Ego Vehicle Localization in Dynamic Environments Using LiDAR-Based Classification of Static and Dynamic Tracks via Probabilistic Graphical Models

Muhammad Adnan

Supervisor: Carlo Regazzoni
David Martín Gómez
Lucio Marcenaro

Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture (DITEN), University of Genoa
Department of Computer Science and Engineering (Departamento de Informática), Universidad Carlos III de Madrid

This dissertation is submitted for the degree of
Doctor of Philosophy

Defense Month: December,2025

Ego Vehicle Localization in Dynamic Environments Using LiDAR-Based Classification of Static and Dynamic Tracks via Probabilistic Graphical Models

Muhammad Adnan

Joint Doctorate in Interactive and Cognitive Environments
JD-ICE

XXXVII Cycle

Acknowledgements

This PhD Thesis has been developed in the framework of, and according to, the rules of the Joint Doctorate in Interactive and Cognitive Environments (JD-ICE) with the cooperation of the following Universities:

Università degli Studi di Genova (UNIGE)
DITEN – Dept. of Electrical, Electronic, Telecommunications Engineering and Naval Architecture
ISIP40 – Information and Signal Processing for Cognitive Telecommunications

Primary Supervisor: Prof. Carlo Regazzoni
Co-supervisor: Prof. Lucio Marcenaro

Universidad Carlos III de Madrid (UC3M)
DIE – Department of Computer Science and Technology
ISL – Intelligent Systems Laboratory

Supervisor: Prof. David Martín Gómez

Acknowledgements

My PhD journey has been a remarkable and transformative experience, and it would not have been possible without the guidance, encouragement, and support of many individuals and institutions. First and foremost, I wish to extend my deepest gratitude to my primary supervisor, Prof. Carlo Regazzoni at the University of Genoa, for his invaluable mentorship, insightful advice, and continuous encouragement throughout my research. His expertise and dedication have played a vital role in shaping my academic development.

I am also privileged to acknowledge the guidance, constructive feedback, and inspiring discussions I received from my secondary supervisor, Prof. David Martin Gomez at the University Carlos III of Madrid. His critical insights have significantly enhanced the quality of this research. I sincerely thank Prof. Lucio Marcenaro for his unwavering support and continuous availability throughout my research journey. My sincere thanks also go to Prof. Pamela Zontone for her insightful discussions and technical assistance during various stages of this project. I am especially grateful to my colleague Abrham Shiferaw, whose technical guidance and support during the initial stages of my research were invaluable. I would also like to thank Dr. Giulia Slavic, my senior colleague and coauthor on a review paper, who has always responded promptly and supported me whenever I faced technical difficulties in our research work.

I am deeply thankful to my parents and family for their unwavering love, encouragement, and support throughout my academic career. Their belief in me has been my greatest source of strength. Finally, I would like to thank the University of Genova for its financial support, which made this PhD possible. This thesis is dedicated to all those who have supported and believed in me throughout this remarkable journey.

Published and Submitted Content

Journal Papers

1. **Adnan M**, Slavic G, Martin Gomez D, Marcenaro L, Regazzoni C.
   "Systematic and Comprehensive Review of Clustering and Multi-Target Tracking Techniques for LiDAR Point Clouds in Autonomous Driving Applications," *Sensors*, vol. 23, no. 13, pp. 6119, 2023.
   doi: 10.3390/s23136119. © 2023 MDPI.

   (This journal paper is partially included in the thesis, in Chapter 3. The material from this source included in this thesis is not singled out with typographic means and references.)

2. **Adnan M**, Zontone P, Martín Gómez D, Marcenaro L, Regazzoni C.
   "A Generative Model Approach for LiDAR-Based Classification and Ego Vehicle Localization Using Dynamic Bayesian Networks," *Applied Sciences*, vol. 15, no. 9, 5181, 2025.
   doi: 10.3390/app15095181. © 2025 MDPI.
   (This journal paper is partially included in the thesis, in Chapters 4, 5, and 6. The material from this source included in this thesis is not singled out with typographic means and references.)

Conference Papers

1. **Adnan M**, Zontone P, Marcenaro L, Gómez DM, Regazzoni C.
   "Classifying Static and Dynamic Tracks for LiDAR-Based Navigation of Autonomous Vehicle Systems," in *Proc. of the 9th Int. Conf. on Frontiers of Signal Processing (ICFSP)*, Sep. 2024, pp. 111–117.
   doi: 10.1109/ICFSP62546.2024.10785465 © 2024 IEEE.

   (This paper is wholly included in the thesis, in Chapter 4. However, the dataset used includes different tracks from those used in the conference paper.)

2. **Adnan M**, Zontone P, Marcenaro L, Gómez DM, Regazzoni C.
   "Autonomous Vehicle Localization via LiDAR-Based Classification of Dynamic and Static Tracks in Dynamic Environments,"in *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, 2025.
   doi: 10.1109/IJCNN64981.2025.11227789.

   (This conference paper is wholly included in the thesis, in chapters 5 and 6.)

Abstract

Autonomous vehicles (AVs) are required to be able to perceive and locate their surroundings accurately in dynamic environments. Using LiDAR and odomotry data, this dissertation proposes a novel framework for classifying tracks into static and dynamic categories and using static tracks as reliable landmarks for ego vehicle localization. This research is based on the principles of multi-target tracking (MTT), Growing Neural Gas (GNG) clustering, and Dynamic Bayesian Networks (DBN), integrating advanced algorithms such as Joint Probabilistic Data Association (JPDA) and Markov Jump Particle Filter (MJPF). In the training phase, LiDAR and odometry data were used to classify tracks based on relative motion patterns, resulting in an 87% classification accuracy. During the localization process, static tracks are identified as invariant landmarks, and dynamic tracks are excluded due to their variability. The framework uses the classified static tracks as reference points for predicting the trajectory of ego vehicle during the testing phase. Localization results are initially obtained based on individual track predictions using MJPF in the testing phase. Afterwards, interaction dictionaries are combined to perform localization under scenarios, such as simultaneous multi-track interactions and periods with no observations and single track interaction. Results from experiments, which validate the framework's adaptability to real-world autonomous navigation scenarios, demonstrate that the framework is capable of obtaining accurate localization without external odometry updates. In this research, reliable classification techniques are combined with an adaptable localization strategy, advancing the development of safe and efficient AVs.

Table of contents

List of figures

List of tables

Nomenclature

Roman Symbols

2D Two-Dimensional

3D Three-Dimensional

ACKF Adaptive Cubature Kalman Filter

ADAS Advanced Driver Assistance Systems

AV Autonomous Vehicle

BEV Bird's Eye View

BN Bayesian Network

BN Bayesian Network

BPNN Back-Propagating Neural Network

CLA Continuous Level Anomaly

CLB Continuous Level Behavior Anomaly

CNN Convolutional Neural Network

DAC DBSCAN-Based Adaptive Clustering

DATMO Detection and Tracking of Moving Objects

DBN Dynamic Bayesian Network

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DNN Deep Neural Network

EgoVM Ego Vehicle Localization Network

GB-DBSCAN Grid-Based DBSCAN

GNG Growing Neural Gas

GPS Global Positioning System

HEVC High-Efficiency Video Coding

HMM Hidden Markov Model

ICP Iterative Closest Point

IMU Inertial Measurement Unit

IoU Intersection Over Union

ISCIF Iterated Split Covariance Intersection Filter

JPDA Joint Probabilistic Data Association

KF Kalman Filter

KLDA Kullback-Leibler Divergence Anomaly

LiDAR Light Detection and Ranging

MJPF Markov Jump Particle Filter

MLP Multi-Layer Perceptron

MTT Multi-Target Tracking

NFF Null Force Filter

OD-C3DL Object Detection and Clustering in 3D LiDAR

OSM OpenStreetMap

PCA Point Cloud Augmentation

PD Persistence Diagram

PEN Pose Estimation Network

PF Particle F ilter

PF Particle Filter

PH Persistent Homology

PI Persistence Image

RANSAC Random Sample Consensus

RBSS Recursive Best Segment Split

ROI Region of Interest

RTMCL Real-Time Monte Carlo Localization

SLAM Simultaneous Localization and Mapping

SRT Scale-Rotation-Translation

SVM Support Vector Machine

TLG Two-Layer-Graphs

UKF Unscented Kalman Filter

VGG16 Visual Geometry Group-16 (a CNN architecture)

Chapter 1

Introduction

1.1 Motivation

AVs are designed to eliminate or significantly reduce human intervention in vehicle operation, ensuring greater efficiency, safety, and autonomy in various real-world applications. The field of AV research has seen rapid advancements since the 1980s, with applications spanning transportation, agriculture, disaster response, security, and surveillance [1, 2]. The ability of AVs to operate autonomously in highly dynamic and unpredictable environments relies on their ability to perceive, interpret, and adapt to their surroundings. Two fundamental paradigms govern AV research: the computationalist approach, which relies on predefined mathematical models and rigid control rules, and the cognitive approach, which allows AVs to dynamically learn and refine their internal models based on real-time sensor data [3]. While computationalist methods provide structured frameworks for decision-making, they often fail in real-world scenarios where unforeseen events and changing conditions necessitate adaptive learning and reasoning. The cognitive approach, inspired by human cognition, enables AVs to develop self-awareness, allowing them to make intelligent decisions based on past experiences and sensory inputs.

Self-awareness in AVs is a crucial capability that enables the system to comprehend its internal state while monitoring external environmental conditions. This is achieved through the integration of exteroceptive sensors (e.g., LiDAR, cameras, radar) and proprioceptive sensors (e.g., Inertial Measurement Units (IMUs), steering angle sensors, wheel speed sensors), enabling AVs to perceive both external obstacles and their own motion [4, 5]. The ability to interpret and fuse data from multiple sensor sources forms the basis of a self-aware AV, allowing it to understand its surroundings, detect anomalies, and adjust its motion accordingly [6].

A key aspect of AV perception is accurate classification and localization, both of which are essential for safe and efficient navigation [7]. Classification allows AVs to distinguish between static (e.g., buildings, traffic signs, poles) and dynamic (e.g.,

vehicles, cyclists, pedestrians) objects. Accurate classification ensures that AVs can reliably interpret which objects remain fixed and which are moving, allowing them to anticipate interactions, make informed decisions, and optimize trajectory planning. On the other hand, localization refers to an AV's ability to accurately determine its position and motion within an environment. Traditional localization methods such as GPS and odometry-based approaches often suffer from signal loss, drift, and inaccuracies in complex environments such as urban canyons, tunnels, and cluttered environments. The absence of GPS and odometry makes localization extremely challenging in such cases. Additionally, GPS signals are vulnerable to intentional interference such as jamming and meaconing, which can severely degrade positioning accuracy or result in spoofed locations—posing critical safety risks for autonomous systems [8]. Therefore, alternative localization techniques are required to enhance reliability and robustness, particularly in dynamic environments where environmental conditions constantly change [9, 10].

In this thesis, LiDAR-based perception is utilized to enhance both classification and localization capabilities. LiDAR generates high-resolution 3D point clouds, which allow AVs to precisely map their surroundings and distinguish between different object categories [11, 12]. LiDAR's ability to capture spatially rich depth information makes it superior to vision-based systems, particularly in adverse lighting conditions, poor weather, and complex urban environments [13]. The classification of LiDAR-based tracks into static and dynamic categories provides landmark references for localization, ensuring that AVs can estimate their position accurately even in the absence of GPS signals or external odometry updates [14]. By leveraging LiDAR-based classification techniques, AVs can use static objects as anchor points for precise localization, while simultaneously tracking and interpreting the motion of dynamic objects.

The motivation for this research extends beyond the intrinsic benefits of LiDAR sensors to its application in the pipeline used to classify and localize AVs. Navigating precisely in dynamic environments requires not only the perception of static and dynamic objects, but also their effective use for localization [15–17]. Using LiDAR's capability to generate 3D point clouds, a classification method is proposed that identifies static and dynamic tracks in the ego vehicle environment [18]. Static tracks provide reliable landmarks for ego vehicle localization because they remain invariant over time. Dynamic tracks, on the other hand, pose difficulties because of their motion [19, 20]. However, the ability to understand interactions with dynamic tracks is essential. Based on this duality, this work focused on accurately classifying tracks during the training phase using algorithms such as JPDA, GNG, and interaction dictionaries. The resulting classified static tracks are then used during testing to determine the ego vehicle's location without relying on external odometry updates, ensuring robustness in real-world situations.

These concepts are explicitly integrated into a cognitive self-awareness cycle. An AV can acquire knowledge of the structure of its environment, generate dictionaries of

potential interactions with a variety of static objects, and characterize the dynamics of moving objects during the offline training phase. Afterward, during the online testing phase, the AV's pose is continuously monitored, anomalies are detected, and the system is able to adapt to unanticipated circumstances by fusing the acquired knowledge with new sensor data using probabilistic filters, specifically an MJPF. In essence, classification and localization serve as two fundamental components of the overarching self-awareness strategy, in which they are perpetually engaged with modules of a higher level.
The design of the proposed framework is driven by the need to model complex temporal interactions between the ego vehicle and its environment using only LiDAR data. A hierarchical DBN structure is adopted to capture multi-level dependencies, enabling both low-level observation modeling and high-level behavior abstraction. This structure allows for efficient integration of continuous observations and discrete semantic states—crucial for interpreting static and dynamic track interactions. The use of interaction dictionaries further complements the DBN by encoding learned spatial-temporal patterns, ensuring robustness during ambiguous or sparse observations. These design choices collectively support a scalable and adaptive localization system that can operate without external odometry, even in real-world conditions.

1.2 Research Questions

This thesis addresses the following key research questions:

- How can LiDAR data be effectively utilized to classify static and dynamic tracks in real-world environments?
- How can ego-vehicle and track interactions be modeled using hierarchical DBNs, incorporating switching variables for enhanced localization?
- How can the ego-vehicle be accurately localized in real-time using only LiDAR data, while integrating odometry information obtained during offline training?

1.3 Thesis Outline

Each chapter in this thesis builds upon the previous one, forming a coherent framework for LiDAR-based classification and localization. Chapter 1 establishes the research context, highlighting the importance of AV self-awareness and LiDAR-based perception. It defines the motivation for this work and introduces the core methodologies used in this thesis.
Chapter 2 presents foundational concepts related to MTT, JPDA, GNG clustering, and Bayesian filtering. It introduces DBNs, which serve as the backbone of probabilistic

modeling for AV decision-making. The chapter also explores Kalman Filters, Particle Filters, and MJPF, all of which play critical roles in classification of objects and localization for ego vehicle.
Chapter 3 provides a comprehensive review of existing classification and localization methods. It explores various LiDAR-based perception techniques, evaluates MTT strategies, and examines the latest advancements in landmark-based localization.
Chapter 4 details the offline phase of the research, where LiDAR and odometry data are processed to create static and dynamic track dictionaries. The chapter presents quantitative accuracy results, confusion matrix, and visual evaluations demonstrating the effectiveness of the classification approach.
Chapter 5 introduces the online testing phase of the proposed localization framework. In this stage, the static tracks previously identified during the training phase are used as reliable landmarks for ego vehicle localization through the Markov Jump Particle Filter (MJPF). The chapter details how the proposed method handles various real-world scenarios by fusing multiple observations, mitigating drift, and enabling robust trajectory estimation. The focus remains on explaining the architecture and logic of the localization process. While it introduces the components and theoretical underpinnings of the approach, it does not yet present experimental results.
Chapter 6 builds upon the methodology described in Chapter 5 by presenting the experimental results. This includes both individual track-based predictions and combined dictionary-based localization outcomes. The results are evaluated across diverse scenarios such as simultaneous observations, absence of observations, and multi-track interactions. Quantitative analyses, anomaly detection metrics, and computational performance evaluations are also included to validate the effectiveness of the proposed framework.
Chapter 7 concludes the thesis by summarizing the main findings and contributions, particularly the integration of LiDAR-based classification with probabilistic localization strategies. It also outlines potential future research directions, such as incorporating multimodal sensor fusion, enhancing the anomaly detection mechanisms, and improving real-time trajectory optimization for autonomous systems.

Notation

This section summarizes the key mathematical symbols and notation conventions used throughout the thesis:

- Scalars are denoted by regular lowercase or uppercase letters, e.g., a, b, A.
- Vectors are represented using bold lowercase letters, e.g., $\mathbf{x}, \mathbf{v} \in \mathbb{R}^n$.
- Matrices are denoted by bold uppercase letters, e.g., $\mathbf{A}, \mathbf{Q}, \mathbf{\Sigma}$.

- $(\cdot)^T$: Transpose of a vector or matrix.
- $\|\cdot\|$: Euclidean norm of a vector.
- $\mathbb{R}^n$: The n-dimensional real vector space.
- $\mathbb{E}[\cdot]$: Expectation operator.
- $\mathrm{Var}[\cdot]$: Variance of a random variable.
- $\det(\cdot)$: Determinant of a matrix.
- $\mathbf{I}_n$: $n \times n$ identity matrix.
- $\sim \mathcal{N}(\mu, \Sigma)$: Gaussian distribution with mean μ and covariance matrix Σ.
- $p(\cdot)$: Probability density function.
- z_t: Observation at time t.
- $\mathbf{x}_t$: Ego vehicle state at time t.
- $\mathbf{z}_{\text{track},t}$: State of an observed track at time t.
- $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{K}, \mathbf{D}, \mathbf{E}$: System matrices (state transition, observation, Kalman gain, correction terms).
- $d_B(\cdot, \cdot)$: Bhattacharyya distance between two distributions.
- t: Timestamp of observation or interaction.
- $C_{\text{ego}}, C_{\text{track}}$: Cluster indices for the ego vehicle and track, respectively.
- $L \in \{0, 1\}$: Binary interaction label (1 = interaction occurred).
- $\mathbf{v}_{\text{ego}}, \mathbf{v}_{\text{track}}$: Velocities of the ego vehicle and track.
- v_{rel}: Relative velocity between ego and track.
- $\mathbf{Q}, \mathbf{R}$: Process and measurement noise covariance matrices.
- $\mathbf{S}$: Switching variable in the Markov Jump Particle Filter (MJPF).
- $\mathbf{I}_{\text{combined}}$: Combined interaction dictionary constructed from static tracks.

All notation specific to individual models or algorithms is introduced contextually in the corresponding chapters.

Chapter 2

Theoretical Background

The purpose of this chapter is to provide an overview of some of the foundational ideas and methodologies that are essential for understanding the approaches developed and applied in this thesis. An emphasis is placed on techniques integral to the classification of static and dynamic tracks, as well as their application to the localization of ego vehicles within dynamic environments. A number of theoretical topics are discussed, including multi-target tracking, unsupervised clustering, dynamic bayesian networks, markov jump particle filter, and probabilistic graphical models. Based on the frameworks we have proposed, these concepts are critical to the effective use of LiDAR data and interaction dictionaries. As a result of elucidating these methodologies, this chapter aims to bridge the knowledge gap for readers, ensuring a comprehensive understanding of the techniques underlying the classification and localization processes outlined in subsequent chapters.

2.1 Multi-Target Tracking

Tracking is necessary for autonomous systems to guide, navigate, and control themselves. Tracking systems are capable of estimating targets (number of targets and their states) and evaluating the situational environment in a particular area by tracking detected targets (including their kinematic parameters and attributes) [21]. Tracking systems employ MTT, which involves generating multiple detections from multiple targets and estimating the state of the targets using one or more tracks [22]. In MTT, once an object is initially detected, subsequent measurements often imprecise and noisy which are incrementally gathered over time. These side measurements are probabilistically associated with existing tracks, allowing the system to refine its understanding of each object's trajectory and state. Rather than assigning detections directly to tracks, the detection stage produces initial candidates, and subsequent observations contribute as rough measurements linked to the evolving object tracks. The functional components of a simple recursive MTT system (as shown in Figure 2.1) play different roles in assigning

detections to tracks, are discussed as follows.

An observation or measurement is considered to be a detection when it is included in a report based on the sensor's output. Observations typically include measurements of kinematic quantities (such as range, line of sight, and range rate) as well as attributes (such as target type, identification number, or shape) [23]. It is also important to include the time at which measurements were made in the detections. Multiple detections per target may be produced by sensors with high resolution, making it necessary to partition the detections before feeding them to assignment-based trackers, such as the Joint Probabilistic Data Association (JPDA) [24].

