POLITECNICO DI TORINO

Master's Degree in ICT FOR SMART SOCIETIES

Master's Degree Thesis

Multi-Object Tracking and Speed Bump Detection based on LiDAR Point Cloud

Supervisors

Prof. ANDREA TONOLI

Prof. ALDO SORNIOTTI

Research Fellow MENG XIE

PhD STEFANO FAVELLI

PhD RAFFAELE MANCA

Candidate

WENXIN SHAO

OCTOBER 2025

Abstract

Multi-object tracking and road feature detection are critical perception tasks in Advanced Driver Assistance Systems (ADAS), as they directly affect perception robustness and driving safety. This paper proposes algorithms for target tracking and speed bump detection based on solid-state LiDAR point clouds.

For multi-object tracking, an Extended Kalman Filter (EKF) with a Constant Turn Rate and Velocity (CTRV) motion model is used to estimate full target states including position, velocity, heading, and yaw rate. By performing inter-frame data association and state estimation on solid-state LiDAR data, the method enables robust tracking of dynamic targets. A Markov chain–based track management strategy ensures reliable initiation, maintenance, and termination of tracks in complex traffic environments.

For speed bump detection, this paper proposes a method based on local point cloud height variations. Edge points are first extracted from ground point clouds using local height differences and then clustered to derive geometric features including position, size, slope, and confidence. These features are incorporated into a geometric decision model designed for sparse LiDAR data, enabling accurate and reliable speed bump localization. Validated detections are published via ROS topics, integrated into the Autoware perception pipeline, and transmitted through CAN messages for vehicle control support.

The algorithms were developed in ROS2 Humble, deployed on NVIDIA® Jetson AGX Orin™ platform, and tested on a PIX-KIT 2.0 platform with verification in the Autoware environment. Real-world experiments demonstrate accurate tracking, robust speed bump detection, and efficient runtime performance. This work provides a practical perception solution for low-cost intelligent driving systems and broadens the application of LiDAR in static road structure recognition.

Keywords: LiDAR Perception, Multi-Object Tracking, Extended Kalman Filter, Speed Bump Detection, ROS2, Autoware

Table of Contents

Chapter 1

Introduction

1.1 Background

In recent years, with the rapid advancement of artificial intelligence, sensor technology, and automatic control theory, intelligent driving technology has emerged as one of the core fields in global technological innovation. Intelligent driving systems aim to achieve automated vehicle operation through multi-sensor fusion for environmental perception, precise decision-making and planning, and stable control execution. Their ultimate objectives are to enhance road traffic safety, improve traffic efficiency, and reduce energy consumption. According to SAE International standards, environmental perception and dynamic target management remain core technological bottlenecks throughout the evolution of intelligent driving systems from Level 1 (L1) driver assistance to Level 5 (L5) full autonomy.[1]

As reported in Figure 1.1, the SAE Levels of Driving Automation consist of six distinct levels, each representing a progressive level of autonomy:

- Level 0 - No Automation: The driver has full control of the vehicle. Warnings and momentary assistance such as emergency braking and blind spot warning are provided, but no intervention.

- Level 1 - Driver Assistance: The vehicle incorporates basic driver assistance features of steering or speed management support, such as adaptive cruise control or lane-keeping assistance. However, the driver remains fully responsible for vehicle operation and must monitor the driving environment.

- Level 2 - Partial Automation: The vehicle can control both steering and speed management under specific conditions. The driver is still responsible for monitoring the driving environment and must be ready to take control at any time. This is made possible by Advanced Driver Assistance Systems(ADAS).

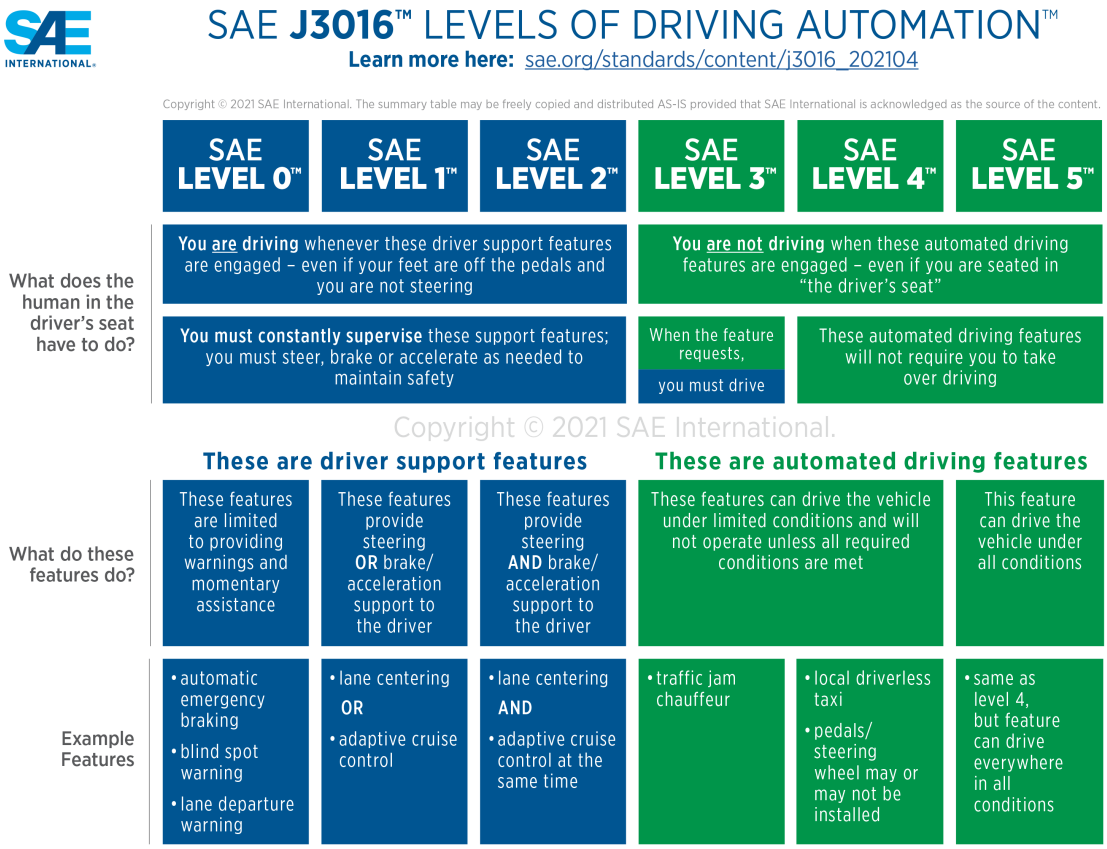

Figure 1.1: SAE Levels of Driving Automation™

- Level 3 - Conditional Automation: The vehicle can manage most aspects of the driving task under specific conditions. However, the driver must be prepared to intervene when alerted by the system to resume control.

- Level 4 - High Automation: The vehicle can perform all driving tasks within defined operational domains and conditions without driver intervention. However, the system may require the driver to take over in exceptional circumstances.

- Level 5 - Full Automation: The vehicle is capable of performing all driving tasks under all conditions, and the driver is not required to be involved in the driving process. Level 5 vehicles are fully autonomous and do not require human intervention.[2]

As the "eyes" of intelligent vehicles, environmental perception systems are required to accurately and reliably identify both static road obstacles—such as

speed bumps, manhole covers, and construction zones—and dynamic objects, including pedestrians, vehicles, and cyclists, in real time. These systems provide essential information regarding object position, velocity, and geometry, which is fundamental to higher-level perception and planning modules. Among the numerous perception tasks, dynamic multi-object tracking and static road obstacle detection represent two critical challenges that directly influence the reliability and robustness of vehicle decision-making processes. To ensure safe navigation, intelligent vehicles must not only detect surrounding objects but also predict their future motion states through consistent tracking. Continuous, stable, and accurate tracking of dynamic targets such as vehicles and pedestrians forms the basis for trajectory prediction and collision risk assessment. In addition to dynamic entities, complex road environments contain numerous static yet safety-critical features. For instance, speed bumps, as common traffic-calming infrastructure, must be detected accurately and in a timely manner to prevent sudden shocks, enable proactive suspension control, or trigger autonomous deceleration when required—thereby enhancing both vehicle stability and passenger safety.[3]

This thesis work relates to the perception tasks of two ADAS projects. One is the MOST – Centro Nazionale per la Mobilità Spostenibile under the SPOKE 2 – Sustainable Road Vehicle project, which consists of the development of algorithms for the optimization of the Energy Flow Management based on the environment perception through ADAS sensors. The activity aims at providing a scalable framework to improve the energy efficiency of Battery Electric Vehicle (BEV) and Fuel Cell Electric Vehicle (FCEV) demonstrators through the optimization of the Energy Management System (EMS) based on ADAS sensors' measurements. Our task aims to develop a LiDAR-based perception module for multi-object tracking and trajectory management based on the existing sensor fusion perception system.[4]

The other one is the Traction Control System with Road Preview Information project, which is an advanced traction control solution for electric, combustion, and hybrid vehicles, based on a nonlinear model predictive controller (NMPC). Unlike conventional TCS, which intervene only once slip has occurred, the system anticipates traction losses by modulating drive torque based on predictive information from ADAS sensors. The system relies on a prediction model that accurately describes the tire–road contact, including vertical load variations caused by road surface irregularities. This allows critical situations that could compromise stability to be prevented. Our task aims to develop a LiDAR-based perception module for detecting and measuring speed bumps ahead of the vehicle in real time, enabling proactive adjustment of parameters in the Active suspension systems (ASS). The system improves vehicle ride comfort by preparing the suspension system in advance for upcoming road irregularities.

The integration of ADAS with EMS and ASS presents a promising direction for

enhancing overall vehicle efficiency, reducing fuel or electricity consumption, and improving both ride comfort and driving safety. For instance, terrain recognition and driving prediction provide essential information for the control algorithms in both systems. Through the use of perception sensors such as radar, LiDAR, and cameras, ADAS is capable of capturing comprehensive information about the surrounding environment, including static features—such as road gradients, curves, speed bumps, and minor obstacles—as well as dynamic objects, including other vehicles, bicycles, and pedestrians. This information can be exploited by the EMS to anticipate energy demands and optimize power utilization, for instance by adapting energy consumption to varying road slopes or anticipated braking events. Furthermore, such data can support the active suspension system in real-time adaptation, including dynamic adjustments of damper stiffness or independent chassis height control at each wheel, thereby enhancing vehicle stability and passenger comfort under diverse driving conditions.

In this study, two core functional modules of ADAS were developed based on LiDAR point cloud data: (1) multi-object tracking and trajectory management using an Extended Kalman Filter (EKF), and (2) speed bump detection through local height difference analysis of ground point clouds. An efficient multi-object state estimation and tracking framework was designed to improve the accuracy of trajectory prediction for dynamic traffic participants. Moreover, optimized point cloud feature extraction and geometric parameter computation methods enable precise detection and characterization of small road obstacles,[5] such as speed bumps, with minimal height variation from the ground surface. Together, these techniques enhance the environmental perception capability of intelligent driving systems and provide reliable inputs for subsequent decision-making and motion planning modules. The proposed methods possess both significant theoretical relevance and strong potential for practical application.

1.2 Motivation

In current ADAS systems, the detection of static obstacles (such as speed bumps, curbs, potholes, etc.) mainly relies on visual data or predefined high-definition maps, failing to fully leverage real-time point cloud data, where ground points are typically filtered out in the preprocessing stage. Meanwhile, dynamic multi-object tracking predominantly focuses on vehicles and pedestrians, lacking robust handling of concurrent multi-object scenarios in complex road structures. Furthermore, most existing systems design static and dynamic perception modules independently, lacking a unified framework from low-level point cloud processing to high-level perception. This results in high system redundancy, poor real-time performance, and limited scalability.

With the rapid advancement of autonomous driving and ADAS technologies, a vehicle's environmental perception capabilities have become the core foundation for achieving safe and efficient autonomous driving. As a key sensor in environmental perception systems, LiDAR demonstrates unique advantages in complex traffic scenarios due to its high-precision 3D spatial detection capabilities.[6] However, single-sensor solutions still face limitations in real-world road environments. Cameras are highly susceptible to lighting and weather conditions, while radar struggles with target classification and geometric detail perception. Consequently, effectively utilizing LiDAR point cloud data to achieve obstacle detection and dynamic target tracking across diverse scenarios has become a prominent research challenge.

This thesis focuses on LiDAR point cloud data processing, addressing two representative road perception tasks: static road feature recognition (e.g., speed bump detection) and dynamic multi-object tracking. Although these tasks serve different application purposes, they share a common technical foundation based on point cloud feature extraction, clustering analysis, and state estimation. Accordingly, both are developed within a unified methodological framework that leverages LiDAR perception for comprehensive environmental understanding in intelligent vehicles.

1.3 Research Objectives

This study aims to establish a unified perception framework based on LiDAR point cloud data. It develops two independent yet technologically complementary algorithmic modules: a speed bump detection system and a multi-object tracking system. Through these modules, the study demonstrates the feasibility and effectiveness of point cloud technology for detecting both static and dynamic obstacles. The specific objectives of this work are as follows:

- Develop a high-precision static obstacle detection model capable of real-time identification and parameter estimation of road features such as speed bumps, based on point cloud height difference analysis, edge clustering, and geometric feature extraction.

- Design an efficient multi-object tracking algorithm using the Extended Kalman Filter (EKF) and the Constant Turn Rate and Velocity (CTRV) motion model to enable continuous tracking and trajectory prediction of dynamic targets.

- Explore common point cloud processing techniques for static and dynamic perception, including filtering, clustering, and feature representation, in order to establish a modular technical foundation for multi-task perception systems.

