Autonomous self-adaptive services for TRansformational personalized inclUsivenesS and resilience in mobiliTy

D3.1 Multimodal data processing. v1

Lead beneficiary	Waveye GmbH	**Lead author**	Gor Hakobyan
Reviewers	Theoktisti Marinopoulou (CERTH), Youjun Choi (KATECH)		
Type	R	**Dissemination**	PU
Document version	V1.0	**Due date**	31/08/2025

Co-funded by the European Union

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

KIAT 한국산업기술진흥원
Korea Institute for Advancement of Technology

AutoTRUST project has received funding from CINEA under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101148123.

Project information

Project title	**Autonomous self-adaptive services for Transformational personalized inclUsivenesS and resilience in mobility**
Project acronym	**AutoTRUST**
Grant Agreement No	101148123
Type of action	Research and Innovation Actions (RIA)
Call	HORIZON-CL5-2023-D6-01
Topic	HORIZON-CL5-2023-D6-01-01 - User-centric development of vehicle technologies and solutions to optimise the on-board experience and ensure inclusiveness (CCAM Partnership)
Start date	1 May 2024
Duration	36 months

Document information

Associated WP	WP3
Associated task(s)	T3.1
Main Author(s)	Gor Hakobyan (Waveye GmbH)
Author(s)	Dimitrios Tsiktsiris, Dimitrios Manolakis, Theoktisti Marinopoulou, Theofilos Christodoulou, Christos Basdekis, Dimitrios Triantafyllou, Eleftherios Blitsis, Theofanis Siamatras, Thomas Kopalidis, Angeliki Zacharaki, Antonios Lalas, Konstantinos Votis (CERTH), Aleksandros Gkillas, Nikolaos-Antonios Piperigkos, Christos Anagnostopoulos, Gerasimos Arvanitis, Aristeidis Lalos (AviSense.AI), Leonidas Katelaris, Christos Kyrkou, Nearchos Stylianides (UCY), Youjun Choi (KATECH), Sooji Yeom (MORAI)
Reviewers	Theoktisti Marinopoulou (CERTH), Youjun Choi (KATECH)
Type	R - Document, report
Dissemination level	PU - Public
Due date	M16 (31/08/2025) – Extended to M17 (30/09/2025)
Submission date	30/09/2025

Document version history

Version	Date	Changes	Contributor (s)
v0.1	28/05/2025	Initial table of contents	Gor Hakobyan (Waveye GmbH)
v0.2	20/06/2025	Contribution in Section 1 and Section 2	All authors
v0.3	04/07/2025	Contribution in Section 2 and Section 3	All authors
v0.4	28/07/2025	Contribution in Section 3 and Section 4, 5	All authors
v0.5	01/08/2025	Finalizing the deliverable for internal review	Gor Hakobyan (Waveye GmbH)
v0.6	26/08/2025	Internal review by CERTH/ITI	Theoktisti Marinopoulou, Antonios Lalas (CERTH/ITI)
V0.7	31/08/2025	Built in the suggestions from the internal review	Gor Hakobyan (Waveye GmbH)
V0.8	02/09/2025	Refinements by CERTH/ITI	Theofilos Christodoulou, Theoktisti Marinopoulou, Antonios Lalas (CERTH/ITI)
V0.9	20/09/2025	Internal review by KATECH	Youjun CHOI (KATECH)
V0.9.5	24/09/2025	Address comments and changes	Theoktisti Marinopoulou (CERTH/ITI)
v0.9.8	29/09/2025	Refinement based on review comments	Antonios Lalas (CERTH) and CERTH team
v1.0	30/09/2025	Final version for submis-sion	Antonios Lalas (CERTH)

Disclaimer

Copyright message

Contents

List of acronyms and abbreviations

Abbreviation	Description
ADAS	Advanced Driving Assistance System
AEC	Acoustic Echo Cancellation
AI	Artificial Intelligent
APE	Absolute Pose Error
ATC	Adapt then Combine
BFS	Breadth-first Search
CAVs	Connected and Autonomous Vehicles
CCAM	Connected and Automated Mobility
CDF	Cumulative Distribution Function
CFAR	Constant False Alarm Rate
CNNs	Convolutional Neural Networks
COCO	Common Objects in Context
CPD	Coherent Point Drift
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DOA	Direction of Arrival
DU	Deep Unrolling
DU-OR	Deep Unrolling with Outlier Removal
DVS	Dynamic Vision Sensor
EAR	Eye Aspect Ratio
EKF	Extended Kalman Filter
EM	Expectation–Maximization
FPFH	Fast Point Feature Histograms
FL	Federated Learning
FPS	Frames Per Second
GICP	Generalized ICP
GMM	Gaussian Mixture Model
GNN	Global Nearest Neighbor
GNSS	Global Navigation Satellite System
GRU	Gated Recurrent Unit
HQS	Half Quadratic Splitting
IAC	Inter-Scale Attention Calibration
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IR	Infrared
JPDA	Joint Probabilistic Data Association
KF	Kalman Filter
LiDAR	Light Detection and Ranging
LeGO-LOAM	Lightweight and Ground-Optimised Lidar Odometry and Mapping on Vari-

Abbreviation	Description
	able Terrain
LLM	Large Language Model
MAR	Mouth Aspect Ratio
MIMO	Multiple-input Multiple-output
mIoU	mean Intersection over Union
MLPs	Multi-Layer Perceptrons
MOT	Multi-Object Tracking
MSCA	Multi-Scale Context Aggregation
MTCNN	Multi-task Cascaded Convolutional Neural Network
NDT	Normal Distributions Transform
NS	Noise Suppression
ONNX	Open Neural Network Exchange
PCA	Principal Component Analysis
RANSAC	Random Sample Consensus
RCS	Radar Cross Section
RIA	Research Innovation Action
SA	Set Abstraction
SLAM	Simultaneous Localisation and Mapping
SNR	Signal Noise Ratio
SR	Super-Resolution
SRAE	Super-Resolution Autoencoder
STFT	Short-Time Fourier Transform
TDoA	Time Difference of Arrival
TSA	Temporal Sample Alignment
TSDF	Truncated Signed Distance Function
V2X	Vehicle to everything
VAD	Voice Activity Detection
VIT	Vision Transformer

List of figures

List of tables

Executive summary

This document presents the foundational methodologies, algorithms, and technical frameworks that support the AutoTRUST project's vision for autonomous, inclusive, and resilient mobility solutions. The deliverable is structured to guide both technical partners and the broader consortium in the integration and advancement of multimodal perception, focusing on the acquisition, synchronisation, and processing of heterogeneous sensor data. Key technologies covered include radar, LiDAR, RGB cameras, and microphone arrays, each selected for their complementary strengths in robust perception under diverse operating conditions. The document provides a critical review of the state of the art, highlighting the need for cost-effective yet accurate solutions that can be deployed at scale, particularly for applications where high-resolution sensors may be cost-prohibitive.

The deliverable introduces a comprehensive suite of algorithms and pipelines for external sensor data processing and fusion. The focus is placed on perception based on sensor modalities such as LiDAR, radar and cameras. A LiDAR super-resolution framework is introduced, where a novel model-based Deep Unrolling (DU) framework is developed to enhance the spatial resolution of affordable, low-channel LiDAR sensors. This approach is designed for real-time execution and is tightly integrated with downstream tasks such as Simultaneous Localisation And Mapping (SLAM) and semantic segmentation. The report demonstrates, through extensive benchmarking, that the proposed Super-Resolution (SR) methods not only bridge the performance gap with high-end LiDAR devices but also ensure computational efficiency, making them suitable for embedded and resource-constrained platforms. For segmentation, a pioneering end-to-end optimisation is presented, in which the SR and segmentation networks are jointly trained using context-aware loss functions to maximize accuracy for both dominant and underrepresented classes. The document further details radar-based environment perception and mapping pipelines, leveraging the resilience of radar to challenging weather and lighting conditions. Additionally, a federated learning approach named FedKalmanNet is introduced, to enable multimodal vehicle localisation. This approach allows vehicles to aggregate local sensor measurements and collaboratively train models for high-accuracy positioning, all while preserving data privacy and reducing communication overhead. Performance evaluations using real-world datasets and simulation environments (such as SemanticKITTI, SemanticPOSS, and CARLA) validate the effectiveness and efficiency of these external perception modules.

For radar, the report introduces a perception pipeline capable of reliably classifying vulnerable road users such as humans in any weather and lighting conditions. The presented pipeline takes the 4D radar point cloud as an input and processes it through multiple steps including clustering, tracking using a Kalman filter with joint probability density association, as well as a deep

neural network-based classification algorithm. For the latter, Pointnet++ based architecture is designed and adopted to the radar modality. Based on the collected dataset, the model is trained to reliably classify humans and distinguish from other objects on the road. Furthermore, radar-based localisation and mapping framework is introduced that achieves a remarkable mapping accuracy based on a single drive, building a highly dense radar map that can serve for localisation in any environmental condition, such as night, fog, snow, or heavy rain.

On the internal monitoring system side, the deliverable addresses the technical groundwork for future in-cabin perception and monitoring applications. Rather than presenting final driver or passenger monitoring systems, this document focuses on the systematic collection, synchronisation, and fusion of visual and acoustic data within the vehicle cabin. The deliverable details the architecture for capturing high-quality visual information using RGB cameras and spatially synchronised audio streams using advanced microphone arrays. These multimodal datasets are curated with attention to the requirements of future analytics, such as driver distraction detection, facial emotion recognition, occupant identification, drowsiness detection, and abnormal sound event recognition. Emphasis is placed on establishing reliable data pipelines, robust synchronisation methods, and flexible interfaces that enable the seamless integration of various sensor modalities. This technical foundation ensures that subsequent work packages can efficiently leverage the collected data to develop advanced in-cabin analytics, virtual assistants, and real-time safety applications. The deliverable also discusses strategies for data annotation, the evaluation of open-source and project-specific datasets, and the challenges inherent in monitoring dynamic in-cabin environments, ensuring that the AutoTRUST project is equipped to address both technical and human-centric needs.

The deliverable further establishes clear best practices for designing scalable, modular, and hardware-agnostic systems, ensuring future-proof integration with evolving sensor technologies and analytics modules. Comprehensive benchmarking and evaluation protocols provide evidence of the project's commitment to high performance and practical deployment. As the initial release of the multimodal data processing framework, this deliverable sets the stage for ongoing development, refinement, and validation throughout the AutoTRUST project lifecycle, with future updates planned to capture advances in technology and feedback from real-world pilot deployments.

1 Introduction

This deliverable, D3.1 “Multimodal Data Processing”, serves as a key technical milestone in the AutoTRUST project, providing a detailed account of the methodologies, algorithms, and integration strategies developed for processing heterogeneous sensor data in automated mobility environments. The document is intended to guide both technical partners and consortium stakeholders in the design, implementation, and benchmarking of robust multimodal perception pipelines. By establishing best practices for sensor selection, calibration, data acquisition, and fusion, this deliverable lays the groundwork for scalable, real-time processing frameworks that support a wide range of perception and monitoring applications—both external (environmental sensing) and internal (in-cabin analytics). The content of D3.1 is structured to ensure alignment with project objectives of inclusiveness, user-centricity, and resilience, and provides the reference architecture for subsequent technical work packages and pilot site deployments.

1.1 Purpose and structure of the document

The purpose of this deliverable, D3.1 “Multimodal Data Processing”, is to provide a comprehensive account of the technical foundations and key decisions guiding the development of multimodal perception systems within the AutoTRUST project. This report documents the essential activities and approaches undertaken in the initial stages of the project, including an in-depth review and selection of sensing technologies and data processing methodologies. The practices outlined herein have been chosen not only for their technical merits, but also for their alignment with AutoTRUST’s vision of fostering reliable, inclusive, and adaptive automated mobility solutions.

Beyond the technical underpinnings, the document places significant emphasis on the systematic integration and synchronisation of heterogeneous sensor data, establishing robust pipelines for external environmental sensing and internal in-cabin analytics. It details the architectural and algorithmic strategies that ensure high performance across diverse operational scenarios, reflecting the collaborative efforts of project partners. These strategies serve as technical guidelines, supporting the creation of scalable and hardware-agnostic systems capable of evolving with future advancements in sensor and AI technologies.

Additionally, the report presents the evaluation criteria and Key Performance Indicators (KPIs) established to assess the effectiveness, efficiency, and scalability of the developed systems, with attention to real-world applicability and user impact.

Following the Introduction, which clarifies the document’s objectives, intended audience, and role within the broader project framework, the structure is organized as follows:

- **Section 2** describes the sensor modalities used in the project, detailing the acquisition pipelines and calibration procedures for each sensing technology.
- **Section 3** introduces the core algorithms for external perception, including LiDAR super-resolution, radar-based detection and Simultaneous Localisation and Mapping (SLAM), and multimodal fusion techniques.
- **Section 4** focuses on internal monitoring, presenting methods for fusing visual and acoustic data for occupant state estimation, along with synchronisation and simulation frameworks.
- **Section 5** summarises the findings and outlines future directions for development and integration in subsequent project phases.

This structure is designed to provide all stakeholders with clear guidance on the technical trajectory of the project, while ensuring traceability and alignment with the overarching goals of AutoTRUST.

1.2 Intended Audience

The AutoTRUST "Multimodal Data Processing" deliverable is intended for both public dissemination and internal use within the AutoTRUST consortium, which includes project members, research partners, industry collaborators, and affiliated stakeholders. This document serves as a comprehensive reference, providing detailed technical guidance on sensor selection, data acquisition, synchronisation, and multimodal data processing methodologies. It is designed to support engineers, researchers, and system architects involved in the development and integration of advanced perception systems, while also offering insight for external parties interested in the project's technical foundations and best practices. Throughout the duration of the project, this deliverable acts as a foundational resource to inform decision-making, ensure methodological consistency, and guide future developments within the AutoTRUST framework.

1.3 Interrelations

The AutoTRUST consortium brings together a diverse range of expertise and resources from leading academic institutions, research organizations, and industry partners across Europe and associated countries. This multidisciplinary collaboration is focused on developing innovative, AI-driven solutions that enhance inclusiveness, resilience, and trust in Cooperative Connected and Automated Mobility (CCAM) systems. With sixteen partners spanning ten EU member states as well as Norway, Switzerland, the United Kingdom, Korea, and Japan, the consortium ensures comprehensive coverage of critical topics such as security, privacy, safety, and user well-being.

AutoTRUST operates as a Research Innovation Action (RIA) project, organized into six Work Packages (WPs) that are further subdivided into targeted tasks. Partners contribute across multiple WPs, enabling a well-coordinated approach to project management, technical development, and dissemination. This structured framework fosters close cooperation and effective knowledge transfer among research institutes, universities, SMEs, and large industry players.

Within this context, the “Multimodal Data Processing” deliverable plays a pivotal role in supporting and informing the technical activities across the project. The methodologies, algorithms, and processing pipelines documented here provide essential input to technical work packages such as WP3 and WP4, while also aligning with the requirements and specifications defined in earlier project stages. Furthermore, the outputs of this deliverable are designed to facilitate evaluation and validation processes in later phases of the project, thereby ensuring a seamless integration of multimodal perception within the broader AutoTRUST architecture.

2 Multi-Modal sensor data acquisition and calibration

This section introduces the sensor modalities integrated into the AutoTRUST framework and outlines the methods used to acquire, synchronise, and calibrate their data. Each sensor modality, i.e., radar, RGB camera, LiDAR, microphone array, and event-based camera, offers complementary strengths that, when fused, enable robust perception across diverse driving conditions. The section provides a concise description of each sensor's operating principles and performance characteristics, followed by calibration techniques essential for spatial and temporal alignment. These procedures ensure accurate data fusion and serve as the foundation for the downstream processing and analytics modules detailed in later sections.

2.1 Radar sensor

The primary function of automotive radar (example shown in Figure 1) is the detection of objects in the vehicle's surroundings and estimation of their parameters such as distance, velocity and direction. Based on this information, environment perception is enabled for higher level functions such as advanced driver assistance systems and autonomous driving. The principle behind radar is the transmission of electromagnetic waves that are reflected from the surrounding objects called radar targets.

By reception and processing of the reflected radar signal, the presence of objects in the radar surroundings is identified (detection) and their parameters are estimated. Radar conventionally measures the target ranges (distances). Radar also enables measurement of targets' relative radial velocities based on the Doppler effect. To localise the radar targets, typically also estimation of target directions (angles) is performed. For a three-dimensional (3D) target localisation, along with the distances both the azimuth and elevation angles of the targets are required.

Radar uses the time-of-flight principle for distance measurement and the Doppler effect for velocity measurement. For measurement of target directions, the Direction of Arrival (DOA) of the reflected electromagnetic waves is estimated. This is typically performed by "scanning" all directions based on the assumption that the antenna or the antenna array has a directive radiation pattern, i.e. receives more power from a certain direction. This dominant direction of the radiation pattern is called main beam. By pointing the main beam subsequently in each possible direction (beamforming), the DOA with the maximum power can be identified. Assuming a single main beam in the entire range of DOA from where the antenna is able to receive sufficient power (i.e. no ambiguities called grating lobes), the beam with the highest power will point to the DOA of the received signal. The narrower the main beam, the more precise DOA estimation can be performed. Today's automotive radars use digital beamforming in combination with

Multiple-Input Multiple-Output (MIMO) radar architecture to determine the angular information of the targets efficiently.

The target detection is performed on the range-Doppler radar spectra using Constant False AlaRm (CFAR) detector. The detected radar targets are added to a radar point cloud, whereas every point has its range, velocity, azimuth and elevation information, as well as its Radar Cross Section (RCS), which is a metric for intensity of the reflected signal from the target. This 4D radar point cloud (3D coordinates and the velocity for every measured point) is fed to the perception module, which detects objects such as cars, pedestrians, cyclists, overridable and underridable objects. This perception output is then fused with other modalities for higher level decision making.

A major strength of radar is the weather and light independence, due to which radar is robust at night, under strong sun, in the presence of heavy rain, snow or fog. This makes radar a critical sensor modality for autonomous driving functions.

Figure 1: Waveye Argus A2 Lightweight Imaging Radar (LIR)

The parameters of Waveye's Argus A2 imaging radar are presented in Table 1 for two range configurations, mid and long range.

Table 1: Argus A2 imaging radar parameters

Parameter	LIR Argus A2: Mid Range	LIR Argus A2: Long Range
Range	50 m	200 m
Range resolution / range separability[1]	4 cm / 8 cm	15 cm / 30 cm

Parameter	LIR Argus A2: Mid Range	LIR Argus A2: Long Range
Range accuracy[1]	1.5 cm	3 cm
Velocity range[1]	(-20, +20) m/s	(-100, +60) m/s
Velocity resolution / separability[1]	0.03 m/s	0.07 m/s
Velocity accuracy[1]	0.015 m/s	0.03 m/s
Azimuth Field of View (10 dB beam-width)	(-55°, +55°)	(-55°, +55°)
Azimuth resolution[3] (native / algorith-mic)	1.15° / 0.55°	1.15° / 0.55°
Azimuth accuracy[2] [3]	0.1°	0.1°
Elevation Field of View (10 dB beam-width)	(-17.5°, +17.5°)	(-17.5°, +17.5°)
Elevation resolution[3] (native / algorith-mic)	1.3° / 0.65°	1.3° / 0.65°
Elevation accuracy[2] [3]	0.1°	0.1°
Cycle time (update time)[1]	100 ms	70 ms

2.2 RGB sensor

The primary function of RGB cameras in automotive and public transport applications is the acquisition of high-resolution visual data in the red, green, and blue spectral bands, enabling dense spatial perception of the vehicle's surroundings or interior environment. This modality supports a range of higher-level functions such as activity monitoring, object recognition, and event detection, which are critical for both safety and operational analytics.

The principle behind RGB imaging is passive sensing through the collection of ambient light reflected off surfaces in the scene. The camera sensor captures the intensity of light in three color channels (red, green, blue), which are then combined to produce a full-color image approximating human visual perception. Unlike active sensing modalities (e.g., radar or LiDAR), RGB cameras do not emit signals; instead, they rely on external illumination, which constrains performance in low-light or adverse lighting conditions unless supported by infrared or auxiliary lighting.

From the captured image stream, spatial features such as edges, textures, and color gradients are extracted and used to infer the presence, identity, and posture of objects and persons. Modern vision-based systems utilise Convolutional Neural Networks (CNNs) or transformer-based architectures to process RGB frames and perform tasks such as person detection, gesture

recognition, and object classification. Temporal continuity in video data further enables motion-based analytics, such as activity recognition or anomaly detection in dynamic scenes.

Camera-based perception provides dense 2D spatial resolution, allowing pixel-level segmentation and fine-grained interpretation of visual scenes. Unlike point cloud–based sensors, RGB cameras do not inherently provide depth information, though stereo vision setups or fusion with depth modalities (e.g., radar, LiDAR, or structured light) can compensate for this limitation. The modality excels at capturing semantic information — including textual content (e.g., signage), clothing color, and facial expressions — which are critical for applications involving human monitoring and behavior understanding.

RGB cameras are sensitive to ambient conditions: performance can degrade under strong backlighting, glare, or at night. However, their passive nature and high information density make them cost-effective and suitable for daytime surveillance tasks where visual context is essential. In multimodal systems, RGB data is typically fused with geometric modalities to balance semantic richness with spatial accuracy, contributing to a comprehensive perception stack for both safety-critical functions and operational intelligence.

2.3 RGB+Depth

The Intel RealSense D435 as depicted in Figure 2 is an RGB-D camera designed to capture both color and depth information simultaneously, making it suitable for 3D perception, gesture recognition, and robotic navigation applications. Unlike conventional RGB cameras that provide only 2D color images, this sensor combines a high-resolution RGB sensor with a stereo infrared (IR) depth camera to generate real-time depth maps as shown in Figure 3.

The D435 integrates dual global-shutter IR cameras for depth sensing, along with a standard RGB sensor, feeding data into an on-board processing pipeline that handles depth computation and alignment. It outputs synchronized RGB and depth streams over a USB 3.0 interface, supporting high frame rates and resolutions for accurate scene reconstruction.

Figure 2: Intel RealSense D435 camera

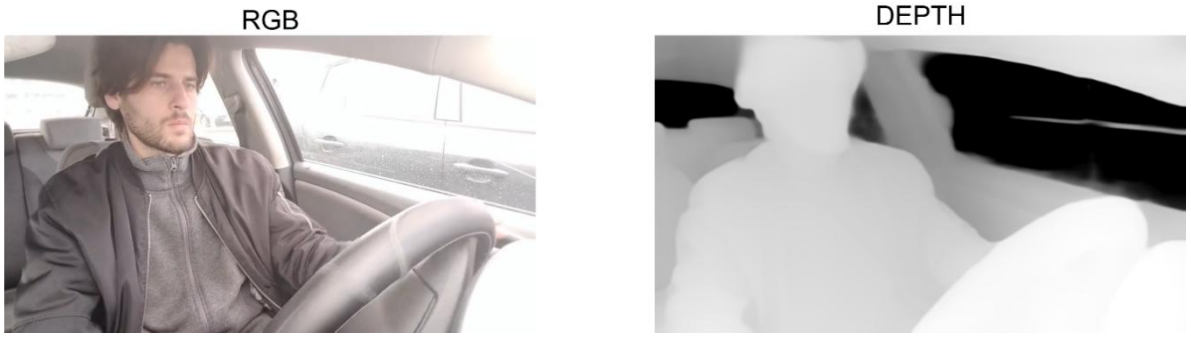


Figure 3: Example RGB and Depth output from the D435

Key features include real-time depth perception, configurable depth range, advanced filtering for noise reduction, and hardware acceleration for efficient processing. These capabilities allow the sensor to accurately capture the 3D structure of environments, detect object positions, and track motion, which is crucial in robotics, augmented reality, and interactive systems.

The camera can provide both raw depth data and pre-processed aligned RGB-D streams, offering flexibility for custom computer vision applications or immediate use in perception pipelines. Its detailed specifications are summarised in Table 2.

Table 2: Intel RealSense D435 – Specifications

Feature	Specification
RGB Sensor	1920 × 1080, 30 FPS
Depth Sensor	Stereo IR, 1280 × 720, 30–90 FPS
Depth Range	0.1 m – 10 m
Field of View	RGB: 69° × 42°, Depth: 87° × 58°
Accuracy	±2% at 1 m
Interface	USB 3.0
Global Shutter	Depth cameras: Yes

2.4 360 Fisheye Panoramic Camera

The Vivotek FE9180-H-V2 as shown in Figure 4 is a 360° fisheye panoramic camera designed to capture a full, hemispherical view of the environment, making it ideal for surveillance, scene monitoring, and immersive video applications. Unlike traditional fixed or PTZ cameras that provide limited fields of view, this camera employs a fisheye lens to deliver a complete panoramic image without blind spots.

The FE9180-H-V2 integrates a high-resolution image sensor with advanced image processing capabilities, allowing real-time video streaming with minimal distortion. The camera supports

multiple viewing modes, including panoramic, dewarped, and quadrant views, which can be configured either through hardware or software for specific monitoring needs.

Key features include high-definition video capture, low-light performance, Wide Dynamic Range (WDR) for challenging lighting conditions, and H.265/H.264 video compression for efficient network transmission. These capabilities enable continuous monitoring of large areas while maintaining high image quality and reducing bandwidth usage.

The camera outputs video streams via standard IP interfaces and supports Power over Ethernet (PoE), simplifying installation and integration into networked surveillance systems. Its flexibility allows both real-time monitoring and recorded analysis of the environment.

Figure 4: Vivotek FE9180-H-V2 360° Fisheye Camera

2.5 LiDAR sensor

Light Detection and Ranging (LiDAR) sensors are essential components in modern perception systems for autonomous vehicles, robotics, and advanced mapping applications. LiDAR operates by emitting rapid laser pulses and measuring the time it takes for each pulse to return after reflecting off surrounding objects. This process enables the sensor to construct a detailed three-dimensional (3D) representation of the environment, known as a point cloud. The spatial accuracy and reliability of LiDAR make it particularly valuable for applications that require precise object detection, environmental mapping, and obstacle avoidance.