Gates are used to identify which detections are eligible for updating existing tracks. Track-to-detection assignment uses gates to reduce unnecessary computations. A validation gate is constructed for a predicted track based on the predicted state and its covariance, so that detections with a high likelihood of association are included in the validation gate. Gating determines which tracks should be assigned to detectionse as a result of gating [25].

Initiating, confirming, and deleting tracks is the responsibility of track maintenance. When a detection cannot be assigned to an existing track, a new track may need to be created. The track confirmation step identifies the status of a tentative track once it has been created. A track is deleted if it is not updated within a reasonable period of time, and the criteria for track deletion are similar to those for confirmation of tracks [26].

A tracking filter is comprised of three main functions: it predicts the tracks for the current moment, calculates the distance between the predicted tracks and detections for gating and assignment, and corrects the predicted tracks with assigned detections [27].

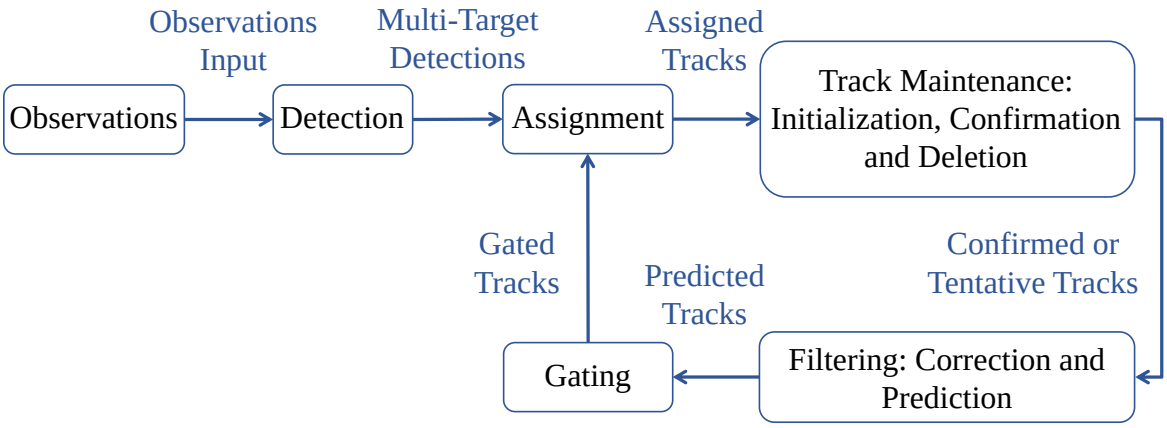


Fig. 2.1 Key components of multi-target tracking, including detection, data association, and trajectory estimation. These components work cohesively to track multiple objects over time, even in dynamic environments.

2.2 Clustering

Clustering is a fundamental technique in machine learning used to group data points based on their inherent similarities, enabling the discovery of patterns and structures in large datasets. It is primarily employed as an unsupervised learning method, allowing the segmentation of data without prior labels. However, clustering can also be integrated into semi-supervised frameworks where partial supervision is available in the form of labeled constraints. This approach enhances its performance in scenarios where limited labeled data exists [28].

In the context of LiDAR-based perception, clustering plays a pivotal role in segmenting point clouds to distinguish between static and dynamic objects [29]. Various clustering algorithms, such as K-Means [30], Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [30], Hierarchical Clustering [31], and GNG [32], have been extensively utilized in autonomous systems.

K-Means is a centroid-based clustering method that partitions data into a predefined number of clusters. Despite its efficiency, K-Means assumes spherical clusters, making it unsuitable for irregularly shaped data distributions often encountered in LiDAR applications. In contrast, DBSCAN is a density-based clustering algorithm that excels in identifying arbitrarily shaped clusters while effectively handling noise. It requires two key parameters: the minimum number of points in a neighborhood to define a dense region and the neighborhood radius. DBSCAN has been successfully applied in LiDAR data processing for object detection and tracking due to its robustness in distinguishing closely spaced objects.

Hierarchical clustering constructs a tree-like hierarchy of clusters, either through an agglomerative (bottom-up) or divisive (top-down) approach. This method provides greater interpretability, enabling researchers to analyze clustering results at multiple levels of granularity. However, its computational complexity poses challenges for large-scale LiDAR datasets. The GNG algorithm dynamically adapts to the topology of data, making it particularly effective for complex, non-linear datasets. Unlike traditional clustering methods that rely on predefined structures, GNG incrementally adds and removes nodes to optimize its representation of input data. This adaptability is beneficial for LiDAR-based object recognition, where the environment continuously evolves with varying densities of data points.

In real-world applications, clustering is instrumental in autonomous navigation, object detection, and behavior analysis. For instance, in self-driving vehicles, clustering helps classify point cloud data into distinct entities such as pedestrians, vehicles, and stationary objects. This segmentation allows vehicles to make informed decisions regarding path planning and collision avoidance. Moreover, clustering methods are integrated into deep learning pipelines to enhance feature extraction and improve classification accuracy [33].

Recent advancements in clustering techniques leverage deep learning architectures to enhance clustering accuracy. Convolutional Neural Networks (CNNs) and Graph Neural Networks (GNNs) have been incorporated into clustering frameworks to learn spatial dependencies and improve segmentation quality [34]. Additionally, hybrid approaches that combine DBSCAN with supervised learning models have demonstrated improved robustness in cluttered environments [35].

In addition to data-driven clustering techniques, model-based approaches have been widely developed in the context of statistical signal processing and Bayesian modeling. One fundamental class of such methods involves latent variable models, where the observed data is assumed to be generated from a mixture of underlying probabilistic distributions. Gaussian Mixture Models (GMMs) are a notable example, where each cluster corresponds to a Gaussian component. The parameters of such models are typically estimated using the Expectation-Maximization (EM) algorithm, which iteratively refines the model by alternating between estimating hidden variables (E-step) and optimizing the likelihood with respect to parameters (M-step). This class of algorithms provides a principled probabilistic framework for clustering and offers benefits such as interpretability and handling of uncertainty [36].

In summary, clustering serves as a fundamental tool in machine learning and autonomous perception, enabling the extraction of meaningful patterns from unstructured data. The selection of an appropriate clustering algorithm depends on the specific requirements of the task, including computational efficiency, scalability, and the ability to handle noise and non-linear structures. Continued research in clustering methodologies will further enhance their applicability in real-world scenarios, contributing to the advancement of autonomous systems [37].

2.3 Dynamic Bayesian Networks

A DBN [38] is an extension of Bayesian Networks (BNs) designed to model stochastic dynamic processes over time. DBNs provide a probabilistic framework for representing sequences of random variables, allowing efficient reasoning about temporal dependencies. They are particularly effective in modeling non-linear, multi-modal, and uncertain environments, making them ideal for applications such as autonomous navigation, object tracking, and decision-making in uncertain scenarios.

DBNs are represented as directed acyclic graphs (DAGs), where nodes represent random variables and edges capture their probabilistic dependencies. Unlike static BNs, which only model dependencies at a single time step, DBNs extend this representation by introducing temporal connections between variables across consecutive time steps. This ability to encode both intra-time slice and inter-time slice dependencies allows DBNs to effectively model dynamic processes with evolving states.

The architecture of a DBN consists of intra-slice dependencies, which describe relationships among variables within the same time step, and inter-slice dependencies, which capture dependencies across time steps, enforcing temporal continuity in the modeled system. The Markov property ensures that a variable's state at a given time step depends only on its state at the previous time step, reducing computational complexity [39]. Mathematically, the probabilistic update of a DBN over time can be expressed using Bayes' theorem as:

$$P(S_t \mid S_{t-1}, O_t) = \frac{P(O_t \mid S_t) \cdot P(S_t \mid S_{t-1})}{P(O_t \mid S_{t-1})}, \tag{2.1}$$

where S_t is the latent state at time t, S_{t-1} is the previous state, and O_t represents the current observation. This equation describes how the posterior belief over the current state is updated based on the likelihood of the observation and the transition dynamics from the previous state. The denominator $P(O_t \mid S_{t-1})$ serves as a normalization constant. Such a formulation enables DBNs to capture temporal uncertainty effectively, which is especially beneficial for dealing with noisy sensor inputs like LiDAR in autonomous systems.

A visual representation of a DBN structure is illustrated in **Figure 2.2**, which depicts the hierarchical relationships between different levels of variables over multiple time steps. The figure demonstrates how DBNs model sequential dependencies by encoding probabilistic connections across various nodes, facilitating a structured decomposition of complex, non-linear dynamics into manageable units.

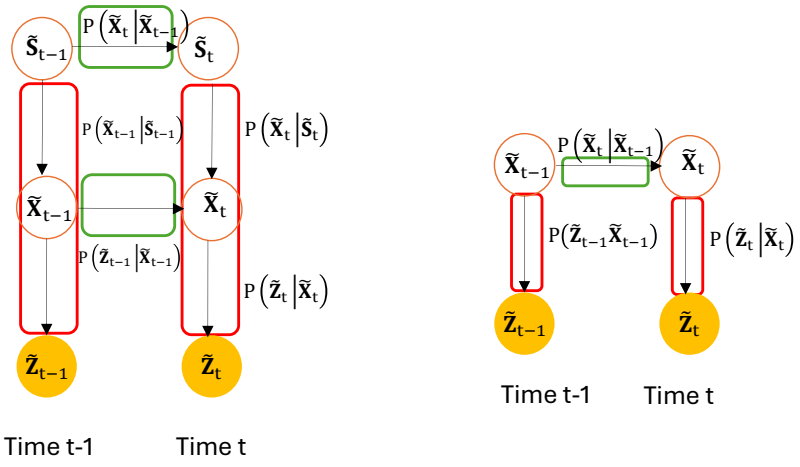


Fig. 2.2 A schematic representation of a Dynamic Bayesian Network (DBN) illustrating inter-slice dependencies (green links) and intra-slice dependencies (red links). This hierarchical structure decomposes complex dynamics into simpler relationships for efficient state estimation.

Learning a DBN involves two primary tasks: parameter learning and structure learning. Parameter learning involves estimating the probabilistic distributions governing the dependencies between variables, typically achieved using techniques such as Maximum

Likelihood Estimation (MLE) or Expectation-Maximization (EM). Structure learning aims to identify the optimal topology of the DBN by determining which variables are directly dependent on each other. Algorithms such as the K2 algorithm, Greedy Search, and Bayesian Model Averaging are commonly used for this purpose [40]. DBNs are widely used in real-world applications where sequential decision-making and state estimation are required. Some of their key applications include object tracking, where DBNs predict and update object trajectories over time by incorporating noisy sensor data [40]. Additionally, DBNs serve as the foundation for modern speech recognition systems by modeling temporal dependencies in audio signals, enabling accurate word prediction. In robotics and human-computer interaction, DBNs facilitate activity recognition by analyzing sequences of human actions and predicting behavioral patterns. Furthermore, DBNs are employed in sensor fusion applications, where multi-sensor data sources (e.g., LiDAR, radar, and cameras) are integrated to enhance situational awareness in autonomous systems. Modern advancements have led to the development of hybrid models that combine DBNs with deep learning techniques. Notable extensions include deep DBNs, which leverage deep neural networks to learn complex temporal dependencies while retaining the interpretability of traditional DBNs. Hierarchical DBNs introduce multi-layered structures that improve robustness in real-time decision-making by modeling abstract high-level states. Additionally, DBNs integrated with reinforcement learning frameworks enhance adaptability in decision-making processes, particularly in autonomous navigation tasks [41].

2.4 Probabilistic Inference

Probabilistic inference is a fundamental principle in Bayesian reasoning that enables dynamic systems, such as AVs, to estimate hidden states from noisy and incomplete observational data [42]. In real-world environments, direct measurement of all relevant system variables is impractical; therefore, probabilistic inference provides a systematic approach to estimate states such as an agent's position, velocity, and other latent variables by leveraging available sensor data.

In Bayesian filtering, the process of inference consists of two key steps: **prediction** and **update** [43]. These steps are recursively applied to refine the estimated state as new observations become available, making the approach particularly useful for real-time applications such as AV perception and motion planning.

The **prediction step** estimates the state at the next time step, s_t, based on the transition probability $P(s_t \mid s_{t-1})$ while conditioning on all previous observations $O_{1:t-1}$. This step is expressed mathematically as:

$$P(s_t \mid O_{1:t-1}) = \int P(s_t \mid s_{t-1}) P(s_{t-1} \mid O_{1:t-1}) \mathrm{d}s_{t-1}. \tag{2.2}$$

The prediction step propagates the uncertainty in the system forward in time, allowing the model to make informed estimates of future states.

In the **update step**, the prior belief from the prediction step is refined by incorporating new observations, O_t, using Bayes' theorem:

$$P(s_t \mid O_{1:t}) = \alpha P(O_t \mid s_t) P(s_t \mid O_{1:t-1}), \tag{2.3}$$

where α is a normalization constant ensuring that probabilities sum to 1. At any time t, the terms in the equation can be interpreted as follows:

- $P(s_t \mid O_{1:t-1})$ represents the *prior*, or the belief about the state before the new observation is considered.
- $P(O_t \mid s_t)$ is the *likelihood*, which quantifies the probability of observing O_t given state s_t.
- $P(s_t \mid O_{1:t})$ is the *posterior*, the updated belief about the state incorporating the latest observation.

This recursive process enables AVs and other intelligent systems to iteratively refine their state estimates over time, continuously improving accuracy and robustness in uncertain environments.

Probabilistic inference is widely applied in various filtering techniques such as Kalman Filters (KF), Particle Filters (PF), and Hidden Markov Models (HMM). Each of these methods builds upon Bayesian filtering principles to optimize state estimation under different assumptions about system dynamics and noise characteristics. For instance, the Kalman Filter assumes linear Gaussian noise, while Particle Filters employ a non-parametric approach that can model highly non-linear systems with multi-modal distributions [44].

Furthermore, probabilistic inference plays a crucial role in sensor fusion, where multiple sources of data (e.g., LiDAR, radar, and cameras) are combined to generate a more accurate and robust estimate of the environment. By leveraging probabilistic reasoning, AVs can infer missing information, reject outliers, and adapt to dynamic scenarios with varying levels of uncertainty.

2.5 Kalman Filter

The KF is one of the most widely used linear estimators for tracking and state estimation in dynamic systems. It provides an optimal recursive solution for estimating the state of a linear system subject to Gaussian noise. The filter is particularly useful in applications where real-time updates are required, such as in AVS, robotics, and control systems [45].

The KF operates on the principle of a **state-space model**, where the system's evolution over time is governed by a transition model and an observation model. These models can be formulated as follows:

Transition model:

$$s_t = As_{t-1} + Bu_{t-1} + \eta_{t-1}, \tag{2.4}$$

where s_t represents the state vector at time t, A is the state transition matrix, B is the control input matrix, u_{t-1} is the control input, and η_{t-1} represents process noise assumed to be zero-mean Gaussian with covariance Q. The noise term η_{t-1} is modeled as a zero-mean Gaussian random variable with covariance matrix Q, which captures the uncertainty in the system dynamics.

Observation model:

$$O_t = Hs_t + \epsilon_t, \tag{2.5}$$

where H is the observation matrix mapping the state to the observation space, and ϵ_t is the measurement noise, also assumed to be Gaussian with covariance R.

The KF operates in two main iterative phases: **prediction** and **update**.

Prediction step: This step estimates the current state based on the previous state and the system dynamics. Note that the matrix multiplication in the following equation implies summation over vector components as per standard linear algebra notation. Specifically, $A\hat{s}_{t-1|t-1}$ and Bu_{t-1} represent linear transformations of the state and control input vectors, respectively.

$$\hat{s}_{t|t-1} = A\hat{s}_{t-1|t-1} + Bu_{t-1}, \tag{2.6}$$

In below equation, $\Sigma_{t|t-1}$ represents the predicted state covariance matrix, capturing the uncertainty in the forecasted state at time t. The matrix $A^\top$ denotes the transpose of the state transition matrix A, and Q is the covariance of the process noise.

$$\Sigma_{t|t-1} = A\Sigma_{t-1|t-1}A^\top + Q, \tag{2.7}$$

where Σ is the state covariance matrix that represents uncertainty in the state estimate.

Update step: Incorporates new observations to correct the predicted state estimate using Bayes' theorem.

$$K_t = \Sigma_{t|t-1}H^\top \left(H\Sigma_{t|t-1}H^\top + R\right)^{-1}, \tag{2.8}$$

$$\hat{s}_{t|t} = \hat{s}_{t|t-1} + K_t\left(O_t - H\hat{s}_{t|t-1}\right), \tag{2.9}$$

$$\Sigma_{t|t} = (I - K_tH)\,\Sigma_{t|t-1}, \tag{2.10}$$

where K_t is the Kalman gain, determining how much the new observation influences the updated estimate.

The KF is optimal for linear systems with Gaussian noise, but in real-world applications, many systems exhibit non-linearity. This limitation led to the development of extensions such as the **Extended Kalman Filter (EKF)** [46] and the **Unscented Kalman Filter (UKF)** [47]. The EKF linearizes the non-linear system using a first-order Taylor series expansion, while the UKF uses a set of sigma points to approximate non-linear transformations more accurately [48].

The KF is extensively used in autonomous navigation, target tracking, and sensor fusion applications. For instance, in AVs, KF-based filtering techniques are employed to estimate the ego-vehicle's position and velocity using Global Positioning System (GPS) and Inertial Measurement Unit (IMU) data. Additionally, in robotic perception, KFs assist in object tracking by refining raw sensor measurements.
Although the KF is effective for linear systems with Gaussian noise, its applicability is restricted in environments that involve non-linear dynamics or non-Gaussian uncertainties, which are prevalent in real-world autonomous navigation tasks. Its reliance on accurate model parameters and noise covariance matrices can lead to poor performance if the system is not well modeled. Moreover, it cannot effectively capture multi-modal distributions, making it less suitable for complex scenarios with ambiguous observations or multiple hypotheses.

2.6 Particle Filter

The PF is a powerful non-parametric method for probabilistic inference that employs a set of weighted particles to approximate the posterior distribution of states. Unlike the KF, which assumes Gaussian noise and linear system dynamics, the PF can effectively model non-linear and non-Gaussian processes, making it highly suitable for real-world dynamic systems [49].

The PF is based on the principle of Sequential Monte Carlo (SMC) methods, where a set of discrete samples (particles) is used to represent the probability distribution of an underlying stochastic process. Each particle carries a state hypothesis, and its corresponding weight indicates its relative likelihood given the observed data. By maintaining a diverse set of particles, the PF provides robust state estimation in systems characterized by multi-modal distributions, occlusions, and abrupt environmental changes.

Mathematically, the PF represents the posterior distribution $P(s_t \mid O_{1:t})$ using a set of N particles $\{s_t^{(i)}\}_{i=1}^{N}$ and their associated weights $\{w_t^{(i)}\}_{i=1}^{N}$. The filtering process consists of four key steps:

1. **Initialization:** The particles are drawn from the prior distribution, and their weights are initialized uniformly:

$$w_0^{(i)} = \frac{1}{N}. \tag{2.11}$$

2. **Prediction:** Each particle is propagated forward according to the system's transition model:

$$s_t^{(i)} \sim P(s_t \mid s_{t-1}^{(i)}). \tag{2.12}$$

   This step captures the system's evolution over time, allowing the particles to represent possible future states.

3. **Weighting:** The weights of the particles are updated based on the likelihood of the new observation O_t:

$$w_t^{(i)} \propto w_{t-1}^{(i)} P(O_t \mid s_t^{(i)}). \tag{2.13}$$

   Higher weights are assigned to particles that better explain the observations, reinforcing their significance in the estimation process.

4. **Resampling:** To mitigate particle degeneracy (where a few particles dominate while others vanish), a resampling step is performed. Particles with low weights are discarded, and those with higher weights are replicated to maintain a representative distribution. The effective sample size (N_{eff}) is used to determine when resampling is required:

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (w_t^{(i)})^2}. \tag{2.14}$$

In contrast to KF-based methods, which approximate state distributions as Gaussian, the PF can capture arbitrary and multi-modal distributions. This makes it particularly advantageous in scenarios where the system dynamics involve sudden changes, occlusions, or non-Gaussian noise. The PF has been successfully applied in various domains, including target tracking, robotic localization, and simultaneous localization and mapping (SLAM) [50].