- Promote the embedded deployment of LiDAR perception in real-vehicle systems to improve system practicality, real-time performance, and robustness in real-world driving environments.

1.4 Thesis Outline

This thesis work is structured as follows:

- Chapter 2 presents the theoretical background of the topics presented, with a particular focus on LiDAR point cloud processing techniques and object tracking algorithms.
- Chapter 3 is dedicated to the design of system architecture and the implementation of the proposed method. Starting from the hardware architecture overview, and the development of the proposed algorithms is discussed, with particular focus on implementation and integration of hardware and software.
- Chapter 4 presents the setup of experimental validation process and the evaluation of the obtained results.
- Chapter 5 is the final chapter, where conclusions and future works are reported.

Chapter 2

Theoretical Background

This chapter aims to explain the core theoretical background relevant to the work presented in this thesis. It begins by introducing fundamental concepts of environmental perception and their significance in autonomous driving systems. Subsequently, it focuses on reviewing the working principles of LiDAR and key technologies for point cloud data processing. Finally, it delves into the theoretical foundations underpinning the two core algorithmic modules: point cloud geometric feature analysis for static obstacle detection, and state estimation algorithms for dynamic target tracking.

2.1 Environmental perception overview

Environmental perception forms the foundation for ADAS and autonomous vehicles to understand their surroundings and make decision-planning. Its core task lies in accurately identifying and locating both static elements such as lane markings, traffic signs, speed bumps and dynamic elements such as vehicles, pedestrians within the environment, while estimating their motion states. A robust perception system must deliver real-time performance, high accuracy, and adaptability to complex scenarios and adverse weather conditions.[7]

LiDAR directly acquires high-precision 3D point cloud data of the surrounding environment by emitting laser beams and measuring their return time. Compared to cameras, LiDAR is less susceptible to lighting variations and provides precise distance information. Compared to radar, LiDAR offers higher angular and distance resolution, enabling better delineation of target geometric contours. Consequently, LiDAR has become one of the key sensors for achieving high-precision environmental perception[8]. This study focuses on utilizing LiDAR point cloud data to address two core challenges: static road element recognition and dynamic object tracking.

2.2 Working Principles of LiDAR sensors

Light Detection and Ranging (LiDAR) is an active optical remote sensing technology that determines the distance, bearing, and surface characteristics of target objects by emitting laser beams and analysing their return signals. Its core principle is the time-of-flight method, which precisely calculates the relative distance between the sensor and the target by measuring the time taken for a laser pulse to travel from emission to reception by the sensor, combined with the constant speed of light. By emitting hundreds of thousands to millions of laser pulses per second and conducting precise measurements of each returning pulse, the system constructs high-density, high-precision three-dimensional point cloud maps of the surrounding environment in real time. This technical capability overcomes the resolution limitations of traditional radar and addresses the shortcomings of camera sensors in depth perception and their dependence on ambient lighting. Consequently, LiDAR has become an indispensable core component for achieving the environmental perception layer in advanced autonomous driving systems. As autonomous driving technology evolves from driver assistance towards full autonomy, the rich three-dimensional geometric information provided by LiDAR forms the cornerstone for achieving precise positioning, obstacle recognition, path planning, and decision control.[9]

The performance of LiDAR systems is determined by a series of key technical parameters, which are interrelated and mutually constraining, forming the core considerations in system engineering design.

Wavelength represents one of the most fundamental design parameters in LiDAR systems, as it directly influences eye safety, power efficiency, and detection performance under various environmental conditions. Modern automotive LiDAR systems primarily operate in two infrared bands: 905 nm and 1550 nm. The 905 nm band benefits from mature and cost-effective photodetector technology. However, its proximity to the visible spectrum means that its energy can be readily absorbed by the human retina. As a result, the emission power must be strictly limited to comply with eye safety standards such as IEC 60825. In contrast, light at 1550 nm is largely absorbed by the aqueous humour and lens before reaching the retina, allowing for significantly higher permissible emission power and thus extended detection range. Nevertheless, this same water absorption characteristic becomes a drawback under adverse weather conditions—such as rain, fog, or snow—where the signal attenuation of 1550 nm lasers is considerably higher than that of 905 nm lasers, severely reducing effective detection range.[10] To compensate, 1550 nm systems generally require higher power consumption. Therefore, wavelength selection in LiDAR design entails a complex trade-off among safety, all-weather performance, and energy efficiency.

Detection range is the most critical performance metric for LiDAR systems,

though its definition requires thorough understanding. Maximum detection range typically denotes the capability to detect highly reflective targets (such as traffic signs with reflectivity >80%) under ideal conditions. However, the metric of greater engineering relevance is the ranging capability based on a specific reflectivity level (usually 10%). A 10% reflectivity level approximates the reflective characteristics of black tyres or dark clothing. This metric more accurately reflects a LiDAR system's capability to detect the most hazardous, low-visibility obstacles within complex road environments. When evaluating products, it is crucial to clearly distinguish between these two distance definitions. A LiDAR unit claiming a maximum detection range of 300 metres may only detect a 10% reflectivity target at 150 metres. The latter distance is the reliable basis for ADAS functional safety design. [11]

The next performance metric we shall discuss concerns the resolution, point frequency, and perception accuracy of LiDAR. LiDAR resolution encompasses angular resolution and point cloud density. Angular resolution determines the minimum angular difference between two adjacent targets that the LiDAR can distinguish; higher resolution yields greater capability in capturing target details. Point cloud density is directly determined by point frequency (the number of laser points emitted per second). A high point frequency enables the acquisition of more data points within the same field of view, thereby generating a finer and more precise environmental model. A high point frequency is crucial for identifying pedestrians and small obstacles at long range, or accurately reconstructing details such as lane markings and kerbs. It forms the foundation for enhancing the accuracy of perception algorithms. Furthermore, the scanning frame rate (typically 10-20 Hz) impacts the system's real-time environmental perception capabilities; a higher frame rate facilitates the close tracking of rapidly moving objects.[11]

In addition, the field of view (FoV) and system layout strategy are crucial factors influencing LiDAR perception performance. The FoV defines the spatial coverage of the sensor, encompassing both horizontal and vertical directions. Mechanical rotating LiDAR systems can achieve a full 360° horizontal FoV, providing comprehensive environmental perception. In contrast, solid-state LiDAR systems are limited by their underlying optical and electronic design, typically offering a horizontal FoV of less than 120°. Consequently, multiple solid-state LiDAR units are often strategically mounted at the front, rear, and sides of the vehicle to ensure complete coverage. Through sensor fusion, the data streams from these units are integrated to construct a continuous and panoramic perception field. The vertical FoV, typically ranging from 25° to 40°, determines the sensor's ability to detect objects at varying heights, such as nearby ground obstacles, distant bridges, and overhead traffic signals..[11]

The technical classification system for LiDAR primarily categorizes it based on beam manipulation methods, dividing it into three major types: mechanical

rotating, hybrid solid-state, and fully solid-state. Each technology path exhibits significant differences in implementation principles, performance characteristics, and applicable scenarios, collectively forming a diverse landscape of LiDAR technological evolution.[12]

Mechanical rotating LiDAR employs a classic rotating mirror optical structure, where an electric motor drives an optical prism or mirror to perform continuous 360-degree rotation, enabling panoramic scanning of the surrounding environment. The core advantage of such systems lies in their unparalleled FOV coverage capability, with a single sensor achieving a complete horizontal field of view of 360 degrees and a vertical field of view ranging from 20 to 45 degrees. Performance-wise, mechanical LiDAR typically delivers high signal-to-noise ratios and extended detection ranges due to its independent transmit/receive channels and large-aperture optical systems. However, its technical limitations are equally pronounced: the presence of precision mechanical components results in bulky, high-cost designs that struggle to meet automotive-grade vibration, shock, and durability requirements. Additionally, mechanical wear on rotating components compromises long-term reliability, while limited scan line counts constrain further improvements in point cloud density. These systems are currently deployed primarily in cost-insensitive professional applications such as autonomous vehicle testing and geographic surveying.[13]

Hybrid solid-state LiDAR, exemplified by micro-electro-mechanical systems (MEMS) technology, strikes a balance between mechanical and fully solid-state approaches. MEMS LiDAR achieves beam scanning by controlling the deflection angle of microscopic mirrors, which measure only millimeters in size. These mirrors are driven electrostatically or electromagnetically to achieve vibration frequencies ranging from tens of thousands to hundreds of thousands of hertz. This design ingeniously replaces macroscopic rotating structures with the limited mechanical motion of micromirrors, preserving beam direction flexibility while significantly reducing system size. The scanning patterns of MEMS systems can be optimized through mirror design to enable intelligent sampling strategies like non-uniform scanning. However, its primary technical challenge lies in the reliability of the micromirrors under the harsh demands of automotive environments, including temperature cycling, mechanical vibration, and shock. Simultaneously, the small mirror size limits the optical aperture, restricting received light energy and affecting detection range performance. Currently, MEMS technology is continuously enhancing its automotive-grade reliability through material innovation and structural optimization.[14]

Pure solid-state LiDAR represents the ultimate direction of technological evolution, primarily encompassing three core technical pathways: optical phased array (OPA), flash-based, and frequency-modulated continuous wave (FMCW) LiDAR. The optical phased array approach emits coherent laser beams through an array of optical antennas, precisely controlling the phase of each antenna element to

synthesize directional beams, thereby enabling electronic beam steering without any mechanical components. This technology achieves microsecond-level scanning speeds with outstanding reliability and stability; however, it faces inherent challenges such as sidelobe suppression and limited scanning angles. Flash-based LiDAR, on the other hand, operates analogously to a camera flash, employing a wide-area light source to illuminate the entire scene in a single pulse while a two-dimensional detector array simultaneously captures the reflected signals. This architecture eliminates all scanning mechanisms, offering excellent vibration resistance and high frame-rate capability, but it also imposes constraints related to peak optical power, environmental light interference, and the high cost of large-scale detector arrays. FMCW LiDAR adopts a continuous-wave laser with frequency modulation, determining distance by measuring the frequency difference between transmitted and received signals and utilizing the Doppler effect to extract velocity information. This technique provides single-photon-level sensitivity and superior resistance to ambient light interference, yet it entails higher system complexity and cost due to its intricate modulation and signal processing requirements.[11]

Various LiDAR technologies exhibit distinct performance gradients. In scanning methods, mechanical systems rely on macro-scale mechanical rotation, MEMS systems depend on micro-mirror vibration, while fully solid-state systems achieve complete electronic operation. Regarding reliability, mechanical systems are constrained by wear of moving parts, MEMS must overcome durability issues in micro-mechanical structures, while pure solid-state technology theoretically offers the highest reliability. In scanning accuracy, mechanical systems are limited by mechanical assembly precision, MEMS depends on micromirror control accuracy, and pure solid-state technology enables electronic-level precision control. Cost structures vary significantly: mechanical systems remain expensive due to precision optics, MEMS costs are expected to decline with semiconductor process maturation, while pure solid-state technology may ultimately achieve chip-level integration cost advantages.

These technical differences directly dictate their respective application scenarios. Mechanical LiDAR, leveraging its panoramic scanning capability, retains value in robotics and intelligent transportation systems. MEMS LiDAR demonstrates strong cost-performance advantages in the automotive OEM market, making it the mainstream choice for current mass-production projects. While pure solid-state technology is not yet fully mature, its significant potential in reliability, size, and cost positions it as a key future development direction for autonomous driving sensors. This diversification of technological pathways is driving LiDAR technology toward continuous evolution toward higher performance, lower cost, and enhanced reliability.

In practical autonomous driving applications, the integration of LiDAR sensors requires a comprehensive evaluation of multiple interrelated factors. Among

these, environmental adaptability constitutes the foremost challenge. Under adverse weather conditions—such as rain, fog, or snow—laser signal attenuation substantially reduces the effective detection range, thereby demanding greater robustness from perception algorithms. To address these limitations, the deep integration of LiDAR with millimeter-wave radar and vision sensors has become an inevitable trend, exploiting the complementary characteristics of multi-source data to construct redundant and reliable perception systems. Millimeter-wave radar, known for its stability in velocity measurement and resilience under poor weather conditions, effectively complements the high-resolution spatial perception provided by LiDAR point clouds. Meanwhile, vision sensors contribute rich texture and semantic information, compensating for LiDAR's inherent limitations in object classification and scene interpretation. Furthermore, industrialization of LiDAR technology must also satisfy stringent automotive-grade requirements while maintaining cost efficiency. The sensors are required to meet rigorous standards in temperature tolerance, vibration resistance, and long-term durability, while continual technological innovation is essential to further reduce production costs and enable large-scale deployment in mass-produced vehicles.[15]

As autonomous driving technology advances to higher levels, LiDAR will continue to play a central role in environmental perception. Its technological evolution focuses on continuous improvements in performance, cost optimization, and system integration. The maturation of solid-state technology will drive LiDAR toward smaller form factors, lower power consumption, and higher reliability, while deep integration with artificial intelligence algorithms will further enhance its ability to understand complex scenarios. Within future intelligent transportation systems, LiDAR—as a key sensor—will deeply integrate with technologies like high-precision positioning and vehicle-infrastructure coordination to jointly build a safe and efficient mobility ecosystem.[12]

2.3 LiDAR point cloud processing principles

2.3.1 LiDAR Point Cloud Characteristics and Preprocessing

Raw LiDAR point cloud data typically contains numerous noise points (e.g., reflections from airborne dust) and points from regions of no interest (e.g., the ground). Therefore, point cloud preprocessing serves as the initial step for subsequent analysis, primarily encompassing:

- Filtering: Employing statistical filtering or radius filtering methods to remove outlier noise points.