Despite their technical strengths, LiDAR sensors face a number of challenges that limit their widespread adoption, particularly in cost-sensitive domains such as mass-market automotive

deployment. A critical issue is the trade-off between resolution and cost. High-resolution LiDAR, which can feature 64 or even 128 vertical channels, provides dense and rich 3D data that greatly improves perception accuracy. However, these high-end sensors remain prohibitively expensive, which restricts their use to premium vehicles and research platforms. Conversely, low-resolution LiDARs, often equipped with just 16 or 32 channels, are far more affordable and thus suitable for broader deployment, but the resulting point clouds are much sparser. This sparsity can lead to missed detections of small or distant objects, reduced scene understanding, and difficulties in tasks such as object segmentation and SLAM. Moreover, the lower density of returns in certain regions can exacerbate problems with occlusions and environmental ambiguities. To overcome these limitations, there is a growing focus on LiDAR super-resolution methods that seek to reconstruct or enhance sparse point clouds using algorithmic or learning-based approaches. These methods aim to bridge the gap between low-cost, low-resolution sensors and the rich perception capabilities of high-end devices, thereby making advanced perception solutions more accessible and scalable.

A representative example of a cost-effective LiDAR sensor is the Velodyne VLP-16, installed also in AVISENSE.AI's vehicle as depicted in Figure 5. This sensor offers a 360-degree horizontal field of view and a vertical field of view of approximately 30 degrees, distributed across its 16 laser channels. The VLP-16 can collect up to 300,000 points per second and provides reliable range measurements up to 100 meters under ideal conditions. Its relatively low weight, compact size, and reduced cost compared to higher-channel-count sensors have made it a popular choice for autonomous vehicle prototypes, robotics platforms, and research applications that require scalable LiDAR deployment.

Figure 5: The Velodyne VLP-16 LiDAR sensor, as deployed on AVISENSE vehicles, provides 360-degree environmental perception with 16 vertical channels, enabling real-time 3D mapping and object detection.

2.6 Microphone sensor array

The ReSpeaker Mic Array v2.0 as shown in Figure 6 is a microphone array sensor designed to capture and process multi-directional sound in real time, making it suitable for voice-interactive systems, audio localisation, and speech enhancement in noisy environments. Unlike traditional single-channel microphones that capture omnidirectional audio without spatial context, this sensor integrates four digital omnidirectional MEMS microphones arranged in a circular array. Each microphone operates independently, feeding digital audio data into a centralised on-board DSP (XMOS XVF-3000), which handles all signal processing locally.

The array outputs processed audio streams via a USB audio class interface and supports advanced features such as beamforming (directional audio focusing), Direction-of-Arrival (DoA) estimation, Voice Activity Detection (VAD), Noise Suppression (NS), dereverberation, and Acoustic Echo Cancellation (AEC). These capabilities allow it to isolate voice signals from multiple speakers and filter out ambient noise or echoes — crucial in smart speaker, conference, or robotic interaction systems.

This array responds in real time to changes in the acoustic environment, providing either a single-channel enhanced audio stream or raw multi-channel microphone data. This allows for flexibility in both pre-processed voice applications and custom audio signal analysis. Its specifications are presented in detail in Table 3.

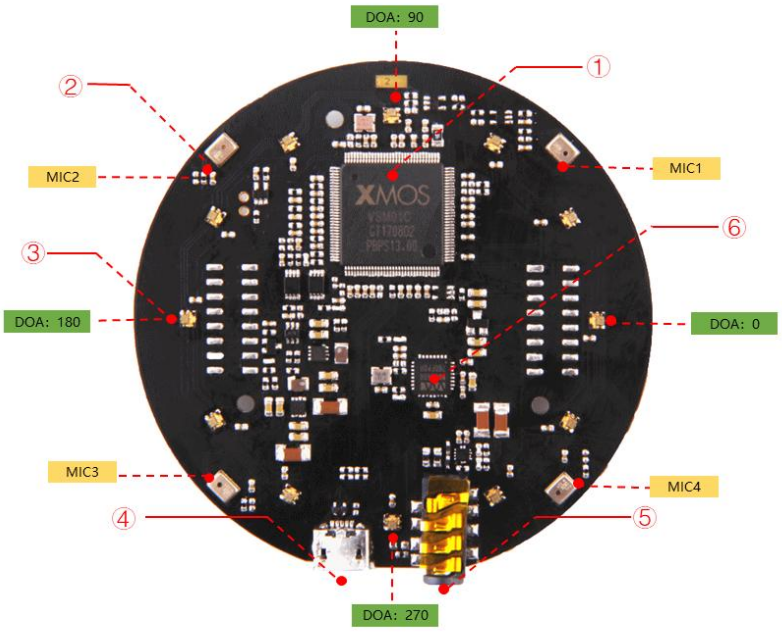


Figure 6: ReSpeaker Mic Array V2.0

Table 3: ReSpeaker Mic Array V2.0 - Specifications

Feature	Specification
Microphones	4 × ST MP34DT01TR-M digital MEMS
Array Geometry	Circular, 70 mm diameter
SNR	~ 63 dB
Frequency Response	100 Hz – 8 kHz
Maximum SPL	~ 120 dB
Sampling rate	Up to 16 kHz (via USB)
LEDs	12 RGB WS2812 LEDs, programmable
Audio Output	3.5 mm headphone jack (via WM8960 codec)

2.7 Event-Based camera sensor

An event-based camera (also known as a neuromorphic camera or dynamic vision sensor, DVS) is a type of image sensor that captures changes in a scene asynchronously, rather than recording full images at fixed intervals like traditional frame-based cameras. Each pixel in an event-based camera operates independently. It monitors changes in brightness (logarithmic intensity) and only sends data when it detects a change. This change is called an event. If nothing changes in the scene at a particular pixel, that pixel remains silent. Instead of outputting full images, the camera produces a stream of events in the form of (x, y, t, p), where x, y are the pixel coordinates, t is the timestamp of the event (with microsecond resolution) and p, polarity of brightness change (+1 for ON events, -1 for OFF events).

Figure 7: Prophesee's event-based sensor Evaluation Kit 3

In comparison with traditional RGB cameras, event-based sensors have high temporal resolution and low power consumption, due to sparse, asynchronous data output. Additionally, since

each pixel only emits events when brightness changes, static background regions (e.g., a wall, ground, or sky) do not generate events. This significantly reduces the amount of data related to unchanging parts of the scene.

Due to their ability of filtering out static or irrelevant background information, event-based cameras are especially valuable in tasks, such as Human Activity Recognition, the pose of human actions in dynamic environments is crucial. In order to capture event data, Prophesee's fourth-generation sensor as illustrated in Figure 7, EVK3, was selected, with a resolution of 1280 × 720 pixels. The sensor ensures that the diffraction patterns are recorded in real-time, offering detailed visualisation without the limitations of traditional frame-based cameras. The camera has a pixel size of 6.3 μm, a dynamic range exceeding 140 dB and operates with latencies below 150 μs.

2.8 Sensor calibration

This section describes the calibration methods that can be used for the cameras sensors in order to optimise their operation. These methods ensure accurate measurements and provide a reliable foundation for subsequent temporal and spatial registration.

2.8.1 Event-based sensor calibration

The settings of the event-based camera are fine-tuned in order to reduce the noise generated by unwanted events (the final values for each bias setting are presented in Table 4). We achieve that by increasing the threshold after which an event occurs and adding a high pass filter to remove the static noise. These configurations help later in the sampling of the dataset and in the training process of the model.

Table 4: Event-based sensor biases settings

Bias	Description	Value
bias diff off	Adjusts the contrast threshold for OFF events	40
bias diff on	Adjusts the contrast threshold for ON events	135
bias fo	Adjusts the low-pass filter	74
bias hpf	Adjusts the high-pass filter	62
bias refr	Adjusts the refractory period	68

2.8.2 Fisheye Camera Calibration Using a Chessboard

Fisheye lenses capture images with a very wide field of view, often greater than 180°, but introduce significant non-linear radial distortion. Accurate calibration of fisheye cameras is necessary to estimate intrinsic parameters and distortion coefficients, enabling geometric correction

of images for precise computer vision tasks. A planar chessboard pattern with known dimensions is commonly used as a calibration target due to its regular, high-contrast feature points.

Camera Model and Distortion

For fisheye cameras, the projection model deviates from the classical pinhole camera. The equidistant projection model is widely adopted, where the radial distance **r** in the image plane relates linearly to the angle θ between the incoming light ray and the optical axis:

$$r = f \cdot \theta \qquad \textit{Equation 1}$$

where f is the focal length in pixels and θ is the angle of incidence, respectively.

To model the lens distortion, the distorted angle θ_d is represented as a polynomial function of the undistorted angle θ:

$$\theta_d = \theta \cdot 1 + (k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \qquad \textit{Equation 2}$$

where, k_1, k_2, k_3, k_4 are the distortion coefficients, respectively.

Normalized image coordinates x, y (obtained by dividing the 3D point coordinates by the depth) are projected to distorted image coordinates x_d, y_d as:

$$x_d = f_x \cdot \frac{x}{r} \cdot \theta_d + c_x \qquad \textit{Equation 3}$$

$$y_d = f_y \cdot \frac{y}{r} \cdot \theta_d + c_y \qquad \textit{Equation 4}$$

where f_x, f_y are focal lengths in pixels, and c_x, c_y represent the principal point coordinates, respectively.

Calibration Methodology

Data Acquisition

A planar chessboard pattern with a known square size is used as the calibration target. Multiple images are captured from different angles and distances to provide diverse perspectives of the pattern.

Feature Extraction

For each captured image, the 2D image coordinates of the internal chessboard corners are detected with sub-pixel accuracy. These points correspond to known 3D positions on the calibration target.

Object Points Definition

The 3D coordinates of the chessboard corners are defined on a plane (typically Z=0):

$$X_j = (i \cdot s, j \cdot s, 0) \qquad \textit{Equation 5}$$

where s is the square size, and i, j index the corner positions, respectively.

Parameter Estimation

Using the correspondences between detected 2D image points and known 3D object points, the intrinsic camera matrix K, distortion coefficients D\mathbf{D}D, and the extrinsic parameters (rotation R_i and translation t_i for each image) are estimated.

The goal is to minimize the total reprojection error:

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\left\|x_{ij} - \pi\left(K, D, R_i, t_i, X_j\right)\right\|^2 \qquad \textit{Equation 6}$$

where:

- N is the number of images,
- M is the number of corners per image,
- x_{ij} is the observed 2D corner location in image iii,
- π (·) represents the fisheye projection function with distortion,
- K contains focal lengths and principal point,
- $D = k_1, k_2, k_3, k_4$ are distortion coefficients,

- R_i, t_i represent the camera pose in image iii, respectively

Non-linear optimisation techniques are applied to find the parameter set minimizing this error.

Image Undistortion

With the calibration parameters estimated, fisheye images can be undistorted by applying the inverse of the distortion model. This process maps each pixel from the distorted image to a corrected location, producing rectified images suitable for downstream vision tasks.

Chessboard-based calibration provides a practical and accurate method for estimating the intrinsic parameters and distortion coefficients of fisheye cameras. By modelling the nonlinear projection and distortion, this process enables effective correction of fisheye images, facilitating their use in precise computer vision applications for in-cabin monitoring.

2.9 Concluding remarks

The sensor suite described in this section establishes the hardware foundation for AutoTRUST's multimodal perception capabilities. Each sensor modality offers unique strengths, i.e., LiDAR for accurate 3D mapping, radar for robustness in adverse conditions, RGB and event-based cameras for rich semantic information, and microphones for acoustic scene understanding. The calibration and synchronisation techniques presented ensure spatial and temporal alignment across modalities, enabling effective downstream fusion and analytics. These preparations are critical for achieving reliable, real-time performance in both external and internal perception pipelines. The next sections build upon this hardware foundation by introducing advanced algorithms for external and internal sensor data processing and multimodal enhancement.

3 Algorithms for external multi-modal sensor data processing and enhancement

This section presents the core algorithms developed for processing and enhancing data from external sensor modalities, specifically focusing on LiDAR and radar as active sensors. The focus is on enabling accurate perception and localisation through cost-effective and scalable methods that operate in real-time. Key contributions include deep learning–based LiDAR super-resolution for SLAM and segmentation, advanced radar-based environment perception pipelines, and federated localisation using GNSS and vehicle odometry. These algorithms are designed to function independently or as part of the external monitoring system, providing robust performance in complex and dynamic driving environments.

3.1 LiDAR-based data enhancement for improved perception

This section describes the use of LIDAR for both improved SLAM and object segmentation.

3.1.1 LiDAR super-resolution for Improved SLAM

LiDAR SLAM is essential for autonomous vehicles and robotics, enabling self-localisation and environmental mapping. Compared to visual SLAM, LiDAR-based methods offer superior reliability in low-light or low-visibility conditions. However, high-resolution LiDAR sensors are costly, making them impractical for many applications. While 64-channel LiDAR provides superior accuracy, lower-cost 16-channel sensors produce sparser point clouds, leading to reduced odometry accuracy, increased drift, and degraded SLAM performance [1].

Inevitably, research has focused on enhancing the resolution of low-cost LiDAR sensors to bridge the performance gap with high-resolution counterparts. This upsampling can be achieved through Super-Resolution (SR) techniques, which refine low-resolution LiDAR data by reconstructing missing details and increasing spatial density. However, most SR approaches fail to account for their impact on SLAM accuracy [2]. These methods typically process either raw 3D point clouds [3] or 2D range images [4] derived from LiDAR scans, yet they introduce several critical limitations that hinder their effectiveness in real-world SLAM applications. Deep learning-based SR algorithms outperform traditional interpolation methods but rely on complex architectures with numerous learnable parameters [4]. Their high computational complexity limits frame rates, making real-time processing impractical for SLAM applications. Additionally, these methods operate independently of SLAM optimisation, leading to inconsistent high-resolution reconstructions and artifacts that propagate errors in the SLAM pipeline. Many SR techniques also introduce outliers, further degrading SLAM accuracy and requiring costly post-processing, preventing real-time execution, resulting in an end-to-end real-time pipeline with

improved accuracy. By leveraging the DU framework, we reformulate a novel optimisation problem into an efficient deep learning architecture, thus resulting in an improved end-to-end SR and SLAM approach.

3.1.2 Preliminaries

LiDAR-based Super-Resolution

LiDAR Super-Resolution (SR) techniques can generally be classified into two main approaches. The first category focuses on performing SR directly on raw 3D LiDAR point clouds. However, these methods require substantial computational resources to identify neighboring point relationships and often necessitate additional processing steps, such as segmentation, due to the inherent sparsity of 3D point clouds [5]. An alternative approach operates in the range image domain by projecting 3D point clouds onto 2D range images [4]. This representation offers a more compact format, effectively mitigating data sparsity issues. However, these methods typically rely on deep learning architectures with a large number of parameters, such as U-Net-based [6]. Despite their potential, the computational demands of these models, combined with the need for post-processing to remove outliers, pose significant challenges for real-time implementation. The high complexity of inference, along with the additional processing overhead, often results in suboptimal frame rates, limiting their applicability in real-world SLAM.

3.1.3 Proposed Methodology

In this section, the basic SR LiDAR optimisation problem is formulated by developing a model-based deep learning network to tackle the LiDAR SR problem, and integrating the SR method into an end-to-end LiDAR SLAM architecture.

Optimisation Problem

To construct the proposed model-based deep learning architecture, we leverage the advantages of projection-based methodologies. Let $\boldsymbol{S_1}$ represent a high-resolution LiDAR sensor with $\boldsymbol{N_1}$channels e.g., 64, and $\boldsymbol{S_2}$ a low-resolution LiDAR sensor with $\boldsymbol{N_2}$ channels e.g., 16, where $\boldsymbol{N_1}, \boldsymbol{N_2} \in \mathrm{Z}$ and $\boldsymbol{N_1} > \boldsymbol{N_2}$. We project the 3D high-resolution point cloud $\boldsymbol{P}$, obtained from $\boldsymbol{S_1}$, into a high-resolution range image $\boldsymbol{H} \in R^{N_1 \times K}$, where K represents the horizontal resolution of the sensor. If $\boldsymbol{L} \in R^{N_2 \times K}$, denotes the range image produced by the low-resolution 16-channel LiDAR sensor, the transformation between the two range images i.e., the low-resolution and high-resolution image can be described by the following mathematical relationship:

$$L = DH + E \qquad \textit{Equation 7}$$

where $\boldsymbol{D} \in R^{N_2 \times N_1}$ denotes the downsampling operator that selects only the N_2 channels from the high-resolution range image and $\boldsymbol{E}$ is a noise term. Our goal is to upscale the quality of the resolution of the point cloud generated from the low-resolution sensor. To achieve this, we formulate an optimisation problem as:

$$\underset{\boldsymbol{H}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{L} - \boldsymbol{DH}\|_F^2 + \mu J(\boldsymbol{H}) \qquad \textit{Equation 8}$$

where the first component ensures consistency with the degradation model defined in Equation 7. The second component $J(.)$ serves as a learnable regularizer, designed to capture the intrinsic features of the high-resolution range image $\boldsymbol{H}$, and μ is the regularization parameter. However, the above formulation of method still faces challenges in ensuring real-time performance, as the learnable regularizer, which aims to capture the intrinsic properties of range images, relies on 2D convolutions. These convolutions, while effective in learning spatial patterns, do not inherently preserve the 3D geometric consistency of the original LiDAR point cloud. As a result, they often introduce outliers, particularly in regions with depth discontinuities, sharp edges, or occlusions. These artifacts often appear as spurious points in free-space regions or distortions along object boundaries, significantly affecting downstream SLAM tasks. To restore consistency, additional post-processing steps are often required to handle outliers. However, these steps increase computational overhead, restricting real-time performance and making the approach impractical for autonomous navigation.

To overcome this, we introduce a novel regularization term that simultaneously preserves structural consistency and removes outliers within the optimisation process. By integrating the outlier removal process directly into the optimisation framework, we eliminate the need for explicit post-processing while simultaneously enhancing the effectiveness of the learnable regularizer $J(.)$. Since $J(.)$ is trained jointly with the outlier removal module, it can learn to better capture the structural properties of the high-resolution range image while being aware of the effects of outlier suppression. This synergistic optimisation leads to more accurate and stable SR outputs, ultimately improving LiDAR SLAM performance. Thus, we propose a novel optimisation problem, defined as:

$$\underset{\boldsymbol{H}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{L}-\boldsymbol{D}\boldsymbol{H}\|_F^2+\mu J(\boldsymbol{H})+\lambda \mathrm{Q}(\boldsymbol{H}) \qquad \textit{Equation 9}$$

where the term $\mathrm{Q}(.)$ represents an outlier removal regularizer. This module guides the solution toward generating geometrically consistent high-resolution range images.

Model-based deep learning SR: To address the proposed optimisation problem in Equation 9, we utilise the Half quadratic splitting (HQS) methodology to decompose the initial problem into more manageable subproblems. Hence, the initial problem can be reformulated as:

where $\boldsymbol{Z} \in R^{N_1 \times K}, \boldsymbol{Y} \in R^{\boldsymbol{N_1} \times K}$ are auxiliary variables. The loss function that HQS seeks to minimize is:

$$L=\frac{1}{2}\|L-DH\|_F^2+\mu J(Z)+\lambda Q(Y)+\frac{b}{2}\|Z-H\|_F^2 + \frac{b}{2}\|Y-H\|_F^2 \qquad \textit{Equation 10}$$

where b denotes a penalty parameter. Based on Equation 10 the sequence of individual subproblems emerges:

$$\boldsymbol{H}^{(k+1)}=\underset{\boldsymbol{H}}{\operatorname{argmin}} \frac{1}{2}\left\|\boldsymbol{L}-\boldsymbol{D}\boldsymbol{H}^{(k)}\right\|_F^2+\frac{b}{2}\left\|\boldsymbol{Z}^{(k)}-\boldsymbol{H}^{(k)}\right\|_F^2 + \frac{b}{2}\left\|\boldsymbol{Y}^{(k)}-\boldsymbol{H}^{(k)}\right\|_F^2 \qquad \text{Data consistency}$$

$$\boldsymbol{Z}^{(k+1)}=\underset{\boldsymbol{Z}}{\operatorname{argmin}} \mu J(\boldsymbol{Z})+\frac{b}{2}\left\|\boldsymbol{Z}-\boldsymbol{H}^{(k+1)}\right\|_F^2 \qquad \text{Denoising}$$

$$\boldsymbol{Y}^{(k+1)}=\underset{\boldsymbol{Y}}{\operatorname{argmin}} \lambda Q(\boldsymbol{Y})+\frac{b}{2}\left\|\boldsymbol{Y}-\boldsymbol{H}^{(k+1)}\right\|_F^2 \qquad \text{Outlier removal}$$

Data consistency Module: This subproblem takes the form of a quadratic regularized least squares problem, which can be solved as:

$$\boldsymbol{H}^{(k+1)} = (\boldsymbol{D}^T\boldsymbol{D} + 2b\boldsymbol{I})^{-1}\left(\boldsymbol{D}^T\boldsymbol{Y} + b\boldsymbol{Z}^{(k)} + b\boldsymbol{Y}^{(k)}\right) \quad \textit{Equation 11}$$

Denoising Module: The denoising module refines the range image by processing the output received from the data consistency module as shown in Equation 11. This refinement step can be implemented using a deep learning-based denoising network that functions as follows:

$$\boldsymbol{Z}^{(k+1)} = G_\theta\left(\boldsymbol{H}^{(k+1)}\right) \quad \textit{Equation 12}$$

where G_θ denotes the denoising neural network that learns to identify and preserve the essential characteristics of range images, effectively serving as a learned prior.

The network architecture follows a U-shaped autoencoder design, where the encoder uses 2D convolutional layers to downsample the input, while the decoder employs deconvolutional layers to perform upsampling. However, since it relies on 2D convolutions, it does not inherently preserve 3D geometric consistency, leading to outliers, particularly in regions with depth discontinuities, sharp edges, or occlusions. These artifacts manifest as spurious points and boundary distortions, which negatively impact SLAM performance. To mitigate this issue, we introduce an outlier removal module, described in the following, which ensures the structural integrity of the estimated range image while maintaining real-time performance.

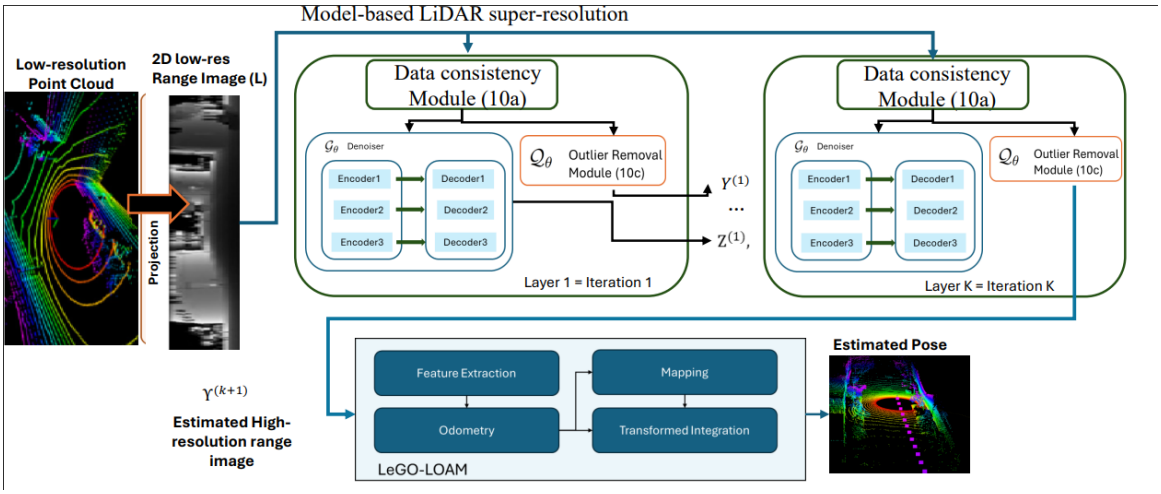


Figure 8: The proposed end-to-end Super-LiDAR-SLAM architecture. The pipeline starts with a low-resolution point cloud captured by a low resolution LiDAR sensor, which is projected into a 2D range image. The 2D range image is then upscaled using the SR module.

Outlier Removal: This module eliminates outliers introduced by the denoising process, refining the estimated range image. The solution corresponds to the proximal operator of the regularizer Q. Instead of using a neural network, we adopt a cluster-based algorithm that operates without learnable parameters. The algorithm processes the structured 2D range image using Breadth-First Search (BFS) to label connected components [7]. It scans the image sequentially, starting from the top-left corner, and initiates BFS for each unlabeled pixel. A neighboring pixel is added to the BFS queue based on a criterion involving range measurements r_1 and r_2 at points p_1 and p_2 measured by a sensor at s. The connectivity angle a_2 is:

$$a_2 = arc\tan\left(\frac{\|sp_2\|\sin a_1}{\|sp_1\| - \|sp_2\|\cos a_1}\right) \qquad \textit{Equation 13}$$

where a_1 is the known angle between laser beams. A threshold θ determines connectivity: if $a_2 < \theta$, the points are in separate clusters due to a significant depth change; otherwise, they belong to the same object. This heuristic yet effective method efficiently segments range images, improving SLAM performance. Final iterative solutions: Hence, the HQS solver consists of three interpretable modules that are the data consistency solution for estimating the high-resolution range image Equation 14, the denoising step in Equation 15 and the outlier removal submodule Equation 16.

$$\boldsymbol{H}^{(k+1)} = (D^T D + b\boldsymbol{I})^{-1}\left(D^T\boldsymbol{Y} + b\left(\boldsymbol{Z}^{(k)} + \boldsymbol{Y}^{(k)}\right)\right) \qquad \textit{Equation 14}$$

$$\boldsymbol{Z}^{(k+1)} = G_\theta\left(\boldsymbol{H}^{(k+1)}\right) \qquad \textit{Equation 15}$$

$$\boldsymbol{Y}^{(k+1)} = Q_\theta\left(\boldsymbol{H}^{(k+1)}\right) \qquad \textit{Equation 16}$$

Model-based SR network: The learnable components within the denoiser enable us to transform the context-aware optimisation problem into an efficient and interpretable deep learning architecture based on established mathematical principles. We accomplish this by implementing the deep unrolling framework. Instead of executing numerous iterations of the HQS solver, we unroll a limited number K of iterations, with each iteration functioning as a distinct layer in the resulting deep network. This creates a K-layer neural network where each layer maps directly to one HQS iteration, creating an optimal balance between computational efficiency, and

model interpretability.