For instance, in autonomous vehicles, the PF is used to track dynamic objects such as pedestrians and other vehicles, where motion patterns can be highly unpredictable. Similarly, in mobile robotics, PF-based localization techniques enable robust estimation of a robot's position within an environment by integrating sensor inputs from LiDAR, cameras, and inertial measurement units (IMUs).

The PF, although powerful in handling non-linear and non-Gaussian systems, suffers from a few operational drawbacks. It can be computationally expensive, especially when a large number of particles are required for accurate state approximation. The resampling step may cause particle impoverishment, where diversity is lost due to repeated

selection of high-weight particles. In low-observation or highly noisy environments, the filter may struggle to maintain accurate estimates unless carefully tuned.

2.7 Markov Jump Particle Filter

MJPF is an extension of the PF which enables discrete mode switching. It is well suited to systems with hybrid dynamics, where state evolution is influenced by both continuous and discrete variables [49]. In the MJPF framework, the state is represented by a combination of a continuous variable s_t and a discrete mode M_t. The mode M_t determines which dynamics govern the system at a given time. The state transition equation is:

$$s_t = A(M_t)s_{t-1} + \eta_t, \tag{2.15}$$

where $A(M_t)$ is a mode-dependent transition matrix and η_t represents process noise.

The MJPF operates through two hierarchical levels: 1. **Continuous state filtering:** Similar to the Particle Filter, continuous states are updated using particles. 2. **Mode inference:** The mode transition probabilities $P(M_t \mid M_{t-1})$ are used to infer the most likely mode sequence.

The filtering process involves: *Prediction:*

$$P(s_t, M_t \mid O_{1:t-1}) = \sum_{M_{t-1}} \int P(s_t \mid s_{t-1}, M_t) P(M_t \mid M_{t-1}) P(s_{t-1}, M_{t-1} \mid O_{1:t-1}) \, \mathrm{d}s_{t-1}. \tag{2.16}$$

Update:

$$P(s_t, M_t \mid O_{1:t}) \propto P(O_t \mid s_t) P(s_t, M_t \mid O_{1:t-1}). \tag{2.17}$$

MJPFs are particularly useful for tracking systems with switching dynamics, such as AV navigation with a variety of modes of operation. As a result, it provides robust state estimation while adapting to discrete mode transitions.

The MJPF framework employs anomaly detection methods at both continuous and discrete levels to evaluate deviations between predicted states and real-world observations. These anomalies—termed Continuous Level Anomaly (CLA), Continuous Level Behavior Anomaly (CLB), and Kullback-Leibler Divergence Anomaly (KLDA)—provide insights into inconsistencies at different hierarchical levels. Below, we detail their formulations and significance.

CLA capture discrepancies at the continuous state level by comparing predicted states $\Phi(\mathbf{X}_{k+1})$ with observed states $\Psi(\mathbf{X}_{k+1})$. This metric evaluates how well the predicted positions align with observations, thus identifying abnormal patterns in the environment. The anomaly is quantified using a probabilistic similarity metric, namely the Bhattacharyya distance, which measures the divergence between two multivariate Gaussian distributions. In this context, it compares the predicted and observed distributions of the ego vehicle state at time $k+1$. The Continuous Level Anomaly (CLA) is formulated

as:

$$\text{CLA} = \Delta_B \left(\Phi(\mathbf{X}_{k+1}), \Psi(\mathbf{X}_{k+1}) \right), \tag{2.18}$$

where $\Phi(\mathbf{X}_{k+1})$ and $\Psi(\mathbf{X}_{k+1})$ denote the predicted and observed state distributions, respectively. Each distribution is assumed to follow a multivariate Gaussian with mean vectors $\mathbf{M}_P, \mathbf{M}_Q$ and covariance matrices $\mathbf{\Sigma}_P, \mathbf{\Sigma}_Q$. The Bhattacharyya distance $\Delta_B(P,Q)$ between the two distributions P and Q is given by:

$$\Delta_B(P,Q) = \frac{1}{8}(\mathbf{M}_P - \mathbf{M}_Q)^\top \mathbf{\Sigma}^{-1}(\mathbf{M}_P - \mathbf{M}_Q) + \frac{1}{2}\ln\left(\frac{\det(\mathbf{\Sigma})}{\sqrt{\det(\mathbf{\Sigma}_P)\det(\mathbf{\Sigma}_Q)}}\right), \tag{2.19}$$

where the intermediate covariance matrix $\mathbf{\Sigma}$ is defined as:

$$\mathbf{\Sigma} = \frac{1}{2}(\mathbf{\Sigma}_P + \mathbf{\Sigma}_Q). \tag{2.20}$$

This formulation captures both the displacement between the means and the similarity in spread (covariance) of the distributions. A larger CLA value indicates a higher divergence between the predicted and observed states, thus signaling a potential anomaly in system behavior.

CLB focuses on the alignment between continuous predictions and discrete superstate-conditioned probabilities. It evaluates the discrepancy between the predicted message $\Phi(\mathbf{X}_{k+1})$ and the transition-informed distribution $\Pi(\mathbf{X}_{k+1} \mid \mathbb{S}_{k+1})$, highlighting any inconsistency within a given superstate.

The CLB metric is expressed as:

$$\text{CLB} = \Delta_B \left(\Phi(\mathbf{X}_{k+1}), \Pi(\mathbf{X}_{k+1} \mid \mathbb{S}_{k+1}) \right), \tag{2.21}$$

where Δ_B is the Bhattacharyya distance. High CLB values typically indicate a conflict between predictions at the continuous level and the superstate probabilities derived from discrete observations [37].

KLDA anomalies are used to assess deviations at the discrete superstate level by comparing the predicted distribution $\Phi(\mathbb{S}_{k+1})$ with the observed distribution $\Psi(\mathbb{S}_{k+1})$. This approach identifies abrupt changes or inconsistencies in cluster assignments or state transitions.

The KLDA is formulated as a symmetrical Kullback-Leibler divergence:

$$\text{KLDA} = \Delta_{KL} \left(\Phi(\mathbb{S}_{k+1}) \| \Psi(\mathbb{S}_{k+1}) \right) + \Delta_{KL} \left(\Psi(\mathbb{S}_{k+1}) \| \Phi(\mathbb{S}_{k+1}) \right), \tag{2.22}$$

where the Kullback-Leibler divergence $\Delta_{KL}(P\|Q)$ is defined as:

$$\Delta_{KL}(P\|Q) = \sum_i P(i) \ln\left(\frac{P(i)}{Q(i)}\right). \tag{2.23}$$

The two terms account for the directional divergences between predicted and observed distributions, providing a robust measure of anomaly at the discrete level. Large KLDA values often indicate mismatches caused by sudden environmental changes or sensor inconsistencies [37].

2.8 Summary

The theoretical foundations outlined in this chapter form the core pillars of the methodologies employed throughout this thesis. However, to effectively position this work within the broader research landscape, it is essential to understand how these concepts have been applied and evolved in recent literature. The following chapter presents a comprehensive review of state-of-the-art techniques in LiDAR-based perception, focusing on clustering, classification, multi-target tracking, and localization. This survey not only contextualizes the chosen methods but also highlights existing gaps that the proposed framework seeks to address.

Chapter 3

State Of the Art

An overview of the state-of-the-art research relevant to our work is presented in this chapter, which includes research in multiple areas. Due to the fact that our work encompasses multiple aspects, we have divided this section into four sections. Each section is designed to reflect a key functional block of the proposed framework, and the literature review provides a foundation for how our method builds upon or diverges from prior work. These components align with the modular design of the proposed framework: clustering and classification form the basis of static/dynamic distinction; multi-target tracking supports interaction modeling; and landmark-based localization methods are contrasted against our DBN+MJPF-based approach. By identifying gaps in existing works, this chapter motivates the need for an integrated, data-driven localization system introduced in subsequent chapters. A different aspect of our study is addressed in each section. This approach allows an in-depth review of the existing literature that pays special attention to studies that apply LiDAR data for multi-target tracking, clustering, classification and ego vehicle localization to autonomous driving. Moreover, a summarized table is shown at the end of this chapter to let readers quickly understand the state of the art relevant to four different areas of research.

3.1 Clustering in LiDAR Data

Several studies have made significant advancements in the field of point cloud clustering for autonomous driving applications which are discussed in this section. The work [51] aimed to automatically estimate the parameters of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) by leveraging the structure of the point cloud. This technique implements a field of view division and empirical relations that allow each point to be independently estimated. A Deep Neural Network (DNN) is used before beginning parameter estimation to remove the background points. Based on a decoder/encoder structure, the network learns features that allow it to differentiate between, for example, foreground and background points. A feature vector is derived

from the resulting points and is employed to classify them as foreground or background. With the background filtration process, the point cloud size can be reduced and clustering can be performed more quickly. This scheme involves dividing the field of view into equal-sized regions and further dividing each region into cells of equal size. By dividing the LiDAR point cloud into cells, local information about the point cloud can be calculated, such as the density of points within the cells.

The study in [52] was carried out in two parts. First, the points were projected onto a grid map, ground points were removed by determining the maximum height difference, and roadsides were detected with the Hough transform to determine the dynamic region of interest (ROI). In the second part, a DBSCAN-based adaptive clustering method (DAC) has been proposed to reduce the risk of over-segmentation and under-segmentation due to the variations in spatial density of point clouds in relation to their positions. The elliptic neighbourhood is designed to match the distribution properties of the point cloud to avoid the possibility of over-segmentation and under-segmentation. To handle the uniformity of points in different ranges, the parameters of the ellipse are adaptively adjusted with respect to the location of the sample point. A dynamic clustering algorithm is proposed in [53] to address the non-uniform spatial distribution of LiDAR point clouds. An elliptical neighbourhood function is introduced in the method, which enhances the robustness of obstacle detection by adapting to the position of each core point. The KITTI dataset is employed to optimise parameters, and the experimental results demonstrate improved clustering efficacy in comparison to conventional methods. In paper [54], "InsClustering" was presented—a fast and accurate method of clustering point clouds for AVs using LiDAR. Consequently, it provides an efficient means of segmenting the ground and clustering the objects within the limited amount of time available. With the use of Velodyne UltraPuck LiDAR range images in spherical coordinates, the method is capable of maintaining clustering accuracy and minimising over-segmentation due to a coarse-to-fine segmentation process.

It has become increasingly important in recent years to compress point cloud data, especially in the context of AVs, where accurate and efficient LiDAR data processing is crucial. Due to their limitations in encoding floating-point numbers and handling distance information inherent in LiDAR data, traditional image and video compression algorithms, such as JPEG2000, JPEG-LS, and High Efficiency Video Coding (HEVC), do not suit point cloud data compression. The proposed method [55] addressed these challenges by employing a lossless compression scheme based on point cloud clustering as well as exploring lossy compression techniques. Rather than relying on traditional image prediction methods, this approach involves a novel prediction method based on correlations between distance information among points. As a result, spatial redundancies can be eliminated without compromising the integrity of the dataset.

A two-stage clustering method for LiDAR data is presented in [56]. First, ground line fitting reduces the data load by extracting the ground plane from the data. In the

case of the non-ground data, a range image-based method is used, in which sub-clusters are initially created through a sliding window approach, and then refined through an adaptive DBSCAN algorithm. It effectively reduces over-segmentation and adapts to variances in object distances. A method of clustering point clouds based on Two-Layer-Graphs (TLG) [57] was proposed to improve the accuracy and speed of segmenting point clouds. This involves dividing the task into storage structures, segmentation standards, and category updates. Range graphs and point cloud set graphs were used to enable fast access to and relationships between neighboring points. Standards for segmentation include distance and angle characteristics in the horizontal direction and distance in the vertical direction. The category update was accomplished through the use of a search algorithm traversing the two layers of the graph. The results of the experiments demonstrated that clustering and differentiation of objects in traffic scenes can be achieved effectively.

3.2 Multi-Target Tracking in LiDAR Data

It is important to track multiple targets in autonomous vehicles and advanced driver assistance systems (ADAS). In addition to path planning, collision avoidance, and precise pose estimation, tracked trajectories are useful for path planning. Object detection and data association are the two major stages of most MTT approaches. Given the critical role of MTT in ensuring reliable localization and navigation, it is essential to discuss the state-of-the-art methods and their advancements in this domain. Therefore, in this section, we present a detailed discussion of the relevant state-of-the-art approaches.

A new neural network architecture is proposed for detecting and tracking objects in 3D point clouds in [58]. The PointTrackNet network converts a sequence of point clouds into results for object detection and tracking by using an end-to-end approach. There are three main components of the network: a point cloud encoder, a module for detecting objects, and a module for tracking. The encoder reduces the point cloud input into a compact representation of features, which is further processed by the detection module to detect objects. As a result, a consistent tracking result is produced by associating the detected objects across frames.

An approach to multi-object detection and tracking in complex urban environments is presented in the [59]. As part of the proposed system, uncertainty and challenges such as occlusions, cluttered scenes, and dynamic changes in the environment are taken into consideration. In this system, three main components are used: object detection, object association, and object tracking. A 3D point cloud is provided for input to the detection part, which is then subdivided into measurements taken on the ground and those taken from an elevated location. The ground is removed using a slope-based approach and filtered thereafter. Additionally, object hypotheses are generated in a

clustering step for the tracking targets. Following a feature-based bounding box fitting and rule-based filtering, the objects of interest are extracted. Using centroid tracking, four main steps are involved in tracking: data association, tracking filters, tracking management, and bounding box correction. An object hypothesis is determined in the association phase based on the already established tracks that correspond to the predicted measurements. The track is updated with an associated measurement if there is a possibility of an association, otherwise a new track is created. Tracking filters are used to perform the prediction and update steps. Track management is responsible for maintaining all tracks, labeling their maturity, and eliminating the infeasible and old ones. As a final step, bounding-box correction assigns valid bounding-box dimensions to mature tracks and updates this information using track history.

In [60], an Adaptive Cubature Kalman Filter (ACKF) is employed to estimate the state of multiple objects in real time, specifically for use in AVs. A 3D detection and tracking network is used to detect and track the objects, and then the ACKF algorithm is used to estimate their position, velocity, and other parameters. In the ACKF algorithm, the state of the objects is calculated using cubature integration, which improves the tracking accuracy. Also included in the algorithm is an adaptive process that updates the parameters of the model in response to the estimated errors. Through this process, the object is able to adapt to changes in its motion and environment over time. SimTrack [61] is a simplified model for 3D multi-object tracking in point clouds which is essential for AVs. The SimTrack system is built on pillar- or voxel-based 3D object detection networks, and its aim is to eliminate heuristic matching steps and manual track life management in tracking-by-detection systems. A hybrid-time centerness map and a motion updating branch are the key components of SimTrack. By using a hybrid-time centerness map, objects are represented based on their first-appearing locations within a given input period, which permits the direct linking of current detections with previously tracked objects without the need for additional matching. Tracked objects are updated with their locations based on the estimated motion of the tracked objects. Using SimTrack, tracking objects are linked, new-born objects are detected, and dead objects are removed in one end-to-end trainable model.

In paper [62], ComplexerYOLO was presented, a real-time system for detecting and tracking 3D objects, using semantic point clouds derived from LiDAR images. It incorporates visual class features from camera-based semantic segmentation, extends Complex-YOLO to process voxelized input features, and predicts 3D box heights and z-offsets. Scale-Rotation-Translation score (SRTs) is a new validation metric introduced by the authors, which is faster than Intersection over Union (IoU) and also takes into account the object's 3DoF pose. Multi-Target Tracking is achieved using an online feature tracker that is separate from the detection network. It achieves state-of-the-art results in the areas of semantic segmentation, 3D object detection, and Multi-Target Tracking, which makes it suitable for the perception of urban self-driving cars. Three-

dimensional MTT, a technology that is crucial for extracting dynamic information from road environments, is discussed in [63], along with its applications in intelligent transportation systems, including autonomous driving and traffic monitoring. There are several challenges associated with current methods for detecting heavily occluded or distant objects, as well as formulating effective pairwise costs for data association. To overcome these challenges, the authors proposed a new 3D tracker based on a data association scheme guided by a prediction confidence scheme. The tracking-by-detection framework of this tracker consists of four steps: detecting objects from point clouds using a deep learning-based 3D object detection algorithm, estimating possible current states of tracked objects based on constant acceleration (CA) motion models, and a prediction confidence model. A correlation between predicted and detected states is formed using prediction confidence and an aggregated pairwise cost, and then the matched pairs are updated, and the unmatched detected states are marked as tracked. It is aimed at improving both the accuracy and speed of tracking objects, as well as using features of objects in point clouds and tracking objects that have been temporarily missed.

3.3 Classification of LiDAR Data

Recent developments in the classification of LiDAR data for autonomous driving have focused on enhancing object detection and semantic segmentation through the use of deep learning and sensor fusion techniques. Aksoy et.al.[64] presented SalsaNet, a deep learning model developed for the efficient semantic segmentation of 3D LiDAR point clouds for autonomous driving. A bird-eye-view (BEV) projection is used to segment roads and vehicles using an encoder-decoder architecture. Moreover, they developed an automatic process for labeling LiDAR data by transfering labels from camera images to LiDAR point clouds. A multimodal 3D object detection framework based on LiDAR and camera data was developed by Zhang et.al.[65] and utilized a feature fusion network to improve detection accuracy in complex driving environments.

An approach to real-time object detection for autonomous vehicles was proposed by [66] based on OD-C3DL, a framework that integrates 3D LiDAR and camera data. The LiDAR provides geometric and depth information, while the camera provides RGB images that are used to classify objects accurately. As part of their approach, they used Point Cloud Augmentation (PCA) to fuse sensor data, Object Region Identification (ORI) for clustering, and a CNN-based architecture using VGG16 with multi-layer ROI pooling for feature extraction and classification. Based on the KITTI dataset, OD-C3DL has achieved superior accuracy (79.13% for cars, 88.76% for pedestrians) with a low processing time of 65 milliseconds, outperforming other methods such as YOLOv5 and CT-Net in terms of accuracy and efficiency.

A real-time object classification method based on 3D LiDAR point clouds was proposed by [67]. It uses the Real AdaBoost algorithm to analyze the size, shape, and velocity of objects and classify them into cars, pedestrians, bicyclists, and background objects. This method, which achieves over 90% accuracy within a 50-meter range and processes objects in 0.07 milliseconds, has been proven to be effective for real-time applications. Further confidence is enhanced by the application of Binary Bayes Filters to the time-series, although challenges persist in detecting long-range objects and partially obscured objects. A back-propagating neural network (BPNN) system was proposed by [68] to classify 3D objects in urban environments using LiDAR point clouds. A histogram-based threshold is used to filter ground points and rasterize the x-z plane to segment remaining points into object clusters. The extracted features, such as volume, density, and eigenvalues, are then used to train a BPNN with 5 input neurons, 20 hidden neurons, and 5 output neurons to categorize objects into walls, poles, pedestrians, trees, and bushes. BPNN achieved 91.5% accuracy, outperforming Decision Trees (89%) and Support Vector Machines (67.5%).This method improved AV's perception and decision making in urban environments.

Based on the fusion of RGB vision and LiDAR data, [69] proposed a deep-learning-based object classification method for AVs. By converting sparse LiDAR point clouds into high-resolution depth maps and integrating them with RGB data, a convolutional neural network (CNN) is trained. A test on KITTI datasets showed that this method was capable of classifying objects (cars, pedestrians, cyclists, trucks) with a 96% accuracy, outperforming RGB-only models. As a result of the fusion of modalities, features are better represented, classification precision is enhanced, and loss is reduced.

An evidential deep learning method for the classification of LIDAR objects in autonomous driving has been proposed by [70], taking into account ambiguity and unknown objects. A multi-task MLP and Dempster-Shafer theory are used to classify objects into vehicles, vulnerable road users, and unknowns, and uncertainty is managed by using a Z-score filter. Based on Velodyne VLP-32C LIDAR data, it achieved high accuracy (89.7%) and improved unknown object detection over a one-class SVM. The lightweight, real-time system paves the way for a more reliable and interpretable AV perception system. The authors of [71] developed a LIDAR-based DATMO (Detection and Tracking of Moving Objects) algorithm that classified objects into three categories: bikes, cars, and trucks based on the Belief Theory. Based on the algorithm, 2D LIDAR point clouds were processed in five stages: clustering, segmentation as recursive best segement split (RBSS), data association, tracking with Kalman filters, and classification. There were several key innovations, including: Segmentation of objects using a RBSS method reduced noise sensitivity, based on Dempster-Shafer theory, pignistic probabilities were generated based on physical attributes such as width, length, speed and lastly incorporating vehicle type-specific motion models to improve precision in

tracking. The algorithm was demonstrated to have a detection rate of 96% and a misclassification rate of 4% in experiments on urban road scenarios.