- Ground Segmentation: Separating ground points from non-ground points (obstacle points) through grid-based or planar model fitting (e.g., RANSAC), significantly reducing data volume and focusing on potential targets.
- Downsampling: Employing methods like voxel grid filtering to reduce point density while preserving the overall shape of the point cloud, thereby enhancing subsequent processing efficiency.[16]

2.3.2 Point Cloud Clustering and Feature Extraction

The preprocessed point cloud must be segmented into distinct object instances. Euclidean clustering is one of the most commonly used unsupervised clustering methods. It groups points within a specified distance threshold into the same object based on their spatial proximity. This method is simple and efficient, suitable for separating discrete obstacles in a scene. Following clustering, geometric features must be extracted from each point cloud cluster to identify its category (e.g., vehicle, pedestrian, speed bump). For static obstacle detection, key geometric features include:

- Size features: Extracting the length, width, and height of the captured point cloud cluster in three-dimensional space.
- Shape features: Eigenvalues calculated via Principal Component Analysis (PCA) or similar methods reflect object geometry (e.g., linear, planar, volumetric).
- Height features: Local point cloud elevation differences relative to the ground surface and slope gradients serve as critical indicators for identifying road surface protrusions like speed bumps.

2.3.3 Static Obstacle Detection

Static obstacle detection in environmental perception is one of the critical tasks for ensuring the safe operation of autonomous vehicles. These obstacles include, but are not limited to, curbs, guardrails, construction cones, and speed bumps—the primary focus of this paper. Object detection algorithms can be broadly categorized into two main types based on the data modality they process: image-based methods and point cloud-based methods. An effective detection approach must accurately identify the geometric properties of these targets and determine their precise locations.

1. Image-Based Object Detection: These methods leverage mature 2D convolutional neural networks (CNNs), such as the YOLO series [17] and Faster R-CNN [18], to directly predict object bounding boxes and categories from camera images.

Their advantage lies in utilizing rich texture and color information. However, this approach heavily relies on lighting and weather conditions. Moreover, accurately recovering an object's 3D position and scale from monocular images is an ill-posed problem, resulting in limited depth estimation accuracy.

2. Point Cloud-Based Object Detection: The three-dimensional point cloud data provided by LiDAR inherently contains geometric and spatial information about objects, enabling precise 3D detection. Current mainstream approaches can be further categorized into:

- Voxel-based methods: Examples include VoxelNet[19], which converts irregular point clouds into regular voxel grids before applying 3D CNNs for feature extraction and detection. These methods effectively handle point cloud disorder, but voxelization introduces quantization errors and incurs high computational costs for 3D convolutions.

- Point-based methods: Examples like PointRCNN[20] operate directly on irregular point clouds, utilizing networks such as PointNet++[21] to extract point features. These methods maximize preservation of the original point cloud's geometric information but are sensitive to point cloud irregularities and sparsity.

- Projection-based methods: Project the point cloud onto a 2D plane (e.g., front view, bird's-eye view), then apply efficient 2D CNNs for detection, such as PIXOR[6]. This approach strikes a good balance between speed and performance, though the projection process inevitably results in some loss of 3D information.[22]

Although the aforementioned general-purpose 3D detectors have achieved significant success on common targets like vehicles and pedestrians, they still face specific challenges when applied to specialized static road elements like speed bumps:

- Unique geometric characteristics: Speed bumps appear in point clouds as elongated, raised regions with specific height and width, rather than enclosed 3D objects. General-purpose 3D detectors, typically designed for cuboid or vehicle-shaped targets, may struggle to optimally capture their flat, elongated geometry.

- Point cloud sparsity: Due to their relatively small size and proximity to the ground, LiDAR scans often yield highly sparse point clouds for speed bumps, particularly at long distances. This poses feature extraction difficulties for deep learning-based detectors.

- Data annotation and model dependency: Deep learning methods typically require large volumes of precisely annotated data for training. For long-tail

distribution targets like speed bumps, annotated data scarcity limits model generalization capabilities.[23, 24]

To address these challenges, some studies have begun exploring model-free or lightweight approaches based on geometric features, directly leveraging the inherent geometric properties of point clouds for recognition.[25] proposed a road obstacle detection method based on point cloud height maps and multi-scale feature fusion. [26]employed point cloud clustering and regular shape fitting to detect curbs and potholes. These studies demonstrate that for specific targets with distinct geometric features, rule-based approaches can serve as an efficient, highly interpretable complement to data-driven methods. Inspired by the aforementioned geometry-based approaches and considering the specificity of speed bumps, this paper proposes a lightweight detection algorithm specifically designed for speed bumps, independent of complex deep learning models. The core idea is that the most prominent feature of speed bumps in point clouds is their local height abrupt changes relative to the flat road surface. The algorithm flow is illustrated as follows: figure added

It primarily consists of the following three steps:

1. Height Difference Edge Point Detection: First, perform radius-based nearest neighbor search on the preprocessed point cloud (e.g., after ground segmentation). For each point, calculate the maximum height difference among its neighboring points. If this height difference exceeds a preset threshold, the point is identified as located within a region of abrupt height change and marked as a candidate edge point. This step effectively captures the front and rear edges of speed bumps.

2. Edge Point Clustering and Analysis: Subsequently, spatially adjacent edge points are grouped using Euclidean clustering to form candidate speed bump instances. For each cluster, geometric properties are calculated, including length, width, average height difference, and dominant orientation.

3. Classification Based on Geometric Rules: Finally, clusters are filtered and classified according to predefined geometric rules. For example, a genuine speed bump should satisfy: length significantly greater than width (elongated shape), average height difference within a specified range, and its primary direction should be roughly parallel to the road direction. Clusters meeting these conditions are ultimately confirmed as speed bumps.

Compared to existing approaches, the speed bump detection algorithm proposed in this paper exhibits the following distinctive features: Designed specifically for the geometric characteristics of speed bumps, it detects them based on the core feature of height difference. The algorithm features clear logic and strong interpretability,

facilitating debugging and optimization. As a model-free method, it does not rely on large amounts of labeled data for training, thereby avoiding the costs associated with data collection and annotation while addressing the scarcity of long-tail target data. It demonstrates excellent generalization capabilities. Furthermore, the algorithm's workflow is based on classical point cloud processing operations (neighborhood search, clustering), resulting in low computational complexity. This makes it highly suitable for real-time operation on embedded platforms with limited computational power, such as in-vehicle computing units. Additionally, the algorithm not only determines the presence of speed bumps but also outputs detailed geometric parameters including precise 3D position, length, width, and height. This information is crucial for vehicle comfort control, such as anticipatory deceleration. In summary, the speed bump detection algorithm presented herein offers an efficient and practical solution distinct from general-purpose deep learning detectors. It is particularly well-suited for stable and reliable detection of specific types of static obstacles in real-world automotive environments.

2.4 Introduction to the Extended Kalman Filter (EKF)

2.4.1 Extended Kalman Filter (EKF) basic introduction

The Extended Kalman Filter (EKF) is an extension of the Kalman Filter (KF), designed to handle nonlinear systems.The standard Kalman Filter assumes that both the state transition model and the observation model are linear.However, in most real-world systems (like robotics, target tracking, or vehicle navigation), the system dynamics are nonlinear.The EKF solves this by linearizing the nonlinear functions around the current estimate using first-order Taylor expansion.So, the EKF is basically a linearized version of the Kalman Filter that can handle nonlinear motion and measurement models.[27]

2.4.2 EKF System Model

A general discrete-time nonlinear system can be described as:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \tag{2.1}$$

$$z_k = h(x_k) + v_k \tag{2.2}$$

where:

- x_k is the state vector at time step k.
- u_{k-1} is the control input.

- z_k is the measurement vector.
- $f(\cdot)$ is the nonlinear state transition function.
- $h(\cdot)$ is the nonlinear measurement function.
- w_{k-1} and v_k are process and measurement noise, assumed to be zero-mean Gaussian:

$$w_{k-1} \sim \mathcal{N}(0, Q_k), \quad v_k \sim \mathcal{N}(0, R_k)$$

2.4.3 EKF Algorithm

The EKF operates in two main stages: **prediction** and **update**.[28]

Prediction Step

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_{k-1}) \tag{2.3}$$
$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^\top + Q_k \tag{2.4}$$

where F_k is the Jacobian of f with respect to the state:

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1|k-1}, u_{k-1}} \tag{2.5}$$

Update Step

When a new measurement z_k is available:

$$\hat{z}_k = h(\hat{x}_{k|k-1}) \tag{2.6}$$
$$H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k|k-1}} \tag{2.7}$$
$$K_k = P_{k|k-1} H_k^\top (H_k P_{k|k-1} H_k^\top + R_k)^{-1} \tag{2.8}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - \hat{z}_k) \tag{2.9}$$
$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{2.10}$$

2.4.4 Discussion

The EKF approximates the nonlinear system by linearizing it locally around the current estimate. This allows it to apply Kalman Filter mathematics to nonlinear systems. However, the approximation accuracy depends on the degree of nonlinearity and the quality of the initial estimate.

2.4.5 Applications

Typical EKF applications include:

- Mobile robot localization and SLAM.
- Target tracking and sensor fusion in autonomous vehicles.
- Navigation in aerospace systems.

2.4.6 Limitations

- Accuracy degrades for highly nonlinear systems due to linearization.
- Requires analytical computation of Jacobians.
- Sensitive to incorrect noise covariance estimates.

2.5 Constant Turn Rate and Velocity (CTRV) Model

2.5.1 Overview

The Constant Turn Rate and Velocity (CTRV) model is a nonlinear kinematic model commonly used to describe the motion of an object that moves with approximately constant speed and constant turn rate. It is particularly suitable for tracking vehicles or moving targets that follow curved trajectories, such as cars in autonomous driving scenarios or aircraft performing coordinated turns.[29]

2.5.2 State Vector

The CTRV model defines the system state vector as:

$$\mathbf{x} = \begin{bmatrix} p_x \\ p_y \\ v \\ \psi \\ \dot{\psi} \end{bmatrix} \tag{2.11}$$

where:

- p_x, p_y: position in Cartesian coordinates,
- v: linear velocity (speed along the heading direction),

- ψ: heading (yaw) angle,
- $\dot{\psi}$: yaw rate (turn rate).

2.5.3 Motion Model

Assuming a time step Δt, the motion model without process noise can be expressed as:

$$\mathbf{x}_{k+1} = \begin{bmatrix} p_x + \frac{v}{\dot{\psi}}\left[\sin(\psi + \dot{\psi}\Delta t) - \sin(\psi)\right] \\ p_y + \frac{v}{\dot{\psi}}\left[-\cos(\psi + \dot{\psi}\Delta t) + \cos(\psi)\right] \\ v \\ \psi + \dot{\psi}\Delta t \\ \dot{\psi} \end{bmatrix} \tag{2.12}$$

for $\dot{\psi} \neq 0$.

When $\dot{\psi} \to 0$ (straight-line motion), the model simplifies to:

$$\mathbf{x}_{k+1} = \begin{bmatrix} p_x + v\cos(\psi)\Delta t \\ p_y + v\sin(\psi)\Delta t \\ v \\ \psi \\ 0 \end{bmatrix} \tag{2.13}$$

2.5.4 Process Noise

To account for system uncertainty such as acceleration and small steering variations, process noise $\mathbf{w}_k$ is added to the velocity and yaw rate:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{w}_k) \tag{2.14}$$

where $\mathbf{w}_k = [a, \nu_{\dot{\psi}}]^T$ represents the process noise:

- a: linear acceleration noise,
- $\nu_{\dot{\psi}}$: yaw acceleration noise.

2.5.5 Applications

The CTRV model is widely used in nonlinear state estimation frameworks such as the Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), and Particle Filter (PF), particularly in:

- Vehicle and pedestrian tracking,

- Autonomous driving and robotics,
- Air and ground target surveillance.

Chapter 3

System Architecture Design and Methodology

This chapter details the overall architecture and implementation of the perception module designed for ADAS. The perception system comprises two core, independent functional modules: the dynamic multi-target tracking module and the static road element (speed bump) detection module. Together, these modules provide the vehicle with comprehensive environmental awareness capabilities. This chapter begins with a system requirements analysis, followed by separate introductions to the hardware and software architectures. It then delves into the design and implementation details of the two core algorithms. Finally, it presents the deployment and integration solution for the system within the Autoware framework.

This research project is dedicated to developing an integrated environmental perception system as a critical component of a comprehensive advanced driver assistance system. Therefore, at the outset of system design, we first focus on requirements analysis. This study does not aim to propose a generic perception solution but rather requires precisely defining task objectives tailored to this specific application scenario. This is achieved by considering the upstream and downstream constraints and specific requirements of the project within the complete workflow, ensuring its outputs seamlessly interface with upper-level decision-making and control modules.

Following the clarification of system requirements, we systematically present the overall system architecture design. The hardware architecture design forms the physical foundation for the entire system's implementation. While ensuring compliance with core project requirements, we thoroughly weighed the system's feasibility, processing efficiency, scalability, and the differences between laboratory environments and real industrial application scenarios. Based on these considerations, we selected the most suitable hardware devices from available resources.

This chapter will focus on detailing the core parameters, operating environments, dependency libraries, and workflow of the selected key hardware components within the system.

Software architecture design and algorithm development form the core of this project and constitute the primary focus of this chapter. First, we will briefly introduce the Robot Operating System (ROS) framework—the foundational system—along with the creation and management of its software packages. Subsequently, we will delve into the design and implementation of two core perception algorithms: the first is a dynamic multi-target real-time tracking algorithm based on , detailing its core workflow encompassing point cloud processing, target clustering, extended Kalman filtering, and data association; the second is a speed bump detection algorithm specifically designed for static road element recognition, illustrating its integrated design for autonomous driving open-source frameworks.