End-to-End Architecture: SR-LeGO-LOAM

Having designed the proposed model-based Super-Resolution (SR) network, we now introduce the complete end-to-end architecture, as illustrated in Figure 8. The proposed architecture extends the Lightweight and Ground-Optimised Lidar Odometry and Mapping on Variable Terrain (LeGO-LOAM) framework by integrating the proposed SR model. The objective of this implementation is to incorporate SR functionality while maintaining the real-time performance of the algorithm. The pipeline begins with a sparse point cloud generated by a low resolution LiDAR sensor, which is transformed into a low-resolution range image. The 2D range image is then upscaled using the SR module, that consists of two denoising processes i.e., the autoencoder denoiser Equation 15 that aims to refines the output of the data-consistency module Equation 14 and the outlier removal module to filter spurious points Equation 16. The remaining steps of the pipeline follow the standard LeGO-LOAM SLAM process.

3.1.4 Numerical Results

Simulation setup

We utilised the ouster LiDAR dataset[1], which captures a 15-minute drive through San Francisco using an OS-1-64 3D LiDAR sensor. The 64-channel LiDAR point clouds were projected into high-resolution range images of size 64 × 1024 as ground truth, while low-resolution images were generated by extracting 16 out of the 64 channels. During training, we used the AdamW optimizer [1] with PyTorch's default settings and unrolled the HQS solver for k = 5 iterations, forming a 5-layer deep learning architecture. The model was implemented in C++ and integrated into the LeGO-LOAM pipeline as a preprocessing step, consistently operating in the 2D image domain. It was tested on an RTX 2080 Ti GPU.

Table 5: Absolute Pose Error (RMSE) w.r.t. translation part (m), FPS, and Number of Parameters

Method	RMSE [m]	FPS	Parameters
LiDAR-16	11.58 / 6717	-	-
SRAE	5.53 / 67.35	5	35M
Simple DU	3.78 / 31.71	5	0.1M
VIT	3.58 / 28.64	1	50M
DU-OR (ours)	**2.35 / 22.64**	**400**	0.2M

[1] *https://github.com/ouster-lidar/yolov5-ouster-lidar-example*

Evaluation study

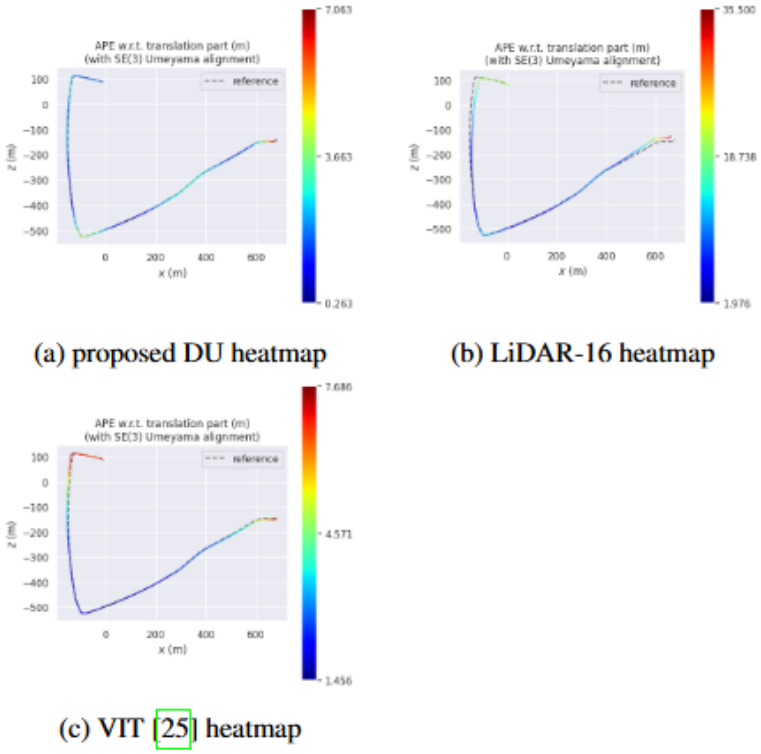
In this section, we present the evaluation results of the proposed SR module with outlier removal using the ouster dataset, which consists of two sequences: a complex 8,000-scan sequence and a smaller 4,000-scan sequence, each sub-sampled to 16 channels. We compare our method Deep Unrolling with Outlier Removal (DU-OR) against state-of-the-art SR LiDAR techniques, including the Super-Resolution AutoEncoder (SRAE) [4], a simple Deep Unrolling (simple DU) SR method [9], and a VIsion Transformer-based (VIT) approach [6]. These methods take the down-sampled ouster sequences as input and generate high-resolution 64-channel scans, acting as a preprocessing step for upscaling low-resolution LiDAR data, which are then used as input to the LeGO-LOAM SLAM pipeline.

Additionally, we also evaluate LeGO-LOAM performance by directly feeding the low-resolution scans (LiDAR-16) without any processing. As a metric, we use the Absolute Pose Error (APE), with the ground truth being the LeGO-LOAM output using the original 64-channel LiDAR. We also report the execution time for each SR technique and the network size in terms of the number of parameters. Table 5 presents the RMSE values for both sequences in a single column, with the results for the 4,000-scan sequence listed first, followed by those for the 8,000-scan sequence, separated by a slash. Figure 9 further illustrates these results through heatmaps for the DU-OR method, the 16-channel LiDAR, and the Transformer-based baseline, using the LiDAR-64 trajectory as reference.

The results highlight the limitations of the 16-channel LiDAR, with an APE RMSE of 11.58m for 4,000 scans, which sharply increases to 6,717m for 8,000 scans, demonstrating the challenges of low-resolution data in complex environments due to the sparsity of point clouds. The proposed DU-OR method consistently outperforms all tested SR approaches, achieving APE RMSE reductions ranging from 34% to 58% for 4,000 scans and 20% to 66% for 8,000 scans. Beyond accuracy, DU-OR excels in real-time performance, achieving 400 fps, compared to 5 fps for SR-AE and 1 fps for VIT, while containing 99% less parameters. The slower speeds of SR-AE and VIT methods result from Monte Carlo-based inference, requiring multiple evaluations per point cloud. In contrast, DU efficiently captures LiDAR structures, reconstructing high-resolution point clouds with a single inference, making it ideal for real-time autonomous applications.

Overall, this section presents a DU-based SR model with an integrated outlier removal module to enhance the accuracy and efficiency of low-resolution LiDAR sensors for SLAM applications. By leveraging a model-based optimisation approach, our method reconstructs high-resolution point clouds while maintaining real-time performance. The proposed SR model was integrated and evaluated within a state-of-the-art LiDAR SLAM framework, demonstrating significant improvements in pose estimation accuracy and efficiency over existing SR methods. These find-

ings highlight the effectiveness of outlier-aware Super-Resolution in improving SLAM performance and its potential extension to other research areas such as object detection.



(a) proposed DU heatmap (b) LiDAR-16 heatmap

(c) VIT [25] heatmap

Figure 9: Heatmaps for the proposed DU, the 16-channel LiDAR and the Transformer based method using as reference trajectory the path derived from the LiDAR-64.

3.1.5 LiDAR super-resolution for Improved Segmentation

Although recent works have increasingly focused on enhancing the resolution of low-cost LiDAR sensors to achieve performance comparable to high-resolution counterparts, existing Super-Resolution (SR) approaches do not explore their impact on perception tasks, such as segmentation [6]. Additionally, these methods operate independently of perception tasks e.g., the segmentation, meaning that the training of SR networks is not influenced by the segmentation process. This independent optimisation approach creates additional challenges when the SR network's output is used as input to a separately trained segmentation network. As a result, the SR methods often produce outputs containing high level of outliers, which heavily degrade segmentation performance. Also, the aforementioned methods fail to preserve details about smaller classes, since the SR process inherently put more emphasis on dominant categories (e.g., buildings). Consequently, the sparse representations of minority classes become distorted, degrading segmentation performance. Hence, optimizing the SR and segmentation networks separately leads to a significant decline in segmentation accuracy, demonstrating the necessity of a joint optimisation approach within an end-to-end framework. Nevertheless, to

achieve a resource-efficient end-to-end architecture, the SR model should be as lightweight as possible to avoid adding computational complexity to the segmentation process. To be more detailed, based on a novel optimisation problem, we design a resource efficient model-based SR network utilizing the deep unrolling strategy [10]. The resulting SR model can be integrated with the segmentation network within a unified system. By jointly optimizing the SR and segmentation net- works during the training phase, the SR model-based network effectively preserves critical contextual details required for accurate segmentation by incorporating semantic information provided by the segmentation network. To the best of our knowledge, this is the first study to propose such an end-to-end framework.

3.1.6 Related Work

LiDAR-based Segmentation

Segmentation methods have been developed, typically categorized into point-based, voxel-based, and projection-based approaches. Point-based methods process raw 3D points directly, preserving spatial structure without transformation of the points utilizing deformable convolutions with an arbitrary number of kernel points to capture local geometric structures [11]. Voxel-based methods tackle the irregularity of 3D point clouds by discretizing the space into uniform voxel grids, allowing for the use of convolutional operations to predict semantic labels [12]. Despite these improvements, voxel-based and point-based methods still suffer from the sparsity of point clouds, leading to redundant calculations and high memory consumption. Projection-based methods transform 3D point clouds into 2D image representations, enabling the use of advanced image feature extraction techniques for semantic segmentation [13]. Methods such as FIDNet [14] and CENet [15] enhance segmentation performance by using ResNet-based encoders and simplifying decoders with interpolation techniques. Similarly, LENet [13], which is a lighweight version of CENet, achieves state-of-the-art results by integrating the Multi-Scale Context Aggregation (MSCA) and Inter-scale Attention Calibration (IAC) modules, which boost the overall performance.

Backbone Segmentation Architecture

Considering computational efficiency and state-of-the-art performance in the 3D segmentation problem, we will focus on projection-based methods operating in the 2D range image domain. In light of this fact, we adopt LENet, as the backbone segmentation network for the proposed end-to-end architecture. In more detail, LENet integrates a multi-scale convolution attention (MSCA) module within its encoder and a lightweight Interpolation And Convolution (IAC) decoder for efficient processing of LiDAR data. The MSCA module consists of three main components: a depth-wise convolution for local information aggregation, multi-branch depth-wise strip convolutions for capturing multi-scale context, and a 1×1 convolution to model inter-channel relationships. The output of the 1×1 convolution acts as attention weights, refining

the input features for enhanced representation. The encoder building block consists of a 3×3 convolution layer followed by the MSCA module. For decoding, LENet incorporates an IAC module, which employs bilinear interpolation for upsampling feature maps from the encoder, a 3×3 convolution to merge information from the encoder and previous IAC stages, and a point-wise convolution to fuse the outputs of the last three IAC modules. The complete encoder-decoder architecture is illustrated in Figure 10.

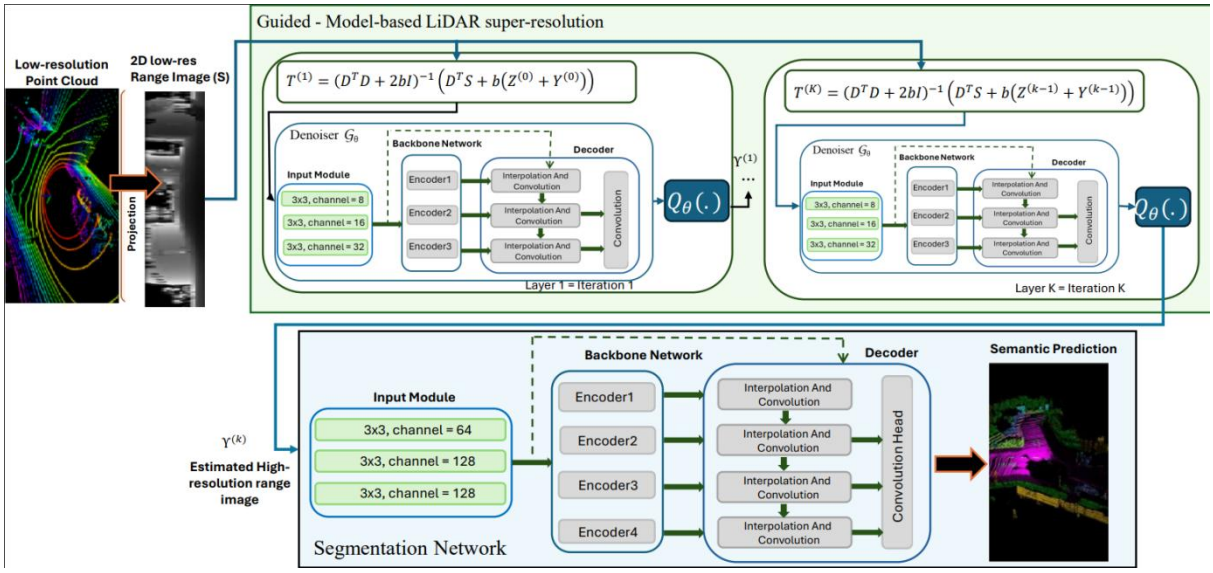


Figure 10: Illustration of the key components of the proposed end-to-end architecture. (a) Segmentation network: The segmentation network employs a hierarchical backbone structure inspired by ResNet34. Its architecture incorporates the IAC module, which progressively upsamples low-resolution feature maps to their original dimensions while integrating outputs from preceding IAC modules. (b) Guided Model-based SR network: A small number of iterations of the solver in are unrolled and treated as a deep learning architecture, consisting of the data-consistency solution Equation 29, the denoiser Equation 30 and the segmentation-guided regularization using the learnable mask Equation 31. (c) Overall end-to-end architecture: A 16-channel LiDAR point cloud is converted into a low-resolution range image, processed by the SR network to generate a high-resolution range image. This high-resolution output is fed into the segmentation network for final 3D segmentation. By jointly optimizing the SR and segmentation networks, the SR process benefits from critical semantic guidance, thus enhancing the segmentation performance

3.1.7 Proposed Methodology

In this section, we present the proposed end-to-end architecture.

Guided Model-based SR

To construct the proposed model-based deep learning architecture, we leverage the advantages of projection-based methodologies. Specifically, we project the 3D high-resolution point cloud $\boldsymbol{P}$ obtained from a 64-channel LiDAR sensor, into a high-resolution range image $\boldsymbol{T} \in R^{64\times N}$. In view of this, the high-resolution data and the low-resolution data sensor are connected as follows:

$$\boldsymbol{S} = \boldsymbol{DT} + \boldsymbol{E} \qquad \textit{Equation 17}$$

where $\boldsymbol{D} \in R^{16\times 64}$ is the downsampling operator that selects the 16 channels from the high resolution range image and $\boldsymbol{E}$ is a term, respectively. A simple optimisation to enhance the resolution of the range image is:

$$\underset{\boldsymbol{T}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{S} - \boldsymbol{DT}\|_F^2 + \mu J(\boldsymbol{T}) \qquad \textit{Equation 18}$$

where the first component ensures consistency with the degradation model defined in Equation 18. The second component $J(.)$ serves as a learnable regularizer, designed to capture the intrinsic features of the high-resolution range image T.

However, the above formulation remains agnostic to segmentation problem, resulting in a lack of influence from the segmentation results during the training and operational phase. This limitation can lead to the loss of structural details for smaller or underrepresented classes. To address this limitation, we introduce an additional regularization term in the optimisation problem that leverages a learnable mask to guide the super-resolution process. This learnable mask is trained using the ground truth segmentation masks during training, ensuring a focus on regions associated with underrepresented classes. The segmentation-guided optimisation problem is defined as:

$$\underset{\boldsymbol{T}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{S} - \boldsymbol{DT}\|_F^2 + \mu J(\boldsymbol{T}) + \lambda Q(\boldsymbol{T}) \qquad \textit{Equation 19}$$

where $\mathrm{Q}(.)$ represents an extra regularizer that guides the estimated high-resolution output to preserve the structure of the classes of interest for the segmentation task.

To tackle the proposed optimisation problem in Equation 19, we employ the Half Quadratic Splitting (HQS) methodology. The problem can be reformulated as follows:

$$\underset{\boldsymbol{T}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{S}-\boldsymbol{D}\boldsymbol{T}\|_F^2+\mu J(\boldsymbol{T})+\lambda Q(\boldsymbol{Y}) \quad s.t.\ \ \boldsymbol{T}=\boldsymbol{Z}, \boldsymbol{T}=\boldsymbol{Y} \qquad \textit{Equation 20}$$

where $\boldsymbol{Z} \in R^{64\times N}, \boldsymbol{Y} \in R^{64\times N}$ are auxiliary variables. The loss function that HQS aims to minimize is:

$$L=\frac{1}{2}\|\boldsymbol{S}-\boldsymbol{D}\boldsymbol{T}\|_F^2+\mu J(\boldsymbol{Z})+\lambda Q(\boldsymbol{Y})+\frac{b}{2}\|Z-T\|_F^2+\frac{b}{2}\|Y-T\|_F^2 \qquad \textit{Equation 21}$$

where b denotes a penalty parameter. Based on Equation 21 a sequence of individual subproblems emerges, that are analysed in the following:

$$\boldsymbol{T}^{(k+1)}=\underset{\boldsymbol{T}}{\operatorname{argmin}} \frac{1}{2}\left\|\boldsymbol{S}-\boldsymbol{D}\boldsymbol{T}^{(k)}\right\|_F^2+\frac{b}{2}\left\|\boldsymbol{Z}^{(k)}-T^{(k)}\right\|_F^2+\frac{b}{2}\left\|\boldsymbol{Y}^{(k)}-\boldsymbol{T}^{(k)}\right\|_F^2 \qquad \textit{Equation 22}$$

$$\boldsymbol{Z}^{(k+1)}=\underset{\boldsymbol{Z}}{\operatorname{argmin}}\, \mu J(\boldsymbol{Z})+\frac{b}{2}\left\|\boldsymbol{Z}-\boldsymbol{T}^{(k+1)}\right\|_F^2 \qquad \textit{Equation 23}$$

$$\boldsymbol{Y}^{(k+1)}=\underset{\boldsymbol{Y}}{\operatorname{argmin}}\, \lambda Q(\boldsymbol{Y})+\frac{b}{2}\left\|\boldsymbol{Y}-\boldsymbol{T}^{(k+1)}\right\|_F^2 \qquad \textit{Equation 24}$$

Data consistency Module: The closed form solution is given as:

$$\boldsymbol{T}^{(k+1)}=(\boldsymbol{D}^T\boldsymbol{D}+2b\boldsymbol{I})^{-1}\left(\boldsymbol{D}^T\boldsymbol{Y}+b\boldsymbol{Z}^{(k)}+b\boldsymbol{Y}^{(k)}\right) \qquad \textit{Equation 25}$$

Denoising Module: The purpose of the denoising module is to produce a refined range image by taking as input the estimated range image from the data consistency module in Equation 25. Thereby, this step can be implemented using a deep learning-based denoiser as follows:

$$\boldsymbol{Z}^{(k+1)} = G_\theta\left(\boldsymbol{T}^{(k+1)}\right) \qquad \textit{Equation 26}$$

where G_θ denotes the neural network responsible for denoising. For the architecture of the deep learning-based denoiser, we draw inspiration from the LENet segmentation network, as illustrated in Figure 10.

Segmentation Guidance: Although this subproblem originally involves the variables Y and T, we substitute T with Z, which represents the denoised version of T obtained from the previous step. This substitution provides a cleaner and more precise representation. The solution of this problem can be replaced again with a learnable function. To this end, we introduce a learnable mask $\mathrm{Q}_\theta(\cdot)$ to incorporate segmentation guidance across both the training and inference phases, based on the solution $\boldsymbol{Z}^{(k+1)}$ from Equation 26. During training, the segmentation mask derived from the ground truth (M_{GT}) supervises the learning of $\mathrm{Q}_\theta(\cdot)$, ensuring accurate segmentation guidance. The loss for the learnable mask is:

$$L_{mask} = \frac{1}{N}\sum_{j=1}^{N}\left\| Q_\theta\left(Z^{(k+1)}(j)\right) - w_{c(j)} M_{\mathrm{GT}}(j)\right\|_1, \qquad \textit{Equation 27}$$

where N is the number of pixels, $Z^{(k+1)}$ is the input at iteration $k+1$, and M_{GT} is the ground truth segmentation mask. The $c(j)$ is class label of pixel j, as indicated by the segmentation mask M_{GT}. The weight $w_{c(j)} = \sqrt{\frac{1}{f_{c(j)}}}$, derived from the class frequency $f_{c(j)}$, ensures higher emphasis on underrepresented classes. During inference, where M_{GT} is unavailable, the learned neural network $\mathrm{Q}_\theta(\cdot)$ generates the mask based on the input $Z^{(k+1)}$ with dimension 64×1024. This estimated mask is then used to enhance context-awareness by applying a pixel-wise multiplication with $Z^{(k+1)}$, ensuring that segmentation guidance is explicitly incorporated. The updated subproblem is formulated as:

$$\boldsymbol{Y}^{(k+1)} = Q_\theta\left(\boldsymbol{Z}^{(k+1)}\right) \odot \boldsymbol{Z}^{(k+1)} \qquad \textit{Equation 28}$$

Final Iterative solutions: Hence, the HQS solver consists of three modules that is the data con-

sistency solution for estimating the high-resolution range image Equation 29, the denoising step in Equation 30 and the learnable segmentation mask that guides the output to preserve critical structure details Equation 31.

$$\boldsymbol{T}^{(k+1)} = (D^T D + b\boldsymbol{I})^{-1}\left(D^T \boldsymbol{S} + b\left(\boldsymbol{Z}^{(k)} + \boldsymbol{Y}^{(k)}\right)\right) \quad \textit{Equation 29}$$

$$Z^{(k+1)} = \boldsymbol{G}_\theta\left(\boldsymbol{T}^{(k+1)}\right) \quad \textit{Equation 30}$$

$$\boldsymbol{Y}^{(k+1)} = Q_\theta\left(\boldsymbol{Z}^{(k+1)}\right) \odot \boldsymbol{Z}^{(k+1)} \quad \textit{Equation 31}$$

Since the denoiser contains learnable parameters, the solutions of the proposed optimisation problem can be reformulated into a computationally efficient architecture. To achieve this, we adopt the Deep Unrolling (DU) framework. We unroll only K iterations and treat each one as a single layer in the resulting deep learning model. The proposed model-based SR network is illustrated in Figure 10 along with the overall end-to-end architecture that we analyse in the next Section.

End-to-End Architecture

Having designed the proposed model-based SR network with low computational complexity, we now introduce the complete end-to-end architecture, as illustrated in Figure 10. The process begins with a low-resolution point cloud captured by a 16-channel LiDAR sensor, which is converted into a low- resolution range image. This range image is then fed into the model-based SR network, to generate a high-resolution range image. The high-resolution range image is subsequently passed to the segmentation network for final 3D semantic segmentation. By jointly optimizing the SR and segmentation networks, the end-to-end architecture incorporates semantic context into the SR process, enhancing the quality of the super-resolved data.

A key element of the proposed architecture is the end-to-end training of the SR and segmentation networks.

Context-Aware LiDAR Loss: To enhance the ability of the SR network to preserve the structure of smaller classes, we introduce a context-aware loss function. This loss function lever-

ages the ground truth segmentation masks to guide the SR model's focus, defined as:

$$L_{sr} = \sum_{i=1}^{p} \sum_{j=1}^{N} w_{c(j)} \left\| \left(Y_i^{(K+1)}(j) \right) - T_i(j) \right\|_1, \qquad \textit{Equation 32}$$

where p denotes the number of range images and N the number of pixels per image. The weight $w_{c(j)}$ is defined similar to eEquation 27. $Y_i^{(K+1)}(j)$ is the predicted pixel j in the i-th image, while $T_i(j)$ is the real value.

Segmentation Loss: A cross-entropy loss L_{wce} is employed defined as:

$$L_{wce}(y, \hat{y}) = -\sum_{i} a_i \cdot p(y_i) \log\left(p(\hat{y}_i)\right) \qquad \textit{Equation 33}$$

where y_i and $\hat{y}_i$ represent the ground truth and predicted class labels, respectively. The total loss is given by:

$$L = w_1 L_{wce} + w_2 L_{ls} + w_3 L_{sr} + w_4 L_{mask} \qquad \textit{Equation 34}$$

where L_{mask} is defined in Equation 27, the w_1, w_2, and w_3 are weights set empirically to $w_1 = 1$, $w_2 = 1.5$ and w_3, $w_4 = 1$, respectively.