Fang et al. [72] introduced, a deep learning framework for identifying roadside objects such as cars, trees, traffic poles, and small objects (pedestrians, bicycles, e-bikes) based on point clouds and multiview images. For the extraction of local geometric features, graph attention convolution was used, while a hierarchical view-group-object architecture was employed for the extraction of global geometric features. In a novel attention-guided fusion module, geometric and multiview features are combined to achieve robust classification in a variety of datasets. This framework outperformed existing methods with an overall accuracy of 95.76% and demonstrated strong generalization when dealing with incomplete and noisy point clouds.

3.4 Localization in LiDAR Data

In recent years, advances in probabilistic models, sensor fusion techniques, and map-matching algorithms have significantly enhanced localization capabilities for AVs. A probabilistic framework for ego-vehicle localization was proposed by [73] by integrating road-level, lane-level, and ego-lane-level localizations. As part of the framework, BNs and Hidden Markov Models (HMMs) are used to fuse GPS, IMU, camera, and LiDAR data, while incorporating prior geometric information from OpenStreetMap (OSM). In this method, computational complexity is reduced and robustness against sensor errors is enhanced. Several key contributions have been made, including a probabilistic map-matching module for road-level localization, a recursive detection for ego-lane marking using top-down information, and lane-level localization by combining multi-sensors and applying BN-HMM filters. Based on KITTI and real-world highway datasets, this method has demonstrated robust localization in challenging conditions such as poor lighting and occlusion.

An algorithm integrating ego-vehicle positioning, heading, and velocity with obstacle detection and tracking using a multisensor framework of radars and LiDARs is presented in the work of [74] for real-time AV state estimation. Using curvilinear coordinates aligned with the road geometry, the system utilizes a Unscented Kalman Filter (UKF) for accurate estimation of ego-vehicle state based on GPS, inertial sensors, and odometry, while a fusion of radar and LiDAR data allows precise tracking of obstacles. Data from LiDAR is converted to a two-dimensional occupancy grid for clustering, while velocity measurements are provided by Radar, with sensor fusion prioritizing data from LiDAR for the purpose of determining position accuracy. A robust data association algorithm is employed via gated and Mahalanobis distances, and tracking continuity is ensured through initialization and removal routines. It has been validated on a prototype at the Monza ENI Circuit, demonstrating low variation in mean square error

and reliable performance in dynamic scenarios. It provides critical state estimates for motion planning.

Persistent Homology (PH) [75] is applied to LiDAR-based ego vehicle localization, with particular attention to loop closure detection, place categorization, and global end-to-end positioning. A persistence diagram (PD), which represents the global structure of a point cloud, is derived from PH, a topological data analysis method that extracts translation and rotation invariant features. As a result, persistence images (PIs) are created, which provide machine learning models with fixed-size, low-dimensional representations. Although PH successfully captures global topological features, the results indicate that it is limited in maintaining local shape details that are essential for registering point clouds accurately, thus limiting its ability to enhance localization accuracy. PH, however, demonstrates the potential for supplementary tasks, such as detecting failures, categorizing places, and understanding global structure relationships, as demonstrated in experiments using the KITTI odometry dataset. According to the study, PIs enhance position estimation slightly in end-to-end localization models, but do not significantly improve orientation prediction. While PH alone cannot drastically increase localization accuracy, it can be used to supplement traditional geometric and neural network-based approaches to address broader localization challenges.

The EgoVM [76] is an end-to-end ego-localization network that achieves centimeter-level accuracy using vectorized maps that reduce map size significantly in comparison to point-based maps. A Bird's Eye View (BEV) feature extraction method is used to extract features from multi-view images and LiDAR, and a transformer decoder with learnable semantic embeddings for cross-modality matching is used along with a histogram-based pose estimation method. A key feature of EgoVM is its integration of semantic supervision to enhance feature consistency and its incorporation of surfel features to improve localization in sparse landmark scenarios. A Real-Time Monte Carlo Localization (RTMCL) method for AVs is presented in [20] which aims to balance the accuracy of pose estimation with the performance of real-time localization. With this algorithm, LiDAR and radar data are integrated using an UKF for improved detection of pole-like landmarks. These landmarks are then clustered using GB-DBSCAN and refined using RANSAC. PF with a probabilistic reference map of landmarks is used for precise localization of the ego-vehicle. A GPS and an IMU data fusion algorithm is used to initialize the system, and the ICP algorithm is used to link detected landmarks with mapped landmarks. This model has been implemented in C++ using optimized libraries, achieving an 11 cm localization error in real-time using only 50 particles, thus demonstrating robustness under varying map uncertainties and computational efficiency suitable for real-world use.

A robust multi-vehicle cooperative localization framework is presented in [77] based on the Iterated Split Covariance Intersection Filter (ISCIF). As a result of this method, several challenges can be addressed in outdoor environments where traditional methods

have been unable to effectively handle simultaneous data correlation and outliers. Using shared point cloud maps and vehicular communication, ISCIF integrates LiDAR-based SLAM for accurate ego-vehicle localization and relative pose estimation for inter-vehicle communication. Experiments in urban and highway simulations demonstrate that ISCIF is more accurate, robust, and efficient than existing methods, particularly in the presence of non-Gaussian noise, process, and observation outliers. ISCIF has the potential to advance cooperative perception and localization in AVs, with additional applications in augmented reality and complex outdoor environments. An ego vehicle and nearby moving obstacles are simultaneously estimated by a LiDAR-based method in [78]. With PointNet++ as the backbone for feature extraction, the proposed Pose Estimation Network (PEN) segments point clouds into static and moving parts and estimates the ego vehicle's pose based on static points as well as obstacle poses based on moving points. Lyapunov theory ensures stability by using a reduced-order observer to estimate velocities. Using this approach, the interpretability of observers is combined with the feature extraction capabilities of deep learning, which results in robust and smooth estimations. A comparison of the accuracy and stability of the proposed method with existing techniques, even in complex traffic scenarios, is demonstrated by simulation and experimental results.

The following table offers an exhaustive comparative summary of the findings that were discussed in the four subsections—clustering, MTT, classification, and localisation. Providing a concise reference to the diverse state-of-the-art approaches in LiDAR-based autonomous driving systems, this tabular overview emphasises the main methodologies, innovations, and performance aspects of the cited works. The categorisation helps the reader understand the evolution and focal areas of each domain, and it also serves as a bridge to the methodological advancements that are proposed in subsequent chapters.

State-of-the-Art: Comprehensive Table

Category	**Reference**	**Methods**
Clustering	[51]	Uses a DNN to remove background points and adapt DBSCAN parameters by exploiting local point densities in each cell. Divides the field of view into subregions, improving speed and avoiding over-segmentation.

(Continued on next page)

Category	Reference	Methods
Clustering	[52]	Two-step approach: (1) project point cloud onto grid map, remove ground via height difference, detect roadsides (Hough); (2) apply DBSCAN-based adaptive clustering (DAC) with elliptical neighborhoods to reduce over-/under-segmentation.
Clustering	[53]	Improves obstacle detection accuracy by proposing a dynamic clustering method that employs an adaptive elliptical neighbourhood to manage non-uniform LiDAR point distributions. .
Clustering	[54]	"InsClustering," a fast and accurate method for segmenting ground and clustering objects in LiDAR data using range images in spherical coordinates. Coarse-to-fine segmentation to minimize over-segmentation.
Clustering	[55]	Proposes a lossless + lossy compression scheme based on point cloud clustering, leveraging correlations between distance information to remove spatial redundancies.
Clustering	[56]	Two-stage clustering: first removes ground via line fitting, then uses a range image-based approach with a sliding window to form subclusters, refined via adaptive DBSCAN. Addresses over-segmentation by adapting to object distance.
Clustering	[57]	Employs Two-Layer-Graphs (TLG) to cluster point clouds. Uses a range graph + point cloud set graph, with distance and angle constraints in horizontal, and distance in vertical. Category updates via a search across the two-layer graph.
Multi-Target Tracking	[58]	"PointTrackNet": end-to-end 3D point cloud detection + tracking. Has a point cloud encoder, detection module, and tracking module to associate objects across consecutive frames.

(Continued on next page)

Category	**Reference**	**Methods**
Multi-Target Tracking	[59]	Addresses multi-object detection/tracking in cluttered, occluded urban scenes. Uses slope-based ground removal, clustering for object hypotheses, bounding box fitting, and centroid tracking with bounding box refinement.
Multi-Target Tracking	[60]	Integrates a 3D detection and tracking network with an Adaptive Cubature Kalman Filter (ACKF) to estimate object states in real time. Adaptive model parameters improve accuracy with changing motion/environments.
Multi-Target Tracking	[61]	"SimTrack": a simplified model for 3D MOT in point clouds. Eliminates heuristic matching steps via a hybrid-time centerness map and motion-updating branch; end-to-end approach for linking and managing tracks.
Multi-Target Tracking	[62]	"ComplexerYOLO": real-time 3D object detection/tracking from semantic LiDAR point clouds. Merges camera-based semantic segmentation for class labels; introduces Scale-Rotation-Translation score (SRTs) for track validation.
Multi-Target Tracking	[63]	Proposes a 3D tracking-by-detection method with a prediction confidence model. Relies on aggregated pairwise cost for data association; unmatched detections become new tracks, improving occlusion handling.
Classification	[64]	"SalsaNet": semantic segmentation of LiDAR point clouds in a bird's-eye view. An encoder-decoder network; uses an automated process transferring camera labels to LiDAR points.
Classification	[65]	A multimodal 3D object detection framework fusing LiDAR and camera data. Uses a feature fusion network to improve detection accuracy in complex driving environments.

(Continued on next page)

Category	Reference	Methods
Classification	[66]	"OD-C3DL": real-time object detection fusing 3D LiDAR + camera. Employs Point Cloud Augmentation, clustering, and a VGG16-based CNN with multi-layer ROI pooling. Achieves high accuracy in 65 ms.
Classification	[67]	Real AdaBoost algorithm on LiDAR data for car, pedestrian, bicyclist classification. Processes size, shape, velocity features. Over 90% accuracy within 50m, 0.07 ms processing per object; uses Binary Bayes Filters for confidence.
Classification	[68]	Back-Prop Neural Network (BPNN) for urban 3D objects. Removes ground points, clusters, extracts volume/density/eigenvalue features. BPNN attains 91.5% accuracy, surpassing Decision Trees and SVM.
Classification	[69]	Deep-learning-based object detection fusing RGB vision + LiDAR. Converts sparse LiDAR points into depth maps combined with RGB images in a CNN. Achieves 96% accuracy on KITTI.
Classification	[70]	Evidential deep learning for LiDAR object classification using Dempster-Shafer to handle unknown classes. Multi-task MLP, 89.7% accuracy, improved detection of unseen/ambiguous objects.
Classification	[71]	LIDAR-based DATMO with Belief Theory (Dempster-Shafer). Clustering + segmentation via RBSS, Kalman filters for tracking, classifies bikes/cars/trucks with 96% detection and 4% misclassification.
Classification	[72]	Graph attention convolution + hierarchical view-group-object architecture for roadside object classification (cars, trees, poles, small objects). Achieves 95.76% accuracy, robust to incomplete/noisy data.

(Continued on next page)

Category	**Reference**	**Methods**
Localization	[73]	Probabilistic (Bayesian Nets + HMM) fusion of LiDAR, GPS, IMU, camera, and OSM geometry for road-, lane-, and ego-lane-level localization. Robust under poor lighting/occlusion.
Localization	[74]	Integrated multi-sensor framework (GPS, IMU, radar, LiDAR) in curvilinear coords. UKF for ego-vehicle pose; radar+LiDAR data association with gating and Mahalanobis distance. Tested on Monza race circuit.
Localization	[75]	Leverages Persistent Homology (PH) for loop-closure detection and place categorization. Converts topological features into persistence images, improving global structure awareness slightly for pose estimation.
Localization	[76]	"EgoVM": end-to-end ego-localization with vectorized maps. Uses BEV feature extraction from images + LiDAR, a transformer decoder with semantic embeddings, and histogram-based pose estimation for cm-level accuracy.
Localization	[20]	"RTMCL": real-time Monte Carlo Localization using LiDAR+radar, merges pole-like landmark detection (GB-DBSCAN + RANSAC) with a Particle Filter. Achieves 11 cm error with only 50 particles.
Localization	[77]	Cooperative localization with Iterated Split Covariance Intersection Filter (ISCIF). Combines LiDAR-based SLAM, shared point-cloud maps, and inter-vehicle communication. Robust to outliers and non-Gaussian noise.
Localization	[78]	Simultaneous estimation of ego-vehicle + moving obstacles in LiDAR. Uses PointNet++ for feature extraction, splits clouds into static/moving parts, and a reduced-order observer for velocities. Stable, robust in complex traffic.

Among the studies summarized in the above table, a few are particularly relevant to the goals of this thesis. Notably, works that rely on DBSCAN or learning-based clustering techniques [51, 52, 57] demonstrate strong segmentation capability but often lack robustness across varying LiDAR densities or environmental noise. Likewise, classical multi-target tracking approaches using KF or PF [60, 62] tend to assume stable measurement noise and suffer under dynamic object interactions. In contrast, this thesis builds on these foundations by integrating GNG for adaptive clustering and a MJPF for robust ego vehicle localization. Moreover, by incorporating interaction dictionaries and a DBN-based structure, the framework proposed here addresses critical limitations related to long-term temporal reasoning and scene variability, which are not effectively tackled in prior literature.

3.5 Summary

While Chapter 3 established a comprehensive overview of existing techniques for LiDAR-based perception and localization, some of these methods face challenges in consistently distinguishing between static and dynamic elements in complex driving environments. Addressing this gap, the next chapter introduces an offline classification framework that utilizes LiDAR and odometry data to pre-label environmental tracks. By leveraging probabilistic filters and unsupervised learning algorithms, this methodology enhances the reliability of downstream localization tasks. The integration of static and dynamic track dictionaries, introduced in Chapter 4, forms the foundational layer for the ego-vehicle localization system developed in subsequent chapters.

Chapter 4

Classification Methodology for Ego-Vehicle Localization Framework

This chapter presents the proposed classification framework for AV navigation and its experimental results. The framework leverages LiDAR and odometry data to classify static and dynamic tracks, enhancing situational awareness and decision-making. The classification process employs the JPDA filter, the NFF, and the GNG algorithm to capture the spatial and temporal dynamics of tracks, generating interaction dictionaries and vocabularies. The results validate the framework's effectiveness in classifying static and dynamic tracks, ensuring robust ego-vehicle localization in dynamic environments.

4.1 Problem statement

Safe and efficient navigation necessitates precise localization in AVs. Effectively classifying environmental objects into static (e.g., buildings, traffic signs) and dynamic categories (e.g., vehicles, pedestrians) is a substantial obstacle to obtaining precise localization. Erroneous map representations can result from misclassifications, which can have a detrimental impact on the decision-making, obstacle avoidance, and navigation processes [79].

Traditional methods frequently emphasise real-time classification during the vehicle's operation. At the same time, this can be computationally expensive and susceptible to inaccuracies as a result of environmental complexities and sensor noise. In order to confront these obstacles, our novel methodology is characterised by an offline training phase during which we pre-classify environmental data into static and dynamic objects. This pre-classified data is used as a foundational layer for subsequent localization tasks, thereby improving the accuracy and robustness of the ego vehicle's positioning.

The objective of this thesis is to enhance the system's resilience to dynamic environmental changes and alleviate the computational burden during real-time operation by incorporating this offline classification into the localization pipeline. This methodology

is consistent with the most recent developments in autonomous driving research, underscoring the significance of precise environmental perception for the reliable localization of vehicles. For example, research has shown that the accuracy of localization can be improved by utilising existing 3D maps to improve dynamic object detection. Moreover, the potential of dynamic object tracking to address the challenges posed by dynamic environments has been demonstrated through the integration of SLAM techniques [80–82].
Overall, the methodology proposed in this chapter emphasises the offline classification of environmental objects to differentiate between static and dynamic entities. This strategy is designed to improve the accuracy and reliability of AV localization, thereby facilitating safer and more efficient navigation in intricate environments.

4.2 Classification Framework as an Offline Training Phase

Figure 4.1 illustrates our proposed classification framework as a block diagram. Our dataset was collected at the University of Carlos III, Madrid, using a vehicle called "iCAB" that was equipped with a Velodyne LiDAR Puck (VLP-16) sensor [83]. A total of 393 frames of point cloud data have been used to capture the environment in detail. There are two vehicles included in this dataset: iCAB1 overtakes iCAB2 at a specific time point. Moreover, the dataset contains static objects, such as buildings, trees, and poles, which are crucial to vehicle navigation and localization. As part of the preprocessing step, it is important to remove ground points and other irrelevant data from the raw point cloud data. This will ensure that subsequent analysis is more accurate. This process also involves filtering out noise and identifying significant data features [84].

4.2.1 Multi-Target Tracking

As part of the multi-target tracking module, a finely tuned detector model is employed with precise axes limits on the X, Y, and Z axes. This ensures that the bounding box is placed accurately. The detection of objects is enhanced by optimizing several parameters, including minimum segmentation distances and minimum detections per cluster. In this module, the thresholds for tracking assignment, confirmation, and deletion have been defined based on the JPDA tracker [85]. Object positions and timings are stored in a data structure, making it possible to analyze the trajectory of each object in detail. JPDA was applied to LiDAR data in order to verify the tracker's performance using ground truth images. The development of MTT techniques has been significantly shaped by foundational work from Bar-Shalom and Blackman. In particular, the PDAF and its extension to the JPDA filter were first introduced and formalized in

Bar-Shalom's seminal work [86]. These methods laid the groundwork for managing uncertainty in data association across multiple targets. Blackman's contributions [87] offered practical perspectives on radar-based tracking systems, including gating strategies, false alarm handling, and track initiation logic. Integrating these principles ensures that our tracking model maintains both theoretical robustness and practical scalability. Seven distinct tracks have been saved, each with its own TrackID and x and y coordinates. The simulator displayed bounding boxes on these tracked objects during the movement of the ego vehicle. In total, seven tracks have been selected: TrackID 14, which represents a car in front of the ego vehicle; TrackID 20, which represents a building on the right side; TrackID 1416, a static pole; TrackID 1682, a tree; TrackID 1929, a small static pole; TrackID 3159 and TrackID 3549, both representing trees. Data from odometry was used to determine the location of the ego vehicle, allowing us to analyze the trajectory in absolute coordinates. In Figure 4.2(a), the ego vehicle's trajectory is illustrated in absolute coordinates., and Figure 4.2(b) shows the position of the seven tracks with their Track IDs relative to the ego vehicle.

4.2.2 Odometry Sensor

During the offline training phase, the odometry sensor plays a critical role in determining the position of the ego vehicle, which is necessary for localizing the ego vehicle during testing. Ego vehicle localization is enhanced by including the (DBNs) derived from the training phase as an additional switching variable. A critical point is that during online testing phase, the ego vehicle localization is performed only using LiDAR data, leveraging the DBNs that have been trained with odometry sensor data.