Building upon the completed hardware and software designs, this chapter will further elaborate on the deployment plan for the entire system on the test vehicle, the rationale for selecting the testing environment, and the key data acquisition process. Finally, we will provide an overview of the integrated pipeline for the entire system, clearly illustrating how this perception system functions as a modular component that integrates and collaborates with a complete advanced driver assistance system.

3.1 Requirement Analysis

Requirements analysis is the process of defining the functional and non-functional requirements of a software system. It is a critical step in the software development lifecycle, laying the foundation for all subsequent development activities, including design, implementation, and testing. This process encompasses requirement gathering and documentation, analysis, and prioritization, ultimately clarifying the system's scope and objectives. This ensures the system meets stakeholder needs and expectations while providing a clear definition of its required functionality and expected performance.

As a core component of the Advanced Driver Assistance System's environmental perception module, this system is designed to provide vehicles with comprehensive, real-time, and reliable perception capabilities of their surroundings. Based on upstream sensor data and specific requirements from downstream decision-planning modules, we defined the system's core mission objectives: the system must process real-time point cloud data streams from LiDAR, accurately detect and continuously track dynamic targets in the surrounding environment while identifying specific static road elements; all perception results must be output stably in a structured format, providing timely and accurate basis for upper-level control

strategy formulation.

To ensure the system's effectiveness in practical applications, we have established key performance indicators it must meet. The system must possess real-time processing capabilities for data streams, synchronized with the sensor's frame rate. Additionally, the algorithms must demonstrate sufficient robustness against point cloud noise, target occlusion, and complex urban environments. Furthermore, the entire software architecture should adhere to modular design principles, enabling independent development, testing, and integration of the two major functional modules: dynamic target tracking and static element detection. Compatibility between the speed bump detection module and open-source autonomous driving frameworks must be ensured to guarantee the system's feasibility and deployability on real vehicle platforms.

3.2 Hardware and Software Environment

3.2.1 NVIDIA® Jetson AGX Orin™

The AI Rugged Computer RML A4AGX is based on the NVIDIA® Jetson AGX Orin™ System-on-Module (SoM), as shown in Figure 3.1. The latest member of NVIDIA's Jetson family features up to 275 TOPS of AI processing power and starts at 32 GB of RAM. The system-on-module (SoM) consumes between 15 and 75 watts of power. This unique combination of form factor, performance, power efficiency and ruggedness opens the door to a new generation of autonomous machines and vehicles.

The fanless AI edge computers from Syslogic's rugged series are among the most robust embedded systems in the world. They are perfectly suited for tough 24/7 use in off-highway, mobile machinery and agriculture. The RPC RML A4AGX comes with eight camera inputs (GMSL2) with PoC. Accordingly, the rugged computer is suitable for inference, edge AI and intelligent vision tasks.

The operating system of the platform is NVIDIA Linux for Tegra (L4T 36.4.3) Ubuntu 22.04 NVIDIA JetPack 6.2 , and ROS2 Humble, The software environment has been meticulously configured over an extended period to ensure operational stability and robust hardware compatibility. Building upon the previously integrated work environment, all necessary dependencies remain fully compatible. [30]

Figure 3.1: Rugged Edge AI Computer RPC RML A4AGX

3.2.2 Falcon K1 LiDAR

The Seyond Falcon K1 LiDAR is a high-performance automotive-grade LiDAR specifically designed for autonomous driving and advanced driver assistance scenarios, as shown in Figure 3.2. This device employs advanced image-level ultra-long-range detection technology to deliver stable and reliable 3D environmental perception data in complex road conditions. Its unique optical system design and signal processing architecture enable outstanding performance in detection accuracy, range, and resolution. [31]

In terms of technical specifications, the Falcon K1 delivers high-resolution point cloud output at 1200x128 resolution, with a maximum detection range of 250 meters under 10% reflectivity conditions. Utilizing a 1550nm laser wavelength, it not only enhances detection range and point cloud quality but also ensures eye safety. With a horizontal field of view of 120 degrees and a vertical field of view of 30 degrees, it effectively covers critical perception areas in front of the vehicle, providing comprehensive environmental information for dynamic target tracking and static obstacle detection.

The device operates at a standard sampling frequency of 10 Hz, meaning the sensor generates and outputs 10 complete frames of point cloud data per second. At the communication protocol level, the Falcon K1 LiDAR employs an optimized User Datagram Protocol (UDP) as the foundation for data transmission. This protocol is encapsulated based on the standard UDP/IP protocol stack. Each data

Figure 3.2: Seyond Falcon K1

packet contains complete point cloud information, including the three-dimensional coordinates of each detected point, reflection intensity data, and precise timestamps. This lightweight protocol design minimizes protocol overhead, ensuring real-time and efficient data transmission that fully meets the stringent real-time requirements for perception data in autonomous driving systems.

To guarantee data communication reliability and integrity, the LiDAR implements multiple safeguards at the application layer. The device incorporates a precision clock synchronization module supporting IEEE 1588-compliant precision timing protocols, ensuring microsecond-level timestamp accuracy for point cloud data. Additionally, the system employs error detection mechanisms such as cyclic redundancy checks (CRC) to effectively safeguard data integrity and accuracy during transmission. These designs enable the Falcon K1 LiDAR to deliver a stable, reliable, and temporally precise point cloud data stream for environmental perception systems.

3.2.3 Ouster LiDAR

Ouster OS1 is a high-resolution, medium-range mechanical LiDAR sensor, as shown in 3.3. Designed for autonomous vehicles, mobile robots, drones, and smart infrastructure applications, it delivers a comprehensive advantage in price, performance, size, weight, and power consumption, operating reliably in indoor and outdoor environments around the clock. Its core feature lies in providing image-level near-infrared intensity and point cloud data, achieving highly reliable perception capabilities within a compact form factor.[32]

Under 80% reflectivity conditions, the maximum detection range reaches 100

Figure 3.3: Ouster OS1

meters (detection rate >90%); under 10% reflectivity conditions, the maximum detection range is 45 meters (detection rate >90%). The minimum detection distance is 0.3 meters. Distance measurement accuracy for standard Lambertian targets is ±3 cm, with a resolution as high as 0.3 cm, ensuring distinguishability of minute objects. It offers three vertical resolution configurations: 32-line, 64-line, or 128-line. The horizontal azimuth resolution can be configured at 512, 1024, or 2048, enabling a fixed and uniform point cloud distribution per frame.

The radar features a 360° horizontal field of view and a 45° vertical field of view (-22.5° to +22.5°), enabling comprehensive environmental perception. Its angular sampling accuracy reaches ±0.01° in both vertical and horizontal directions, guaranteeing precise point cloud orientation. The scanning rotation frequency supports configurable modes of 10 Hz or 20 Hz to accommodate varying application speed requirements.

For data transmission, the OS1 outputs data via Gigabit Ethernet using the UDP protocol. At maximum configuration (128 lines, 2048 horizontal resolution), the data output rate reaches 2.62 million points per second, with a data bandwidth of approximately 254 Mbps. Each data point contains rich information including distance, signal strength, reflectivity, near-infrared light, channel number, azimuth, and timestamp. Additionally, the device integrates an IMU (ICM-20948) that outputs triaxial gyroscope and accelerometer data at 100 Hz to assist with motion

compensation.

The sensor features an extremely compact design with a diameter of just 85 mm, a height (without heat sink cover) of 58.35 mm, and a weight of only 377 grams, facilitating easy integration. It demonstrates robust environmental adaptability with an operating temperature range of -40°C to +60°C and an IP68/IP69K protection rating, resisting dust ingress and high-pressure water jets. The product has also passed rigorous shock and vibration testing, ensuring stable operation in harsh mechanical environments.

3.2.4 PIX-KIT 2.0 Drive-by-wire Chassis

PIX-KIT 2.0 Drive-by-wire Chassis is a one-stop integrated development platform designed for autonomous driving research and education. By offering highly integrated hardware configurations and an open-source software ecosystem, this kit aims to establish a comprehensive autonomous driving algorithm verification environment for university, research institute, and corporate R&D teams. This significantly lowers the technical development threshold, making it suitable for diverse applications including algorithm development, prototype validation, and teaching practice. As shown in Figure 3.4, its core hardware architecture adopts a modular design philosophy, integrating a pure-electric open-source drive-by-wire chassis, a multi-sensor system, and an industrial-grade remote controller. The drive-by-wire chassis features an independent four-wheel hub motor drive system, enabling precise control over steering, braking, and acceleration. It provides a standard CAN bus interface for upper-level algorithms to directly drive actuators, laying a solid foundation for vehicle control algorithm research.[33]

Figure 3.4: PIX-KIT 2.0 Drive-by-wire Chassis

Regarding software and development environments, the platform is tightly integrated with mainstream Robot Operating System (ROS) and open-source autonomous driving frameworks. The platform comes pre-installed with complete drivers and development interfaces, ensuring seamless compatibility with open-source systems like Autoware and Apollo. Users can directly develop and test core algorithms such as perception, planning, and control based on this foundation. Additionally, the kit provides a comprehensive development toolchain and detailed technical documentation covering sensor calibration, map building, and simulation testing. This forms a complete technical support system spanning hardware operation to software deployment, effectively ensuring the smooth progress of research work.[33]

The PIX-KIT 2.0 platform delivers performance parameters tailored for research applications: a maximum speed of 30 km/h, a maximum climbing gradient of 30%, and a standard range between 90 and 120 kilometers. With a standard payload capacity of 500 kg and an optional upgrade to 800 kg, it provides ample space for mounting various computing devices and customized functional modules. Regarding control precision, its steering accuracy is better than 1 degree, achieving a minimum turning radius of 3 meters in four-wheel steering mode. This provides excellent hardware conditions for precise path tracking and maneuver control research. Since its launch, this development kit has been deployed globally in autonomous driving curricula at universities, SLAM and multi-sensor fusion research at scientific institutions, and rapid prototyping projects at enterprises. Its proven value as an efficient and reliable research and teaching platform is well-established.[33]

3.2.5 ROS2 environment

ROS 2 (Robot Operating System 2) represents a major architectural overhaul of the robot operating system, designed to address core challenges faced by ROS 1 in production-level and commercial applications. The ROS 2 project was formally launched around 2015, led by Open Robotics. Its design goal is to provide a reliable and secure software framework for robots, applicable throughout the entire lifecycle from research prototypes to product deployment. Unlike a simple functional upgrade over ROS 1, ROS 2 represents a fundamental restructuring. Its most significant change lies in replacing the underlying communication middleware, delivering exceptional real-time performance, cross-platform support, robust network communication, and industrial-grade safety. This enables it to thrive in complex scenarios demanding extreme reliability and performance, such as autonomous driving and industrial automation.

The cornerstone of the ROS 2 architecture is the Data Distribution Service (DDS). DDS itself is a mature industrial standard defined by the Object Management Group

(OMG), specifically designed for distributed systems demanding high reliability, real-time performance, and scalability. ROS 2 does not define its own communication layer but acts as a higher-level abstraction over DDS, leveraging it to handle discovery, connection, and messaging between all nodes. This design allows ROS 2 to inherently inherit all advanced features of DDS, including anonymous, asynchronous communication based on the publish/subscribe pattern and fine-grained control over Quality of Service (QoS) policies. Developers can configure distinct QoS policies for each topic or service based on application requirements. For instance, they can specify whether messages should be transmitted "best effort" or must be delivered "reliably," and whether data should be persistently retained or only made available to current subscribers. This capability is crucial for ensuring critical data integrity in unreliable network environments.

In terms of node models and communication mechanisms, ROS 2 inherits core concepts from ROS 1—nodes, topics, services, and actions—while significantly enhancing implementation details and functionality. Nodes remain the fundamental computational units, communicating asynchronously via topics for publish/subscribe interactions, synchronously via services for request/response calls, and utilizing the Action Library (ActionLib) to handle long-running, preemptible tasks. However, leveraging DDS's robust capabilities, ROS 2 communication no longer relies on a single centralized ROS Master node. In ROS 2, node discovery and connection are fully distributed, accomplished through DDS's built-in discovery mechanism. This critical improvement eliminates the single point of failure risk present in ROS 1. Even if some nodes crash or network partitions occur, communication between other nodes continues, greatly enhancing the system's overall robustness and fault tolerance.

To address the complexity of modern robotics software development, ROS 2 also achieves significant advancements in system design and cross-platform support. It is designed from the ground up to support real-time operating systems (such as VxWorks and NuttX) and microcontrollers (MCUs), enabling developers to run ROS 2 nodes directly on resource-constrained embedded devices. For process lifecycle management, ROS 2 introduces the concepts of Executables, Components, and Containers. A Component is a node designed to be dynamically loaded into a Container process at runtime. This model reduces system memory footprint and inter-process communication overhead, making it highly suitable for deployment in large-scale systems. Furthermore, ROS 2 natively supports multiple programming languages, including C++ and Python, and provides a consistent experience for type systems and threading models through its Client Libraries.

Another core strength of ROS 2 lies in its robust security features. It integrates modern cybersecurity standards, supporting Transport Layer Security (TLS) and Data Link Layer Security (DDS-Security). Through DDS-Security, ROS 2 enables

communication authentication (ensuring trusted node identities), encryption (preventing data eavesdropping), and access control (restricting node access to specific topics or services). These capabilities are crucial for protecting robotic systems from malicious attacks and preventing sensitive data leaks, particularly when robots connect to public networks or perform safety-critical tasks.