3.1.8 Numerical results

Simulation setup

Dataset: We utilise two widely used LiDAR segmentation benchmarks: the SemanticKITTI dataset [16] and the SemanticPOSS dataset [17]. SemanticKITTI is a large- scale dataset collected using the Velodyne HDL-64E LiDAR sensor, specifically designed for point cloud segmentation in autonomous driving scenarios. It is split into three subsets: sequences 00–04 for training, se-

quence 05 for validation, and sequences 06–08 for testing. SemanticPOSS, uses a PANDORA 40 channel LiDAR sensor. For SemanticKITTI, we project point clouds from the 64-channel sensor into high- resolution range images of size 64×1024, which serve as ground truth. Corresponding low-resolution range images of size 16×1024 are generated by selecting 16 out of the 64 channels, simulating data from a low-cost 16-channel LiDAR. A similar process is applied to the SemanticPOSS dataset, producing high-resolution range images of size 40×1024 and low-resolution versions of size 10×1024.

Implementations Details: For the proposed model-based SR network, we unroll the derived HQS solver in Equation 29-Equation 31 for k = 4 iterations, resulting in a 4-layer deep learning architecture. In the end-to-end framework, we use the AdamW optimizer with default optimizer with default PyTorch settings. The initial learning 2×10^{-3} and dynamically adjusted using a cosine annealing scheduler over 80 epochs. Also, we utilise the mean Intersection over Union (mIoU) metric to evaluate the performance. Finally, all experiments were conducted on an NVIDIA RTX 4090 GPU.

Evaluation study

Table 6 and

Table 7 show the quantitative results on the SemanticKITTI and SemanticPOSS benchmarks for the LeNET segmentation network across different configurations: using the high-resolution LiDAR sensor, the low-resolution LiDAR sensor, the state-of-the-art (SOTA) Transformer-based LiDAR SR method combined with the LeNET segmentation net- work, and the proposed end-to-end architecture.

*(a) **Impact of Resolution on Segmentation performance**: The results in Table 6 and*

Table 7 clearly demonstrate how LiDAR resolution significantly influences segmentation performance. Using low-resolution LiDAR data leads to a noticeable decline in segmentation accuracy. The reduced point density in the low-resolution data results in poorer segmentation. For example, in KITTI dataset [2] the bicycle class shows a substantial drop of 66%, while the person class experiences a significant decrease of 74%.

(b) ***Independent Training with SOTA Transformer-Based SR*** [6]: This scenario utilises a SOTA LiDAR SR model based on the Swin Transformer architecture, independently trained alongside the segmentation network. The results indicate that while this framework improves segmentation performance for dominant classes, such as cars, it fails to deliver similar benefits for smaller classes like bicycles. The SR model struggles to reconstruct key regions necessary for accurate segmentation of these smaller classes, as the lack of joint optimisation between the SR and segmentation networks limits the ability to preserve their details. Furthermore, the Transform-

er-based model contains over 50 million parameters, significantly surpassing the complexity of the segmentation network. This imbalance in complexity highlights the challenges of integrating such models into an end-to-end framework.

(c) ***Proposed End-to-End Framework***: The proposed end-to-end framework achieves superior results with a significant 99% reduction in parameters compared to the Transformer-based SR model. This is attributed to the lightweight design of the proposed model-based SR framework, making it well-suited for real-time applications. Our model achieves 23 fps, outperforming the 6 fps of baseline method, as we can see in Table 6. Also, the framework delivers segmentation performance that is on par with a segmentation network utilizing high-resolution data from a high-cost LiDAR sensor.

Impact of the Proposed Context-Aware Loss Function: The results in Table 8 highlight the impact of the context-aware SR loss on segmentation performance for smaller or underrepresented classes. It is evident that the proposed loss guides the SR model to preserve finer structural details critical for accurate segmentation.

Different Segmentation Architectures: We adopt LeNet as the primary segmentation backbone due to its strong performance. However, our method is architecture-agnostic and can be integrated with other range-view segmentation networks such as CENet and FIDNet. As shown in Table 9, the consistent performance across architectures demonstrates the generalizability of our approach.

Overall**,** this section introduced an end-to-end framework for LiDAR SR and semantic segmentation, addressing the limitations of existing methods that often fail to integrate SR and segmentation tasks effectively. Using a novel joint optimisation process and a carefully designed context-aware SR loss function, the proposed framework achieves high segmentation accuracy, particularly for the underrepresented classes, while maintaining computational efficiency. The proposed lightweight LiDAR SR network reduces complexity by 99% compared to state-of-the-art SR models, enabling its seamless integration into an end-to-end system. Experimental results highlight that the framework delivers segmentation performance similar to that of high-resolution LiDAR data, demonstrating its potential to bridge the gap between affordable low-resolution sensors and high-performance perception systems.

Table 6: Performance comparison on SemanticKITTI benchmark

	SR Params (M)	Segmentation Params (M)	FPS	Car	Bicycle	Motorcycle	Truck	Person	Road	Parking	Building	Fence	Trunk	Terrain	Traf-Sign
LiDAR 64-channel	-	4.7	27	0.97	0.38	0.72	0.61	0.47	0.94	0.62	0.89	0.63	0.64	0.65	0.48
LiDAR 16-	-	4.7	29	0.81	0.1	0.35	0.3	0.12	0.8	0.54	0.73	0.47	0.4	0.54	0.2

	SR Params (M)	Segmentation Params (M)	FPS	Car	Bicycle	Motorcycle	Truck	Person	Road	Parking	Building	Fence	Trunk	Terrain	TrafSign
channel					3		7		4				8		2
Transformer + LeNET	50	4.7	6	0.88	0.07	0.40	0.39	0.16	0.89	0.53	0.81	0.49	0.50	0.58	0.24
Proposed End-to-End	**0.1**	4.7	23	**0.90**	**0.37**	**0.60**	**0.60**	**0.44**	**0.93**	**0.59**	**0.83**	**0.58**	**0.56**	**0.63**	**0.45**

Table 7: Performance comparison on SemanticKITTI benchmark

	Person	Rider	Car	Trunk	Plants	Traffic sign	Pole	Building	Bike	Ground
LiDAR 40-channel	0.76	0.24	0.77	0.67	0.74	0.54	0.32	0.81	0.53	0.80
LiDAR 10-channel	0.52	0.05	0.43	0.25	0.66	0.15	0.28	0.72	0.34	0.70
Transformer + LeNET	0.61	0.07	0.56	0.26	0.68	0.08	0.29	0.73	0.40	0.72
Proposed	**0.69**	**0.16**	**0.75**	**0.63**	**0.72**	**0.53**	**0.31**	**0.76**	**0.43**	**0.75**

Table 8: Impact of the proposed context-aware SR loss on segmentation performance - SemanticKITTI

Class	Without Context-Aware SR Loss	With Context-Aware SR Loss	Improvement (%)
Person	0.36	0.44	+22.22
Bicycle	0.30	0.37	+23.33
Traffic Sign	0.39	0.45	+15.38

Table 9: Impact of different segmentation architectures

	IoU avg
Proposed + LeNET	**0.573**
Proposed + CENET	0.567
Proposed + FIDNET	0.558

3.2 Radar based environment perception

Radar sensors are a cornerstone of reliable surround sensing, employed across diverse fields such as agriculture, aviation, robotics, and critically, the automotive industry. Unlike optical

sensors, radar technology offers inherent advantages, including insensitivity to varying lighting conditions and adverse weather. Furthermore, radar uniquely provides accurate measurements of both relative velocity and range to detected objects. Historically, traditional radar systems exhibited lower information density and resolution compared to benchmark technologies like LiDAR, a difference particularly noticeable in point cloud representations.

However, recent advancements have led to the emergence of **imaging radars**, which significantly overcome these limitations. Through sophisticated concepts such as multi-chip cascading, multiple-input and multiple-output (MIMO) processing, and advanced signal processing methods, imaging radars have achieved substantial improvements in resolution and information density [18]. These innovations are expanding the application scope within the radar domain, positioning imaging radars as independent and primary sensors, rather than mere complements to cameras or LiDARs, especially for applications requiring robust perception in challenging environments.

Waveye's "ultra-high resolution radar imaging" boasts an angular resolution of 0.5° in azimuth and elevation, capable of generating over 5000 detections in a typical urban scene. This high-resolution radar data serves as the foundation for establishing a robust object detection and classification pipeline, crucial for advanced perception tasks, particularly in the context of autonomous driving and human interaction.

3.2.1 4D Imaging Radar Data and Point Cloud Characteristics

The input to the perception pipeline is a point cloud, based on radar detections, critically provided by a 4D imaging radar. The imaging radar provides the following 4 dimensions, leading to the name “4D”: range, azimuth angle, elevation angle, and relative velocity (Doppler). These four intrinsic measurements enable a comprehensive understanding of detected objects' positions and dynamics.

Each detection point thus carries rich, multi-dimensional information. From these primary measurements and the intensity of the reflection, additional valuable features can be derived or are also part of the point cloud data:

- **Range:** The direct distance of the detection from the radar sensor.
- **Azimuth Angle (ϕ):** The horizontal angle of the detection relative to the radar's boresight.
- **Elevation Angle (θ):** The vertical angle of the detection relative to the radar's boresight.
- **Relative Velocity:** The speed of the detected point towards or away from the radar.

- **X, Y, Z coordinates:** The precise spatial location of the detection in three dimensions, derived from range, azimuth, and elevation angles.
- **Radar Cross Section (RCS):** Indicating the reflectivity of the object, which can provide clues about its material and size.
- **Signal to Noise Ratio (SNR):** A measure of the signal strength relative to background noise, indicating detection quality.

These comprehensive features, coupled with synchronised camera images, form the basis for subsequent perception tasks. The camera images primarily serve to facilitate the accurate labelling of radar data during ground truth generation.

Figure 11 illustrates an example of the input data from a typical scenario within the dataset. The left panel displays a synchronised camera image, which roughly covers the same area of interest as the radar. The right panel displays the point cloud data, where the single detections are colored by range. One can perceive the robot and the human in the point cloud data.

Figure 11: Point cloud data provided by the imaging radar visualized in Foxglove Studio

3.2.2 Clustering: Grouping Radar Detections

Following the acquisition of raw 4D imaging radar data, the initial crucial step in the perception pipeline is **point clustering**. This process involves grouping individual radar detections into coherent clusters, with the primary objective that each cluster ideally corresponds to a distinct physical object present within the radar's field of view. This is crucial for:

- **Robust Object Feature Extraction:** This allows for the extraction of aggregated features (e.g., cluster size, centroid velocity, orientation) that are vital for subsequent object

classification and tracking stages, enabling a comprehensive understanding of object behavior and presence.

- **Simplified Object Representation:** By consolidating numerous individual detections into fewer, meaningful clusters, the complexity of the raw data is reduced. This simplified representation facilitates more efficient downstream processing for object recognition and classification.

The inherent complexity of real-world scenarios, including varying surface reflectivity and dynamic object density, imposes specific requirements on clustering algorithms. According to [19], effective clustering solutions for radar point clouds must satisfy the following criteria:

- **Invariance to Object Count:** The algorithm should not require prior knowledge of the number of objects in the scene.
- **Accurate Object Representation:** Clusters should ideally contain only points truly belonging to a single physical object, minimizing extraneous points or merging distinct objects.
- **Robustness to Density Fluctuations:** The ability to handle varying point densities across different spatial locations is crucial, as detection density can differ for objects at various ranges.
- **Computational Feasibility:** The algorithm must be efficient enough for real-time or near real-time processing in automotive systems.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [20] is a prominent algorithm that fundamentally satisfies most of these requirements, making it well-suited for point cloud processing in radar and LiDAR domains. A slightly modified version of the DBSCAN algorithm is used in this perception pipeline.

3.2.3 Bounding Box Estimation

Building upon the clustered radar detections, the next logical step in the perception pipeline is the **estimation of bounding boxes**. This process encapsulates each identified point cluster within a geometrically defined shape, typically a cuboid, thereby providing a concise and interpretable representation of the detected object. Bounding box fitting serves two primary purposes:

- **Enhanced Visualisation and Intuition:** Bounding boxes significantly improve the visual clarity of objects detected within the radar point cloud, making it easier for human observers and visualisation tools to understand the scene. This is particularly valuable for validating system performance and understanding object presence, such as pedestrians.

- **Compact Object State Representation:** The parameters of a fitted bounding box (e.g., center position, dimensions, and orientation) offer a compact and informative representation of an object's spatial state. This aggregated information is a more robust input for subsequent stages like object tracking, as it abstracts away individual point-level noise and provides stable features for motion estimation.

For estimating these bounding boxes from point clouds, algorithms often leverage methods such as the L-shape based detection, which has proven effective in both LiDAR and radar domains due to the similar nature of point cloud representation. These algorithms work by fitting an optimal rectangular or cuboid shape that encompasses the clustered points, often by minimizing discrepancies between points and the fitted shape's edges. Optimisation criteria typically focus on geometric properties like area or volume minimization and maximizing the fit to the clustered points.

The output of such a function typically includes:

- **Center:** The Cartesian coordinates (X, Y, Z) of the cuboid's central point.
- **Dimensions:** The lengths (length, width, height) of the cuboid along its principal axes.
- **Orientation:** The yaw angle defining the cuboid's pose in space.

This structured bounding box information is crucial for feeding into downstream modules like object trackers and classifiers, providing a higher-level abstraction of the scene content. Figure 12 illustrates examples of fitted bounding boxes over clustered radar detections. The different colors show the object category of the underlying clustered group of points.

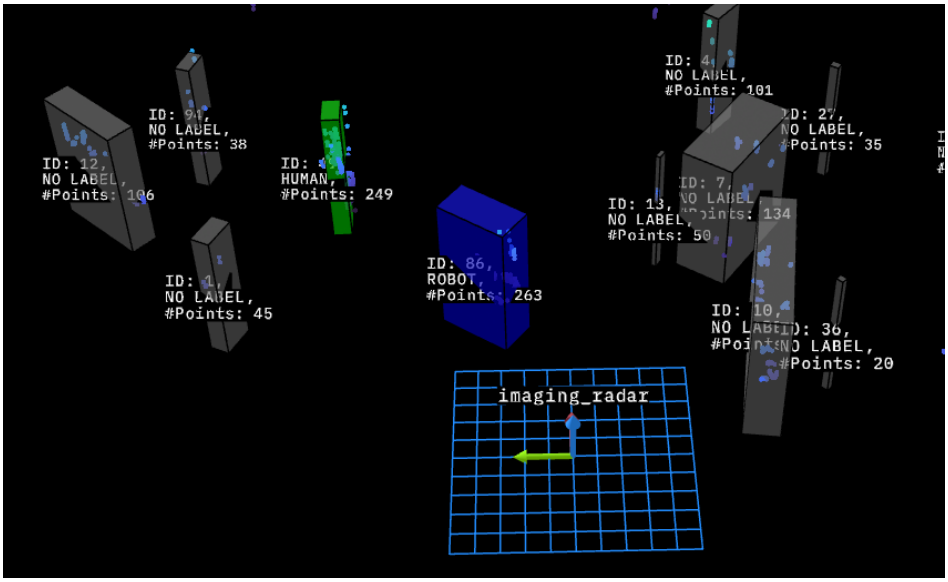


Figure 12: Visualisation of the bounding boxes fitted to the clusters

3.2.4 Object Tracking: Understanding Dynamic Environments

Following the clustering of raw radar detections and the estimation of bounding boxes, the next critical stage in the perception pipeline is **Multi-Object Tracking (MOT)**. While clustering provides a snapshot of objects at a single moment, tracking establishes their temporal continuity. This means identifying which bounding box in the current frame corresponds to the same physical object detected in previous frames.

Object tracking is fundamental for understanding the dynamic behavior of objects, particularly humans, in complex environments like the automotive domain. It allows autonomous systems to:

- **Predict Future States:** By understanding an object's past trajectory, its future position and velocity can be estimated, enabling safe path planning and interaction. This is crucial for anticipating movements of pedestrians or other vehicles.
- **Maintain Object Identity:** Assigning a unique ID to each persistent object allows for consistent observation and analysis over time, even if detections are sporadic or occluded momentarily.
- **Simplify Downstream Tasks:** A stable track provides a more reliable input for subsequent decision-making processes than fluctuating raw detections or single-frame clusters.

3.2.5 State Estimation with the Kalman Filter

A core component of any object tracking system is the ability to estimate and predict an object's state (e.g., position, velocity) in the presence of sensor noise. Radar measurements, like all sensor data, are inherently subject to uncertainties stemming from various sources, such as thermal noise or environmental reflections. To mitigate these effects and achieve lower uncertainties in state estimation, the **Kalman Filter (KF)** [21] is widely employed.

The KF is a recursive algorithm that provides an optimal estimate of a system's state by fusing predictions from a motion model with noisy measurements. It operates in two main steps:

- **Prediction Step:** Based on the object's estimated state from the previous time step and a defined motion model (e.g., constant velocity), the filter predicts the object's current state and its associated uncertainty. This step effectively forecasts where the object *should* be.
- **Update Step:** When a new measurement becomes available, the filter combines this measurement with the predicted state. The Kalman gain, a key parameter, determines the weight given to the new measurement versus the prediction. This balance allows

the filter to refine its state estimate, reducing uncertainty and correcting predictions based on actual observations.

In the underlying perception pipeline, we use a constant velocity motion model.

The state of an existing track is represented by the following vector:

$$x^{\mathrm{T}} = [x, v_{\mathrm{x}}, y, v_{\mathrm{y}}, z, v_z, \theta, l, w, h] \qquad \textit{Equation 35}$$

Measurements consist of the estimated bounding boxes from the bounding box fitting and have the following format:

$$z^{\mathrm{T}} = [x, y, z, \theta, l, w, h] \qquad \textit{Equation 36}$$

Where:

- **x, y, z**: Cartesian coordinates of the bounding box center point
- **v_x, v_y, v_z**: Estimated velocities of the center point in the respective dimension
- **θ**: Yaw angle of the bounding box
- **l, w, h**: Box dimensions: length, width, height

3.2.6 Data Association with Joint Probabilistic Data Association

A fundamental challenge in Multi-Object Tracking (MOT) is **data association**: determining which new measurements (i.e., newly detected bounding boxes) correspond to which existing tracks, or if they represent a new object or clutter. This problem becomes particularly complex when multiple objects are in proximity or when detections are sparse.

The process typically begins with **gating**, where a region is defined around each track's predicted position. Only measurements falling within this 'gate' are considered potential candidates for association with that track, significantly reducing computational load [22].

Figure 13 shows the data association problem, where the red triangle and rectangle show the predicted states of the tracks at a certain timestep. Around these predicted positions the gating regions are spanned. The black points are showing measurements.

Following gating, an **assignment algorithm** decides how to link measurements to tracks. While simpler methods like Global Nearest Neighbor (GNN) assign each track to its single closest measurement, this can lead to errors in dense or crossing scenarios. For robust tracking in such environments, more sophisticated multi-hypothesis algorithms are often preferred.

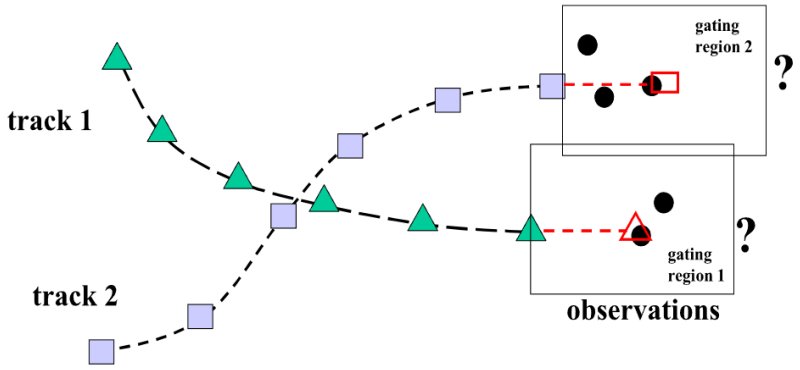


Figure 13: Illustration of the data association problem and the gating principle [23]

The **Joint Probabilistic Data Association (JPDA) algorithm** is a multi-hypothesis approach that strikes a good balance between computational complexity and performance in challenging scenarios, particularly where objects are close to each other. Instead of making a hard assignment, JPDA considers *all* valid measurements within a track's gate. It calculates a probability for each possible association (including the possibility of no association or clutter) and then combines these weighted probabilities to form a single, more robust update for the track's state. This allows the filter to incorporate information from multiple potential measurements, making it more resilient to ambiguities arising from closely spaced objects, such as multiple pedestrians in a crowd.

The robust object tracking provided by a KF coupled with JPDA is essential for reliably monitoring human activity, enabling critical functions like collision avoidance.

3.2.7 Object Classification: Categorizing Perceived Entities

Following the clustering of radar detections and the subsequent estimation of bounding boxes, the next critical step in the perception pipeline is **object classification**. This process assigns semantic labels (e.g., "human," "vehicle," "robot") to the detected and tracked objects, enabling the autonomous system to understand the nature of its surroundings and interact appropriately. For instance, distinguishing a pedestrian from a static pole is paramount for safe navigation and decision-making in automotive applications.

Given that object localisation and tracking are already handled by the preceding pipeline stages, the classifier's input consists of point cloud data grouped by clusters and associated with tracks.

3.2.8 PointNet Fundamentals

PointNet [24] often serves as the foundational architecture for directly processing unordered point clouds. Unlike Convolutional Neural Networks (CNNs) that require structured grid data

(e.g., images), PointNet can directly ingest a set of points (each with its features like X, Y, Z coordinates, velocity, RCS, SNR). Its key innovation lies in using a **max pooling layer**, which acts as a symmetric function. This enables the network to achieve **permutation invariance** – meaning the order of input points does not affect the output. This property is crucial for point clouds, where points inherently lack a fixed order.

The PointNet architecture as shown in Figure 14 first applies shared Multi-Layer Perceptrons (MLPs) independently to each input point to extract point-wise features. After a feature transformation to align spatial dependencies, another series of MLPs leads to the global max pooling layer. This layer aggregates all point features into a single global feature vector, which then passes through dense layers to produce classification scores.

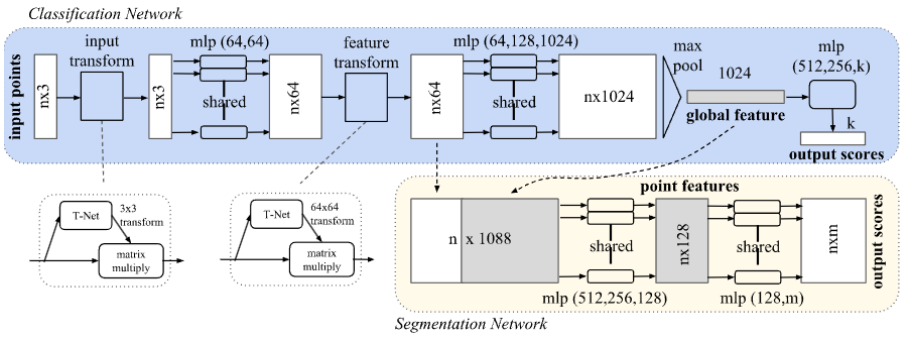


Figure 14: Illustration of the PointNet architecture [24]

3.2.9 PointNet++ Architecture

PointNet++ [25] extends PointNet by addressing its limitation in capturing local structures. While PointNet extracts global features, PointNet++ incorporates a hierarchical architecture that progressively learns features at different scales, enabling it to model local point neighborhoods.

As shown in Figure 15, PointNet++ operates through multiple **Set Abstraction (SA) modules**. Each SA module performs three key operations:

- **Sampling:** Selects a subset of points (centroids) from the input point cloud. Farthest point sampling is commonly used for uniform coverage.
- **Grouping:** For each sampled centroid, it groups neighboring points within a defined radius. A standard ball-query algorithm is typically employed here.
- **PointNet Feature Encoding:** A mini-PointNet instance is applied to the grouped points to extract local features for that neighborhood.

This iterative process of sampling, grouping, and feature encoding within SA modules effectively downsamples the point cloud while enriching the features of the remaining points, allowing the network to capture contextual information from local regions. The features learned at different levels of the hierarchy are ultimately aggregated for final classification.

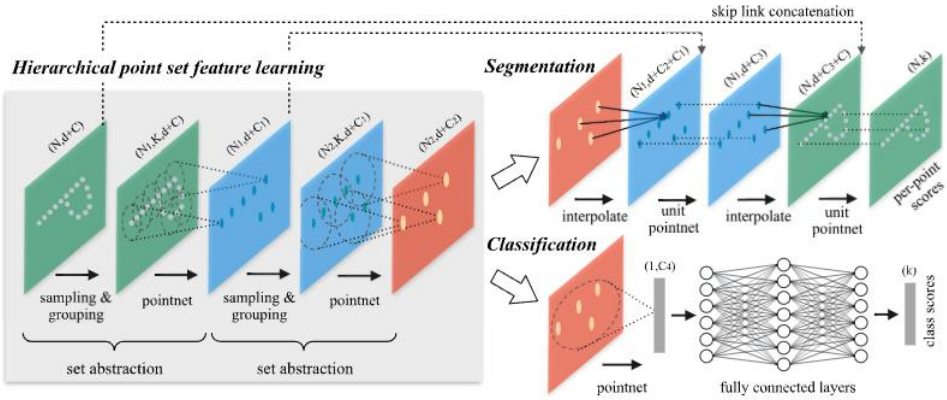


Figure 15: Illustration of the PointNet++ architecture [25]

For this study, the PointNet++ model was adapted to leverage the rich 4D features available from radar data, specifically including Cartesian coordinates (X, Y, Z), relative velocity, Radar Cross Section (RCS), and Signal-to-Noise Ratio (SNR) as input for each point. This multi-dimensional input significantly enhances the network's ability to differentiate between object classes. Input data undergoes normalization to ensure consistent feature distributions and account for varying cluster sizes.

3.2.10 Numerical results

This section provides insights into the experimental results and key findings observed during the development and evaluation of the perception pipeline, encompassing aspects of object formation, tracking, and classification.

3.2.11 Object Formation and Tracking Process Findings

The object formation and tracking methodology demonstrates robust performance with minimal susceptibility to clutter, effectively grouping radar detections into clusters and tracking them over time. This provides a consistent representation of objects in the scene.

