rendezvous Laboratory
-	SMW	– Safexplain MiddleWare
-	SSD	– Single Shot MultiBox Detector
-	TCC	– Time To Collision
-	WP	– Work Package
-	XAI	– eXplainable Artificial Intelligence

References

[1] https://kelvins.esa.int/satellite-pose-estimation-challenge/, (2019)

[2] https://kelvins.esa.int/pose-estimation-2021/, (2021)

[3] Park, Tae Ha, et al. "SPEED+: Next-generation dataset for spacecraft pose estimation across domain gap." 2022 IEEE Aerospace Conference (AERO). IEEE, 2022., https://arxiv.org/abs/2110.03101

[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), Computer Vision – ECCV 2014, pp. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1

[5] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. CoRR, abs/1805.04687, 2018. URL http://arxiv.org/abs/1805.04687.

[6] Rangi Lyu. Super fast and lightweight anchor-free object detection model. real-time on mobile devices. 2020. URL https://github.com/RangiLyu/nanodet

[7] Honda H, Uchida Y. CLRerNet: improving confidence of lane detection with LaneIoU. InProceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision 2024 (pp. 1176-1185).

[8] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In AAAI, February 2018. 1, 3, 5, 6

[9] Ranftl R, Bochkovskiy A, Koltun V. Vision transformers for dense prediction. InProceedings of the IEEE/CVF international conference on computer vision 2021 (pp. 12179-12188).

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In CVPR, 2012.

[11] CARLA Simulator - https://carla.readthedocs.io/en/latest/

[12] Ge Z, Liu S, Wang F, Li Z, Sun J. Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430. 2021 Jul 18.