In summary, ROS 2 represents a new phase in the evolution of robotic software frameworks. By embracing the mature industrial standard DDS, it builds a decentralized, secure, reliable, and high-performance distributed computing platform. Not only does it seamlessly inherit ROS 1's flexibility and rich ecosystem for research applications, but more importantly, it overcomes numerous limitations of ROS 1. This provides a robust technical foundation for advancing robotics from the laboratory to real-world deployment and enabling large-scale commercial product implementation. With ROS 1 officially reaching its end-of-life in May 2022, ROS 2 has become the undisputed mainstream choice and de facto standard for current and future robotics development.[34]

This chapter presents the design and implementation of two core perception modules based on 3D LiDAR point clouds: (i) a multi-object tracking framework powered by an Extended Kalman Filter (EKF) under a Constant Turn Rate and Velocity (CTRV) motion model, and (ii) a speed bump detection framework based on local height discontinuities and geometric reasoning. Both systems are implemented as ROS 2 nodes with real-time performance on embedded hardware platforms.

3.3 Multi-Object Tracking Algorithm

3.3.1 Overview

This work develops a real-time object tracking and track management framework based on clustered LiDAR point clouds. The proposed system integrates spatial clustering, sensor fusion, and probabilistic motion estimation through an EKF. The entire algorithm is implemented as a ROS2 node that receives synchronized LiDAR point clouds, camera detections, and image data, and outputs tracked objects with continuous IDs, semantic labels, and estimated motion states. The system is designed for use in intelligent vehicles and advanced driver-assistance systems (ADAS), where stable and reliable perception of surrounding objects is crucial.

3.3.2 System Architecture

The architecture of the proposed tracking system is composed of four main stages:

1. **LiDAR Point Cloud Preprocessing**: The raw LiDAR data is filtered, downsampled, and spatially segmented to obtain individual object candidates.

2. **Camera-LiDAR Fusion**: The segmented clusters are projected onto the image plane using calibrated extrinsic and intrinsic parameters and associated with 2D detections from a YOLO network.

3. **Multi-target Tracking**: The resulting fused detections are passed to a multi-target tracker, which applies the EKF for state estimation and uses Mahalanobis distance for data association.

4. **Track Management and Visualization**: Targets are dynamically created, updated, or removed based on state transitions, and the tracking results are visualized and logged for evaluation.

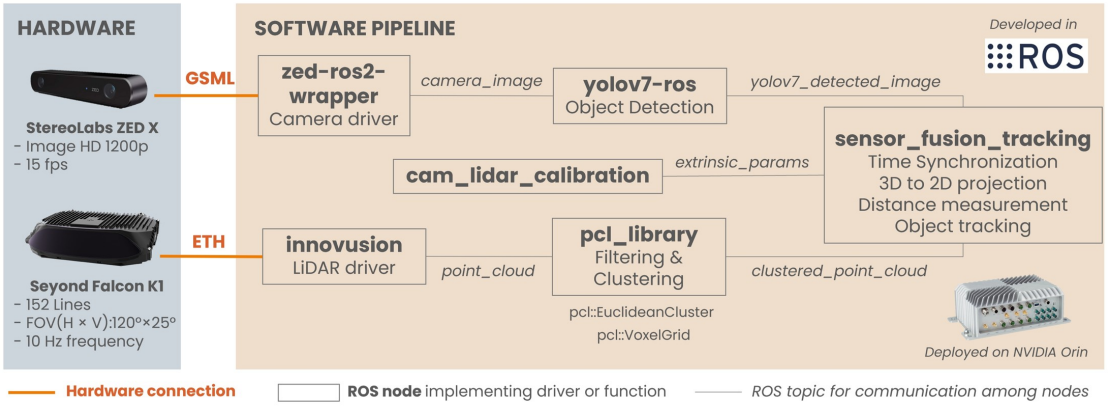

Figure 3.5: System Architecture of Multi-Object Tracking

As illustrated in Figure 3.5, each stage is modularly integrated within the ROS2 node, ensuring synchronization, scalability, and compatibility with other perception modules.

3.3.3 LiDAR Point Cloud Preprocessing

Coordinate Transformation

Upon receiving the point cloud message (`sensor_msgs::PointCloud2`), the raw LiDAR points are transformed to align with the camera coordinate system. The axes are rearranged to match the conventional camera frame, facilitating projection and data fusion.

Region Filtering

Passthrough filters are applied along the x and z axes to retain points within a predefined spatial region, removing irrelevant sky points and low ground reflections. This reduces computational load and improves clustering robustness.

Adaptive Downsampling

A voxel grid filter is employed to reduce the number of points while preserving structural features. The voxel size ℓ is determined dynamically based on the average nearest-neighbor distance d_{avg} in the filtered cloud:

$$\ell = k \cdot d_{avg} \tag{3.1}$$

where k is an empirical scaling factor (typically $k = 3.5$).This adaptive strategy ensures consistent clustering behavior across varying point densities.

3.3.4 Object Clustering and Feature Extraction

The filtered cloud is segmented into object clusters using **Euclidean Cluster Extraction**. Two points are grouped into the same cluster if their Euclidean distance is below a threshold ε:

$$\|p_i - p_j\| < \varepsilon \tag{3.2}$$

where ε is proportional to the voxel size.

For each detected cluster C_n, the centroid is computed as:

$$\mathbf{c}_n = \frac{1}{|C_n|} \sum_{p_i \in C_n} p_i = \begin{bmatrix} \bar{x}_n \\ \bar{y}_n \\ \bar{z}_n \end{bmatrix} \tag{3.3}$$

Other attributes such as bounding limits $(x_{min}, x_{max}, y_{min})$ and RGB color are also recorded. The centroids are projected to the 2D image plane via:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

where $\mathbf{K}$ is the intrinsic matrix, and $[\mathbf{R}|\mathbf{t}]$ represents the camera's rotation and translation with respect to the LiDAR frame.

3.3.5 Sensor Fusion and Class Assignment

After projection, each LiDAR centroid is checked for overlap with YOLO bounding boxes in the image plane. If the centroid (u, v) lies within the bounding box $[u_{min}, u_{max}] \times [v_{min}, v_{max}]$, the cluster inherits the associated class label c and confidence score s_c:

$$(u, v) \in B_c \Rightarrow \text{cluster.class\_id} = c, \quad \text{cluster.score} = s_c \tag{3.5}$$

This step assigns semantic meaning to spatially precise LiDAR clusters, producing a set of labeled 3D measurements for the tracking module.

3.3.6 Multi-target Tracking Framework

Each tracked object is represented by a state vector:

$$\mathbf{x} = [p_x, p_y, v, \psi, \dot{\psi}]^T \tag{3.6}$$

where p_x, p_y denote the 2D position, v the linear velocity, ψ the yaw angle, and $\dot{\psi}$ the yaw rate.

Motion Model: CTRV (Constant Turn Rate and Velocity)

The motion model assumes a constant turn rate and velocity between consecutive frames. The predicted state $\mathbf{x}_{k+1}^-$ is given by:

$$\mathbf{x}_{k+1}^- = \begin{cases} \begin{bmatrix} p_x + \frac{v}{\dot{\psi}}(\sin(\psi + \dot{\psi}\Delta t) - \sin(\psi)) \\ p_y + \frac{v}{\dot{\psi}}(-\cos(\psi + \dot{\psi}\Delta t) + \cos(\psi)) \\ v \\ \psi + \dot{\psi}\Delta t \\ \dot{\psi} \end{bmatrix}, & \text{if } |\dot{\psi}| > \epsilon \\ \begin{bmatrix} p_x + v\cos(\psi)\Delta t \\ p_y + v\sin(\psi)\Delta t \\ v \\ \psi \\ \dot{\psi} \end{bmatrix}, & \text{otherwise} \end{cases} \tag{3.7}$$

The corresponding Jacobian matrix $F_k = \frac{\partial f}{\partial x}$ is derived from the nonlinear motion function $f(\mathbf{x}, \Delta t)$ and used for covariance propagation:

$$P_{k+1}^- = F_k P_k F_k^T + Q \tag{3.8}$$

where Q is the process noise covariance.

Measurement Model and Update

The measurement vector consists of 2D planar positions derived from LiDAR centroids:

$$\mathbf{z}_k = [p_x, p_y]^T \tag{3.9}$$

and the observation model is linear:

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{v}_k \tag{3.10}$$

with

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{v}_k \sim \mathcal{N}(0, R) \tag{3.11}$$

The Kalman gain and updated state are computed as:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{3.12}$$

$$\mathbf{x}_k = \mathbf{x}_k^- + K_k(\mathbf{z}_k - H\mathbf{x}_k^-) \tag{3.13}$$

$$P_k = (I - K_k H) P_k^- \tag{3.14}$$

3.3.7 Data Association

For multiple objects, data association is performed using the **Mahalanobis distance**, which measures how well a measurement matches a predicted track given the covariance of uncertainty:

$$d_M = \sqrt{(\mathbf{z} - \hat{\mathbf{z}})^T S^{-1} (\mathbf{z} - \hat{\mathbf{z}})} \tag{3.15}$$

where $S = HPH^T + R$ is the innovation covariance. The measurement with the smallest Mahalanobis distance below a predefined threshold is associated with the corresponding track; otherwise, a new track is initialized.

3.3.8 Track Management and State Transitions

As illustrated in Figure 3.6 [35], The tracking framework employs a state-machine-based management strategy with four discrete states:

- **ACTIVE**: Represents a newly initialized track created from an unassociated LiDAR measurement. The track awaits confirmation from future detections to verify its validity.
- **TRACKED**: Denotes a confirmed and continuously updated track. Measurements are successfully associated across frames, and the EKF correction step refines its state estimate.

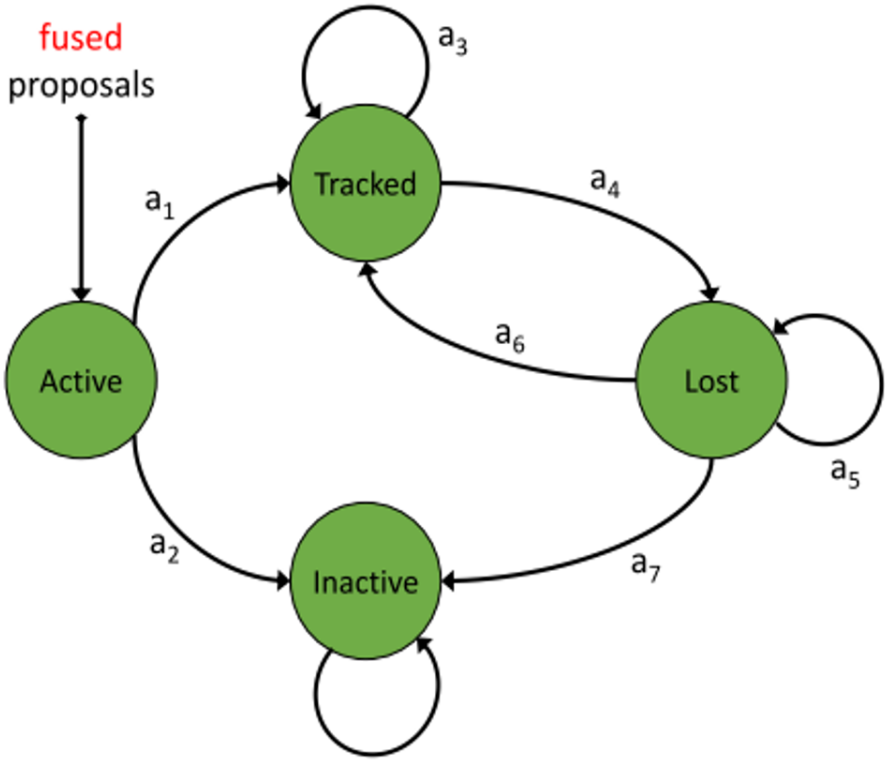

Figure 3.6: Target state transitions based on Markov Decision Process

- **LOST**: Indicates a temporarily unmatched track. The EKF prediction continues without measurement correction to preserve trajectory continuity during short-term occlusions or detection gaps.
- **INACTIVE**: Marks a track that has remained unmatched for longer than a predefined threshold. It is removed from active tracking to prevent resource waste and false positives.

This finite-state logic ensures temporal continuity of object trajectories and suppresses false positives caused by noise or occlusion.

3.3.9 Data Output and Visualization

All intermediate and final results are published through dedicated ROS2 topics, including:

- `/pcl_points` for clustered clouds

- `/pcl_centroids` for object centroids
- `/pcl_targets` for tracked objects
- `/projectedImg` for fused 2D–3D visualization

Additionally, CSV logs are generated for both YOLO detections and tracker outputs, containing timestamps, class IDs, confidence scores, estimated positions, and tracking states. These files serve as the basis for offline evaluation and quantitative analysis.

3.3.10 Summary

In summary, the developed methodology combines 3D geometric reasoning, 2D visual classification, and probabilistic motion estimation within a unified ROS2 framework. The system effectively bridges raw sensor data and semantic perception, enabling robust object tracking in dynamic environments with variable visibility and partial occlusions. The EKF-based CTRV model ensures smooth trajectory estimation, while the Markov-style track management guarantees reliable target lifecycle control.

3.4 Speed Bump Detection Algorithm

3.4.1 Overview

This research develops a speed bump detection framework based on 3D LiDAR point clouds, implemented as a ROS2 node named `speed_bump_detector`. The system identifies speed bumps by analyzing local height discontinuities on the road surface, distinguishing raised regions from flat ground through geometric and spatial analysis. The proposed node integrates feature extraction, clustering, geometric reasoning, and multi-channel output publishing to achieve real-time detection performance suitable for advanced driver-assistance systems (ADAS) and autonomous vehicles.