A notable finding relates to the variability in object representation within point cloud data. Depending on an object's geometry and orientation relative to the radar, its reflections can sometimes be segmented into multiple distinct clusters, e.g. for lift truck objects. While this poses a challenge for consistent object representation, the robust tracking system is designed to handle such complexities and maintain track continuity.

Furthermore, the detectability of objects can be influenced by their material properties and surface orientation. For instance, flat, metallic surfaces may deflect radar waves away from the sensor, leading to intermittent or temporarily missing detections. Critically, the tracking algorithm proves resilient in these scenarios, capable of bridging short-term detection gaps and maintaining track continuity. If detections are absent for an extended period, tracks are appropriately deleted and re-initialized upon re-detection.

The overall ability of the tracking system to maintain persistent object identities for dynamically moving entities like humans, even under challenging conditions and with varying object appearances, is a strong foundation for the subsequent classification task. This reliable tracking performance for human actors is paramount for safety-critical applications in complex automotive environments.

3.2.12 Classifier Performance

The PointNet++ classifier, chosen for its ability to directly process unordered point cloud data, was evaluated using a dedicated test dataset comprising tracks entirely excluded from the training data, ensuring unbiased performance assessment. The split is 70/10/20 between training, validation and test datasets.

The primary metric for comparison was **Balanced Accuracy**, which is particularly relevant for datasets with class imbalances, as it averages the True Positive Rates of all classes (diagonal entries in the row-normalized confusion matrix). The current results are obtained for human vs. non-human classification, whereas other classes such as cars, trucks, motorcycles, overridable and underridable objects will be added over the course of the project.

A detailed analysis of the single snapshot confusion matrix, as depicted in Figure 16, provides deeper insights into the classification performance for human perception. The current perception pipeline achieves 89.3% human classification performance in a single snapshot. When combined over multiple snapshots, this performance can be significantly improved and reaches over 96% in less than half a second.

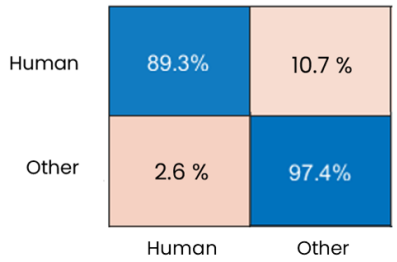


Figure 16: Classification performance between human and other class

Clearly, the developed pipeline based on imaging radar input shows the ability to detect and protect vulnerable road users such as humans.

3.3 Radar-based localisation and mapping

For autonomous vehicle navigation, there are multiple solutions to locate vehicles, such as those based on satellite positioning techniques (GPS, Galileo, etc.). However, these techniques are not always robust and exhibit some important limitations: areas of low satellite coverage, GPS-denied environments or covered zones. To overcome these weaknesses, they must be fused with other types of sensors, such as cameras, lidars or radars.

Visual-based methods are the most common, but they also present the greatest challenges when operating in adverse weather conditions (rain, snow or fog), in low-visibility settings or under high-dynamic range lighting. On the other hand, lidars (light detection and ranging), are sensitive to a lesser extent than cameras to adverse weather effects, vibrations or physical obstruction of the sensor.

Alongside these well-known methods, radars can be used for localisation based on SLAM techniques. Due to its competitive price compared to lidars, it could achieve greater market penetration, transforming the industry quickly and deeply. Radar navigation based on SLAM exhibits highly robust weather and lighting behavior.

3.3.1 Overview of RADAR-based SLAM

A fully autonomous vehicle must be able to map its surroundings while simultaneously localising within them, it must also be capable of referencing a global frame of reference, i.e. global localisation. This set of problems can be solved in a unified way using SLAM, where not only is the vehicle's odometry extracted but also a map is built in which it can later localise itself. If a prior map exists, global localisation to that map can be performed and the vehicle's pose subsequently updated using odometry techniques. To address this challenge with radar, there are multiple approaches.

The mapping and localisation methods consist of several components. The algorithm's frontend is responsible for inferring the vehicle's movement and orientation, there are multiple options for this, and today the most advanced for radar are those based on point-cloud registration. By solving an optimisation problem that finds the maximum overlap between temporally adjacent point clouds, one can determine the transformation matrix that yields the minimum error. That transformation matrix describes the vehicle's motion. Some of the most representative algorithms include:

- Iterative Closest Point (ICP) [26]. This algorithm starts from an initial rigid transformation (rotation + translation). At each iteration, it finds the nearest neighbor in the target cloud for each source point, then solves a least-squares problem to update the transform by minimizing the sum of squared Euclidean distances between matched pairs. Converges to a local optimum, highly sensitive to the initialization.
- Generalized ICP [27], enhances ICP by incorporating local surface structure. Each point is associated with a covariance matrix estimated from its neighborhood. Rather than point-to-point distances, GICP minimizes a Mahalanobis distance between local Gaussian distributions, effectively blending point-to-point and point-to-plane metrics for improved robustness to noise and complex geometries.
- Normal Distributions Transform (NDT) [28]. NDT performs a partition the target cloud into a voxel grid, fitting a Gaussian distribution to the points within each voxel. The source cloud is aligned by maximizing the likelihood of its points under these Gaussian models. The optimisation directly updates the transformation parameters using analytical derivatives of the Gaussian density.
- Coherent Point Drift (CPD) [29]. Treats the source cloud as centroids of a Gaussian Mixture Model (GMM) and the target cloud as observed data. Uses an Expectation–Maximization (EM) algorithm to estimate both the GMM correspondence probabilities and the transformation (for the radar case rigid). A regularization term penalizes non-smooth deformations, ensuring coherent (“drift-free”) motion of the points.

Once the transformation between vehicle frames is estimated, it remains to define a way to structure and store data known as the algorithm’s backend. Currently, the most accepted and best-performing method is the pose graph [30], which provides a structure in which constraints and uncertainties can be added easily, allowing subsequent optimisation of the calculated trajectory. This method has been shown to outperform traditional backend approaches such as the Extended Kalman Filter and particle filters. Similarly, the de facto library for performing pose-graph optimisation is General Graph Optimisation (the g2o algorithm) [31].

The works in [32], [33], [34] demonstrate how to perform radar based odometry and extract key points to subsequently carry out localisation.

Works such as those presented in [35], [36], [37], [38], [39], [40] show a complete SLAM pipeline in which the authors adopt different solutions for the algorithms used in odometry extraction, point-cloud processing and information storage; these represent the reference methods for performing SLAM. Figure 17 shows the main differences among the most powerful SLAM methods.

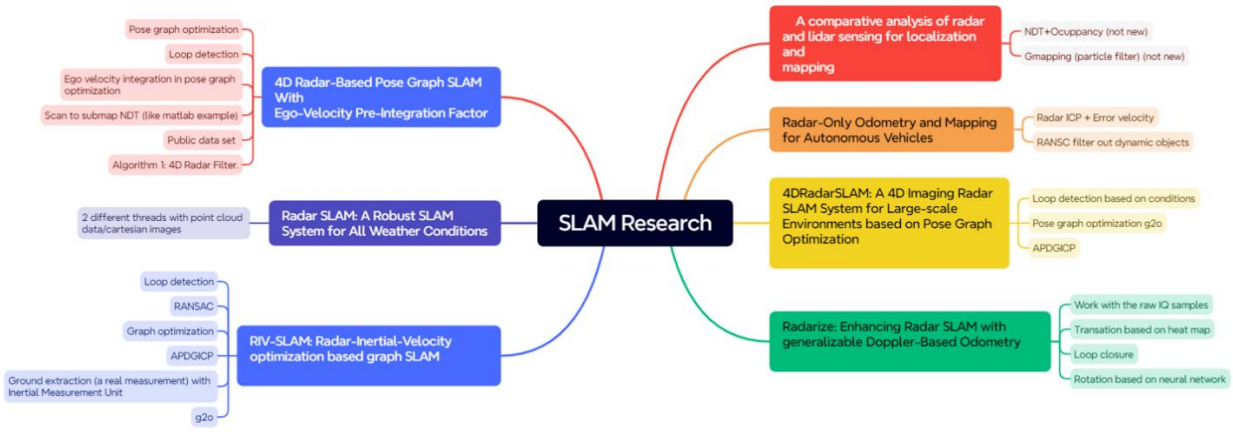


Figure 17: Research classification

3.3.2 Waveye's RADAR based localisation and mapping stack

In this section we will define the most representative algorithms used by the Waveye radar navigation stack.

3.3.2.1 Data pre-processing

Given the temporal computing requirements (the algorithms must be suitable for real-time processing), a down sampled of the point cloud must be performed to obtain a cloud that is sufficiently descriptive while reducing computation time. Likewise, for mapping it is necessary to eliminate all dynamic targets so that the constructed map has temporal persistence; to this end, outlier detection is performed using a Random Sample Consensus (RANSAC) algorithm on the radial relative-velocity profile and the azimuth angle of the detections. Finally, the point cloud is conditioned by removing clutter, ghost targets and artifacts so that the mapping is cleaner.

A highly efficient way to perform down sampling is to retain only the peak detections in the range-angle-Doppler processing, this ensures a low-density yet highly descriptive subset of detections.

At this stage of the pipeline, the received scan is evaluated for redundancy. If it is deemed redundant (e.g., the vehicle is stationary and that area has already been mapped), the scan is discarded to reduce computational load.

3.3.2.2 Front end algorithms based on point cloud matching

For radar-based odometry we use a robust variant of the Normal Distributions Transform (NDT). In this approach, each voxel in the reference scan is modelled as a Gaussian distribution, and we compute the likelihood that each point in the new scan was generated by those Gaussians. Maximizing this likelihood yields the rigid-body transformation matrix between the two-

point clouds. To ensure reliable convergence, the voxel-grid resolution is adjusted dynamically based on scene characteristics (e.g. point density, expected motion).

We also implement Coherent Point Drift (CPD) in the front end. The CPD treats one scan as centroids of a Gaussian Mixture Model (GMM) and aligns the other by maximizing the overall correspondence probability. Its probabilistic formulation makes it more robust in highly ambiguous or low-structure environments.

With the implemented algorithms, we have achieved good results in terms of both efficiency and accuracy. Table 10 shows the RMSE results for point cloud matching.

Table 10: Root Mean Square Error front end

Algorithm	Mean	Variance	Min	Max
NDT	0.6068	0.2295	0.0041	3.9692
Recursive NDT	0.5219	0.1547	0.0040	3.2868
CPD	0.4944	0.1096	0.0039	4.0898
CPD Peaks	0.5020	0.1375	0.0208	4.7275

3.3.2.3 Front end algorithms based on doppler

Radar, unlike any comparable sensor, can measure the instantaneous radial relative velocity of each detection. Exploiting this capability, we can apply methods such as those proposed by Kellner et al. [16], [17]. First, it is essential to include only static targets in the estimation process using the pre-processing steps described above to filter out dynamic objects. Once the static points are identified, we estimate velocity via least squares; if we know each point's signal-to-noise ratio or uncertainty, we can instead use weighted least squares.

This method requires knowledge of the radar's orientation: the measured radial velocities must be back-projected to the vehicle's center of rotation and then integrated over the radar's sampling interval to construct the vehicle's odometry. When multiple radars are available, additional degrees of freedom can be added to the system of equations to estimate both yaw rate and lateral velocity independently, although cars follow the Ackermann steering constraint (i.e. no pure lateral velocity), this additional complexity is often unnecessary.

This method of obtaining odometry is more complex but offers a key advantage: it does not require distinguishable features in the environment, making it suitable for use in featureless areas such as deserts or completely empty roads.

3.3.2.4 Back end

In the backend, we use a pose graph architecture where the poses obtained from radar internal odometry and radar point cloud odometry are fused, and measurement uncertainties are also

incorporated. A fundamental component of the backend is loop recognition: when the vehicle revisits a known part of the map, a loop is introduced into the graph. It is assumed that the point is the same (accounting for the relative transformation, since it's never possible to revisit the same point), and this constraint in the graph helps reduce accumulated drift and smooth the estimated trajectory.

The logic required to incorporate loop recognition involves several steps. First, scene similarity must be verified as place recognition. Place recognition is computationally expensive. To reduce its cost, two non-exclusive strategies can be applied:

- Dimensionality reduction of the data by extracting scene descriptors. Common algorithms for this include:
    - Scan Context
    - Fast Point Feature Histograms (FPFH)
    - Principal Component Analysis (PCA) based descriptors
    - Deep learning–based methods such as RadarNet or RING
- Intelligent loop search, where the search space is constrained based on the vehicle's orientation and position, thereby reducing the number of candidate locations to check for revisits.

Together, these techniques enable efficient loop closure, improving map consistency and long-term localisation accuracy.

3.3.2.5 Mapping

Once odometry is accurately estimated, each Radar frame is projected into a shared world coordinate system and merged in two stages: a coarse alignment provided by the odometry estimate, followed by local refinement provide by NDT, CPD or RIO depending the structure of the environment, this refinement is done to remove residual pose errors and ensure tight scan-to-scan registration. For advance 4D Imaging radars the mapping problem must be performed in the 3D space, for mapping in 3D: a uniform voxel grid would explode in memory (typical 2D method), so a probabilistic octree (e.g. OctoMap) is used. The space is adaptively subdivided: nodes only spawn children if the contained volume exhibits mixed occupancy, and homogeneous regions remain as coalesced leaves. Each leaf holds a log-odds value updated via the same Bayesian fusion but along volumetric beams rather than planar rays. To keep the tree balanced and efficient, "lazy" insertion strategies defer deep subdivision until information gain justifies the extra resolution, and periodic pruning collapses subtrees whose children converge to the same occupancy state.

Contemporary enhancements layer on top of this core: Truncated Signed Distance Function (TSDF)-based octrees fuse signed distance measurements from multiple sweeps to yield smooth surface meshes in real time; semantic occupancy maps attach learned class probabilities (e.g. vehicle vs. building) to each node; and GPU-accelerated ray-marching pipelines exploit massive parallelism to sustain the high rate of modern radars. Together, these techniques maximize map fidelity while respecting the tight memory and computing budgets of automotive platforms. The effectiveness of the proposed SLAM pipeline is illustrated in Figure 18, which shows the mapping results obtained with the Waveye Radar in a representative driving environment.

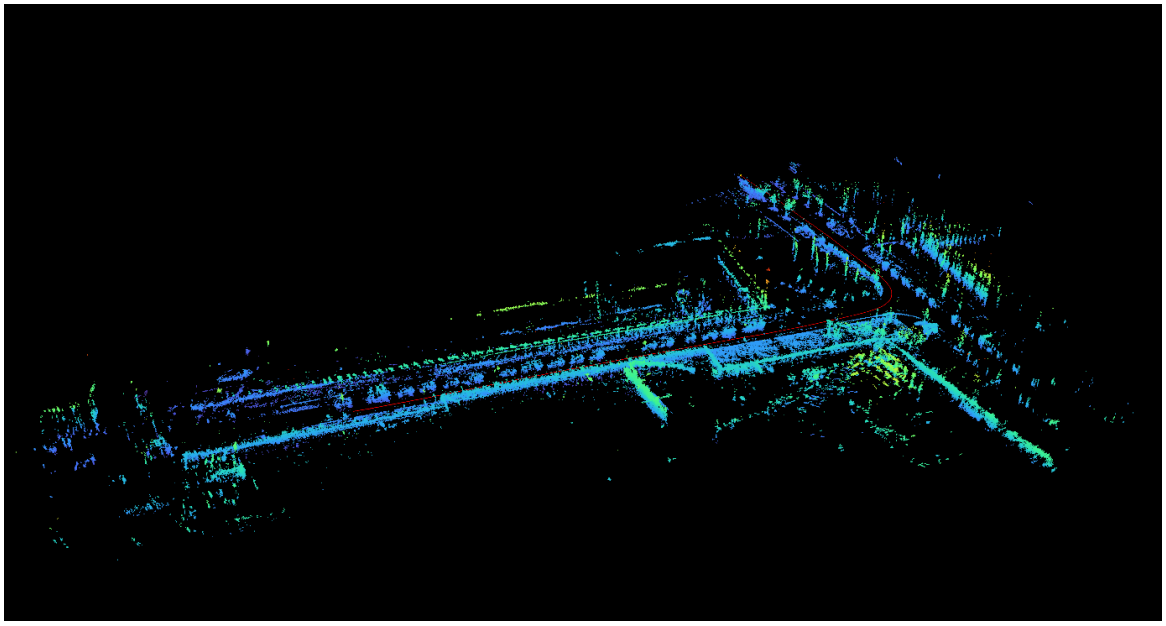


Figure 18: Mapping results based on the presented SLAM pipeline on Waveye Radar

3.3.2.6 Logic block diagram

In summary, Figure 19 shows a flow diagram of the SLAM pipeline. Over the further course of the project, the presented framework will be extended to be able to localise based on already recorded radar map.

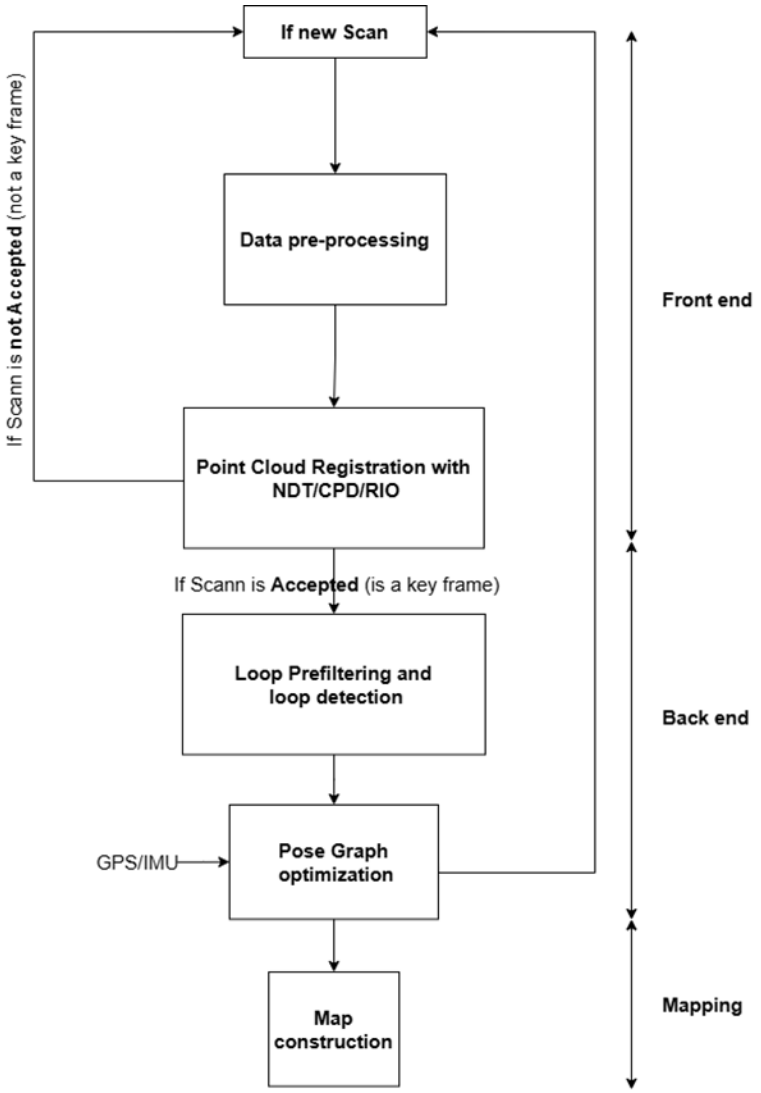


Figure 19: Logic block diagram for the presented radar-based SLAM

3.4 In-Cabin External Awareness Module

External awareness (Traffic signs detection and road condition assessment) is performed on RGB images captured from the in-cabin outward-facing camera. The datasets are organised into YOLO format with train, validation, and test splits, and bounding box annotations expressed in normalised coordinates. At runtime, no custom preprocessing is applied; YOLOv8 internally handles input conversion to RGB, resizing to 640 × 640 pixels with letterbox padding to preserve aspect ratio, and normalization of pixel values to the [0, 1] range. During training, the framework also applies its default augmentations, including horizontal flips, HSV color jitter, scaling, translations, and composite strategies such as mosaic and mixup. In validation and inference, only deterministic resizing and normalisation are applied. Once preprocessed, the images are processed by YOLOv8 to detect traffic signs and assess road conditions, outputting

bounding boxes and confidence scores for the relevant categories (e.g., specific sign classes or potholes), enabling the system to maintain continuous awareness of the external driving environment. The overall data input pipeline for the in-cabin external awareness module is illustrated in Figure 20.

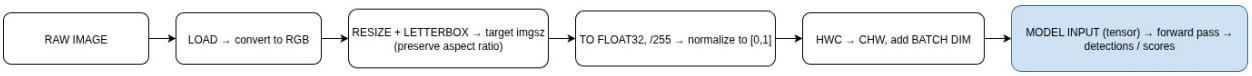


Figure 20. Data input pipeline for the in-cabin external awareness module

3.5 Multimodal Data Fusion for external monitoring system

This section introduces FedKalmanNet, a novel FL-based approach to autonomous vehicle localisation, designed to fuse heterogeneous sensor inputs—such as GNSS, IMU, and wheel odometry—in a privacy-preserving and communication-efficient manner. By distributing the training of Kalman filter–inspired neural networks across vehicles, this framework enables each agent to learn from multimodal sensor data locally, while still benefiting from collective knowledge across the fleet.

Unlike traditional centralised localisation pipelines, FedKalmanNet avoids raw data sharing and instead aggregates model updates, ensuring scalability and data confidentiality. The approach supports real-time inference and adapts to sensor variability, making it well-suited for diverse vehicle platforms and dynamic operational contexts. By integrating temporal dynamics, sensor fusion, and distributed learning into a unified architecture, FedKalmanNet achieves robust, multi-modal localisation accuracy, even in GNSS-degraded environments.

3.5.1 Preliminaries

System Model

Vehicle i at time instant t, needs to autonomously navigate using its own sensor capabilities. Its state is characterized by the 3D position, i.e., $\boldsymbol{x_i^{(t)}} = [x_i^{(t)}\ y_i^{(t)}\ z_i^{(t)}]^T \in \mathbb{R}^3$. Utilizing a suite of visual, satellite, and mechanical sensors, the vehicle can gather both self and relative multi-modal observations or measurements concerning its own state and that of a nearby vehicle j. More specifically, we can define the state transition and self-positioning models using data from IMU and GNSS sensors. These models, which are assumed to be degraded by Gaussian noise, denoted as $\mathcal{G}(\mu, \boldsymbol{\Sigma})$, can be expressed as follows [41]:

- State transition model:

$$\boldsymbol{x_i^{(t)}} = f\left(\boldsymbol{x_i^{(t-1)}}, \boldsymbol{u_i^{(t)}}\right) + \boldsymbol{e_i^{(t)}}$$

Equation 37

where $\boldsymbol{e_i^{(t)}} \sim \mathcal{G}(0, \sqrt{\boldsymbol{R_i^{(t)}}})$. Function $f(\cdot)$ employs a constant velocity motion model: $f = \boldsymbol{Ax_i^{(t-1)}} + \boldsymbol{Bu_i^{(t)}}$. Here, $\boldsymbol{A} = \mathbb{I}_3$ and $\boldsymbol{B} = diag(dt, dt, dt)$. Control input vector $\boldsymbol{u_i^{(t)}} = [u_i^{(x,t)}\ u_i^{(y,t)}\ u_i^{(z,t)}]^T \in \mathbb{R}^3$ consists of 3D velocity as recorded by the IMU sensor.

- Self positioning measurement model

$$\hat{\boldsymbol{z}}_{\boldsymbol{i}}^{(\boldsymbol{t})} = \boldsymbol{x_i^{(t)}} + \boldsymbol{n_p},\ \boldsymbol{n_p} \sim \mathcal{G}(0, \boldsymbol{\Sigma_p}) \qquad \textit{Equation 38}$$

Collaborative decision-making approaches [42], [43] based on traditional optimisation, exploit the V2X connectivity links among nearby vehicles by fusing self and relative vehicular measurements, in order to localise ego vehicle and its neighbors. Relative measurements or observations include distance, azimuth and inclination angles with respect to nearby vehicles, extracted by visual sensors like camera or LiDAR. Instead of real-time measurement transmission and fusion between vehicles in challenging environments, the proposed collaborative learning scheme in the context of FedKalmanNet will perform offline local models aggregation using only self measurements. Afterwards, the trained FedKalmanNet will be exploited by each individual vehicle in order to localise itself highly accurate and much more efficient than a collaborative decision-making approach.