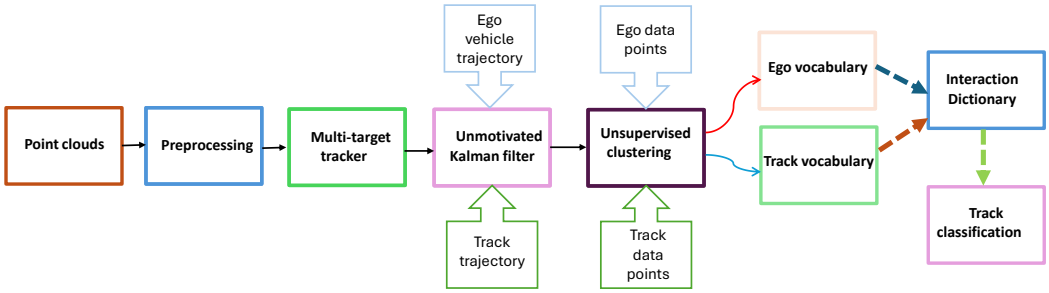
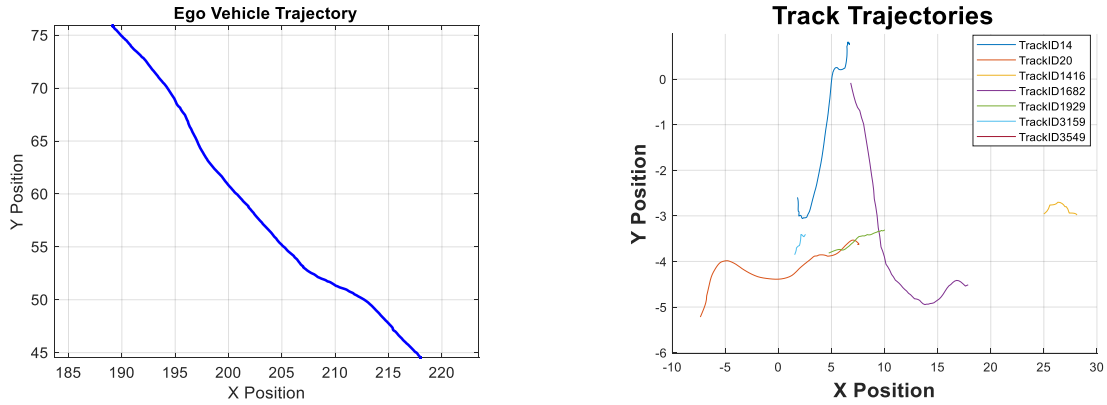


Fig. 4.1 Our framework providing a visual representation of the steps involved in the classification of tracks, allowing the generation of interaction dictionaries.

4.2.3 Null Force Filter

Afterward, the seven tracks and the trajectory of the ego vehicle are filtered using the NF filter. In this filter, it is assumed that no external forces are affecting the motion, as a result of which the system can continue to move at its previous velocity without being affected by external forces [88]. Using this approach, we were able to obtain



(a) Trajectory of the ego vehicle

(b) Positions of the seven tracks

Fig. 4.2 (a) Trajectory of the ego vehicle, derived from odometry sensor data, in absolute coordinates. (b) Positions of the seven tracks in relative coordinates.

generalized states, including positions and velocities, for each of the seven tracks as well as the ego vehicle. The GNG algorithm was applied to cluster the generalized states (positions and velocities along x and y) of the ego vehicle and the tracks [89]. By clustering, vocabularies with mean positions, mean velocity, and covariance between positions and velocity were generated. The parameters of the GNG algorithm were selected according to the size of the data points for each trajectory, ensuring that smaller trajectories have fewer nodes and larger ones have more. Each cluster of ego vehicles and tracks has a mean position and mean velocity defined as follows:

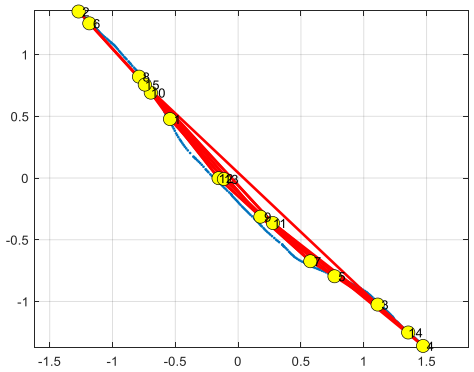
$$\mu_{\text{pos}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{4.1}$$

$$\mu_{\text{vel}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{v}_i \tag{4.2}$$

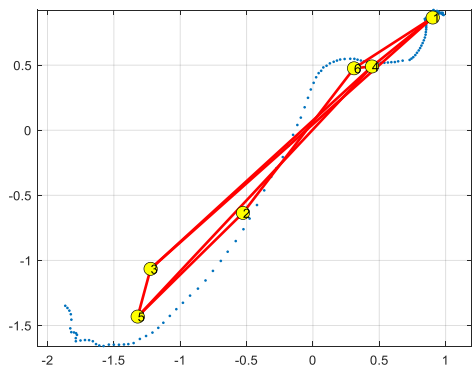
where $\mathbf{x}_i$ and $\mathbf{v}_i$ are the position and velocity vectors of the i-th node, respectively, and N is the number of nodes for that cluster. Using these mean vectors, proposed mathematical equations have been used to create dictionaries for the classification of tracks. As shown in Figure 4.3 (a-h), the clustered generalized states of the seven tracks and the ego vehicle's trajectory are shown.

4.2.4 Classification Models

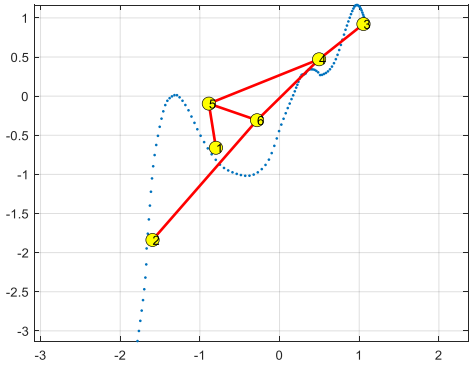
Thus, the interaction dictionary is constructed based on the mean position and velocity of each cluster, considering both the ego vehicle and tracks. For each timestamp, a cluster index is calculated using the Euclidean distance to the cluster mean. As part of the initialization of the interaction array, the positions of the ego vehicle as well as time, cluster indices, and interaction labels of both the ego vehicle and tracks are recorded.



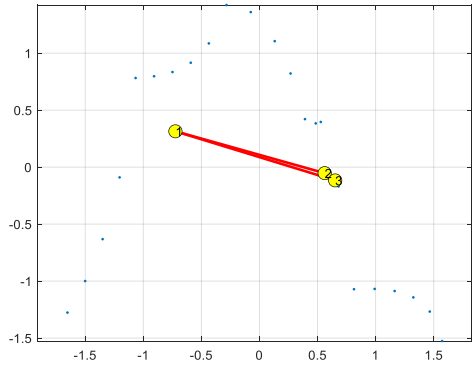
(a) Ego vehicle clustering.



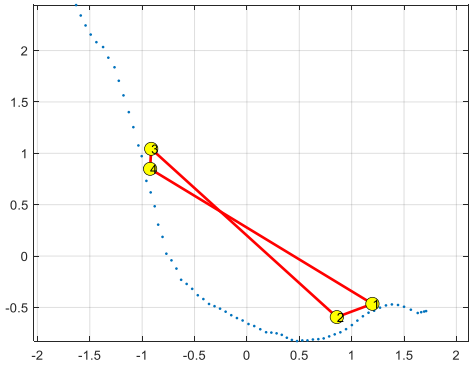
(b) Track 14 clustering (Car).



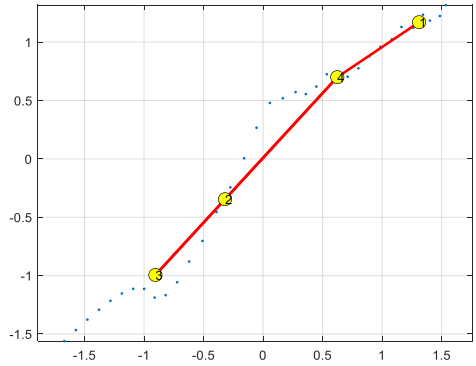
(c) Track 20 clustering (Building).



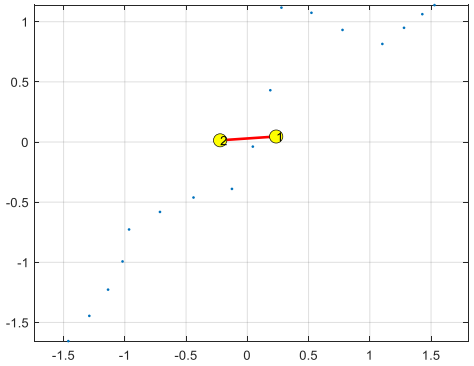
(d) Track 1416 clustering (Static pole).



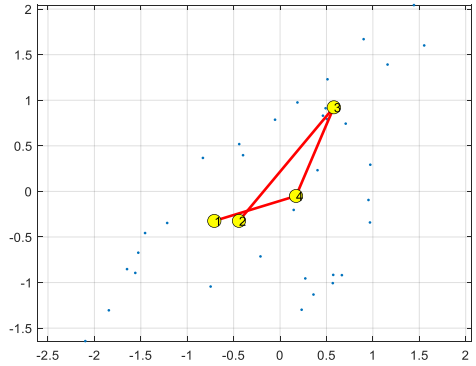
(e) Track 1682 clustering (Tree).



(f) Track 1929 clustering (Static pole).



(g) Track 3159 clustering (Tree).



(h) Track 3549 clustering (Tree).

Fig. 4.3 Clustering of ego vehicle and different track types. (a) Ego vehicle clustering, (b) Track 14 (Car), (c) Track 20 (Building), (d) Track 1416 (Static pole), (e) Track 1682 (Tree), (f) Track 1929 (Static pole), (g) Track 3159 (Tree), and (h) Track 3549 (Tree).

The timing analysis were performed for interactive clusters in which the ego vehicle and each track were simultaneously present in the space at the same time. Track 14 interacted with the ego vehicle from 1 to 14.7 seconds. Similarly, Track 20 had interactions with the ego vehicle from 0.7 to 14.5 seconds. Interactions for Track 1416 began at 15.8 seconds and ended at 18 seconds. Track 1682's interaction took 18 to 24.8 seconds. Track 1929 interacted with ego vehicle from 20 to 23.4 seconds. Interaction times between track 3159 and ego vehicle 32.1 to 33.7 seconds. Finally, Track 3549 interacted between 36 and 39.3 seconds. Table 4.1 illustrates the initial part of the interaction array belongs to the interactions between track 14 and ego vehicle. The track cluster index of 0 indicates that there was no interaction at that time, so the first interaction began at 1.0000. Note that this interaction continues until the end of a particular interaction, maintaining separate interaction arrays for the ego vehicle and for each track.

Table 4.1 Example of the interaction array between the ego vehicle and Track 14.

Time	**Track Cluster Index**	**Label**	**Ego Vehicle Cluster Index**	**Ego Vehicle x Position**	**Ego Vehicle y Position**
0.1000	4	0	0	218.0021	44.5398
0.2000	4	0	0	218.0051	44.5382
0.3000	4	0	0	218.0092	44.5362
0.4000	4	0	0	218.0013	44.5405
0.5000	4	0	0	217.9974	44.5423
0.6000	4	0	0	217.9942	44.5441
0.7000	4	0	0	217.9893	44.5475
0.8000	4	0	0	217.9881	44.5494
0.9000	4	0	0	217.9875	44.5504
1.0000	4	1	5	217.9887	44.5512
⋮	⋮	⋮	⋮	⋮	⋮

The hierarchical DBN model is illustrated in Figure 4.4. Odometry data is used to determine the position of the ego vehicle on the left side of the DBN. There are seven different tracks on the right side, provided by the LiDAR sensor. A layer of observation is located at the bottom, where raw sensor data has been collected. The middle level of the model (i.e., a continuous level) represents our generalized states. Through the use of the GNG algorithm, the third upper layer is a discrete layer that allows us to group the generalized states into clusters. Lastly, dictionary interactions occur at the top of the DBNs, where clusters derived from GNG are considered, enabling interaction between tracked objects and the ego vehicle. A switching variable is incorporated into the DBN to distinguish static tracks from dynamic tracks according to predefined mathematical equations. As a result, these classifications play a crucial role in extending learned

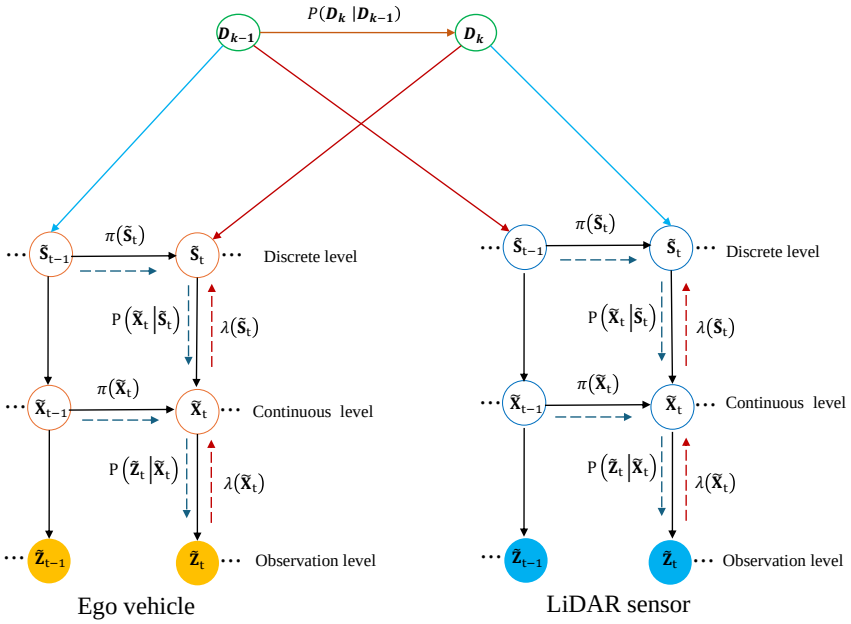


Fig. 4.4 Dynamic Bayesian Network (DBN) of the model.

vocabularies during testing, particularly for localizing the ego vehicle. For this purpose, separate dictionaries are maintained for each track and ego vehicle, each containing six columns of information: time, the cluster index of the ego vehicle, the cluster index of the track, the interaction label, as well as the x and y positions of the ego vehicle. It is especially useful for predicting the position of the ego vehicle to have the interaction label, which indicates whether it is interacting with a specific track at a given time. Using these dictionaries to analyze interaction labels and spatial data, the model can more accurately predict the ego vehicle's position relative to static tracks. It is achieved using a MJPF, which combines the strengths of Kalman filters and particle filters. Utilizing the consistent patterns in the static tracks' data, the MJPF refines the positional estimates by comparing predictions from the DBN with real-time observations. Whenever the switching variable indicates a static state, the DBN leverages the corresponding dictionary entries to provide more accurate and robust localization of the ego vehicle. The use of this approach ensures that the localization process is both data-driven and mathematically based, greatly improving the accuracy and reliability of navigation on the ego vehicle.

During the final stage of offline training phase, we classify tracks based upon the interaction between the clustered states of the ego vehicle and the clusters of the tracks. The interaction between an ego vehicle and a dynamic track cluster should be classified as dynamic, while the interaction with a static track cluster should be classified as static. The relative velocities of the ego vehicle and the track clusters determine whether a track is static or dynamic. It is done by measuring the difference between the ego vehicle's velocity and those of each track cluster over a period of time. To classify

whether a track is static or dynamic, the mathematical equations have been proposed to calculate the relative velocities of interacting clusters.

Let p_{track} denote the current position of a track cluster, p_{ego} the current position of the ego vehicle, p'_{track} the position of the track cluster at the next time step (after a short interval Δt), and v_{ego} the velocity of the ego vehicle. Assuming the track is static in the world (absolute) frame, its apparent motion in the ego-centric frame is induced solely by the ego vehicle's motion. The relationship between these positions is:

$$p_{\text{track}} - p_{\text{ego}} = p'_{\text{track}} - p_{\text{ego}} + v_{\text{ego}} \cdot \Delta t \tag{4.3}$$

This equation expresses the relative displacement of the static track cluster over time, accounting for the ego vehicle's movement during Δt.

By rearranging Equation (4.3), we obtain:

$$p_{\text{track}} = p'_{\text{track}} + v_{\text{ego}} \cdot \Delta t \tag{4.4}$$

Subtracting both sides by p_{track} and isolating the displacement yields:

$$p'_{\text{track}} - p_{\text{track}} = -v_{\text{ego}} \cdot \Delta t \tag{4.5}$$

Dividing both sides by Δt results in the apparent velocity of the track cluster in ego-centric coordinates:

$$v_{\text{track}} = -v_{\text{ego}} \tag{4.6}$$

Equation (4.6) indicates that a static track appears to move with the same speed but in the opposite direction to the ego vehicle. The relative velocity between the ego vehicle and the track cluster is then computed as the Euclidean norm of their velocity difference:

$$v_{\text{rel}} = \|v_{\text{ego}} - v_{\text{track}}\| \tag{4.7}$$

Substituting $v_{\text{track}} = -v_{\text{ego}}$ into Equation (4.7), we obtain:

$$v_{\text{rel}} = \|2v_{\text{ego}}\| \tag{4.8}$$

Equation (4.8) shows that in the ego-centric frame, a track that is truly static in the world frame will appear to move at a velocity equal and opposite to that of the ego vehicle, resulting in a relative velocity magnitude of twice the ego vehicle's speed.
This derivation is central to the classification logic: by measuring the relative velocity v_{rel}, we determine whether a track is static (i.e., $v_{\text{rel}} \approx 2\|v_{\text{ego}}\|$) or dynamic (i.e., deviates from this pattern) using a threshold.

4.2.5 Construction of the Interaction Dictionary

The interaction dictionary encodes the time-dependent relationship between the ego vehicle and each observed track during training. It is constructed using the step-by-step procedure summarized in Table 4.2. Each dictionary is constructed for each track during training, and is then combined for simultaneous interactions during testing.

Table 4.2 Steps followed to construct the interaction dictionary between the ego vehicle and the observed tracks. Each step transforms raw frame-level data into structured interaction entries.

Step	Description
1	Temporal Alignment: Synchronize LiDAR and odometry timestamps to associate ego and track observations frame-wise.
2	Track Association: Identify active TrackIDs using the JPDA filter at each frame.
3	Cluster Assignment: Assign ego and track positions to their respective GNG clusters to extract cluster indices.
4	Interaction Labeling: If ego and track positions fall within a defined spatial proximity, label the interaction as $L = 1$; otherwise $L = 0$.
5	Dictionary Population: Store the interaction entry as $I = \{t, C_{\text{ego}}, C_{\text{track}}, L, x, y\}$, where (x, y) is the ego vehicle position.

4.2.6 Parameter Configuration for JPDA and GNG

To ensure reproducibility, we detail here the configuration of the JPDA tracker and the GNG algorithm. These components support interaction-based clustering and continuous ego vehicle motion prediction under partial observability.

The GNG network dynamically adjusts its number of nodes based on the complexity of the trajectory during training, with an upper bound of 100 nodes to control overfitting and runtime. The NFF predicts the ego vehicle's motion by assuming no external forces influence its trajectory resulting in a constant velocity movement. It is a simplified Kalman filter that allows us to estimate future states under a zero acceleration assumption [37]. JPDA parameters were tuned for stability across multi-object scenes. All parameters were empirically selected using the training dataset and are summarized in Table 4.3.

Table 4.3 Parameter settings used in the JPDA tracker and GNG clustering algorithm. All values were empirically tuned using the training dataset.

Module	**Parameter**	**Description and Value**
JPDA Tracker	Gating threshold	0.8 (based on Mahalanobis distance)
	Confirmation count	3 consecutive detections required
	Missed detection tolerance	Up to 5 missed updates allowed
GNG Clustering	Maximum number of nodes	100 (adaptive per trajectory length)
	Learning rate (winner) ϵ_b	0.05
	Learning rate (neighbor) ϵ_n	0.0006
	Maximum edge age	50 iterations
	Node insertion interval	Every 100 steps
	Error decay factor β	0.0005

These values were validated during development and align with established practices in LiDAR-based trajectory modeling.