3.4.2 System Architecture

The `speed_bump_detector` node subscribes to the ground point cloud topic (`/ground_points`), which represents the filtered surface layer obtained from LiDAR data. It processes each frame in real time and publishes the following outputs:

- `/perception/speed_bump_detector/edge_points` – edge points indicating abrupt height changes for debugging and visualization.

- `/perception/speed_bump_detector/markers` – visual markers representing detected bumps in 3D space.
- `/perception/speed_bump_detector/objects` – standardized detection messages compatible with the Autoware Perception framework.
- `can_tx` – encoded CAN messages conveying detection and position information for vehicle-level decision-making.

The node's internal structure is composed of five key modules:

1. Edge point detection – identifies local height discontinuities.
2. Feature clustering – groups adjacent edge points belonging to the same structure.
3. Geometric feature extraction – computes width, length, height, and slope parameters.
4. Bump validation and classification – applies threshold-based criteria to confirm valid bumps.
5. Data publishing and CAN communication – outputs the results to visualization and control interfaces.

3.4.3 Edge Point Detection

The core idea of the detection framework is that speed bumps create local height discontinuities in the ground surface, which can be detected through elevation changes within a small neighborhood.

For each point in the ground point cloud, its neighboring points within a given radius are retrieved using a Kd-tree search. The algorithm then evaluates the local vertical variation:

$$\Delta z_i = \left(\max_{j \in \mathcal{N}_i} z_j\right) - \left(\min_{j \in \mathcal{N}_i} z_j\right) \tag{3.16}$$

If $\Delta z >$ threshold, the point is marked as an edge point, signifying a potential boundary of an elevated structure.

This procedure, implemented in the function `detectEdgePointsByHeightRange()`, operates using two configurable parameters:

- `radius`: neighborhood search radius (default 0.3 m);
- `min_height_diff`: minimum required height difference (default 0.05 m).

The resulting edge points capture all areas exhibiting significant elevation changes, including potential speed bump boundaries.

3.4.4 Feature Clustering and Geometric Analysis

Detected edge points are spatially grouped into clusters using a radius-based region growing algorithm (`extractSpeedBumpFeatures()`). Each cluster is analyzed to extract characteristic geometric features that describe the potential bump.

Within each cluster, five representative points are determined:

- Closest point – nearest to the vehicle.
- Highest point – maximum elevation within the cluster.
- Leftmost and rightmost points – lateral boundaries.
- Centroid – geometric center of all cluster points.

From these key points, several geometric descriptors are computed:

$$\text{Width} = |y_{\text{rightmost}} - y_{\text{leftmost}}| \tag{3.17}$$

$$\text{Length} = 2 \times |x_{\text{highest}} - x_{\text{closest}}| \tag{3.18}$$

$$\text{Height difference} = z_{\text{highest}} - z_{\text{closest}} \tag{3.19}$$

The length estimation is adaptively refined based on the range of x-coordinates in the cluster to ensure consistent measurements across different bump sizes. Each cluster with a sufficient number of points and a height difference exceeding the predefined threshold is retained as a speed bump candidate.

3.4.5 Speed Bump Characterization

Each confirmed cluster is encapsulated in a `BumpDetection` data structure that summarizes its geometric and spatial attributes:

- Center coordinates (x, y, z);
- Dimensions (width, length, height);
- Height variance (represented by height difference);
- Type (fixed as “speedbump”);
- Confidence score (default value 1.0 for confirmed detections).

These parameters are used both for visualization and downstream control tasks. Detections failing to meet the confidence or geometric constraints are filtered out to reduce false positives caused by uneven ground or curbs.

3.4.6 Data Publishing and Integration

The detection results are distributed through multiple data channels to ensure compatibility with visualization, perception, and control systems.

3.4.7 Visualization

Each detected bump is visualized using 3D markers (`visualization_msgs::MarkerArray`) published to `/perception/speed_bump_detector/markers`. Markers are represented as cuboids centered at the bump's estimated position, with dimensions scaled according to measured width, length, and height. Color intensity corresponds to detection confidence, allowing quick qualitative assessment in RViz.

3.4.8 CAN Bus Communication

For direct interaction with vehicle electronic control units (ECUs), detection results are encoded into CAN messages. Two message types are defined:

- Detection message (`SPEED_BUMP_DETECTION_ID`): includes width, length, height difference, and slope information.
- Position message (`SPEED_BUMP_POSITION_ID`): encodes the detected bump's position relative to the vehicle coordinate frame.

If no bumps are detected, a default "no-detection" CAN message is sent to maintain communication consistency. This interface supports real-time alerts or automatic suspension adjustments in intelligent vehicles.

3.4.9 Data Logging and Evaluation Support

To facilitate offline analysis and system validation, all detections are automatically recorded in CSV format.Each log entry contains timestamps, coordinates of the key points, geometric dimensions, and detection confidence. The log files are time-stamped and stored in a designated directory for later processing. This structured data enables evaluation of detection stability, repeatability, and environmental robustness under different road conditions.

3.4.10 Summary

The proposed ROS2-based Speed Bump Detection Node applies a height-difference-driven geometric analysis to identify raised road structures from LiDAR point clouds.By combining edge detection, feature clustering, and rule-based validation, the system achieves accurate and interpretable detection of speed bumps.Its modular

design allows seamless integration into the proposed ADAS perception stacks, providing reliable geometric cues for navigation safety and vehicle dynamics control.

Chapter 4

Experiments and Results

This chapter outlines the complete execution process of the test plan and the evaluation of the results. To validate the effectiveness and robustness of the proposed LiDAR-based perception framework, a series of real-world experiments were conducted focusing on two representative perception tasks: multi-object tracking and speed bump detection. All experimental data were collected using the integrated perception platform equipped with a high-precision LiDAR sensor and stored in ROS bag format for offline analysis. The results presented in the following sections demonstrate the feasibility, reliability, and real-world applicability of the proposed perception algorithms.

4.1 Testing Plan

The primary objective of this study is to verify the feasibility of the two proposed LiDAR-based perception functions. Each function corresponds to an individual project with its own deployment plan and testing scenarios. Therefore, the algorithms were evaluated using real-world data collected from actual sensor deployments, ensuring that the results reflect both algorithmic performance and the potential for integration into the overall system.

For the multi-object tracking algorithm, the goal of the test is to assess the stability of the tracking process and the functionality of the track management system. To achieve this, experiments were conducted in a controlled environment using real vehicles as dynamic targets. The test platform consists of our research vehicle equipped with the LiDAR sensor suite, as shown in Figure 4.1, while surrounding vehicles serve as tracked objects. The experiment was carried out in a parking lot, providing a safe and structured environment that allows precise control of the experimental variables and accurate ground-truth measurement.

For the speed bump detection algorithm, the objective is to evaluate whether

Figure 4.1: Test vehicle for Multi-object tracking

the proposed method can successfully detect road speed bumps and measure their distance in real time. The experiment was conducted on a closed testing ground, where a LiDAR sensor was mounted on the front platform of the PIXKIT 2.0 drive-by-wire chassis, and a standard speed bump was fixed on the ground, as shown in Figure 4.2. This setup reflects the intended application scenario of the overall system and allows a deeper understanding of the algorithm performance, guiding future optimization for system-level integration.

4.2 Data collection

4.2.1 Multi-object Tracking

According to the testing plan, raw data were collected from the LiDAR and camera sensor suite, which had been precisely calibrated within the project's perception pipeline. To support subsequent analysis and processing, all acquired sensor data were recorded and stored in ROS bag files. Although the multi-object tracking algorithm is designed for real-time execution, offline validation was performed to systematically evaluate different parameter configurations and conduct detailed data analysis. This approach enables fine-tuning of the algorithm and optimization of its overall performance.

As illustrated in Figure 4.3, the experimental setup for testing the multi-object tracking algorithm involved a stationary target vehicle positioned in a parking space,

Figure 4.2: PIXKIT 2.0 drive-by-wire chassis

with several other vehicles parked on its right side. The test vehicle, equipped with the LiDAR–camera sensor set, was parked three spaces away from the target vehicle, facing its left side at an initial distance of approximately 8.5 meters. In this configuration, the LiDAR maintained a clear and stable view of the main target while also capturing the two vehicles located behind it. This arrangement not only facilitated a multi-object tracking scenario, but also simulated typical driving conditions where targets in the same lane may become temporarily occluded by preceding vehicles, yet still remain relevant to the perception system.

During the experiment, the ego vehicle slowly reversed, gradually increasing the distance from the target vehicle until it reached the end of the parking lot, with a final distance of approximately 21 meters. By progressively increasing the distance and analyzing the corresponding tracking results, we evaluated the stability of the tracking algorithm and the performance of the track management module.

4.2.2 Speed Bump Detection

The experimental data used to evaluate the speed bump detection algorithm were collected using the PIXKIT 2.0 drive-by-wire chassis, which serves as the primary test platform in this study. To ensure high-quality LiDAR point cloud acquisition, the LiDAR sensor was mounted on the front platform of the vehicle, as shown in Figure 4.4. This configuration enables more effective capture of ground-level point

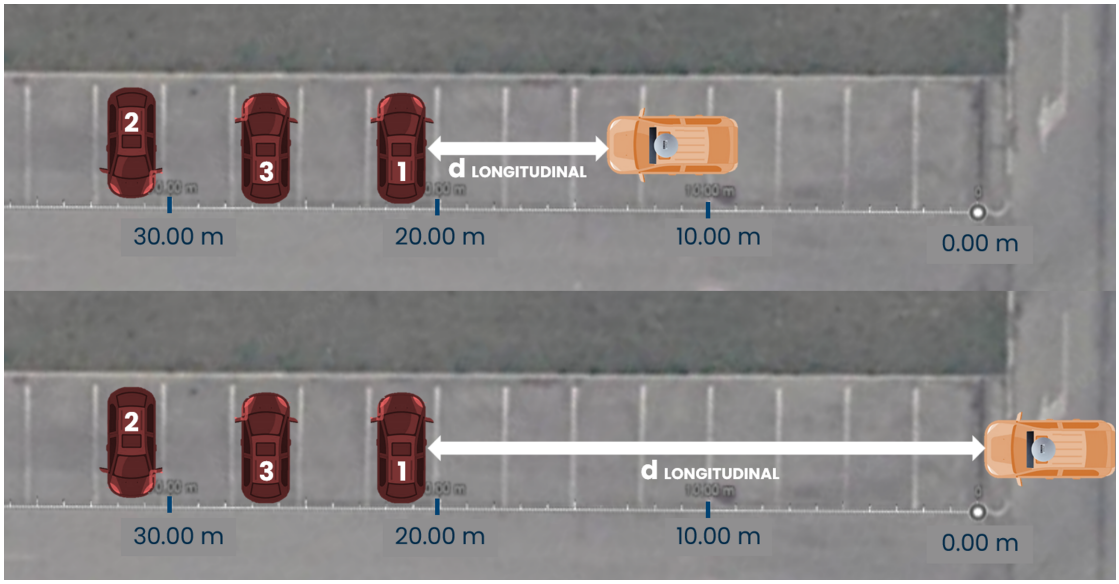

Figure 4.3: Test scenario for multi-object tracking

cloud data. The LiDAR was positioned 0.59 m above the ground, with a horizontal distance of 0.41 m between the sensor's center and the front bumper of the vehicle.

The experiment was conducted in an enclosed test yard at the Aero Club Torino. During testing, the vehicle was remotely controlled to move forward from an initial distance of 9 m and stop approximately 1 m before the speed bump. This scenario simulates a typical driving situation in which the vehicle must adjust parameters of the active suspension and traction control systems based on the distance information of an upcoming speed bump. The LiDAR point cloud data were recorded and stored in ROS bag files for offline processing. The collected data were then used to fine-tune algorithm parameters and to validate the detection accuracy, measurement stability, and real-time distance estimation capability of the proposed method.

4.3 Result Evaluation

In this section, the results of the offline validation for the two tested functions are presented and discussed. For each function, specific evaluation criteria have been developed to assess the algorithm's capability and performance.

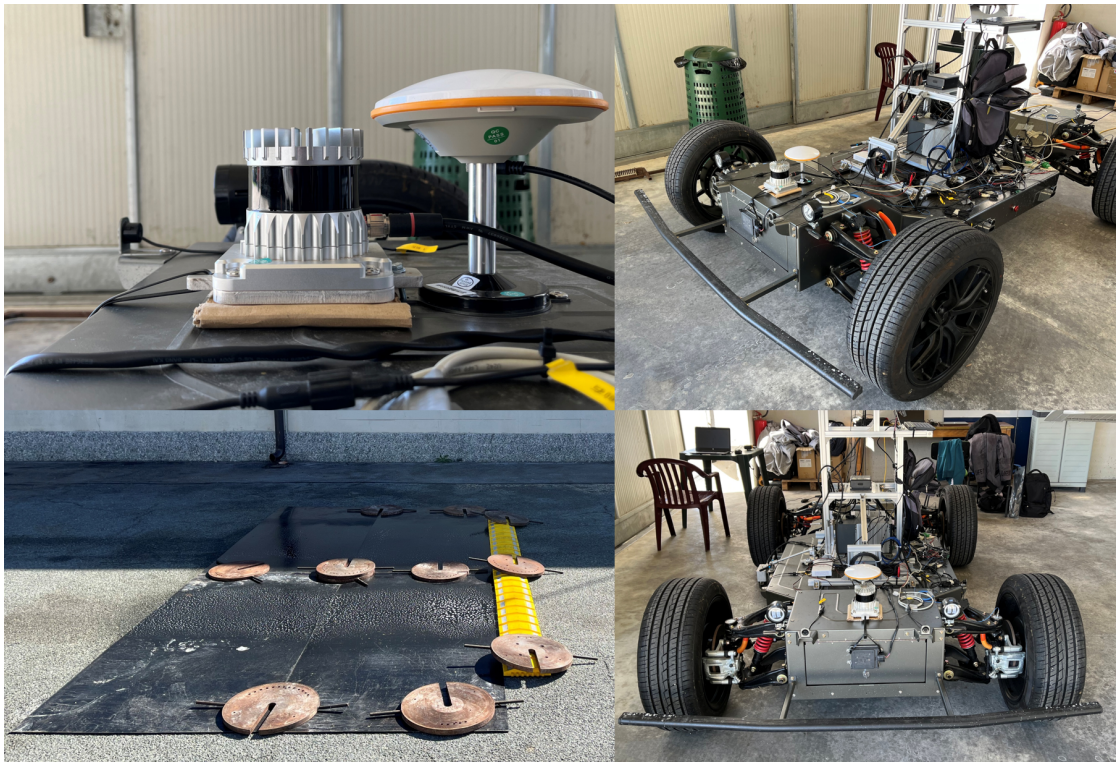

Figure 4.4: Test scenario for speed bump detection

4.3.1 Multi-object Tracking

In this scenario, the evaluation focuses on the stability of the tracking algorithm and the effectiveness of the track management module. Figure 4.5 illustrates the visualization of the sensor fusion and multi-object tracking process, providing an intuitive understanding of the working principles of the algorithm.