Data-Driven Kalman Filtering for state estimation

Before we proceed with the presentation of our framework, we will revisit the fundamental equations of the EKF, used for state or location estimation in autonomous driving. This review will help illustrate how standard KalmanNet enhances ego vehicle localisation through two key features: the representation of non-linear system dynamics and the estimation of covariance matrices for state and measurement noise using an explainable deep learning approach. Therefore, the steps for estimating state $\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})} \in \mathbb{R}^3$ and its covariance matrix $\hat{\boldsymbol{S}}_{\boldsymbol{i}}^{(\boldsymbol{t})} \in \mathbb{R}^{3x3}$ using the EKF can be described as follows:

$$\overline{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})} = \boldsymbol{A}\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t-1})} + \boldsymbol{Bu_i^{(t)}} \qquad \textit{Equation 39}$$

$$\overline{\boldsymbol{S}}_{\boldsymbol{i}}^{(\boldsymbol{t})} = \boldsymbol{A_i^{(t)}}\tilde{\boldsymbol{S}}_{\boldsymbol{i}}^{(\boldsymbol{t-1})}\boldsymbol{A_i^{(t)T}} + \boldsymbol{R_i^{(t)}} \qquad \textit{Equation 40}$$

$$\boldsymbol{K_i^{(t)}} = \overline{\boldsymbol{S}}_{\boldsymbol{i}}^{(\boldsymbol{t})}\boldsymbol{H_i^{(t)T}}\left(\boldsymbol{H_i^{(t)}}\overline{\boldsymbol{S}}_{\boldsymbol{i}}^{(\boldsymbol{t})}\boldsymbol{H_i^{(t)T}} + \boldsymbol{Q_i^{(t)}}\right)^{-1} \qquad \textit{Equation 41}$$

$$\hat{\boldsymbol{x}}_i^{(t)} = \overline{\boldsymbol{x}}_i^{(t)} + \boldsymbol{K}_i^{(t)}(\hat{\boldsymbol{z}}_i^{(t)} - g\left(\overline{\boldsymbol{x}}_i^{(t)}\right)) \quad \textit{Equation 42}$$

$$\hat{\boldsymbol{S}}_i^{(t)} = \left(\mathbb{I} - \boldsymbol{K}_i^{(t)}\boldsymbol{H}_i^{(t)}\right)\overline{\boldsymbol{S}}_i^{(t)}, \quad \textit{Equation 43}$$

where, $\boldsymbol{R}_i^{(t)} \in \mathbb{R}^3$ is the state transition covariance matrix, $\boldsymbol{Q}_i^{(t)} \in \mathbb{R}^{3x3}$ denotes the measurement covariance matrix and $\hat{\boldsymbol{z}}_i^{(t)} \in \mathbb{R}^3$ represents the measurement vector required to estimate the state or location of i, respectively. Additionally, $\boldsymbol{H}_i^{(t)} \in \mathbb{R}^{3x3}$ corresponds to the jacobian matrix of some generic function $g(\cdot)$ with respect to $\overline{\boldsymbol{x}}_i^{(t)}$. In case where $\hat{\boldsymbol{z}}_i^{(t)}$ contains the GNSS position, i.e., direct measurement of i's state, then $g\left(\overline{\boldsymbol{x}}_i^{(t)}\right) = \overline{\boldsymbol{x}}_i^{(t)}$ and $\boldsymbol{H}_i^{(t)} = \mathbb{I}_3$. As such, EKF turns to Kalman filter (KF), i.e., its linear counterpart.

The KalmanNet architecture is designed to estimate the uncertainty matrices of KF algorithm. These include the Kalman gain matrix $\boldsymbol{K}_i^{(t)} \in \mathbb{R}^{3x3}$, the state transition covariance matrix $\boldsymbol{R}_i^{(t)}$,the predicted state covariance matrix $\overline{\boldsymbol{S}}_i^{(t)} \in \mathbb{R}^{3x3}$, and the matrix $\boldsymbol{W}_i^{(t)} \in \mathbb{R}^{3x3}$. The latter is defined as $\boldsymbol{W}_i^{(t)} = \boldsymbol{H}_i^{(t)}\overline{\boldsymbol{S}}_i^{(t)}\boldsymbol{H}_i^{(t)\mathrm{T}} + \boldsymbol{Q}_i^{(t)}$. Using these estimated matrices, KalmanNet computes the updated state estimate $\hat{\boldsymbol{x}}_i^{(t)}$ and its covariance $\hat{\boldsymbol{S}}_i^{(t)}$ following the standard KF equations. The network takes as input the current and previous measurement vectors $\hat{\boldsymbol{z}}_i^{(t)}$and $\hat{\boldsymbol{z}}_i^{(t-1)}$, the previous state estimates $\hat{\boldsymbol{x}}_i^{(t-1)}$, $\overline{\boldsymbol{x}}_i^{(t-1)}$, and $\overline{\boldsymbol{x}}_i^{(t-2)}$, the control input vector $\boldsymbol{u}_i^{(t)}$, and the time interval dt. Therefore, the equations of the KF can be formulated as:

$$\boldsymbol{K}_i^{(t)}, \boldsymbol{R}_i^{(t)}, \overline{\boldsymbol{S}}_i^{(t)}, \boldsymbol{W}_i^{(t)} = KalmanNet_{\theta,i}(\cdot) \quad \textit{Equation 44}$$

$$\hat{\boldsymbol{x}}_i^{(t)} = \overline{\boldsymbol{x}}_i^{(t)} + \boldsymbol{K}_i^{(t)}(\hat{\boldsymbol{z}}_i^{(t)} - \overline{\boldsymbol{x}}_i^{(t)}) \quad \textit{Equation 45}$$

$$\hat{\boldsymbol{S}}_i^{(t)} = \overline{\boldsymbol{S}}_i^{(t)} - \overline{\boldsymbol{S}}_i^{(t)}\boldsymbol{K}_i^{(t)} \quad \textit{Equation 46}$$

where $KalmanNet_{\theta,i}(\cdot)$ represents the KalmanNet network, which is used to estimate the corresponding uncertainty matrices.

The architecture of KalmanNet consists of three gated recurrent unit (GRU) layers interconnected with fully connected layers. These layers are structured to progressively estimate the required matrices: The first GRU layer estimates $\boldsymbol{R}_i^{(t)}$. The second layer uses the output of the first to estimate $\overline{\boldsymbol{S}}_i^{(t)}$. The third layer utilises the outputs of the previous layers to estimate $\boldsymbol{W}_i^{(t)}$.

Finally, $\overline{\boldsymbol{S}}_{\boldsymbol{i}}^{(\boldsymbol{t})}$ and $\boldsymbol{W}_{\boldsymbol{i}}^{(\boldsymbol{t})}$ are used to compute $\boldsymbol{K}_{\boldsymbol{i}}^{(\boldsymbol{t})}$. This sequential structure allows the network to capture the dependencies between these matrices in the Kalman filtering process.

3.5.2 Federated Data-Driven localisation: FedKalmanNet

In this Section, the proposed FedKalmanNet methodology will be presented. Based on the distributed learning theory [44], the proposed FL scheme can be realized by an ATC strategy, motivated by the fact that each vehicle employs a local KF algorithm utilizing the corresponding KalmanNet, and subsequently the local KalmanNet models are fused at the server side in order to derive a more robust and accurate global KalmanNet. Initially, we will formulate the general approach of FedKalmanNet and then present the proposed adaptation and combination steps of this methodology.

Federated KalmanNet

To establish the FL (federated learning) framework, we consider a network of $\mathcal{N}$ vehicles. In this distributed learning framework, each vehicle i participating in the proposed collaborative learning process, utilises its local dataset $\mathcal{D}_i = \{\boldsymbol{Z}_{\boldsymbol{i}}^{\boldsymbol{1:T_i}}, \boldsymbol{X}_{\boldsymbol{i}}^{\boldsymbol{1:T_i}}\}$, containing an input trajectory as measured over time by its own sensors (GNSS, IMU, etc.), as well as the corresponding ground truth (or target) trajectory. More specifically, T_i is the length of training trajectories, input $\boldsymbol{Z}_{\boldsymbol{i}}^{\boldsymbol{1:T_i}} = [\boldsymbol{z}_{\boldsymbol{i}}^{(\boldsymbol{1})} \dots \boldsymbol{z}_{\boldsymbol{i}}^{(\boldsymbol{T_i})}] \in \mathbb{R}^{6\mathrm{x}T_i}$ contains the noisy 3D positions and velocities for the corresponding training trajectory, while target $\boldsymbol{X}_{\boldsymbol{i}}^{\boldsymbol{1:T_i}} = [\boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{1})} \dots \boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{T_i})}] \in \mathbb{R}^{3\mathrm{x}T_i}$ contains the corresponding ground truth 3D trajectory. Note that in order to generate input $\boldsymbol{Z}_{\boldsymbol{i}}^{\boldsymbol{1:T_i}}$, we add white Gaussian noise to ground truth position and velocity of the training dataset following Equation 37 and Equation 38 For simplicity, we assume that each local dataset consists of a single pair of input and ground truth trajectories.

Each vehicle employs a local KF algorithm and trains its corresponding KalmanNet model using its private dataset, following Equation 44 and Equation 46. The fact that each agent employs only its local dataset to train a local model may lead to limitations in the model's ability to generalize across various environmental conditions. The local KalmanNet model ($KalmanNet_{\theta,i}(\cdot)$) might only capture the uncertainties in sensor measurements specific to the local environment, hence failing to capture the system dynamics across different scenarios (e.g., weather conditions, trajectories in rural or urban areas). To address this limitation, we propose the FedKalmanNet framework. This approach enables agents to collaborate under the coordination of a central server. Through this collaboration, vehicles can learn a more robust KalmanNet model that demonstrates enhanced generalization capabilities across diverse environmental conditions.

Federated Data-driven localisation: Adaption Step

During the adaptation step, each vehicle i employs a local KF which utilises a local KalmanNet $KalmanNet_{\theta,i}(\cdot)$ to estimate the Kalman gain:

$$\boldsymbol{K_i^{(t)}} = KalmanNet_{\theta,i}(\cdot) \quad \textit{Equation 47}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})} = \overline{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})} + \boldsymbol{K_i^{(t)}}(\hat{\boldsymbol{z}}_{\boldsymbol{i}}^{(\boldsymbol{t})} - \overline{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})}) \quad \textit{Equation 48}$$

In the proposed framework, the local KalmanNet can be trained end-to-end using the local dataset. In more detail, let $\boldsymbol{\theta_i}$ denote the trainable parameters of the local KalmanNet, and γ_i be a regularization coefficient. Each agent employs an ℓ_2-regularized mean-squared error (MSE) loss to optimize its local model, defined as follows:

$$\ell_i(\boldsymbol{\theta_i}) = \frac{1}{T_i}\sum_{t=1}^{T_i}\left\|\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})}\left(\boldsymbol{z}_{\boldsymbol{i}}^{(\boldsymbol{t})};\boldsymbol{\theta_i}\right) - \boldsymbol{x}_{\boldsymbol{i}}^{(\boldsymbol{t})}\right\|^2 + \gamma_i\|\boldsymbol{\theta_i}\|^2 \quad \textit{Equation 49}$$

where $\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})}\left(\boldsymbol{z}_{\boldsymbol{i}}^{(\boldsymbol{t})};\boldsymbol{\theta_i}\right)$ is the output of the local KF parametrized by $\boldsymbol{z}_{\boldsymbol{i}}^{(\boldsymbol{t})}$ and $\boldsymbol{\theta_i}$. The fact that the KalmanNet model is optimized using only the local dataset of each agent may lead to limitations in the model's ability to generalize across various environmental conditions. Hence, after all participating vehicles $i \in N$ have updated their local KalmanNet using Equation 49, the next step is the combination phase.

Federated Data-driven localisation: Combination Step

The goal of this step is to develop a model that captures underlying system dynamics and accurately estimates covariance matrices for state and measurement noise using local data from the vehicles. Given the structure of KFs, agents upload to the central server only $KalmanNet_{\theta_n}(\cdot)$, as defined in Equation 44. The server then aggregates the local KalmanNets using a fusion rule:

$$KalmanNet_{\theta_g}(\cdot) = \sum_{i=1}^{N} a_i KalmanNet_{\theta_i}(\cdot) \quad \textit{Equation 50}$$

where $KalmanNet_{\theta_g}(\cdot)$ denotes the global KalmanNet and a_i are the combination weights. Afterwards, the server broadcasts the global KalmanNet model back to all clients. Each vehicle then initializes its local KalmanNet model (Equation 44) with the received global model. This iterative process is repeated for M communication rounds, enabling the global model to continuously improve, incorporating diverse data as well as ensuring the privacy of local training datasets. Thus, the FedKalmanNet method can be realized by a twofold process: adaptation, i.e., training the local KalmanNet using KF concept and the local private dataset, and combina-

tion, i.e., aggregating the local models at the server side and broadcasting the global model back to the agents:

$$\boldsymbol{K}_{\boldsymbol{i}}^{(\boldsymbol{t})} = KalmanNet_{\theta,i}(\cdot) \quad \textit{Equation 51}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})} = \overline{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})} + \boldsymbol{K}_{\boldsymbol{i}}^{(\boldsymbol{t})}(\hat{\boldsymbol{z}}_{\boldsymbol{i}}^{(\boldsymbol{t})} - \overline{\boldsymbol{x}}_{\boldsymbol{i}}^{(\boldsymbol{t})}) \quad \textit{Equation 52}$$

$$KalmanNet_{\theta_g}(\cdot) = \sum_{i=1}^{N} a_i KalmanNet_{\theta_i}(\cdot) \quad \textit{Equation 53}$$

$$KalmanNet_{\theta,i}(\cdot) = KalmanNet_{\theta_g}(\cdot) \quad \textit{Equation 54}$$

The proposed FedKalmanNet approach is demonstrated as depicted in Figure 21.

3.5.3 Numerical results

Simulation setup

The simulations were carried out using dataset[2] which contains the trajectories of 60 vehicles moving in CARLA simulator's environment [45]. The dataset contains ground truth 3D position, linear velocity, acceleration, etc. For the testing evaluation, we have chosen vehicle with index 0 as the ego vehicle over the simulation horizon of $T = 900$ time instances and sampling interval $dt = 0.1$ sec.

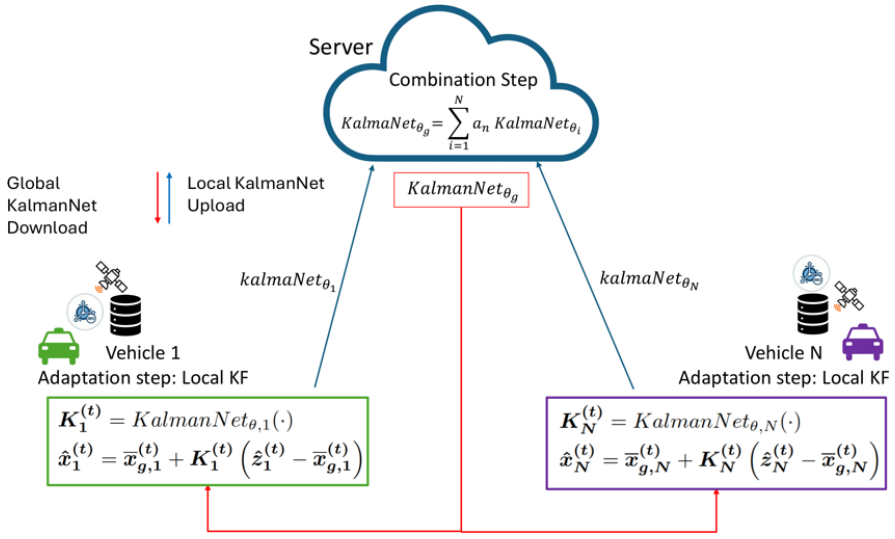


Figure 21: The proposed end-to-end FedKalmanNet approach

[2] https://dx.doi.org/10.21227/511y-4s83

The evaluation study will consider two scenarios in order to assess the proposed FL approach: i) comparing **FedKalmanNet** with the traditional **KalmanNet** when all the training data are available to the global server, as well as when only the dataset from an individual agent is used during training, ii) how the collaborative training paradigm presented in this work can outperform traditional optimisation based CL approaches, which require much larger amount of information from nearby vehicles to localise ego vehicle. Baseline methods, apart from standalone GNSS, include **MSMV** [42] and **LKF-SA** [43], which set the covariance matrix of sensor measurements equal to identity. The error metrics include Root Mean Square Localisation Error (RMSL) over Time, i.e., $RT - LE$, in order to evaluate ego vehicle i's ability to localize itself. Additionally, the Cumulative Distribution Function (CDF) of instantaneous localisation errors demonstrates the probability of location error to be lower or equal than a specific threshold. For the training of networks, we have exploited four vehicles/clients with trajectories from TownMap10 of CARLA, and varying lengths (T =1550, 4600, 1420), which are shown in Figure 22. Although they seem similar, vehicles actually move with different velocities. For example, the minimum and maximum velocity of agents 1 and 2 are between $8.9 - 10.8\ m/sec$ and $9.2 - 10.9\ m/sec$, respectively, while those of 3 and 4 are between $0.7 - 10.7\ m/sec$ and $3.6 - 10.7\ m/sec$, respectively. The input to the network is the ground truth 3D trajectory degraded by additive white Gaussian noise of zero mean and standard deviation $\boldsymbol{\Sigma_p} = diag(1.5m, 1.5m, 1.5m)$.

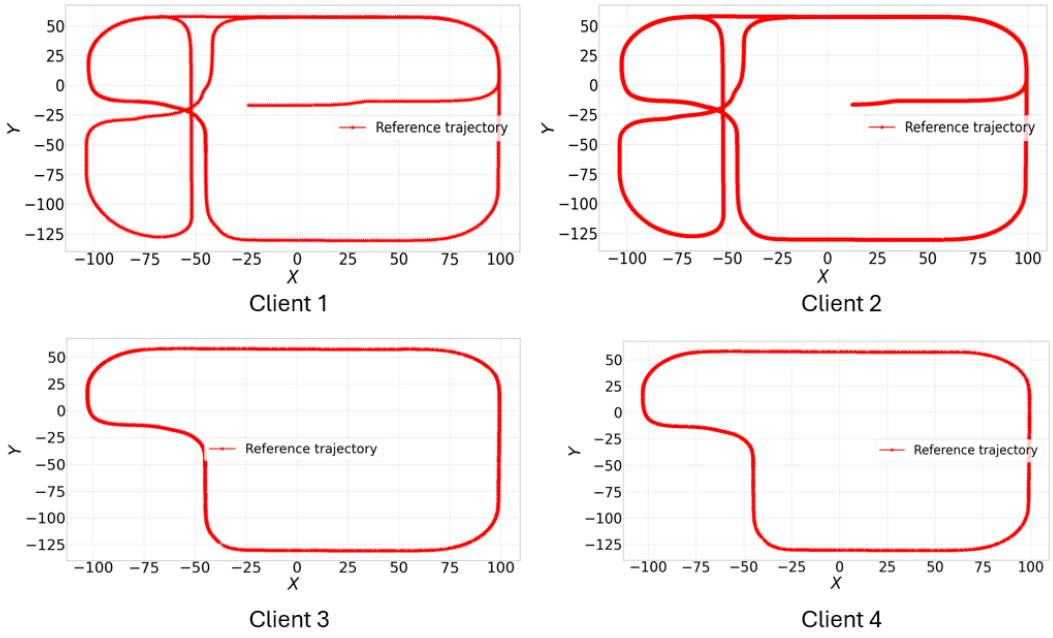


Figure 22: Client s trajectories from TownMap10 of CARLA

Furthermore, additive white Gaussian noise is used for the 3D velocity inputs in order to realistically capture state transition noise. The standard deviation of velocity noise is set to 10% of the ground truth velocity of the specific vehicle, as stated in [46]. Each trajectory is splitted in subtrajectories of length equal to 100, while 80% and 20% out of them are used for training and cross validation. For the centralized training, all four datasets are used to train **CentrKalmanNet** for 1500 epochs. For the individual training, we exploit the trajectories only from agent 0 and train the network for 500 epochs. For the FL implementation, we adopt the FedAvg [47], and use 20 communication rounds. Batch size is equal to 1, while learning rate and weight decay are set to 0.3.

Evaluation study

1) Impact of federated learning vs centralized and individual training: In this testing scenario, we will evaluate the performance of the proposed data-driven FL framework with respect to centralized and individual implementation of **KalmanNet's** training. More specifically, Figure 23 demonstrates the convergence of **FedKalmanNet** to **CentrKalmanNet** after 20 communication rounds. To be more detailed, after each round both **FedKalmanNet** and **CentrKalmanNet** are evaluated in terms of $RT-LE$ with the GNSS based trajectory of ego vehicle as input, which has been generated by adding white Gaussian noise of zero mean and standard deviation $\boldsymbol{\Sigma_p} = diag(1.8m, 1.8m, 1.8m)$, as well as a bias drawn from uniform distribution $\mathcal{U}[0.5, 1]$ to the ground truth. In that way, we will show that the performance of networks is still high enough, regardless of the fact that we have conducted training with input trajectory degraded by Gaussian noise. Furthermore, the standard deviation of velocity noise is set to 15% of the ground truth velocity. Clearly, centralized training achieves superior performance in terms of $RT-LE$ due to the availability of all training data. However, the distributed framework of **FedKalmanNet** reduces $RT-LE$ after each round, as it is expected from the relevant theory, reaching $RT-LE$ at round 20 lower than $1.5m$, with respect to $1.43m$ of **CentrKalmanNet**. Additionally, Figure 24 highlights the CDF of ego vehicle localisation error using **FedKalmanNet** (after 20 rounds of training), **CentrKalmanNet**, **IndKalmanNet**, a traditional KF taking as input the GNSS and without any knowledge of system uncertainty, and, finally, GNSS. For each one of the curves, we indicate the maximum error attained by each approach. For example, we see that the simple KF reduces GNSS error by almost $3m$, while the **IndKalmanNet** reduces it by $5m$, clearly showing the benefits of estimating the underlying uncertainty. Most importantly, the proposed **FedKalmanNet** reduced maximum GNSS error by $6.3m$, reaching the same accuracy with that of **CentrKalmanNet**. As such, we conclude that the proposed data-driven FL framework efficiently converges to the centralized model's accuracy, significantly improving at the same time the localisation accuracy of ego vehicle.

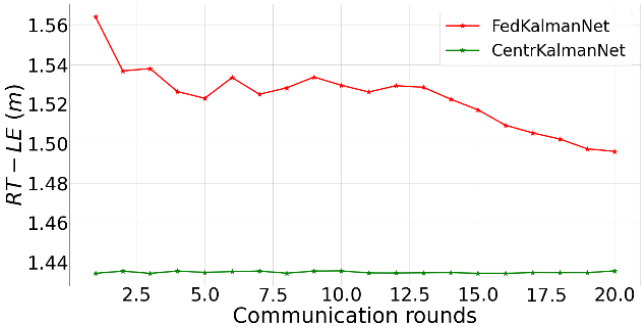


Figure 23: Convergence of FedKalmanNet to CentrKalmanNet after 20 communication rounds.

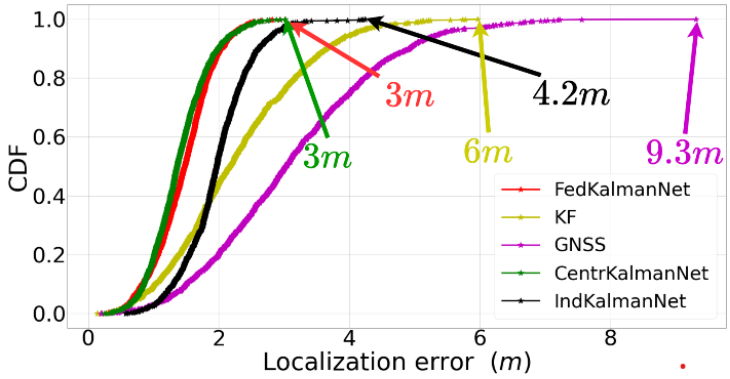


Figure 24: Cumulative distribution function of ego vehicle localisation accuracy

2) Impact of collaborative training vs collaborative decision-making for ego vehicle localisation: In the second testing scenario, we will investigate the performance of **FedKalmanNet** versus CL techniques **LKF-SA** and **MSMV**. Our goal is to demonstrate that federated data driven localisation which exploits only self measurements of ego vehicle, is capable of outperforming collaborative decision-making solutions which require to fuse information coming from nearby vehicles. As such, by estimating the underlying uncertainty through the proposed collaborative training scheme, we will accurately and cost-efficiently localise ego vehicle. Results are summarised in Figure 25. To simulate relative and self measurements that have to be fused by the ego vehicle, we set standard deviation of distance, azimuth and inclination angle measurement noise equal to $\sigma_d = 1\,m$, $\sigma_{az} = \sigma_{in} = 4^{\circ}$, respectively, as well as $\boldsymbol{\Sigma_p} = diag(3.5m, 3.5m, 3.5m)$. Furthermore, each vehicle establishes a connected neighborhood with its nearby vehicles within a range of $30m$, consisting of a maximum number of vehicles (based on shortest distance). In Figure 25, we demonstrate $RT - LE$ of each technique with respect to maximum number of connected neighbors. Clearly, GNSS experiences the lowest accuracy, while both **FedKalmanNet** and GNSS are actually independent of the number of neighbors. **LKF-SA** improves its accuracy as the size of neighborhood grows larger, reaching $RT - LE$ equal to $1.62m$, requiring at the same time richer V2X communication resources in order to perform the fusion. **MSMV** performs more or less the same in all cases ($2.1m$ of $RT - LE$). We

see in all cases that **FedKalmanNet** significantly outperforms the other solutions, reaching $1.5m$ of accuracy, exploiting only the self measurements (GNSS and velocity) of ego vehicle. On the other hand, **LKF-SA** has to exploit larger neighborhoods in order to enhance the localisation performance.

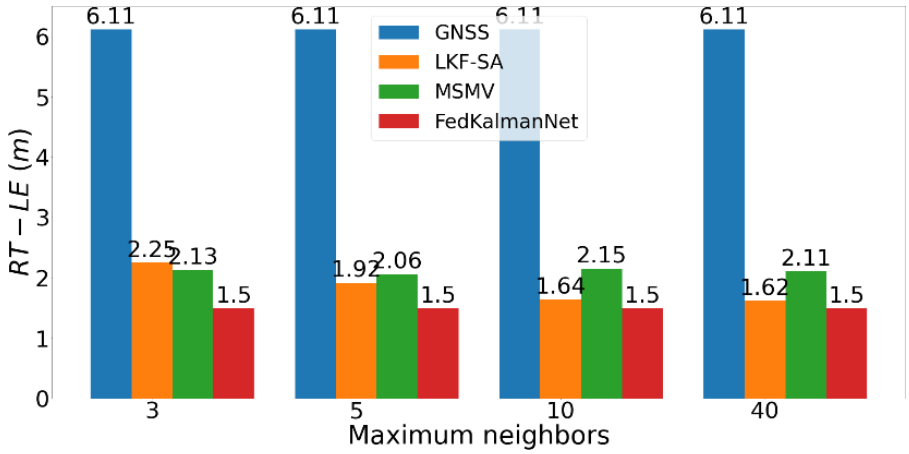


Figure 25: FedKalmanNet outperforms the baseline methods, exploiting only self GNSS and velocity. LKF-SA has to integrate greater amount of information from neighbors to reach FedKalmanNet's accuracy.