4.3 Results and Discussions

Using the minimum distance between the current positions of the ego vehicle and predefined cluster centroids of track positions, the nearest cluster indices were determined for both the ego vehicle and the tracks at each time stamp. According to the Figures from 4.5 to 4.11, cluster indices vary over time, illustrating how the ego vehicle and a particular track behave over time. The red line represents the cluster assignments for the ego vehicle over time, while the blue line represents the cluster assignments for the track during interactions. A comprehensive overview of the dynamics of the interaction between the ego vehicle and the surrounding tracks is shown in Figure 4.12. At various time instants, it details when each track interacts with the ego vehicle. This figure depicts both scenarios in which multiple tracks interact with the ego vehicle simultaneously, as well as scenarios in which separate tracks engage at different intervals.
Figure 4.5 presents a dynamic track interaction with the ego vehicle, i.e. a car in front of ego vehicle, while figure 4.6 shows a building to the right, and the rest show static tracks, i.e. static poles and trees. For the purpose of distinguishing static clusters from dynamic clusters, the threshold is calculated by accumulating all relative velocities of interactions in a histogram and calculating the mean plus standard deviation. An adaptive threshold method for detecting human motion is based on the method proposed in [90]. A histogram is shown in Figure 4.13: dynamic velocities appear to the right of the threshold, and static velocities appear to the left. Some dynamic points are misclassified as static points and vice versa, but our model correctly classifies most

of the points.
The validation is illustrated in Figure 4.14 and has been validated over time, demonstrating both static and dynamic classification accuracy. We are able to distinguish between static clusters (red points below the threshold line) and dynamic clusters (blue points above the threshold line). Using this validation, we are able to confirm that the threshold we chose reliably classifies interactions, as we are already familiar with the expected behavior of track clusters over time. As a result, the majority of points are correctly classified, demonstrating the effectiveness of our model.

Our classification model is further validated by computing the confusion matrix [91] shown in Figure 4.15. It compares the true labels (dynamic or static) with the predicted labels. A true class is based on known interactions, such as Track 14 is being dynamic, whereas other remaining tracks are static. The predicted classes are determined using our classification algorithm, which utilizes threshold values derived from the mean and standard deviations of relative velocities. The results of the matrix are shown:

- True positive (TP): The 317 dynamic interactions are correctly classified.
- True negative (TN): The 85 static interactions are correctly classified.
- False positive (FP): 43 static interactions are misclassified as dynamic.
- False negative (FN): 17 dynamic interactions are misclassified as static.

The results of the performance metrics are shown in Table 4.4.

Table 4.4 Performance metrics for iCAB and KITTI datasets

Metric	**iCAB**	**KITTI**
Accuracy	87%	82%
Precision	88%	80%
Recall	94%	89%
F1 Score	91%	85%

The classification model we developed achieved an accuracy of 87% in classifying static and dynamic tracks, demonstrating its effectiveness. Based on the high recall of 94%, most dynamic interactions are correctly identified. A precision of 88% indicates a good balance between true positives and false positives. The F1 score of 91% indicates that our model exhibits a strong balance between precision and recall, which confirms the robustness of the model.

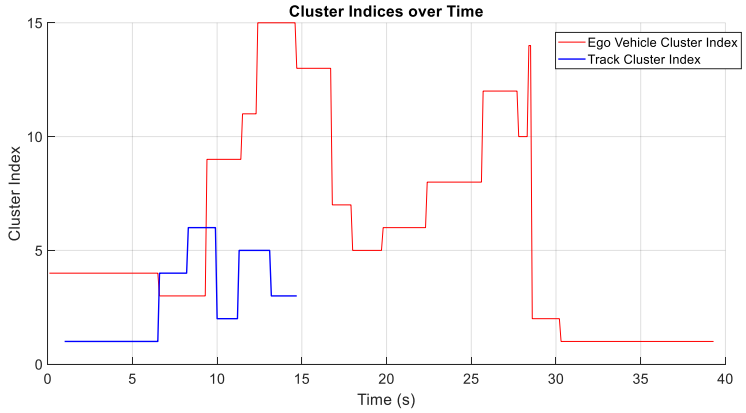


Fig. 4.5 Interaction of ego vehicle and Track 14 clusters over time.

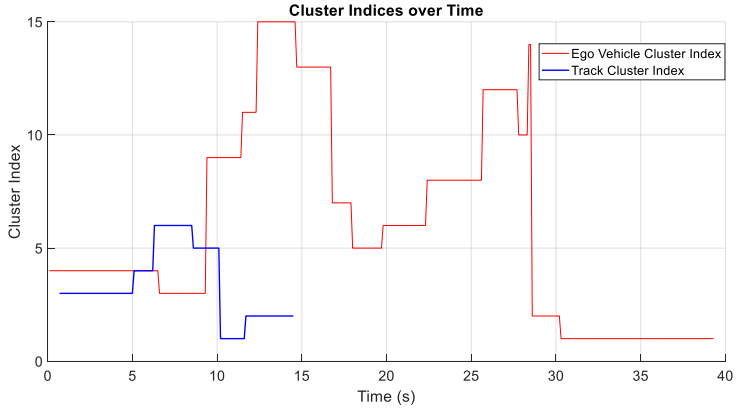


Fig. 4.6 Interaction of ego vehicle and Track 20 clusters over time

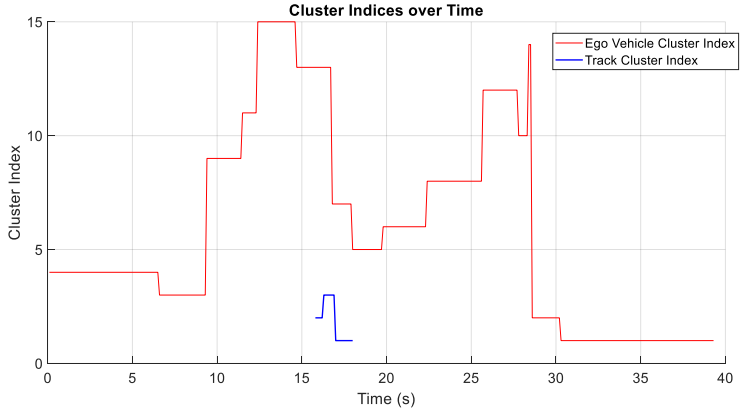


Fig. 4.7 Interaction of ego vehicle and Track 1416 clusters over time

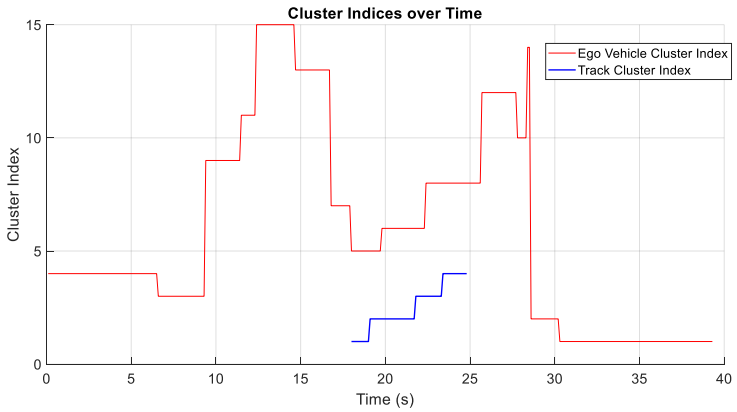


Fig. 4.8 Interaction of ego vehicle and Track 1682 clusters over time

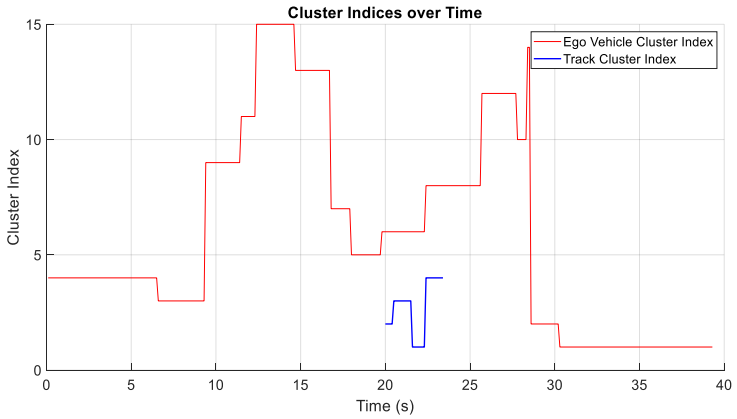


Fig. 4.9 Interaction of ego vehicle and Track 1929 clusters over time

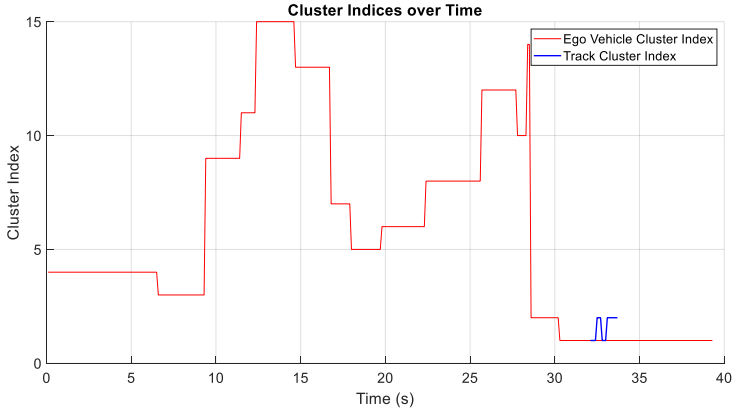


Fig. 4.10 Interaction of ego vehicle and Track 3159 clusters over time

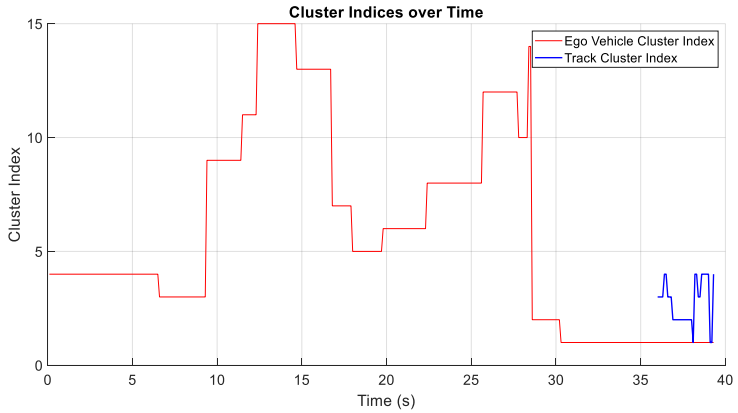


Fig. 4.11 Interaction of ego vehicle and Track 3549 clusters over time

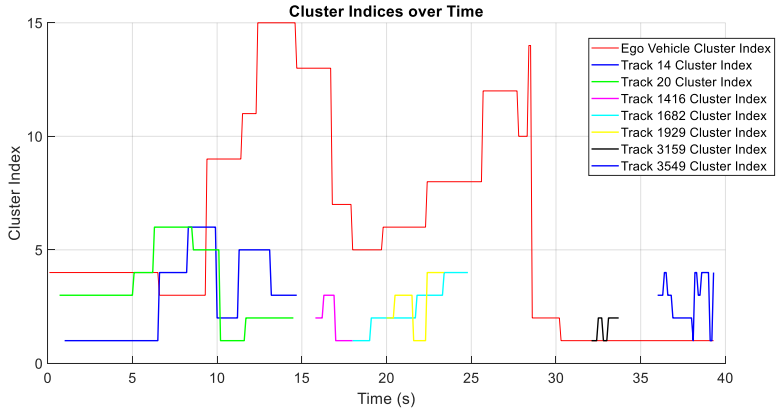


Fig. 4.12 Interaction of ego vehicle and Track 1052 clusters over time

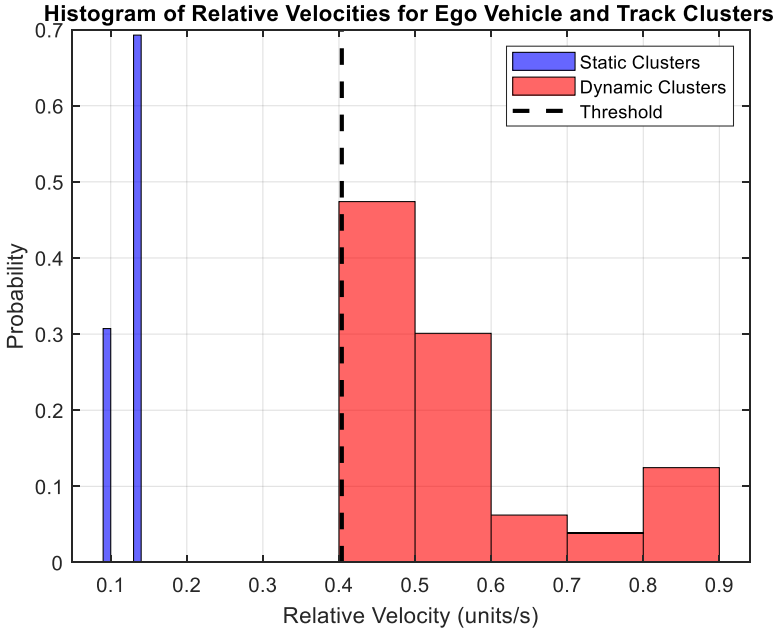


Fig. 4.13 Histogram of relative velocities

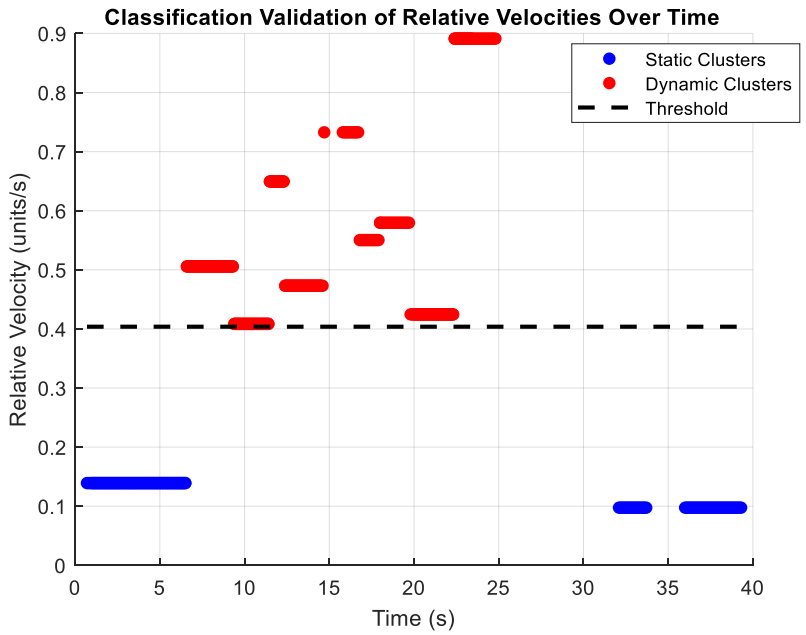


Fig. 4.14 Classification validation

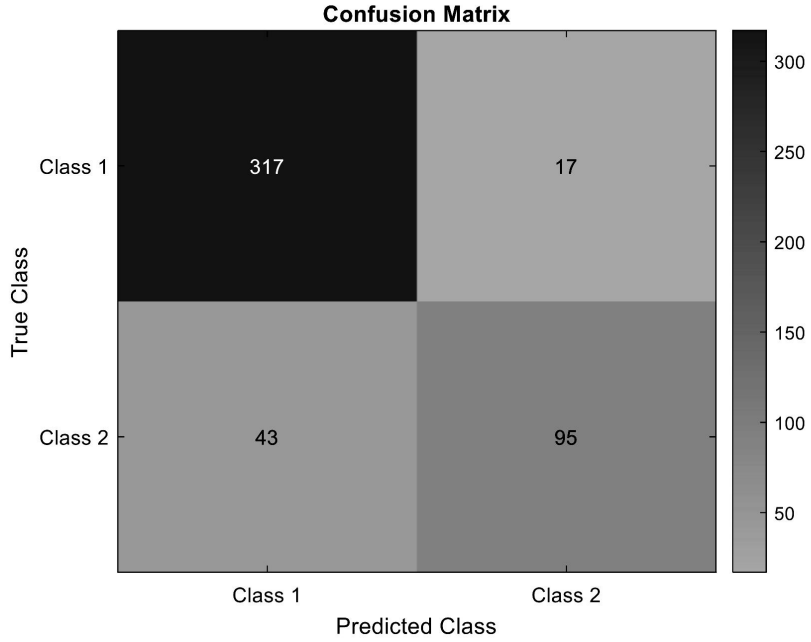


Fig. 4.15 Confusion matrix for classification

4.3.1 Misclassifications and Practical Implications

The overall classification accuracy of 87% demonstrates the effectiveness of the proposed framework. However, a small number of misclassifications were observed in the confusion matrix, mainly involving the dynamic car track. In some frames, the car's motion appeared similar to a static object due to temporary stops or low relative velocity, which may have caused occasional confusion during classification.

It is important to note that both the training and testing datasets were collected in the same environment, ensuring consistency in scene structure and sensor conditions. This significantly reduced the likelihood of domain-related misclassification. Furthermore, the testing phase included only one dynamic object, limiting the impact of these rare classification errors.

Although these misclassifications did not lead to any significant degradation in localization results in this controlled setting, they highlight the importance of considering temporal motion consistency and object behavior patterns in more diverse and dynamic environments.

4.4 Summary

This chapter introduced a classification framework that accurately distinguishes between static and dynamic tracks using LiDAR and odometry data. By employing algorithms such as JPDA, the NFF, and GNG, a structured offline pipeline was developed to construct interaction vocabularies crucial for understanding ego vehicle's environment dynamics. The experimental evaluation demonstrated an 87% classification accuracy, validating the robustness of the proposed method in diverse interaction scenarios. These pre-classified static tracks serve as reliable environmental landmarks, enabling a shift from perception to precise localization.

Although this chapter focuses on offline processing, its outputs directly influence how the system performs during the online phase. The use of static tracks as persistent landmarks ensures robustness against transient dynamic changes, anchoring localization to stable features. Moreover, this framework in the subsequent stage leverages these dictionaries while incorporating probabilistic gating and anomaly detection metrics (e.g., CLA, KLDA) to flag mismatched or unexpected observations. A trade-off is evident between the duration and resolution of offline processing and the accuracy of the online estimation: richer modeling leads to better predictive performance, but at the cost of potential overfitting. To balance this, the framework retains only salient and frequently observed interactions that generalize well to new conditions.

Building upon this foundation, the next chapter details an online localization framework that utilizes the generated interaction dictionaries and probabilistic graph-

ical models to estimate the ego vehicle's position in real time, even in complex and observation-sparse conditions.

Chapter 5

Localization Framework using LiDAR via Probabilistic Graphical Models

This chapter presents a novel framework for ego vehicle localization in dynamic environments using LiDAR-based observations. Building upon the classification of tracks into static and dynamic categories, this framework integrates MJPF with interaction dictionaries and learned vocabularies to enhance localization accuracy. The framework's foundation lies in a dual-phase localization approach, which leverages pre-classified static tracks to enhance the reliability of state estimation. The initial phase employs separate track-based predictive models, each designed to characterize the interaction patterns between the ego vehicle and individual static tracks. These predictions are represented using MJPF, enabling the system to effectively model both discrete and continuous state transitions across time. In the second stage, the framework transitions from individual predictions to a more comprehensive approach that incorporates combined interaction dictionaries. These dictionaries are constructed during the offline training phase and encode the dynamics of multi-object interactions, enabling the ego vehicle to perform accurate localization in complex scenarios, such as simultaneous tracks interactions, no observation periods and finally the interactions of single tracks with the ego vehicle.

5.1 Problem Statement

To ensure safety and reliability, AVs must be accurately localized. There are, however, significant errors in traditional methods that rely on GPS and odometry sensors in environments with no GPS (e.g., urban canyons, tunnels) and when the sensors are subjected to adverse conditions [92]. This presents a challenge that must be addressed by leveraging reliable environmental features. Static tracks, such as buildings, poles,

and trees, provide stable reference points for accurate localization. There are several critical gaps that prevent their effective utilization, despite their potential:

- Classified static tracks are often not properly incorporated into existing frameworks for predicting the state of ego-vehicles [20].
- The interaction information between the ego vehicle and static tracks, which would enhance trajectory accuracy, is neglected [16].
- The simultaneous interactions between the ego vehicle and multiple static tracks at precise time instants are not systematically modeled, resulting in inaccurate state updating [16].

Using interaction dictionaries, this study aims to model temporal and spatial relationships between ego vehicle and static tracks to resolve these issues.

$$\mathcal{I} = \{t, C_{\text{ego}}, C_{\text{track}}, L, x, y\} \tag{5.1}$$

where t is the time of interaction, C_{ego} and C_{track} represent the ego vehicle's and track's cluster indices, L is the interaction label, and (x, y) are the ego vehicle's positions.