From the visualized point cloud, it can be observed that Vehicle 1, the primary target, is clearly segmented as a vehicle-shaped point cloud cluster. Its centroid trajectory maintains a consistent color, as do the trajectories of the two following vehicles. This indicates that the same targets are correctly identified across consecutive frames and are stably tracked, even though a few frames were lost at the beginning of the test due to minor data jitter. These results demonstrate the robustness of the track management module.

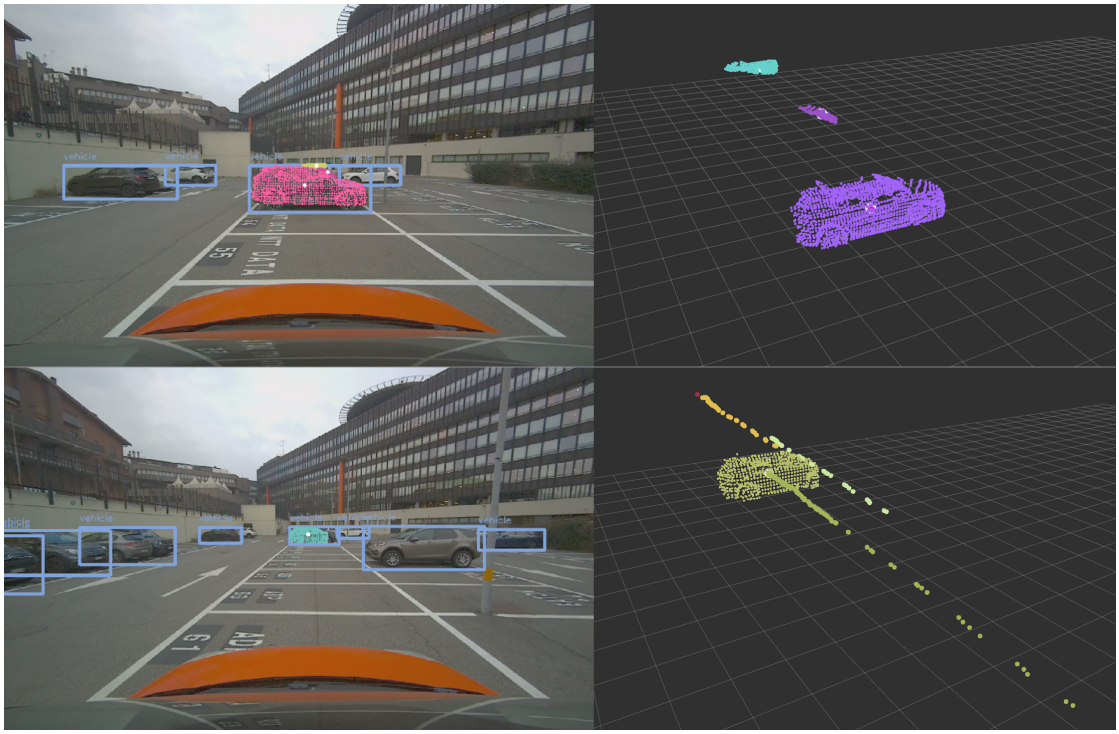

Figure 4.5: Visualization of multi-object tracking data

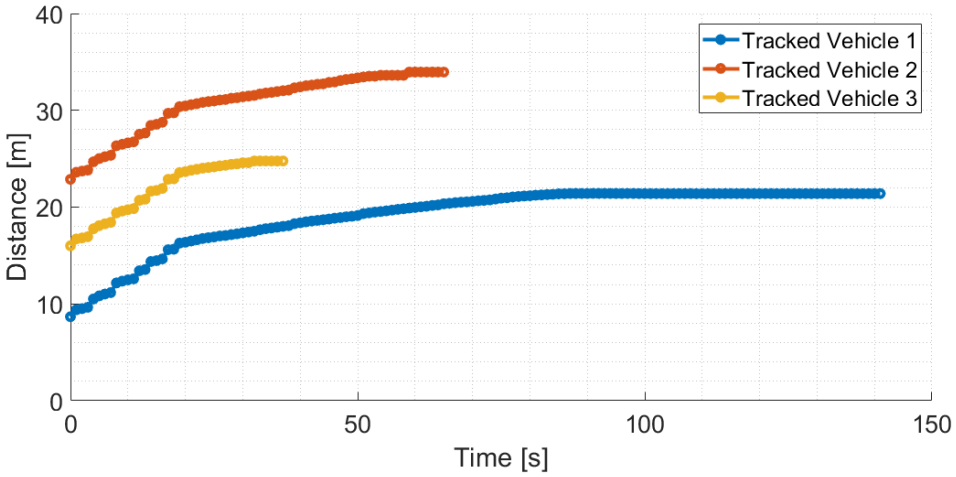

Figure 4.6: Distance measurement of multi-object tracking

Furthermore, as shown in Figure 4.6, the distances between the ego vehicle and the three tracked targets are accurately measured. The three vehicles were parked in alternating parking spaces, leaving approximately 7 meters between each other,

which aligns well with the measured distances.

4.3.2 Speed Bump Detection

In this scenario, the evaluation focuses on assessing the functionality of the proposed speed bump detection algorithm, as well as the accuracy and stability of the distance measurements.

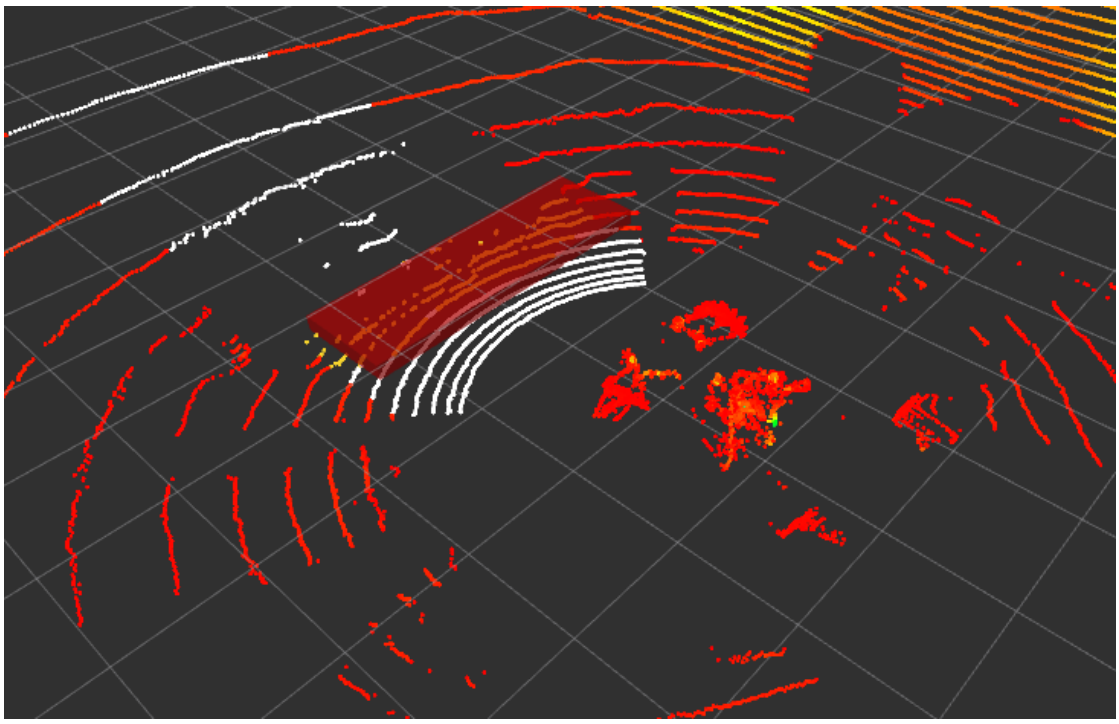

Figure 4.7: Visualization of speed bump detection

Figure 4.7 illustrates the identified and segmented speed bump within the LiDAR point cloud, offering an intuitive visualization of the algorithm's operating principles and confirming its effectiveness in feature recognition. As shown in Figure 4.8, the yellow points representing the speed bump are clearly separated from the ground points rendered in white, demonstrating the algorithm's capability to accurately distinguish subtle height variations between road surfaces and elevated structures.

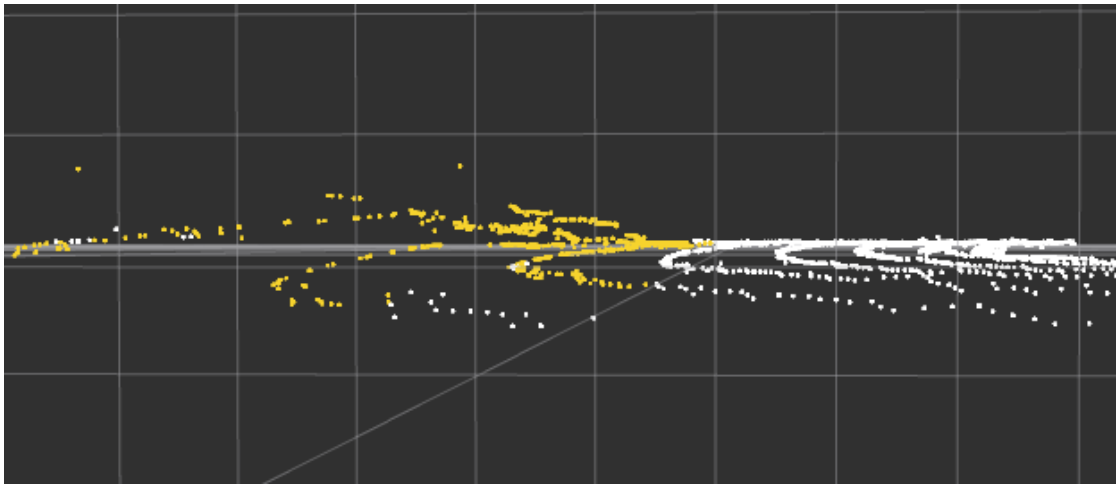

Figure 4.8: Detected speed bump in yellow points

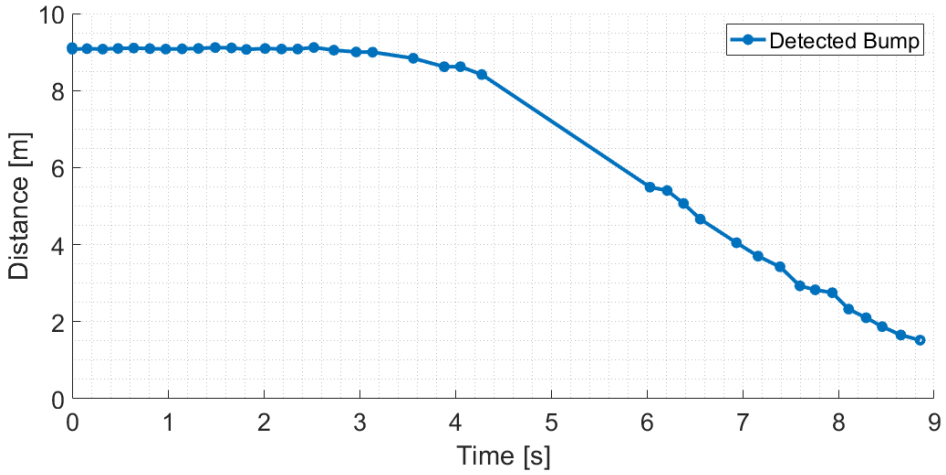

Figure 4.9: Distance measurement of detected speed bump

Figure 4.9 illustrates the experimental results obtained during the validation test. In this experiment, the vehicle started from rest at approximately 9 meters from the speed bump and gradually accelerated forward. As it approached the obstacle, the vehicle decelerated and eventually stopped immediately in front of the bump. This motion pattern is clearly reflected in the detection results, where the measured distance shows a smooth and continuous decrease corresponding to the vehicle's approach trajectory.

A short discontinuity is observed in the mid-range portion of the curve, mainly caused by temporary sparsity of LiDAR returns near the surface of the ground. This

phenomenon typically occurs when the scanning angle becomes shallow, resulting in a reduced number of valid reflections. Despite this brief absence of data, the algorithm quickly recovered detection once sufficient surface points were captured again, maintaining a coherent and physically consistent trend.

These results demonstrate the strong capability of the proposed algorithm to accurately identify and localize the speed bump, even under partially missing or noisy point cloud conditions. The estimated distances remain stable across consecutive frames, showing high temporal consistency and robustness against random measurement noise.