Concluding remarks

In this Section, we have introduced FedKalmanNet, the FL counterpart of KalmanNet, in order to enable a collaborative training paradigm among a group of vehicles, aiming to enhance vehicle localisation through self measurements uncertainty estimation. The distributed learning scenario among a group of vehicles has been formulated through the ATC strategy, where each vehicle exploits initially its local private dataset to train a local KalmanNet, which is then updated by a global aggregation operation at the server side. Evaluation results in CARLA simulator demonstrate that the FL model features almost similar performance with its centralized counterpart, while significantly outperforms the model trained with the data coming from an individual vehicle. Most importantly, we have shown that the proposed FL data-driven localisation framework exploiting only self measurements, performs much more efficient than collaborative decision-making schemes, which fuse data from large neighborhoods of connected vehicles in order to localise ego vehicle. In deliverable D3.2 and based on the previous formulation, we will investigate how different aggregation rules at the server side influence the FL model accuracy, as well as the potential of enabling a serverless and personalized training strategy to enhance the overall accuracy of multi-modal fusion for localisation not only of single autonomous vehicle, but a swarm of CAVs.

3.6 Concluding remarks

The algorithms introduced in this section demonstrate how advanced processing techniques can significantly enhance the utility of low-cost sensors in real-world perception tasks. By applying model-based super-resolution, federated learning, and deep radar processing, AutoTRUST enables high-performance environmental understanding without reliance on expensive hardware. These methods collectively support scalable deployment of autonomous capabilities and form a core part of the perception stack for robust and efficient mobility systems. The next section shifts focus from external to internal monitoring, detailing the fusion of in-cabin sensor data to assess driver and occupant state.

4 Algorithms for internal sensor data processing and fusion

The development of robust internal sensing systems is a central component of the AutoTRUST architecture, aiming to enhance safety, personalization, and inclusiveness in automated mobility by monitoring the vehicle cabin environment and its occupants. This section presents the foundational methodologies and system architecture that enable in-cabin multimodal perception using synchronised visual and acoustic data streams. It focuses on sensor deployment, data acquisition, and fusion techniques that serve as the basis for future behavioral and context-aware analytics.

The internal monitoring system, as described in this deliverable (D3.1), emphasizes the modular design and integration of complementary sensing modalities—including RGB cameras and microphone arrays—for capturing rich, high-resolution visual and audio cues. These sensor streams support a wide range of downstream analytics tasks, such as driver distraction detection, facial emotion recognition, drowsiness estimation, occupant identification, and abnormal sound event recognition. At this stage, the system prioritizes the establishment of robust data pipelines, multimodal synchronisation, and preliminary signal processing, laying the groundwork for real-time interpretation and intelligent decision-making in the cabin context.

Most importantly, the current deliverable provides a high-level overview of the algorithmic strategies, calibration methods, and architectural design principles, which will be further elaborated and extensively evaluated in deliverable D3.2 “Advanced internal and external sensing system”.

4.1 Multimodal data fusion for internal monitoring system

One of the foundational challenges in multimodal sensor fusion is temporal synchronisation. Piatkowski et al. [48] proposed a lightweight algorithm for synchronised multimodal data acquisition based on Temporal Sample Alignment (TSA), addressing the desynchronisation problem across sensor streams with heterogeneous sampling rates. Their algorithm achieves alignment in software without hardware modification, supporting real-time acquisition from sensors such as RGB cameras, LiDARs, microphones, and IMUs. The authors demonstrate that TSA can reduce drift and maintain alignment under realistic operating conditions, thereby increasing the robustness of downstream fusion pipelines. While originally validated with visual and spatial sensors, the proposed method is generalizable to in-cabin modalities where fine-grained temporal correlation between, for instance, voice cues and facial expressions is critical.

Cameras remain a cornerstone of in-cabin monitoring due to their high spatial resolution and compatibility with deep learning-based perception. Cieśla and Ostermayer [49] investigated multimodal distraction detection by combining camera data with vehicle telemetry and inertial measurements. Using a recurrent neural network architecture, they demonstrated that fusing temporal sequences of RGB video frames with inertial data significantly improved the detection accuracy of driver distraction compared to unimodal baselines. Their results support the hypothesis that spatiotemporal fusion of behavioral cues across sensor domains enables finer-grained classification of cognitive and physical driver states.

Complementing visual information, audio sensing contributes an additional channel for activity and intent recognition. Jiang et al. [50] implemented a microphone array system integrated with high-definition maps to enhance vehicle perception through sound. Although their system focused on detecting external acoustic events (e.g., sirens, vehicle localisation via Time Difference of Arrival (TDoA)), their framework validates the feasibility of real-time multi-microphone fusion in vehicular settings. The techniques employed—beamforming, spatial filtering, and map-based acoustic localisation—can be extended to in-cabin environments to capture speech, detect stress, or monitor interactions, especially when fused with visual inputs.

To bridge the gap between visual and acoustic sensing, event-based cameras offer a novel modality that encodes dynamic visual changes with microsecond latency. These sensors produce sparse, high-frequency asynchronous data, ideal for capturing rapid movements without motion blur or lighting artifacts. Savran et al. [51] introduced a fully convolutional neural network for Voice Activity Detection (VAD) using an event camera alongside microphone signals. By converting event streams into event-intensity maps and integrating them with audio spectrograms, their system achieved high accuracy in detecting speech onsets and offsets. This work illustrates how event-based vision can enhance audio perception tasks, particularly in low-light or visually ambiguous conditions, making it suitable for in-cabin monitoring scenarios involving speech or gesture recognition.

Further supporting the utility of event-based cameras, Feng et al. [52] developed a real-time object detection framework that integrates event-based vision into the CARLA simulator and ROS pipeline. Their method adapts YOLOv8 to handle asynchronous event streams and demonstrates high detection accuracy on dynamic objects. While the primary application is external object perception, the underlying architecture—transfer learning from event-driven input, integration with conventional pipelines, and low-latency inference—provides a transferable methodology for applying event-based vision to in-cabin settings, such as fast driver head-turn detection or interaction monitoring.

Beyond individual sensor capabilities, the effectiveness of in-cabin monitoring systems increasingly relies on the design of fusion strategies that integrate heterogeneous sensor data into semantically meaningful representations. Shariff et al. [53] provide a comprehensive taxonomy of fusion methodologies that can be directly applied to the in-cabin domain. These include fusion based on discernible units, where sensor streams are aligned by semantically meaningful segments—such as synchronizing lip movements from RGB or event cameras with voice onset from microphones to detect speaking behavior. Fusion based on feature complementarity is particularly relevant for integrating spatial-temporal features from event cameras with audio intensity or environmental sensor data (e.g., cabin temperature or CO_2 levels), enabling richer inference of passenger states such as fatigue or stress. In fusion based on target attributes, different sensors monitor distinct dimensions of the same subject; for instance, voice pitch (microphones) and facial tension (visual sensors) jointly inform emotional or cognitive load estimations. Finally, fusion based on multi-source decision-making aggregates individual sensor judgments—such as distraction scores from vision, acoustic, and physiological modalities—into a unified driver state classification. These strategies, although widely applied in external Advanced Driver Assistance Systems (ADAS) perception systems, are now being adapted for internal monitoring, where asynchronous and noisy sensor signals make fusion both a technical challenge and an opportunity for more robust modeling of driver and passenger behavior [32].

Together, these studies converge on the conclusion that effective in-cabin monitoring relies not only on sensor diversity but on the fusion mechanisms that integrate them. Temporal alignment methods, recurrent and convolutional architectures, and cross-modal representations all play critical roles in enabling robust interpretation of driver and passenger behavior. As sensor technologies evolve, particularly in asynchronous domains like event-based vision, continued innovation in fusion strategies will be essential to support safe, adaptive, and intelligent in-cabin systems.

Table 11: Fusion process for in-cabin monitoring

Stage	Component / Tool	Description
Input sensor	RGB Camera	Captures color images of the in-cabin environment.
	Depth Sensor	Provides depth information for spatial understanding.
Fusion Processing	ZED SDK (Neural Depth Engine + Body34)	Combines RGB and depth data to estimate 3D skeleton keypoints
	Body Tracking Module	Extracts 34 keypoints (upper body) in real time from fused input.

Stage	Component / Tool	Description
Output	3D Skeleton Keypoints	Used for behavior recognition, risk detection, and simulation input.
	RGB + Depth Frames	Stored for visual reference and dataset enrichment.

The in-cabin monitoring system integrates visual and spatial data streams through a unified skeleton-based representation. 3D keypoints representing the occupant's upper-body posture are extracted using stereo vision and depth information. These skeleton outputs serve as the integration backbone across modules. They provide a consistent input format for behavior recognition, simulation control, and dataset structuring. By centering the fusion architecture on 3D skeletal representations, the system simplifies the combination of multimodal inputs and ensures modularity for future sensor extensions or pipeline upgrades. The overall framework is summarised below in Table 11.

The functionality of in-cabin monitoring will focus on performing crucial tasks for internal perception, like driver distraction, facial emotion recognition, occupant identification, etc., that are analyzed in the following. The extensive evaluation and results, as well as the assessment of the in-cabin monitoring system performance will be provided in deliverable D3.2.

In the following, we describe the algorithms used for driver distraction detection and facial emotion recognition.

4.1.1 Driver Distraction Detection

Captured video data from front-facing and mid-to-side view cameras were first standardized to ensure consistency across all inputs. Each video frame was resized to a resolution of 224 × 224 pixels, which allowed uniform processing in downstream modules. To enhance the robustness of the data and account for variability in real-world driving conditions, frames were further augmented. Random cropping simulated slight camera misalignment and variations in zoom, while brightness adjustments compensated for different lighting conditions encountered during driving, as illustrated in Figure 26. Temporal variations were mimicked using frame jitter, and horizontal flipping was applied to account for left-right variations in driver behaviors. These preprocessing steps ensured that the raw video data was transformed into a more diverse and reliable representation for analysis.

After preprocessing, the frames were organized into sequential clips, forming video batches suitable for input into the distraction detection pipeline. This sequencing maintained temporal continuity across frames, which is crucial for capturing dynamic driver behaviors. The resulting batches represented a fully processed, augmented, and model-ready form of the raw camera

data, enabling efficient and effective downstream analysis while preserving the temporal and spatial characteristics of driver actions.

Figure 26: Example augmentations of a single resized camera frame.

4.1.2 Facial Region Extraction

All facial analysis tasks in this system, including emotion recognition, occupant identification and drowsiness detection, start from raw frames captured from front faced camera. To ensure consistency and reliability across different analyses, the following preprocessing steps are applied to all faces:

1. Face Detection: Facial regions are located in each frame using a Multi-task Cascaded Convolutional Neural Network (MTCNN)-based pipeline, which identifies key landmarks and produces bounding boxes around detected faces. This ensures that only relevant facial regions are analyzed, removing background and irrelevant information.
2. Cropping: Detected faces are cropped and, if necessary, resized to match the input requirements of the next processing stage.

By applying these preprocessing steps, the system produces clean, standardized facial images that can be used for various tasks—whether predicting emotions, identifying individual occupants or detecting drivers drowsiness—without requiring task-specific adjustments at this stage. An indicative example of the aforementioned procedure is depicted in Figure 27.

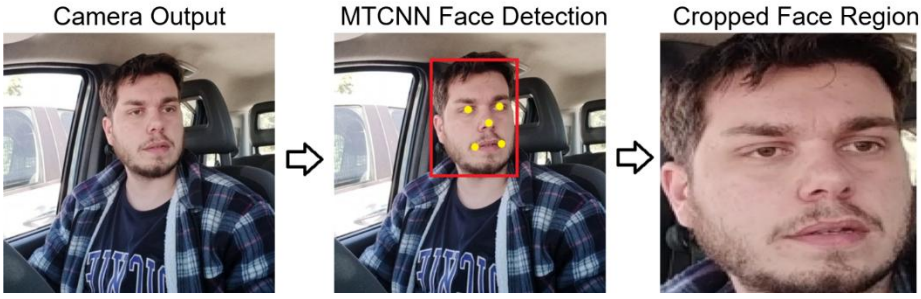


Figure 27: Face Extraction Pipeline

4.1.3 Drowsiness Detection

Drowsiness detection is performed on the frontal camera, where recorded frames are resized and normalized to account for variations in lighting and pose. Face detection follows the approach described previously, and the pipeline then focuses on landmark detection. Facial landmarks are extracted using two Open Neural Network Exchange (ONNX) models: one for face detection (SCRFD_10G_KPS) and one for landmark localization (2d106det). Input frames are processed, aligned via affine transformations, and cropped before landmarks are mapped back into the original frame coordinates. These landmarks are then used to calculate the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR), which represent eye closure and yawning, respectively. Acting as key indicators of drowsiness, EAR and MAR values are passed into a lightweight classifier, with temporal smoothing applied through a sliding window to ensure classification stability and reduce false positives. The pipeline, as shown in Figure 28, is optimized for real-time inference with ONNX Runtime, making it suitable for edge deployment.

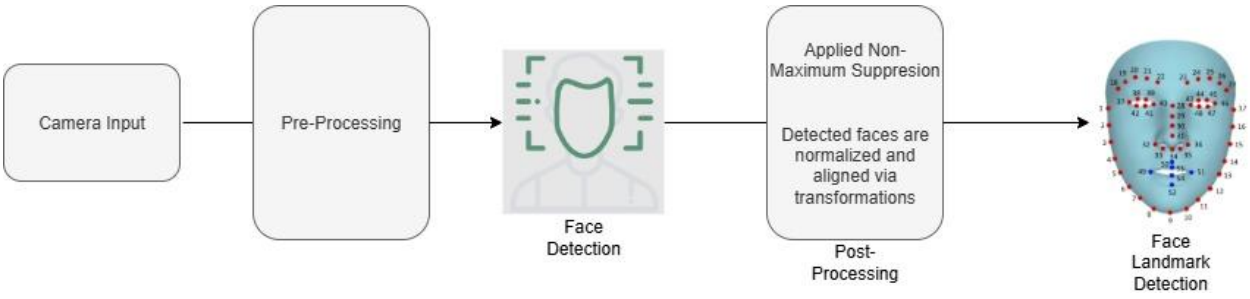


Figure 28: Data processing pipeline for drowsiness detection

4.1.4 Abnormal Sound Event Detection

The Abnormal Sound Event Detection (ASED) is performed in real-time using the ReSpeaker Mic Array v2.0 with audio captured at a 16,000 Hz sampling rate and segmented into 1-second chunks for processing. Each 1-second audio segment then undergoes a preprocessing pipeline where the raw waveform is transformed into a structured input suitable for model inference. This includes the computation of a log-mel spectrogram using Short-Time Fourier Transform (STFT) with 25 ms windows and a 10 ms hop size. The frequency content is then projected onto 64 mel filter bands, covering the range of 125 Hz to 7500 Hz. The resulting spectrogram is expanded in dimensionality and cast to float32 to match the input format expected by the acoustic model. The data processing pipeline of the acoustic input is depicted in Figure 29.

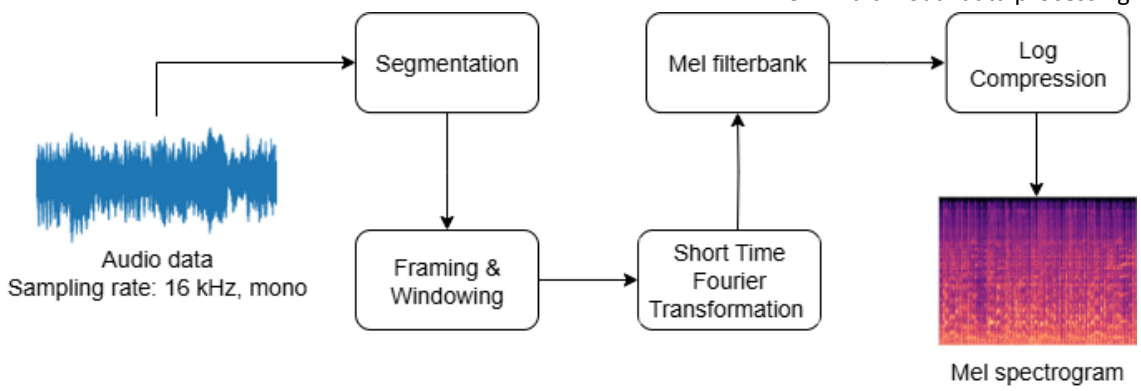


Figure 29: Data processing pipeline for abnormal sound event detection.

The preprocessed audio is fed into a neural acoustic event classification model, converted to the ONNX format and executed via ONNX runtime for efficient real-time inference. The model outputs a vector of class logits, representing the likelihood of various predefined acoustic events: *Baby cry, Noise, Scream, Siren, Snoring, Speech, Traffic noise, and Vehicle Horn.*

The dominant event class is identified and stored along with its confidence score and timestamp using UltraDict, a shared-memory dictionary that supports real-time inter-process communication. This enables downstream modules — such as the Large Language Model (LLM) process — to access and integrate the current acoustic context into multimodal reasoning and decision-making workflows.

4.1.5 Virtual assistant for multimodal data fusion

The virtual assistant integrates multimodal data fusion to continuously monitor the internal state of the vehicle cabin by combining information from visual, auditory, and emotional inputs. Specifically, it retrieves real-time data from three sources: facial emotion detection, camera-based distraction monitoring, and acoustic event detection. Each modality provides individual labels (e.g., emotions like anger or fear, visual distractions like texting, and sounds such as snoring or horns), which are processed in parallel and stored using shared memory structures.

The system aggregates short-term observations (e.g., the last 10 frames per modality) and performs statistical filtering to determine the most frequent (and therefore likely) state for each source. It then performs event-level fusion by checking whether any of the newly detected dominant states have changed and are deemed critical. Only significant events—such as a shift from a neutral to a dangerous emotion or a new distracting behavior—trigger a response. Under such circumstances, the assistant agent is dynamically generated a context-aware textual prompt, which prompts it to produce brief, distinct warnings for every issue it detects. By using fusion logic, this method reduces false positives and improves robustness by guaranteeing that only significant multimodal combinations result in interventions.

4.2 Data Synchronisation

This section describes the communication protocol and synchronisation approach used for in-cabin monitoring.

4.2.1 Communication and Synchronisation via UltraDict

The internal monitoring system brings together several key components—like acoustic event detection, facial emotion recognition, face identification, and an analysis of driver distraction. At its core, it relies on a lightweight inter-process communication tool in Python, called UltraDict, which allows the different modules to run independently yet stay in sync by smoothly sharing data. Technically, the main goal here is to create a fast and dependable method for combining multiple types of input. This ensures the AI assistant always has a clear, up-to-date picture of both the driver's behaviour and the surrounding environment.

4.2.1.1 Architecture Overview

Each module operates as an independent process, complete with its own pipeline for data acquisition, inference, and output. Rather than direct communication, all interaction between the modules and the AI assistant happens through UltraDict, a lightweight shared-memory interface that acts as the system's communication backbone.

This modular design allows new sensors or upgraded components to be integrated without disrupting existing workflows. Everything stays loosely coupled, which means maintenance is easier, and scaling becomes a matter of plug-and-play rather than full rewrites.

Modules remain focused on their specific responsibilities, while the assistant takes charge of integration and reasoning. At regular intervals, it queries the shared memory segments, retrieves the most recent outputs from each perception module, and fuses the data into a unified, time-aligned snapshot of the system state.

By keeping responsibilities clearly divided, the architecture stays robust and flexible. It supports fast iteration cycles and makes room for continuous system improvements—without compromising stability.

4.2.1.2 Inter-Process Communication and Synchronisation

UltraDict functions as the central glue holding the system’s components together. It provides a fast, shared-memory interface where each module writes its predictions into uniquely named memory segments. These segments can be accessed in parallel by other processes—most notably, the assistant. This setup eliminates the overhead and complexity often seen with socket-based or message queue communication, making data exchange both simpler and faster.

Interestingly, synchronisation across different data modalities doesn't rely on explicit coordination. Each module updates its respective memory space independently, at regular intervals. Meanwhile, the assistant continuously polls these segments, pulling the latest prediction values. It uses polling rates and contextual logic to verify the relevance and timing of each value, effectively aligning inputs in time without tight coupling.

By adopting a shared-memory model for communication, the system sidesteps many of the pitfalls of traditional IPC mechanisms. UltraDict streamlines the architecture, removing the need for handshakes or complex messaging protocols. As long as each module maintains its expected update rate.

4.2.2 Future Iterations

While the existing system utilises UltraDict for rapid shared-memory communication between perception modules and the assistant, in the final version of WP3 deliverables, we will implement ROS2 to establish a more modular, maintainable, and introspectable framework. In this configuration, every perception module (such as emotion recognition, distraction detection or sound event detection) will operate as a ROS2 node, transmitting structured messages to specific topics. The virtual assistant will subscribe to these subjects to create an internal, real-time awareness of the driver's condition, and for the in-cabin environment in general. ROS2 offers a solid framework for managing these interactions, featuring tools for message visualisation, debugging, and lifecycle management. This facilitates scaling the system, replacing models, or introducing new features (such as gesture recognition) without changing the fundamental architecture. The assistant can now utilise ROS2 to receive sensor-based predictions and broadcast speech outputs or alerts, creating a complete event-driven cycle among perception and reasoning.

4.3 Simulation-Based Multimodal Fusion Framework for In-Cabin Monitoring

As part of AutoTRUST's internal monitoring system, the MORAI's simulation platform provides a scalable, configurable, and high-fidelity environment for testing and validating multimodal data fusion strategies. This simulation-based framework is designed to complement real-world sensing by enabling controlled experimentation with virtual sensors and occupant behaviors, accelerating the development of robust in-cabin monitoring systems.

Virtual Sensor Architecture and Methodology

The simulation platform as shown in Figure 30, built on Unreal Engine 5.4, adopts a modular, class-based architecture that allows precise configuration and control of virtual in-cabin sensors. These sensors, implemented as unreal engine components, can emulate RGB, infrared

(IR), and depth camera modalities. Users can configure each sensor's parameter, including field of view, focal length, orientation, etc., and position them flexibly within the vehicle cabin (e.g., near the rear-view mirror, on the seatback, or side pillars). The current implementation includes a single RGB camera with a 2.1 mm focal length and a polarizing lens mounted at the rear-view mirror position, following industry best practices for unobtrusive driver and occupant monitoring.

The architecture includes interactive User Interface (UI) components and supports both manual control and automated pipelines for scenario execution. Through a set of specialized widgets and managers, users can trigger occupant animations, adjust environmental conditions (e.g., lighting), configure sensor settings, and initiate synchronised data capture. This flexibility enables rapid prototyping of sensor configurations and behavior scenarios.

Synthetic Multimodal Data Generation

MORAI's simulation platform supports the generation of synchronised synthetic datasets comprising RGB video, depth maps, and 3D skeleton keypoints as shown in Figure 31. These datasets are critical for training and evaluating multimodal behavior recognition models under diverse conditions. Behavior scenarios are enacted using animated virtual occupants, with predefined risk-related actions such as reaching, phone use, distraction, fatigue, and absence of the driver.

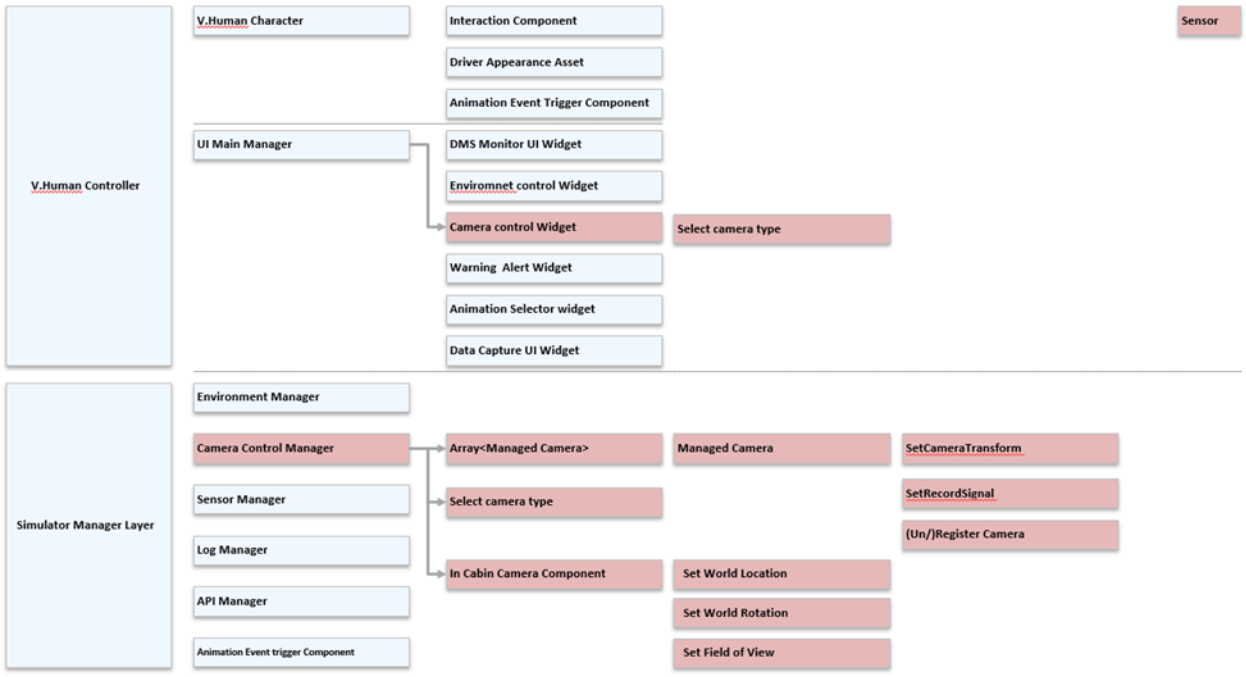


Figure 30: Sensor fusion architecture for in-cabin

Each recorded sequence includes:

- RGB and depth streams, captured from configured virtual sensors.
- 3D skeleton keypoints, extracted using the ZED SDK's Body38 model.
- Scenario metadata, including behavior labels, involved body parts, timing annotations, and camera configurations.