5.2 Proposed Framework

Figure 5.1 illustrates the proposed framework, which consists of two major phases: offline training and online testing. During the offline training phase, already discussed in Chapters 4, data from LiDAR and odometry sensors was used to generate vocabulary and interaction dictionaries. This phase included preprocessing, MTT, and clustering in order to determine whether tracks were static or dynamic, with static tracks serving as reliable landmarks. This chapter focuses on the online testing phase, which is specifically designed to assess the ego vehicle's localization based on separate tracks predictions and a combined dictionary approach . In the online phase, the learned vocabulary and interaction dictionaries are used to estimate the positions of the ego vehicle solely based on the observations from LiDAR testing dataset. To ensure clarity in the probabilistic formulation, we briefly summarize the key notations used in this chapter:

- $\mathbf{X}_k \in \mathbb{R}^4$: Ego vehicle state at time k, defined as $\mathbf{X}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]^\top$, where position is in meters and velocity in m/s.
- $\mathbb{S}_k \in \{1, \ldots, S\}$: Discrete superstate at time k, indexing one of the S learned interaction classes.
- $\Phi(\mathbf{X}_k) \in \mathbb{R}^d$: Classifier-based likelihood vector for $\mathbf{X}_k$, derived from GNG+NFF feature embeddings.

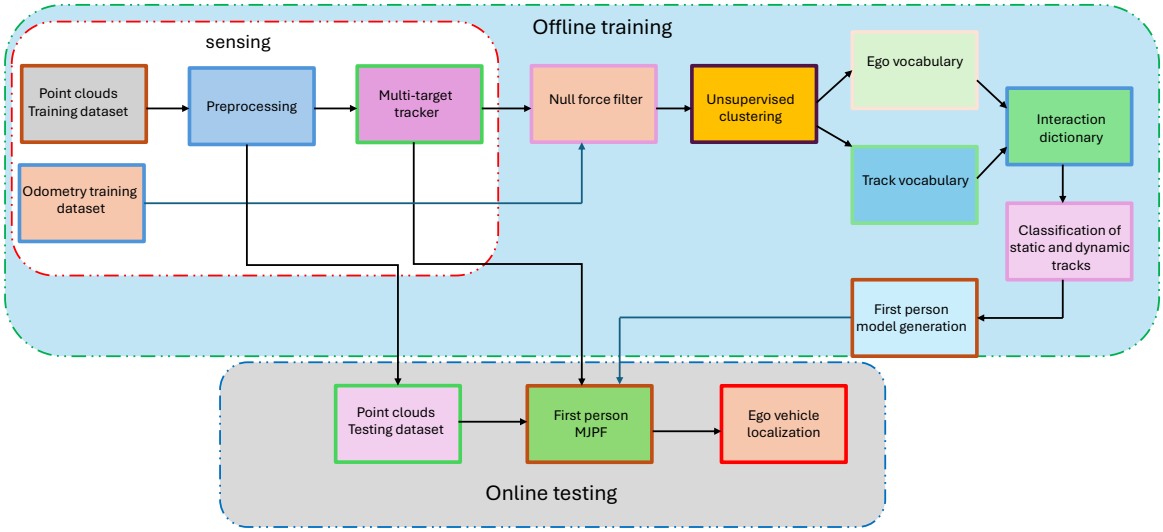


Fig. 5.1 Our framework providing a visual representation of the steps involved in the classification of a tracks, allowing the generation of interaction dictionaries as first person model and perform localization as first person MJPF.

- $\Pi(\mathbf{X}_k \mid \mathbb{S}_k) \in \mathbb{R}^d$: Prior likelihood conditioned on $\mathbb{S}_k$, computed from interaction dictionaries.
- $\Delta_B(P,Q) \geq 0$: Bhattacharyya distance between two Gaussians $P = \mathcal{N}(\mu_P, \Sigma_P)$ and $Q = \mathcal{N}(\mu_Q, \Sigma_Q)$, used for anomaly detection:

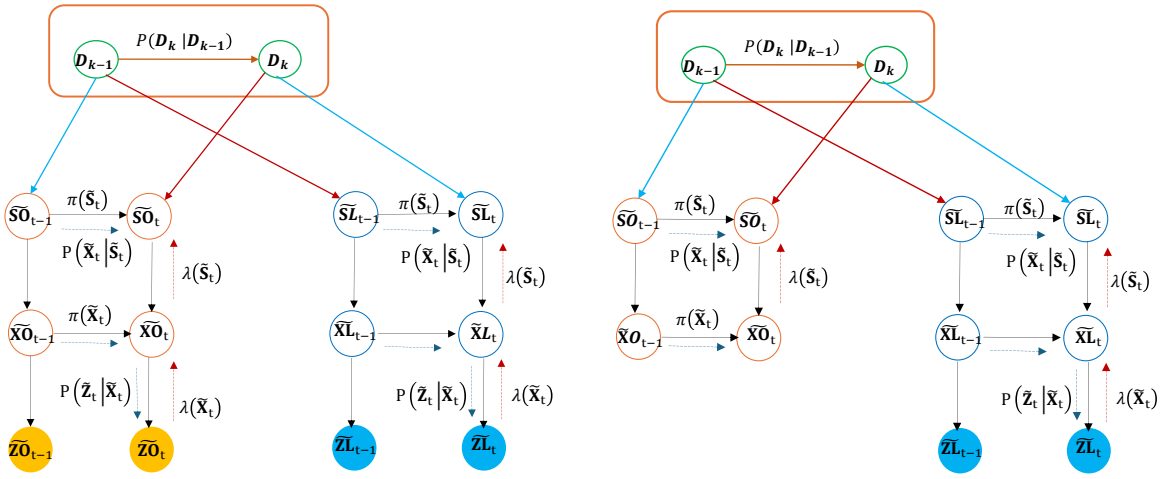
$$\Delta_B = \frac{1}{8}(\mu_P - \mu_Q)^\top \Sigma^{-1}(\mu_P - \mu_Q) + \frac{1}{2}\ln\left(\frac{\det\Sigma}{\sqrt{\det\Sigma_P \cdot \det\Sigma_Q}}\right)$$

  where $\Sigma = \frac{1}{2}(\Sigma_P + \Sigma_Q)$.
- $P(\mathbb{S}_k \mid \mathbb{S}_{k-1})$: Transition probability between superstates.
- $\tilde{Z}_k \in \mathbb{R}^2$: Noisy observation of ego position at time k, derived from LiDAR-based static landmarks.
- $\hat{\mathbf{X}}_k^{(i)}$: i-th particle representing a state hypothesis in the MJPF at time k.

Following are a description of the theoretical concepts and methodologies underlying the online testing phase.

5.2.1 Online Testing Phase as Separate Prediction Models

The objective of the online phase is to localize the ego vehicle using track observations derived from a testing dataset of LiDAR-based environment. It is important to note that the localization process utilizes the learned vocabulary and interaction dictionaries developed during the offline training phase. Using a combination of track cluster



(a) Dynamic Bayesian network during training

(b) Dynamic Bayesian network during testing

Fig. 5.2 Dynamic Bayesian networks representing (a) training and (b) testing phases. The models illustrate how the system evolves and predicts the ego vehicle's trajectory in different scenarios.

matching, particle initialization, state predictions, and update phases, we estimate the trajectory of the ego vehicle using the MJPF. MJPF incorporates separate prediction models for tracks and ego vehicle as well as interaction dictionaries for state refinement during updates. The DBN model during testing phase is shown in figure 5.2 (b).

Matching Phase: Bhattacharyya Distance for Track-Cluster Association

In this phase, the observed tracks are matched to their corresponding learned clusters using the Bhattacharyya distance d_B. This distance measures the similarity between the observed track state $\mathbf{z}_t^{\text{obs}}$ and the learned cluster state $\mathbf{z}_t^{\text{cluster}}$, ensuring accurate association for ego vehicle localization. The Bhattacharyya distance is computed as:

$$d_B(\mathbf{z}_t^{\text{obs}}, \mathbf{z}_t^{\text{cluster}}) = \frac{1}{8}(\mathbf{z}_t^{\text{obs}} - \mathbf{z}_t^{\text{cluster}})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z}_t^{\text{obs}} - \mathbf{z}_t^{\text{cluster}}) + \frac{1}{2}\ln\left(\frac{\det(\boldsymbol{\Sigma})}{\sqrt{\det(\boldsymbol{\Sigma}_{\text{obs}})\det(\boldsymbol{\Sigma}_{\text{cluster}})}}\right) \tag{5.2}$$

Here, $\boldsymbol{\Sigma}_{\text{obs}}$ and $\boldsymbol{\Sigma}_{\text{cluster}}$ are the covariance matrices of the observed track and the learned cluster, respectively, which quantify the uncertainty in the state estimates. The use of the Bhattacharyya distance in Equation (5.2) requires the definition of an intermediate covariance matrix Σ, which is constructed as the arithmetic mean of the observed track covariance Σ_{obs} and the learned cluster covariance Σ_{cluster}, that is:

$$\Sigma = \frac{1}{2}(\Sigma_{\text{obs}} + \Sigma_{\text{cluster}}).$$

This formulation equally weighs both uncertainty sources to ensure a fair and symmetric comparison between the observed track and each potential cluster. The rationale for adopting equal weighting stems from the practical design of the framework: both Σ_{obs} and Σ_{cluster} are estimated from real sensor data — the former derived from incoming LiDAR observations and the latter from previously clustered interaction trajectories during training. Since no external confidence metric is available to justify a dominant contribution from either side, an unbiased average avoids introducing any implicit preference. Moreover, the symmetric mean is a standard convention in Bhattacharyya-based similarity metrics when neither distribution is known to be inherently more reliable. In our case, this approach allows the framework to remain generalizable across environments and makes the matching process robust to variations in sensor quality or training set size. The intermediate covariance matrix represents the joint uncertainty by averaging the covariances from both the observed track and the cluster, ensuring that the similarity measurement reflects the variability in both sources. The best matching cluster S_{vt} is selected by finding the one with the minimum Bhattacharyya distance:

$$S_{vt} = \arg\min_{S} d_B(\mathbf{z}_t^{\text{obs}}, \mathbf{z}_t^{\text{cluster}}) \tag{5.3}$$

This matching process ensures that each observed track is correctly associated with its corresponding learned cluster, enabling precise localization of the ego vehicle based on track observations. Equation (5.3) does not represent a closed-form solution but instead defines how the ego vehicle state is corrected using dictionary-based interaction terms. This computation is carried out as part of the recursive update step in the MJPF. Specifically, the dictionary entries provide the matrices $D(S_{t,n})$ and $E(S_{t,n})$, which are pre-learned from offline training data and used to adjust each particle's state estimate based on its associated track cluster. Thus, Equation (5.3) is resolved numerically through particle-wise updates during filtering, not analytically.

Initialization Phase as Particle Initialization

Once the matching phase has been completed, particles are initialized to represent hypotheses about the states of the ego vehicle and tracks. During the initialization process, particle states are sampled from matched cluster distributions, as for track states

$$\mathbf{z}_{t=1,n}^{\text{track}} \sim \mathcal{N}(\mathbf{M}_{S_{vt}}, \mathbf{Q}_{S_{vt}}) \tag{5.4}$$

and ego vehicle states are

$$\mathbf{z}_{t=1,n}^{\text{ego}} \sim \mathcal{N}(\mathbf{M}_{S_{vt}}^{\text{ego}}, \mathbf{Q}_{S_{vt}}^{\text{ego}}) \tag{5.5}$$

Here $\mathbf{M}_{S_{vt}}$ and $\mathbf{Q}_{S_{vt}}$ are the mean and covariance of the matched cluster for tracks. $\mathbf{M}_{S_{vt}}^{\text{ego}}$ and $\mathbf{Q}_{S_{vt}}^{\text{ego}}$ represent the same for the ego vehicle.

Prediction Phase: Separate Predictions for Tracks and Ego Vehicle

Each track is predicted using the tranistion matrix $\mathbf{A}^{\text{track}}(S_{t,n})$ corresponding to the matched cluster:

$$\mathbf{z}^{\text{track}}_{t+1|t,n} = \mathbf{A}^{\text{track}}(S_{t,n})\mathbf{z}^{\text{track}}_{t|t,n} + \mathbf{w}^{\text{track}}_{t} \tag{5.6}$$

where $\mathbf{z}^{\text{track}}_{t+1|t,n}$ is the predicted state of the track at time $t+1$ and $\mathbf{w}^{\text{track}}_{t}$ represents the process noise for the track. The state of the ego vehicle is predicted using the the transition matrix $\mathbf{A}^{\text{ego}}(S_{t,n})$:

$$\mathbf{z}^{\text{ego}}_{t+1|t,n} = \mathbf{A}^{\text{ego}}(S_{t,n})\mathbf{z}^{\text{ego}}_{t|t,n} + \mathbf{B}^{\text{ego}}(S_{t,n}) + \mathbf{w}^{\text{ego}}_{t} \tag{5.7}$$

In this case, $\mathbf{B}^{\text{ego}}(S_{t,n})$ accounts for velocity-related terms derived from ego vehicle dynamics and $\mathbf{w}^{\text{ego}}_{t}$ is the process noise for the ego vehicle. The dual prediction approach is capable of handling the distinct dynamics of tracks and ego vehicle.

Update Phase as Refining State Estimates

Each particle's track state has been refined using the observed track state $\mathbf{a}^{\text{track}}_{t}$ and the observation matrix $\mathbf{C}^{\text{track}}(S_{t,n})$:

$$\mathbf{z}^{\text{track}}_{t|t,n} = \mathbf{z}^{\text{track}}_{t|t-1,n} + \mathbf{K}^{\text{track}}_{t}\left(\mathbf{a}^{\text{track}}_{t} - \mathbf{C}^{\text{track}}(S_{t,n})\mathbf{z}^{\text{track}}_{t|t-1,n}\right) \tag{5.8}$$

here, $\mathbf{K}^{\text{track}}_{t}$ is the Kalman gain for the track state. Using the interaction dictionaries, the state of the ego vehicle is updated:

$$\mathbf{z}^{\text{ego}}_{t|t,n} = \mathbf{z}^{\text{ego}}_{t|t-1,n} + \mathbf{D}(S_{t,n})\mathbf{z}^{\text{track}}_{t|t,n} + \mathbf{E}(S_{t,n}) \tag{5.9}$$

where $\mathbf{D}(S_{t,n})$ adjusts the ego vehicle's state based on the current track state while $\mathbf{E}(S_{t,n})$ is a correction term derived from the interaction dictionaries, utilizing the ego vehicle's x, y positions.

5.2.2 Online Testing Phase as Combined Dictionary Predictions

In the section 5.2, MJPF approach was introduced for localizing an AV using static LiDAR-based landmarks. Specifically, in section 6.1 of chapter 6, in-depth analysis of results is presented where *individual* track dictionaries (e.g., buildings, poles, trees) were used to estimate ego-vehicle trajectories. It was demonstrated that each track, treated separately, could yield highly accurate localization results if treated as a stable landmark.

However, real-world driving scenarios are far more complex. Multiple static tracks can appear *simultaneously*, and certain time intervals might exist with *no observations*.

This motivates the need to *combine* the information contained in the different dictionaries (one dictionary per track) into a unified framework. By doing so, we can better handle the following situations:

1. **Simultaneous Track Interactions:** When multiple static tracks are observed at the same instant, combining dictionaries can refine ego-vehicle position estimates by leveraging cross-track constraints.

2. **Extended No-Observation Periods:** In some time intervals, no new LiDAR observations of static tracks may be available (e.g., occlusion or sensor dropout). Incorporating combined dictionaries can help maintain stable localization by relying on prior learned interactions across multiple tracks.

3. **Isolated Track Interactions:** Even when only one track is observed, the combined dictionary framework can cross-validate with the other track dictionaries (now unused) to check consistency, thereby reducing ambiguity.

Combined Dictionaries Creation for Different Scenarios

As shown from the results in section 5.2, each static track (e.g., building, pole, tree) was associated with a *separate* interaction dictionary, denoted generically by:

$$\mathcal{I}_k = \left\{ t, C_{\text{ego}}, C_{\text{track}}, L, x, y \right\}$$

where k indexes the track ID or track cluster. Each $\mathcal{I}_k$ encodes the learned interaction relationship between the ego vehicle and track k, including the time t, cluster indices C_{ego} and C_{track}, an interaction label L, and the corresponding ego positions (x, y).

Motivation for Combined Dictionaries

Although single-track interaction dictionaries (e.g., $\mathcal{I}_{\text{building}}$, $\mathcal{I}_{\text{pole}}$, $\mathcal{I}_{\text{tree}}$) can independently provide good localization, real driving often observes *multiple* static tracks simultaneously. Hence, combining these dictionaries into a **single** representation $\mathcal{I}_{\text{combined}}$ can yield synergy and redundancy. Specifically:

- **Redundancy**: If one track becomes occluded, the other tracks still refine the localization [93] [94].

- **Consistency**: Conflicting track estimates can be reconciled by weighting each track's reliability in a joint update [95].

- **Completeness**: Using multiple tracks can better cover all possible motion states in the environment [96].

Steps to Create a Combined Dictionary

Step 1: Concatenate All Track Dictionaries. We initialize $\mathcal{I}_{\text{combined}}$ as the union of all $\mathcal{I}_k$:

$$\mathcal{I}_{\text{combined}} = \bigcup_{k=1}^{K} \mathcal{I}_k \tag{5.10}$$

where K is the total number of six static tracks considered.

Step 2: Remove Duplicate or Overlapping Entries. Certain time instants t may have interactions with *multiple* tracks. These are *simultaneous* interactions. If the same ego-vehicle cluster C_{ego} is repeated with minor variations, we merge or remove duplicates to ensure the dictionary remains concise.

Step 3: Store Track-Specific Indices. To preserve which track each interaction came from, we store an additional track ID label in each dictionary entry:

$$\Big\{t, C_{\text{ego}}, C_{\text{track}}, L, x, y, \text{track\_ID}\Big\}.$$

This helps in quickly referencing the relevant transition matrices or covariance structures for that track.

Step 4: Recompute Interaction Statistics. For each unique pair $(C_{\text{ego}}, C_{\text{track}})$, we can compute:

$$(\mu_x, \mu_y, \Sigma_x, \Sigma_y)$$

to represent the mean and covariance of the ego positions (x, y) encountered in those interactions. These statistics become vital when updating or predicting the ego state from the combined dictionary.

Bhattacharyya Distance to Match Observed Tracks

Just as in equation 5.2, whenever a new *observed track* $\mathbf{z}_t^{\text{obs}}$ appears in the LiDAR data, we must determine *which* track cluster it corresponds to in the combined dictionary $\mathcal{I}_{\text{combined}}$. We use the Bhattacharyya distance d_B:

$$d_B\Big(\mathbf{z}_t^{\text{obs}}, \mathbf{z}_t^{\text{cluster}}\Big) = \frac{1}{8}\Big(\mathbf{z}_t^{\text{obs}} - \mathbf{z}_t^{\text{cluster}}\Big)^{\top} \mathbf{\Sigma}^{-1} \Big(\mathbf{z}_t^{\text{obs}} - \mathbf{z}_t^{\text{cluster}}\Big) + \frac{1}{2} \ln\left(\frac{\det(\mathbf{\Sigma})}{\sqrt{\det(\mathbf{\Sigma}_{\text{obs}})\,\det(\mathbf{\Sigma}_{\text{cluster}})}}\right), \tag{5.11}$$

where

$$\mathbf{\Sigma} = \tfrac{1}{2}\Big(\mathbf{\Sigma}_{\text{obs}} + \mathbf{\Sigma}_{\text{cluster}}\Big).$$

We assign the observed track to the cluster S_{vt} that yields the *minimum* Bhattacharyya distance:

$$S_{vt} = \arg\min_{S \in \mathcal{I}_{\text{combined}}} d_B\Big(\mathbf{z}_t^{\text{obs}}, \mathbf{z}_t^{\text{cluster}}\Big). \tag{5.12}$$

Thus, the combined dictionary still uses the same track-cluster matching but across *all* possible track entries from 1 to K.

Extended MJPF Equations for Combined Dictionaries

Recalling from section 5.2.1, the basic MJPF steps are:

1. **Particle Initialization.**
2. **Prediction Phase (Track States + Ego State).**
3. **Update Phase (Measurements + Interaction Dictionaries).**

When combining dictionaries, the core *structure* is unchanged, but we must handle *multiple* track observations at once. Below we provide the extended equations as follows.