Overall, the experiment confirms that the LiDAR-based detection method can robustly capture the geometric characteristics of speed bumps and provide continuous, stable measurements during vehicle approach. Even without any external correction or reference data, the system exhibits strong resilience to real-world imperfections such as point sparsity and partial occlusion.

Chapter 5

Conclusions and Future Works

This paper presented two LiDAR-based perception algorithms designed for ADAS: a multi-object tracking algorithm based on the Extended Kalman Filter (EKF) with a Constant Turn Rate and Velocity (CTRV) motion model, and a speed bump detection algorithm based on ground point cloud analysis. Although these two functions serve distinct perception purposes, both share a unified methodological framework built upon feature extraction, cluster segmentation, and state estimation from LiDAR point clouds.

For multi-object tracking, the proposed algorithm effectively maintains track continuity and demonstrates robust performance of the track management module, even in scenarios with partial occlusion or temporary measurement loss. The tracking results show that targets are stably and consistently associated across consecutive frames, confirming the reliability and temporal stability of the developed approach.

For static road feature detection, the speed bump detection algorithm successfully identifies and localizes the bump structure in real driving environments. The experimental results verify its ability to provide smooth, continuous, and stable distance measurements during vehicle motion, even in the presence of sparse or noisy point cloud data. The algorithm shows strong resilience to common LiDAR sensing imperfections, confirming its robustness and applicability in real-world ADAS applications.

In summary, the proposed framework demonstrates that the perception system based on LiDAR point cloud can effectively support both dynamic and static environmental understanding in ADAS systems, contributing to safer and more intelligent vehicle operation.

Future Works

Future work will focus on the following directions:

- **Real-time deployment and integration:** implementing the algorithms in the onboard perception pipeline to validate performance under real driving conditions.
- **Sensor fusion:** integrating LiDAR with camera and radar data to enhance perception accuracy and robustness under various environmental conditions.
- **Complex road scenarios:** extending the methods to handle irregular road geometries and unstructured environments to improve generalization capability.
- **Adaptive optimization:** applying learning-based or data-driven techniques for automatic parameter tuning across different driving contexts.

These directions will further advance the practical applicability of LiDAR-based perception algorithms and contribute to the development of more reliable autonomous and assisted driving systems.

List of Figures

Bibliography

[1] Jamil Fayyad, Mohammad A. Jaradat, Dominique Gruyer, and Homayoun Najjaran. «Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review». In: *Sensors* 20.15 (2020). ISSN: 1424-8220. DOI: 10.3390/s20154220. URL: https://www.mdpi.com/1424-8220/20/15/4220 (cit. on p. 1).

[2] *SAE Levels of Driving Automation™ Refined for Clarity and International Audience?* URL: https://www.sae.org/news/blog/sae-levels-driving-automation-clarity-refinements (cit. on p. 2).

[3] Shuqi Wang and Meng Chen. «A LiDAR Multi-Object Detection Algorithm for Autonomous Driving». In: *Applied Sciences* 13.23 (2023). ISSN: 2076-3417. DOI: 10.3390/app132312747. URL: https://www.mdpi.com/2076-3417/13/23/12747 (cit. on p. 3).

[4] Stefano Favelli, Meng Xie, and Andrea Tonoli. «Sensor Fusion Method for Object Detection and Distance Estimation in Assisted Driving Applications». In: *Sensors* 24.24 (2024). ISSN: 1424-8220. DOI: 10.3390/s24247895. URL: https://www.mdpi.com/1424-8220/24/24/7895 (cit. on p. 3).

[5] Yujie Shen, Kai Jing, Kecheng Sun, Changning Liu, Yi Yang, and Yanling Liu. «Review of Uneven Road Surface Information Perception Methods for Suspension Preview Control». In: *Sensors* 25.18 (2025). ISSN: 1424-8220. DOI: 10.3390/s25185884. URL: https://www.mdpi.com/1424-8220/25/18/5884 (cit. on p. 4).

[6] Wei Wang, Jun Liu, Chenjie Wang, Bin Luo, and Cheng Zhang. «DV-LOAM: Direct Visual LiDAR Odometry and Mapping». In: *Remote Sensing* 13.16 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13163340. URL: https://www.mdpi.com/2072-4292/13/16/3340 (cit. on p. 5).

[7] Hao Zhu, Ka-Veng Yuen, Lyudmila Mihaylova, and Henry Leung. «Overview of Environment Perception for Intelligent Vehicles». In: *IEEE Transactions on Intelligent Transportation Systems* 18.10 (2017), pp. 2584–2601. DOI: 10.1109/TITS.2017.2658662 (cit. on p. 7).

[8] Jingmeng Zhou. «A Review of LiDAR sensor Technologies for Perception in Automated Driving». In: *Academic Journal of Science and Technology* 3.3 (Nov. 2022), pp. 255–261. DOI: 10.54097/ajst.v3i3.2993. URL: https://drpress.org/ojs/index.php/ajst/article/view/2993 (cit. on p. 7).

[9] *LIDAR 101: What is lidar?* URL: https://velodynelidar.com/what-is-LiDAR/. (accessed: 25.06.2023) (cit. on p. 8).

[10] Mark A. Richards. «Fundamentals of Radar Signal Processing». In: *Fundamentals of Radar Signal Processing*. 2005. URL: https://api.semanticscholar.org/CorpusID:56692486 (cit. on p. 8).

[11] Nanxi Li, Chong Pei Ho, Jin Xue, Leh Woon Lim, Guanyu Chen, Yuan Hsing Fu, and Lennon Yao Ting Lee. «A Progress Review on Solid-State LiDAR and Nanophotonics-Based LiDAR Sensors». In: *Laser & Photonics Reviews* 16.11 (2022), p. 2100511. DOI: https://doi.org/10.1002/lpor.202100511. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/lpor.202100511 (cit. on pp. 9, 11).

[12] *An Introduction to Automotive Lidar*. URL: https://www.ti.com/lit/wp/slyy150d/slyy150d.pdf?ts=1760291832545. (accessed: 2025) (cit. on pp. 10, 12).

[13] Saiful Islam, Md Shahnewaz Tanvir, Md Habib, Tahsina Tashrif, Md Apu Ahmed, Tafannum Ferdous, Md Arefin, and Sanim Alam. «Autonomous Driving Vehicle System Using LiDAR Sensor». In: "Springer Nature Singapore", Feb. 2022, pp. 345–358. ISBN: 978-981-16-7609-3. DOI: 10.1007/978-981-16-7610-9_25 (cit. on p. 10).

[14] Xinyu Sun, Lisheng Jin, Yang He, Huanhuan Wang, Huo Zhen, and Yewei Shi. «SimoSet: A 3D Object Detection Dataset Collected from Vehicle Hybrid Solid-State LiDAR». In: *Electronics* 12 (May 2023), p. 2424. DOI: 10.3390/electronics12112424 (cit. on p. 10).

[15] Saiful Islam, Md Shahnewaz Tanvir, Md. Rawshan Habib, Tahsina Tashrif Shawmee, Md Apu Ahmed, Tafannum Ferdous, Md. Rashedul Arefin, and Sanim Alam. «Autonomous Driving Vehicle System Using LiDAR Sensor». In: *Intelligent Data Communication Technologies and Internet of Things*. Ed. by D. Jude Hemanth, Danilo Pelusi, and Chandrasekar Vuppalapati. Singapore: Springer Nature Singapore, 2022, pp. 345–358. ISBN: 978-981-16-7610-9 (cit. on p. 12).

[16] Huan Zhang and Zunpei Wei. «Research on Point Cloud Data Preprocessing for Unmanned LiDAR». In: *2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. Vol. 7. 2024, pp. 617–621. DOI: 10.1109/IAEAC59436.2024.10504008 (cit. on p. 13).

[17] Riadh Ayachi, Yahia Said, Mouna Afif, Aadil Alshammari, Manel Hleili, and Abdessalem Ben Abdelali. «Assessing YOLO models for real-time object detection in urban environments for advanced driver-assistance systems (ADAS)». In: *Alexandria Engineering Journal* 123 (2025), pp. 530–549. ISSN: 1110-0168. DOI: https://doi.org/10.1016/j.aej.2025.03.077. URL: https://www.sciencedirect.com/science/article/pii/S1110016825003850 (cit. on p. 13).

[18] Yan Zhou, Sijie Wen, Dongli Wang, Jinzhen Mu, and Irampaye Richard. «Object Detection in Autonomous Driving Scenarios Based on an Improved Faster-RCNN». In: *Applied Sciences* 11.24 (2021). ISSN: 2076-3417. DOI: 10.3390/app112411630. URL: https://www.mdpi.com/2076-3417/11/24/11630 (cit. on p. 13).

[19] Yin Zhou and Oncel Tuzel. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. 2017. arXiv: 1711.06396 [cs.CV]. URL: https://arxiv.org/abs/1711.06396 (cit. on p. 14).

[20] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. «PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud». In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 770–779. DOI: 10.1109/CVPR.2019.00086 (cit. on p. 14).

[21] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: 1706.02413 [cs.CV]. URL: https://arxiv.org/abs/1706.02413 (cit. on p. 14).

[22] Fenglin Pang, Yutian Chen, Yan Luo, Zigui Lv, Xuefei Sun, Xiaobin Xu, and Minzhou Luo. «A Fast Obstacle Detection Algorithm Based on 3D LiDAR and Multiple Depth Cameras for Unmanned Ground Vehicles». In: *Drones* 8.11 (2024). ISSN: 2504-446X. DOI: 10.3390/drones8110676. URL: https://www.mdpi.com/2504-446X/8/11/676 (cit. on p. 14).

[23] Peng Xie, Hongcheng Wang, Yexian Huang, Qiang Gao, Zihao Bai, Linan Zhang, and Yunxiang Ye. «LiDAR-Based Negative Obstacle Detection for Unmanned Ground Vehicles in Orchards». In: *Sensors* 24.24 (2024). ISSN: 1424-8220. DOI: 10.3390/s24247929. URL: https://www.mdpi.com/1424-8220/24/24/7929 (cit. on p. 15).

[24] Jung-Hwan Lee, Hak-Jin Kim, Bong-Jin Cho, Jin-Ha Choi, and Young-Joo Kim. «Road Bump Detection Using LiDAR sensor for Semi-Active Control of Front Axle Suspension in an Agricultural Tractor». In: *IFAC-PapersOnLine* 51.17 (2018). 6th IFAC Conference on Bio-Robotics BIOROBOTICS 2018, pp. 124–129. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.

2018.08.074. URL: https://www.sciencedirect.com/science/article/pii/S240589631831190X (cit. on p. 15).

[25] Jing Li, Rui Li, Junzheng Wang, and Min Yan. «Obstacle information detection method based on multiframe three-dimensional lidar point cloud fusion». In: *Optical Engineering* 58.11 (2019), p. 116102. DOI: 10.1117/1.OE.58.11.116102. URL: https://doi.org/10.1117/1.OE.58.11.116102 (cit. on p. 15).

[26] Hong Lang, Yuan Peng, Zheng Zou, Shengxue Zhu, Yichuan Peng, and Hao Du. «Multi-Feature-Filtering-Based Road Curb Extraction from Unordered Point Clouds». In: *Sensors* 24.20 (2024). ISSN: 1424-8220. DOI: 10.3390/s24206544. URL: https://www.mdpi.com/1424-8220/24/20/6544 (cit. on p. 15).

[27] Luca Rosafalco, Paolo Conti, Andrea Manzoni, Stefano Mariani, and Attilio Frangi. «EKF–SINDy: Empowering the extended Kalman filter with sparse identification of nonlinear dynamics». In: *Computer Methods in Applied Mechanics and Engineering* 431 (Nov. 2024), p. 117264. ISSN: 0045-7825. DOI: 10.1016/j.cma.2024.117264. URL: http://dx.doi.org/10.1016/j.cma.2024.117264 (cit. on p. 16).

[28] Riccardo Pieroni, Simone Specchia, Matteo Corno, and Sergio Matteo Savaresi. *Multi-Object Tracking with Camera-LiDAR Fusion for Autonomous Driving*. 2024. arXiv: 2403.04112 [cs.RO]. URL: https://arxiv.org/abs/2403.04112 (cit. on p. 17).

[29] Oscar Montañez, Marco Suarez, and Eduardo Avendano Fernandez. «Application of Data Sensor Fusion Using Extended Kalman Filter Algorithm for Identification and Tracking of Moving Targets from LiDAR–Radar Data». In: *Remote Sensing* 15 (July 2023), p. 3396. DOI: 10.3390/rs15133396 (cit. on p. 18).

[30] *NVIDIA Jetson Orin*. URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/. (accessed: 10.10.2025) (cit. on p. 23).

[31] *Falcon K1*. URL: https://www.seyond.com/products/falcon-k1/. (accessed: 10.10.2025) (cit. on p. 24).

[32] *ouster Os1*. URL: https://ouster.com/products/hardware/os1-lidar-sensor. (accessed: 10.10.2025) (cit. on p. 25).

[33] *PIXKIT 2.0*. URL: https://www.pixmoving.com/pixkit. (accessed: 10.10.2025) (cit. on pp. 27, 28).

[34] *ros2*. URL: https://docs.ros.org/en/humble/index.html. (accessed: 10.10.2025) (cit. on p. 30).

[35] Akshay Rangesh and Mohan Manubhai Trivedi. «No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles Using Cameras and LiDARs». In: *IEEE Transactions on Intelligent Vehicles* 4.4 (2019), pp. 588–599. DOI: 10.1109/TIV.2019.2938110 (cit. on p. 34).