These synthetic datasets are annotated using a dedicated labeling pipeline that aligns multi-modal frames and metadata, ensuring consistency and reproducibility across scenarios.

Figure 31: Example RGB (left) and IR (right) data from the in-cabin simulation platform

Comparative Assessment of Sensor Modalities

In the context of internal monitoring, we have conducted a detailed evaluation of available sensing modalities. While thermal, radar, and IMU sensors offer advantages in specific contexts, the simulation framework focuses on vision-based sensing due to its balance between spatial fidelity, semantic richness, and integration feasibility. RGB-depth stereo cameras, particularly the ZED 2i, were selected based on criteria such as skeleton tracking capability, SDK flexibility, and physical integration within vehicle cabins.

Integration with Multimodal Fusion Objectives

The MORAI simulation platform serves as an experimental backbone for multimodal data fusion by:

- Enabling controlled and repeatable generation of rich sensor data under varied configurations.
- Supporting design exploration of in-cabin sensor layouts and fusion architectures.
- Providing annotated, synchronised multimodal datasets that bridge the gap between simulated and real-world deployments.

- By aligning synthetic data generation with the fusion models' needs, the platform facilitates early validation of perception pipelines and offers a cost-effective way to simulate edge cases that are difficult to capture in real-world driving.

In the final version of the deliverable (deliverable D3.3) will extend the simulation capabilities to include multi-camera configurations, audio sensing, and low-light environments using virtual IR sensors. These enhancements will further support the development of comprehensive in-cabin analytics and reinforce AutoTRUST's commitment to inclusive and resilient mobility solutions.

4.4 Visual data collection

Open-source datasets for monitoring passengers on buses are limited. However, some relevant datasets exist for related applications. We provide a description of relevant datasets that were identified herein while model development will be described in section of deliverable D3.2. It is worth noting that not all datasets will be used in model development only those that provide high quality data that aligns with the pilot setup and scenario.

People Detection: The Common Objects in Context (COCO) dataset[3] is a large-scale image recognition, segmentation, and captioning dataset widely used in computer vision research. It contains over 330,000 images, more than 200,000 of which are labelled, with annotations for over 80 object categories. Among these, the *person* class is one of the most frequent and densely annotated categories. Images in COCO often depict people in diverse real-world contexts, including varied poses, occlusions, interactions with objects, and in groups, which introduces significant intra-class variability. The dataset includes detailed instance-level annotations such as bounding boxes, segmentation masks, and key points (for body joints), enabling fine-grained tasks like human pose estimation. The richness and complexity of the *person* class annotations make COCO particularly valuable for evaluating algorithms on human-centric tasks under challenging conditions. COCO provides well-annotated bounding boxes and segmentation masks for people in diverse scenes, which can be used to train object detection models (e.g., YOLO, Faster R-CNN) to identify passengers boarding, alighting, or standing near the bus.

Purpose: COCO's *person* class serves as a robust base for detecting and localizing humans in complex scenes and can be used effectively for initializing a bus monitoring system. However, domain-specific fine-tuning on data captured from actual bus environments is essential to bridge the gap between generic detection capabilities and the specific operational needs of

[3] https://cocodataset.org/#home

transit safety, crowd control, and behavioural monitoring. We anticipate commencing this activity once the pilot setup is complete.

Bus Violence Dataset

The Bus Violence dataset[4] is a large-scale collection of videos depicting violent and non-violent situations in public transport environments as shown in Figure 32. This benchmark was gathered from multiple cameras located inside a moving bus where several people simulated violent actions, such as stealing an object from another person, fighting between passengers, etc. It 1,400 video clips manually annotated as having or not violent scenes, making it one of the biggest benchmarks for video violence detection in the literature. Specifically, videos are recorded from three cameras at 25 Frames Per Second (FPS) --- two cameras located in the corners of the bus (with resolution 960 x 540 px) and one fisheye in the middle (1280 x 960 px). The clips have a minimum length of 16 frames and a maximum of 48 frames, capturing a very precise action (either violence or non-violence). The dataset is perfectly balanced, containing 700 videos of violence and 700 videos of non-violence. The Bus Violence dataset is intended as a test data benchmark. However, for researchers interested in using our data also for training purposes, we provide training and test splits. The 1,400 video clips are divided into two folders named Violence /No Violence, containing clips of violent situations and non-violent situations, respectively; two txt files containing the names of the videos belonging to the training and test splits, respectively.

Purpose: The Bus Violence dataset can be used in a bus monitoring system to train and evaluate models for automated violence detection in real-time onboard video feeds. By providing short, annotated video clips of both violent and non-violent interactions recorded from multiple camera angles inside a moving bus, the dataset enables the development of supervised learning models (e.g., CNN-RNN hybrids, 3D CNNs, transformers) that can learn temporal and spatial patterns associated with aggressive behaviours. Its balanced class distribution and high-quality labels make it suitable for both training and benchmarking violence detection algorithms under realistic conditions, such as crowded scenes, occlusions, and camera motion. This can support automated alerts to security personnel, enhance passenger safety, and assist in post-incident analysis.

[4] *Bus Violence: a large-scale benchmark for video violence detection in public transport,* https://zenodo.org/records/7044203

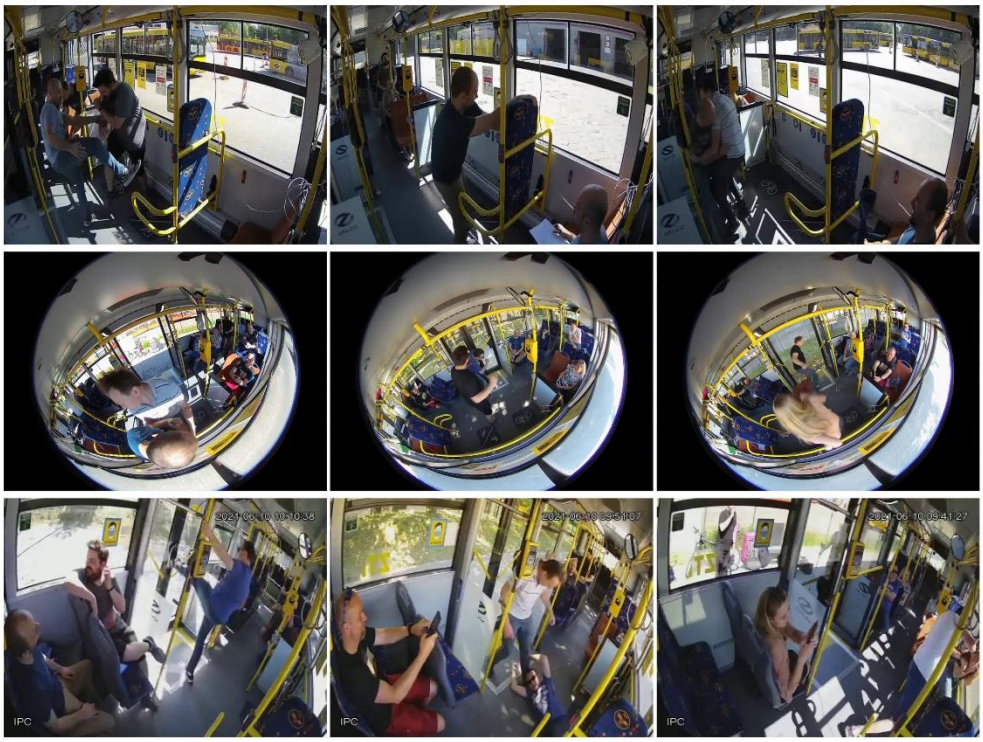


Figure 32: Samples of our Bus Violence benchmark belonging to the violence class, where the actors simulated violent actions, such as fighting, kicking, or stealing an object from another person. Each row corresponds to a different camera having a different perspective.

Multimodal Synthetic Bus Dataset

Given the scarcity of open-source datasets as mentioned above, an alternative approach is to generate synthetic images of passengers inside a bus as shown in Figure 33. By creating annotated synthetic images that include objects such as empty seats, occupied seats, and standing or seated passengers, it is possible to train a model effectively for detecting these classes in real scenarios. The Figure below illustrates an example of such a scene.

Purpose: This dataset can be used to fine-tune existing models or train models that utilise both RGB and Depth images.

Figure 33: (left) depth image generated from a generative AI model (right) Image generated from photorealistic simulator.

Real time people counting in cluttered scenes

People Counting Dataset (PCDS)[5] is real-world RGB-D dataset comprises over 4,500 videos recorded at the entrance doors of buses in both normal and crowded conditions in China as shown in Figure 34. The dataset includes footage captured from a ceiling-mounted camera positioned above the bus doors, recording passengers as they enter and exit.

Purpose: This dataset can be used to train models for monitoring the on-boarding and off-boarding phase. This of course assumes a suitable camera is placed at the top of the bus door so that the views are aligned.

Abnormal behaviour detection

Wang and Xia, in their study [54] explored abnormal behaviour detection using the public BOSS dataset. This dataset contains videos recorded on a train, showcasing both normal and abnormal activities. The abnormal activities include mobile phone theft, fighting, newspaper theft, harassment, fainting, and panic.

Purpose: This dataset can be used to train models to detect abnormal activities.

[5] https://github.com/paperswithcode/paperswithcode-data

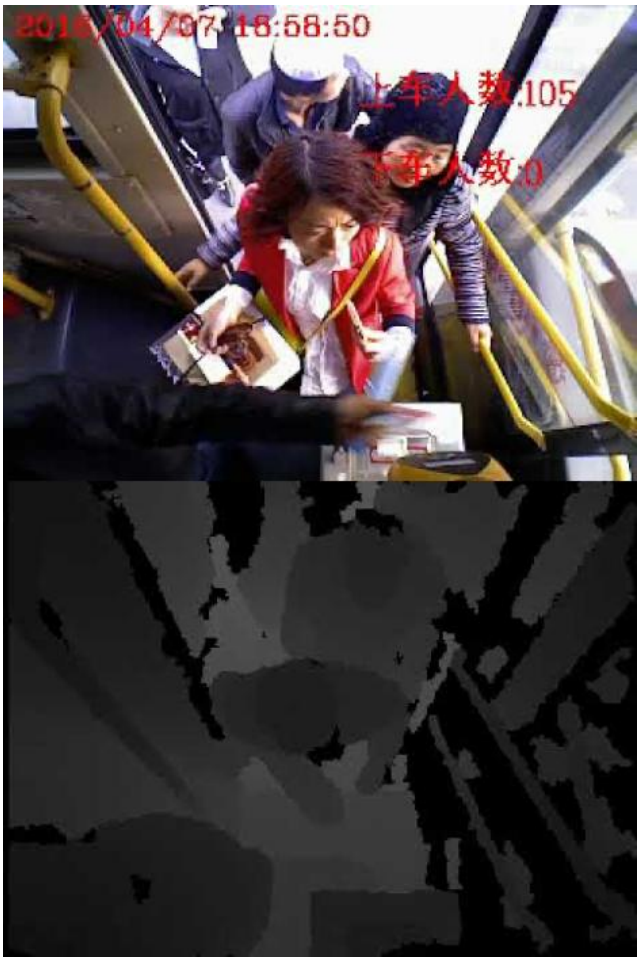


Figure 34: Multi-modal RGB and depth dataset for bus entrance.

4.5 Concluding remarks

The internal monitoring strategies outlined here provide a robust framework for capturing and interpreting driver and occupant states using synchronised multimodal inputs. By fusing RGB, audio, and event-based data, the system supports a wide range of safety and comfort applications—from distraction detection to abnormal sound recognition. The use of synthetic simulation and real-world data ensures adaptability and generalization. These contributions lay the groundwork for future enhancements in personalized in-cabin intelligence and user-centric vehicle behavior. The conclusion section summarises these developments and outlines the roadmap for upcoming AutoTRUST deliverables and pilot deployments.

5 Conclusion

This deliverable has presented the initial release of the multimodal data processing framework developed within the AutoTRUST project. It outlines the methodologies and algorithmic pipelines that underpin both external perception and internal monitoring systems, emphasizing real-time performance, cost-effectiveness, and adaptability to diverse mobility environments.

On the external sensing side, we introduced advanced techniques such as model-based deep unrolling for LiDAR super-resolution, enabling low-cost sensors to approximate the performance of high-resolution LiDAR in tasks like SLAM and segmentation. Complementary radar-based perception and mapping pipelines were also developed, leveraging 4D radar point clouds for robust object tracking and classification in challenging environmental conditions. Additionally, the integration of a federated learning approach, FedKalmanNet, offers a privacy-preserving solution for multi-modal localisation, validated through simulation datasets.

Additionally, the document detailed a flexible architecture for in-cabin monitoring using RGB cameras, microphone arrays, and neuromorphic event-based sensors. These modalities are synchronised through an extensible interface to support analytics such as emotion recognition, distraction detection, and abnormal sound event detection. To supplement real data and support model development, a simulation framework was introduced, allowing the generation of synthetic multimodal datasets within a photorealistic virtual environment.

Collectively, these contributions provide a modular, scalable, and future-proof basis for the AutoTRUST's external and internal monitoring system. This foundational work sets the stage for future development phases, which will expand these methods with graph-based reasoning, distributed intelligence, and personalized user modeling. The outcomes will directly support upcoming deployments and validations in pilot sites, ultimately contributing to the realization of safe, inclusive, and trustworthy mobility services.

References

[1] Y. Li and J. Ibanez-Guzman, “Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems,” *IEEE Signal Process Mag*, vol. 37, no. 4, pp. 50–61, 2020, doi: 10.1109/MSP.2020.2973615.

[2] X. Yue, Y. Zhang, and M. He, “{LiDAR}-based {SLAM} for robotic mapping: state of the art and new frontiers,” Nov. 2023, *arXiv*. doi: 10.48550/arXiv.2311.00276.

[3] J. Yue, W. Wen, J. Han, and L.-T. Hsu, “3D Point Clouds Data Super Resolution-Aided LiDAR Odometry for Vehicular Positioning in Urban Canyons,” *IEEE Trans Veh Technol*, vol. 70, no. 5, pp. 4098–4112, 2021, doi: 10.1109/TVT.2021.3069212.

[4] T. Shan, J. Wang, F. Chen, P. Szenher, and B. Englot, “Simulation-based lidar super-resolution for ground vehicles,” *Rob Auton Syst*, vol. 134, p. 103647, Dec. 2020, doi: 10.1016/J.ROBOT.2020.103647.

[5] A. Savkin, Y. Wang, S. Wirkert, N. Navab, and F. Tombari, “Lidar Upsampling With Sliced Wasserstein Distance,” *IEEE Robot Autom Lett*, vol. 8, no. 1, pp. 392–399, 2023, doi: 10.1109/LRA.2022.3214791.

[6] B. Yang, P. Pfreundschuh, R. Siegwart, M. Hutter, P. Moghadam, and V. Patil, “Tulip: Transformer for upsampling of lidar point clouds,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15354–15364.

[7] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, “RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4213–4220. doi: 10.1109/IROS40897.2019.8967762.

[8] J. Cheng, L. Zhang, Q. Chen, X. Hu, and J. Cai, “A review of visual SLAM methods for autonomous driving vehicles,” *Eng Appl Artif Intell*, vol. 114, p. 104992, 2022.

[9] A. Gkillas, A. S. Lalos, and D. Ampeliotis, “An Efficient Deep Unrolling Super-Resolution Network for Lidar Automotive Scenes,” in *2023 IEEE International Conference on Image Processing (ICIP)*, 2023, pp. 1840–1844. doi: 10.1109/ICIP49359.2023.10222856.

[10] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, “Model-Based Deep Learning,” *Proceedings of the IEEE*, vol. 111, no. 5, pp. 465–499, 2023, doi: 10.1109/JPROC.2023.3247480.

[11] Q. Hu *et al.*, “RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[12] R. Cheng, R. Razani, E. Taghavi, E. Li, and B. Liu, "(AF)2-S3Net: Attentive Feature Fusion with Adaptive Feature Selection for Sparse Semantic Segmentation Network," 2021. [Online]. Available: https://arxiv.org/abs/2102.04530

[13] L. Kong *et al.*, "Rethinking Range View Representation for LiDAR Segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023, pp. 228–240.

[14] Y. Zhao, L. Bai, and X. Huang, "FIDNet: LiDAR Point Cloud Semantic Segmentation with Fully Interpolation Decoding," 2021. [Online]. Available: https://arxiv.org/abs/2109.03787

[15] H. Cheng, X. Han, and G. Xiao, "Cenet: Toward Concise and Efficient Lidar Semantic Segmentation for Autonomous Driving," in *2022 IEEE International Conference on Multimedia and Expo (ICME)*, 2022, pp. 1–6. doi: 10.1109/ICME52920.2022.9859693.

[16] J. Behley *et al.*, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9296–9306. doi: 10.1109/ICCV.2019.00939.

[17] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, and H. Zhao, "SemanticPOSS: A Point Cloud Dataset with Large Quantity of Dynamic Instances," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 687–693, Feb. 2020, doi: 10.1109/IV47402.2020.9304596.

[18] Cohen, "RADAR On Steroids: Why the Imaging RADAR is the future. ," Retrieved from https://www.thinkautonomous.ai/blog/imaging-radar.

[19] N. S. Scheiner, "Machine Learning Approaches for Identifying Moving Road Users. ," *Kassel: Universität Kassel.*, 2024.

[20] K. S. Ester, "A density-based algorithm for discovering. In Knowledge Discovery and Data. ," 1996.

[21] R. E. Kalman, "A new approach to linear filtering and prediction problems. ," *Transactions of the ASME–Journal of Basic Engineering* , pp. 35–45, 1960.

[22] MathWorks., "Tracking EKF distance. Retrieved from https://de.mathworks.com/help/fusion/ref/trackingekf.distance.html#mw_," *2024*.

[23] R. Collins, "Retrieved from CSE598C Vision-Based Tracking," Lecture Notes: https://www.cse.psu.edu/~rtc12/CSE598C/.

[24] C. R. , S. H. , M. K. , & G. L. J. Qi, "Retrieved from PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *https://arxiv.org/abs/1612.00593*, 2016.

[25] C. R. , Y. L. , S. H. , & G. L. J. Qi, “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space: ,” in *https://arxiv.org/abs/1706.02413*, 2017.

[26] F. Wang and Z. Zhao, “A survey of iterative closest point algorithm,” in *2017 Chinese Automation Congress (CAC)*, Oct. 2017, pp. 4395–4399. doi: 10.1109/CAC.2017.8243553.

[27] A. N. C. L. A. J. L. and J. H. M. Magnusson, “Evaluation of 3D registration reliability and speed - A comparison of ICP and NDT,” *IEEE International Conference on Robotics and Automation*, pp. 3907–3912, May 2009.

[28] P. Biber and W. Strasser, “The normal distributions transform: a new approach to laser scan matching,” in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, Oct. 2003, pp. 2743–2748 vol.3. doi: 10.1109/IROS.2003.1249285.

[29] A. Myronenko and X. Song, “Point-Set Registration: Coherent Point Drift,” *IEEE Trans Pattern Anal Mach Intell*, vol. 32, no. 12, pp. 2262–2275, Dec. 2010, doi: 10.1109/TPAMI.2010.46.

[30] A. Jurić, F. Kendeš, I. Marković, and I. Petrović, “A Comparison of Graph Optimization Approaches for Pose Estimation in SLAM,” in *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, Sep. 2021, pp. 1113–1118. doi: 10.23919/MIPRO52101.2021.9596721.

[31] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3607–3613. doi: 10.1109/ICRA.2011.5979949.

[32] D. Barnes and I. Posner, “Under the Radar: Learning to Predict Robust Keypoints for Odometry Estimation and Metric Localisation in Radar,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 9484–9490. doi: 10.1109/ICRA40945.2020.9196835.

[33] S. H. Cen and P. Newman, “Radar-only ego-motion estimation in difficult settings via graph matching,” Apr. 2019, *arXiv*. doi: 10.48550/arXiv.1904.11476.

[34] S. H. Cen and P. Newman, “Precise Ego-Motion Estimation with Millimeter-Wave Radar Under Diverse and Challenging Conditions,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6045–6052. doi: 10.1109/ICRA.2018.8460687.

[35] J. Zhang *et al.*, “4DRadarSLAM: A 4D Imaging Radar SLAM System for Large-scale Environments based on Pose Graph Optimization,” in *2023 IEEE International Conference on*

Robotics and Automation (ICRA), May 2023, pp. 8333–8340. doi: 10.1109/ICRA48891.2023.10160670.

[36] Z. Hong, Y. Petillot, A. Wallace, and S. Wang, "Radar SLAM: A Robust SLAM System for All Weather Conditions," Apr. 2021, *arXiv*. doi: 10.48550/arXiv.2104.05347.

[37] D. Wang, S. May, and A. Nuechter, "RIV-SLAM: Radar-Inertial-Velocity optimization based graph SLAM," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, Aug. 2024, pp. 774–781. doi: 10.1109/CASE59546.2024.10711511.

[38] M. Mielle, M. Magnusson, and A. J. Lilienthal, "A comparative analysis of radar and lidar sensing for localization and mapping," in *2019 European Conference on Mobile Robots (ECMR)*, Sep. 2019, pp. 1–6. doi: 10.1109/ECMR.2019.8870345.

[39] D. C. Herraez, M. Zeller, L. Chang, I. Vizzo, M. Heidingsfeld, and C. Stachniss, "Radar-Only Odometry and Mapping for Autonomous Vehicles," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 10275–10282. doi: 10.1109/ICRA57147.2024.10610311.

[40] E. Sie, X. Wu, H. Guo, and D. Vasisht, "Radarize: Enhancing Radar SLAM with Generalizable Doppler-Based Odometry," Jun. 2024, pp. 331–344. doi: 10.1145/3643832.3661871.

[41] R. M. Buehrer, H. Wymeersch, and R. M. Vaghefi, "Collaborative Sensor Network Localization: Algorithms and Practical Issues," *Proceedings of the IEEE*, vol. 106, no. 6, pp. 1089–1114, 2018, doi: 10.1109/JPROC.2018.2829439.

[42] P. Yang et al., "Multi-Sensor Multi-Vehicle (MSMV) Localization and Mobility Tracking for Autonomous Driving," *IEEE Trans. on Vehicular Technology*, vol. 69, no. 12, pp. 14355–14364, 2020, doi: 10.1109/TVT.2020.3031900.

[43] N. Piperigkos et al., "Extending Online 4D Situational Awareness in Connected and Automated Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 8, pp. 5316–5335, 2024, doi: 10.1109/TIV.2023.3335605.

[44] J. Chen and A. H. Sayed, "Diffusion Adaptation Strategies for Distributed Optimization and Learning Over Networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, 2012, doi: 10.1109/TSP.2012.2198470.

[45] C. Anagnostopoulos, C. Koulamas, A. Lalos, and C. Stylios, "Open-Source Integrated Simulation Framework for Cooperative Autonomous Vehicles," in *2022 11th Mediterranean Conference on Embedded Computing*, 2022, pp. 1–4. doi: 10.1109/MECO55406.2022.9797115.

[46] M. Elazab, A. Noureldin, and H. S. Hassanein, “Integrated cooperative localization for Vehicular networks with partial GPS access in Urban Canyons,” *Vehicular Communications*, vol. 9, pp. 242–253, 2017, doi: 10.1016/j.vehcom.2016.11.011.

[47] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[48] J. Piatkowski, L. Karbowiak, and F. Depta, “A lightweight algorithm for synchronized multimodal data acquisition using temporal sample alignment,” *Sci Rep*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-07797-7.

[49] M. Ciesla and G. Ostermayer, “A Multimodal Recurrent Model for Driver Distraction Detection,” *Applied Sciences (Switzerland)*, vol. 14, no. 19, Oct. 2024, doi: 10.3390/app14198935.

[50] K. Jiang *et al.*, “Adding ears to intelligent connected vehicles by combining microphone arrays and high definition map,” *IET Intelligent Transport Systems*, vol. 15, no. 10, pp. 1228–1240, Oct. 2021, doi: 10.1049/itr2.12091.

[51] A. Savran, “Fully Convolutional Event-camera Voice Activity Detection Based on Event Intensity,” in *2023 Innovations in Intelligent Systems and Applications Conference, ASYU 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ASYU58738.2023.10296754.

[52] J. Feng, P. Zhao, H. Zheng, J. Konpang, A. Sirikham, and P. Kalnaowakul, “Enhancing Autonomous Driving Perception: A Practical Approach to Event-Based Object Detection in CARLA and ROS,” *Vehicles*, vol. 7, no. 2, p. 53, May 2025, doi: 10.3390/vehicles7020053.

[53] W. Shariff, M. S. Dilmaghani, P. Kielty, M. Moustafa, J. Lemley, and P. Corcoran, “Event Cameras in Automotive Sensing: A Review,” *IEEE Access*, vol. 12, pp. 51275–51306, 2024, doi: 10.1109/ACCESS.2024.3386032.

[54] J. Wang and L. Xia, “Abnormal behavior detection in videos using deep learning,” *Cluster Comput*, vol. 22, no. S4, pp. 9229–9239, Jul. 2019, doi: 10.1007/s10586-018-2114-2.