Particle Initialization for Multiple Tracks

At $t = 1$, if the AV observes M different static tracks from LiDAR, we sample N particles for *each* track $m = 1, \ldots, M$, following:

$$\mathbf{z}_{1,n}^{\text{track},m} \sim \mathcal{N}\Big(\mathbf{M}_{S_{v1}^{(m)}}, \mathbf{Q}_{S_{v1}^{(m)}}\Big), \tag{5.13}$$

where $S_{v1}^{(m)}$ is the matched dictionary entry for track m. For the *ego vehicle*, we combine all relevant dictionary means:

$$\mathbf{z}_{1,n}^{\text{ego}} \sim \mathcal{N}\Big(\overline{\mathbf{M}}^{\text{ego}}, \overline{\mathbf{Q}}^{\text{ego}}\Big), \tag{5.14}$$

where

$$\overline{\mathbf{M}}^{\text{ego}} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{M}_{S_{v1}^{(m)}}^{\text{ego}}, \quad \overline{\mathbf{Q}}^{\text{ego}} = \text{diag}\Big(\sigma_x^2, \sigma_y^2, \ldots\Big)$$

is an averaged or fused initial guess.

Prediction Phase (Per Track + Ego Vehicle)

Each track m evolves via:

$$\mathbf{z}_{t+1|t,n}^{\text{track},m} = \mathbf{A}^{\text{track}}\Big(S_{t,n}^{(m)}\Big)\, \mathbf{z}_{t|t,n}^{\text{track},m} + \mathbf{w}_t^{\text{track},m}, \tag{5.15}$$

while the ego vehicle evolves as:

$$\mathbf{z}_{t+1|t,n}^{\text{ego}} = \mathbf{A}^{\text{ego}}\Big(\mathbf{S}_{t,n}^{(1)}, \ldots, \mathbf{S}_{t,n}^{(M)}\Big)\, \mathbf{z}_{t|t,n}^{\text{ego}} + \mathbf{B}^{\text{ego}}\Big(\mathbf{S}_{t,n}^{(1)}, \ldots, \mathbf{S}_{t,n}^{(M)}\Big) + \mathbf{w}_t^{\text{ego}}, \tag{5.16}$$

where $\mathbf{A}^{\text{ego}}$ and $\mathbf{B}^{\text{ego}}$ may now depend on *all* matched track indices $(1, \ldots, M)$, so that we integrate *multiple dictionary* information if available.

Update Phase (Measurements + Combined Interaction)

Upon receiving LiDAR measurements $\mathbf{a}_t^{\text{track},m}$ for each observed track m, we update track m:

$$\mathbf{z}_{t|t,n}^{\text{track},m} = \mathbf{z}_{t|t-1,n}^{\text{track},m} + \mathbf{K}_t^{\text{track},m}\left(\mathbf{a}_t^{\text{track},m} - \mathbf{C}^{\text{track}}\left(S_{t,n}^{(m)}\right)\mathbf{z}_{t|t-1,n}^{\text{track},m}\right), \tag{5.17}$$

where $\mathbf{K}_t^{\text{track},m}$ is the Kalman gain for track m and $\mathbf{C}^{\text{track}}\left(S_{t,n}^{(m)}\right)$ is the observation matrix derived from the combined dictionary.

For the ego vehicle, we combine all track updates via the dictionaries:

$$\mathbf{z}_{t|t,n}^{\text{ego}} = \mathbf{z}_{t|t-1,n}^{\text{ego}} + \sum_{m=1}^{M}\left[\mathbf{D}^{(m)}\left(S_{t,n}^{(m)}\right)\mathbf{z}_{t|t,n}^{\text{track},m} + \mathbf{E}^{(m)}\left(S_{t,n}^{(m)}\right)\right], \tag{5.18}$$

where $\mathbf{D}^{(m)}\left(S_{t,n}^{(m)}\right)$ and $\mathbf{E}^{(m)}\left(S_{t,n}^{(m)}\right)$ are the track-specific adjustments for track m, extracted from the *combined* dictionary.

DBN Structure During Training

The training phase as shown in figure 5.3 incorporates data from both the **odometry sensor** and the **LiDAR training dataset** to build a robust model of the environment. The odometry sensor provides precise ego vehicle positions, forming a reliable foundation for motion estimation, while the LiDAR dataset captures interactions with surrounding objects. From this data, six static tracks—corresponding to stationary objects such as buildings, poles, and trees—were classified and stored for future reference.

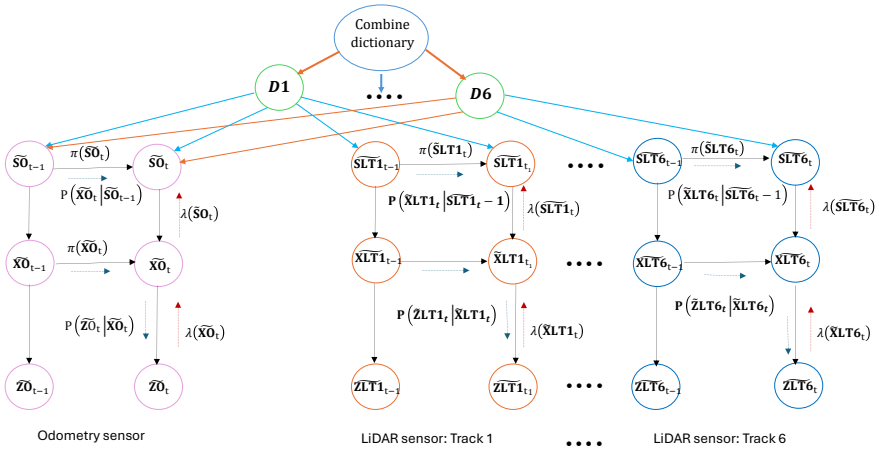


Fig. 5.3 DBN structure during the training phase. Odometry sensor data and LiDAR observations are used to classify and learn tracks.

The DBN in the training phase operates as follows:

1. **Odometry Sensor Integration:** The lower-level nodes in the DBN represent the ego vehicle's state, including its hidden state $\tilde{S}_0$, predicted position $\tilde{X}_0$, and

observed position $\tilde{Z}_0$. These nodes are connected via transition probabilities $P(\tilde{X}_0|\tilde{S}_0)$, modeling the odometry-based motion dynamics.

2. **LiDAR-Based Track Classification:** At the track level, the DBN models individual static tracks, such as $S\tilde{L}T_1$, where $S\tilde{L}T_1$ represents the hidden state of the track and $X\tilde{L}T_1$ represents its observed position. Transition probabilities $P(X\tilde{L}T_1|S\tilde{L}T_1)$ capture the temporal evolution of these static objects in the LiDAR dataset.

3. **Dictionary Fusion:** A higher-level node integrates the outputs of all track-level nodes into a unified *combined dictionary*. This dictionary serves as a probabilistic repository of interactions between the ego vehicle and the classified tracks. It encodes cross-track consistency and enables robust ego position estimations across different scenarios.

This approach ensures that the training phase produces a comprehensive model of the environment, capturing both the static structure and the temporal dynamics of track interactions. The combined dictionary becomes a central component for downstream testing.

DBN Structure During Testing

The testing phase shown in 5.4 is designed to operate without the **odometry sensor**, relying solely on the **LiDAR testing dataset** for ego vehicle localization. The testing DBN builds on the insights learned during training, focusing on matching observed tracks to the pre-classified tracks and using this information to estimate the ego vehicle's position.

The DBN for testing operates as follows:

1. **Track Matching:** Observed tracks in the LiDAR testing dataset are matched with the learned static tracks using the *Bhattacharyya distance*. This step ensures that the six static tracks identified during training are accurately re-associated with their counterparts in the testing environment, which shares a similar structure.

2. **Prediction with MJPF:** Once the tracks are matched, the MJPF predicts the ego vehicle's positions. The MJPF leverages the matched tracks as static references, using the learned motion dynamics and interaction models to estimate the ego state.

3. **Dictionary Fusion for Robust Estimation:** The matched track dictionaries are combined into a unified dictionary, as in the training phase. This fused dictionary incorporates all static tracks and provides probabilistic corrections

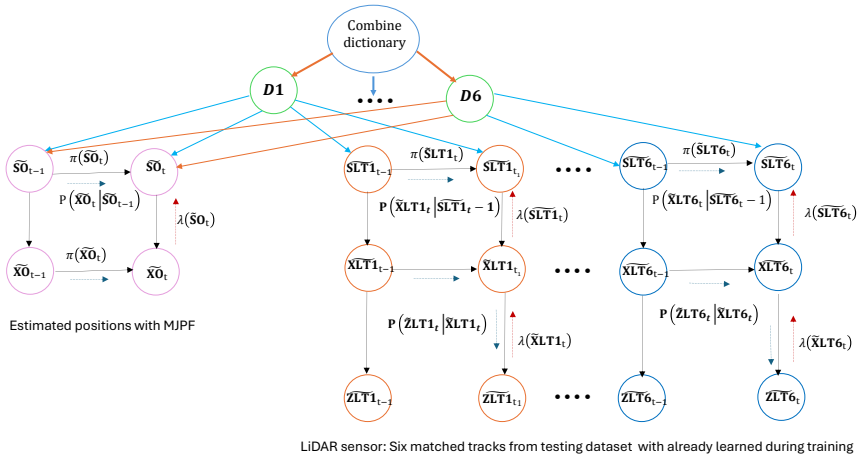


Fig. 5.4 DBN structure during the testing phase. Only LiDAR observations are used to match tracks and estimate ego vehicle positions.

to the MJPF predictions, ensuring that the ego vehicle's trajectory remains consistent with the observed environment.

4. **Final Ego Position Estimation:** The leftmost nodes of the DBN represent the final estimated positions of the ego vehicle. These nodes aggregate probabilistic influences from all matched tracks and the fused dictionary, enabling accurate localization even in the absence of direct odometry input.

Key Differences Between Training and Testing DBNs

- **Sensor Dependencies:** The training DBN utilizes both odometry and LiDAR data, whereas the testing DBN relies solely on LiDAR observations for localization.
- **Role of Tracks:** In the training phase, tracks are classified and learned as static landmarks. In the testing phase, these learned tracks are matched and used as references for localization.

The use of DBNs in both training and testing phases enables a seamless transition from learning to inference. By leveraging multi-sensor data during training and focusing on LiDAR-based localization during testing, the framework demonstrates the ability to achieve robust and accurate ego vehicle localization in dynamic environments. The results of this approach are analyzed in chapter 6, providing a detailed evaluation of the continuous and discrete anomaly measures and their implications for real-world applications.

5.3 Summary

This chapter presented an ego localization framework that leverages MJPF in combination with previously learned interaction vocabularies derived from LiDAR observations,

eliminating the reliance on odometry. Two predictive strategies were introduced: one based on isolated track models, where each static object contributes independently to the ego state estimation, and another utilizing a combined interaction dictionary that aggregates multiple track influences. The track-based approach employs Bhattacharyya distance for matching, followed by particle initialization, motion prediction, and correction using transition matrices specific to each interaction. In contrast, the combined dictionary integrates contextual information through a Dynamic Bayesian Network (DBN), providing enhanced robustness during overlapping interactions or sparse observation periods. The effectiveness of both strategies is further examined in the next chapter through comprehensive evaluations using real-world datasets and controlled interaction scenarios.

Chapter 6

Results and Discussions

6.1 Localization: Results and Discussions Based on Separate Models Predictions

This chapter presents the results obtained using the proposed localization framework, which employs separate MJPF predictions for individual tracks as well as a combined dictionary approach. A comprehensive analysis was conducted to evaluate the interaction dynamics between each track and the ego vehicle, leveraging the respective interaction dictionaries and the learned vocabularies from the offline training phase. The individual MJPF predictions yielded estimated ego vehicle positions, offering critical insights into the accuracy and robustness of the framework. The first part of this analysis assesses the framework's performance in estimating the ego vehicle's trajectory for each track independently. Subsequently, the second part presents the results derived from the combined dictionary approach, highlighting its comparative advantages and limitations.

6.1.1 Experimental Dataset

Experimental dataset comprises LiDAR point clouds collected at the University of Carlos III, Madrid using a real vehicle, iCAB [3]. A straight trajectory was followed by the iCAB while it collected environmental data along the way. A total of 393 frames were collected during the training phase, while 415 frames were collected during the testing phase. In both the training and testing datasets, the starting and ending points, as well as the surrounding environment, were identical. Due to this consistency, the same tracks are reliably identified during the matching process in the MJPF. Furthermore, unlike the training phase, which included odometry data, the testing phase relies solely on the testing dataset to predict the ego vehicle's location without relying on odometry.

6.1.2 Results and Discussion

Analyses of the localization results for each track and their corresponding transition matrices are presented in this section. The timings of the interactions between each track with the ego vehicle are shown in Table 6.1. Note that the y-axis in the error plots is normalized to a maximum of 0.15 for better visualization of error variations across tracks, whereas the mean errors reported in Table 6.2 represent the absolute values in meters. Specifically, the low mean errors as shown in this table confirm the accuracy and robustness of the proposed localization framework based on static tracks. Based on the localization results, it is shown that the ego vehicle positions were accurately predicted using LiDAR-only observations without the use of odometry. Below is an analysis of each track's localization performance:

TrackID 20: Building Localization results for TrackID 20 as shown in Figure 6.1, representing a building on the right side, indicate an accurate match between the predicted trajectory and the ground truth trajectory. As can be seen from the transition matrix in Figure 6.7, which exhibits some off-diagonal values, indicating occasional deviations in the predicted states. Despite the off-diagonal elements, the mean localization error for Track 20 is one of the lowest among all tracks at 0.12 m. This outcome can be attributed to the static nature of the building, providing a stable reference for localization, and the high-quality match between observed and learned clusters. Furthermore, the low mean error with off-diagonal elements suggesting occurance of minor state transitions but they do not effect the overall localization precision.

TrackID 1416: Static Pole A prediction of ego vehicle trajectory for TrackID 1416 shown in Figure 6.2, representing a static pole, is aligned well with ground truth. Static poles provide highly stable and fixed reference points, resulting in a low mean error. This track shows a relatively diagonal-dominant transition matrix with minimal off-diagonal values demonstrated in Figure 6.8. The mean error for this track is 0.18 m. The pole's narrow geometry and limited interaction area can introduce minor prediction inaccuracies, leading to slightly higher errors compared to Track 20. Nonetheless, the consistent diagonal dominance reflects reliable track-to-cluster associations.

TrackID 1682: Tree A tree represented by TrackID 1682 also provides accurate localization results displayed in Figure 6.3. The transition matrix for this track display moderate off-diagonal values shown in Figure 6.9, indicating variability in the observed states during the interactions. However, the mean error for this track is 0.22 m which is higher than TrackID 20 and TrackID1416, but due to their static nature, trees serve as robust landmarks, and the overall mean error remains low.

TrackID 1929: Static Pole The mean error for static pole exibited in Figure 6.4 is higher than all the tracks may be the smaller size of this track but the localization framework successfully predicts the ego vehicle trajectory. Although, moderate off-diagonal values found in its transition matrix (Figure 6.10) but its fixed position helped in predicting the ego vehicle positions.

TrackID 3159 and TrackID 3549: Trees Both Tracks 3159 and 3549 also involve tree interactions with the ego vehicle, with mean errors of 0.15 m and 0.19 m, respectively. Both produce accurate localization results, which can be demonstrated from the Figures 6.5 and 6.6. The predicted ego vehicle trajetory is well aligned with the ground truth trajectory, with minimal mean errors. Their transition matrices (Figures 6.11 and 6.12) reveal relatively consistent diagonal dominance, suggesting that despite the inherent variability of trees, the learned clusters provided a reasonably accurate basis for localization. The slightly lower error for Track 3159 may be due to its more distinct structural features compared to Track 3549. Finally, the transition matrices confirms the effectiveness of static trees as landmarks by confirming smooth state updates.

Table 6.1 Timing Analysis of Track Interactions with Ego Vehicle

Track ID	**Start Time (s)**	**End Time (s)**
Track 14	1.0	14.7
Track 20	0.7	14.5
Track 1416	15.8	18.0
Track 1682	18.0	24.8
Track 1929	20.0	23.4
Track 3159	32.1	33.7
Track 3549	36.0	39.3

Table 6.2 Mean Error Analysis for Different Track IDs

Track ID	**Mean Error (m)**	**Ground Truth Description**
Track 20	0.12	Building
Track 1416	0.18	Static Pole
Track 1682	0.22	Tree
Track 1929	0.26	Static Pole
Track 3159	0.15	Tree
Track 3549	0.19	Tree

The presence of non-zero mean localization errors across tracks may suggest a small bias in the estimator. This is expected and can be attributed to several sources. First, the testing phase relies exclusively on LiDAR data, without odometry updates to correct for cumulative drift. This design choice, while intentional, naturally introduces a fixed spatial offset in some sequences. Second, the Bhattacharyya-based matching process can cause slight localization shifts when the observed track's geometry does not

fully align with its corresponding cluster centroid in the dictionary. Third, temporal misalignments due to LiDAR sparsity and asynchronous object detection can lead to trajectory lag effects [97, 98].

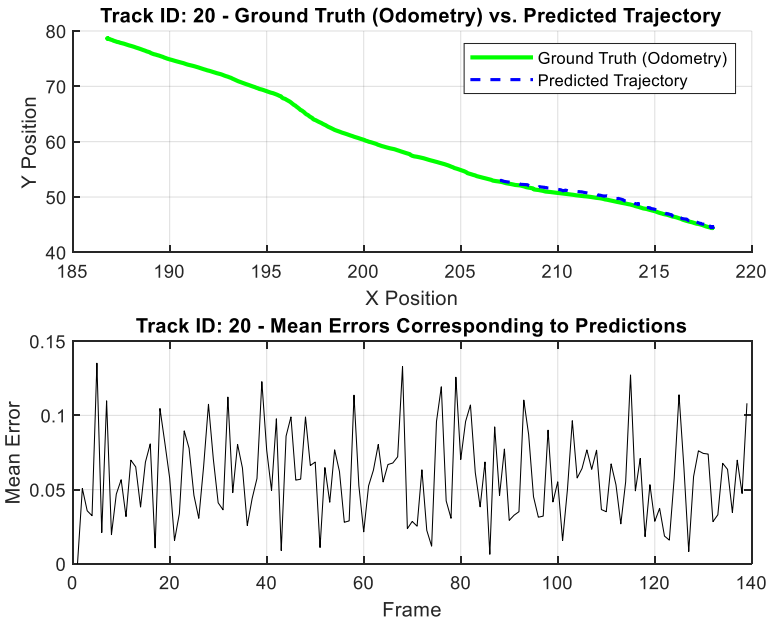


Fig. 6.1 Localization results comparing the predicted ego vehicle trajectory (blue) with the ground truth (green) for track 20

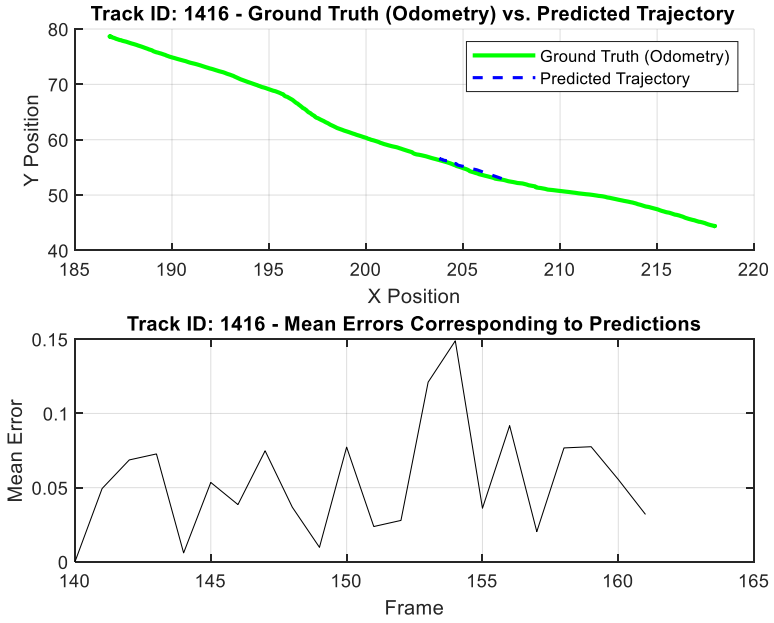


Fig. 6.2 Localization results comparing the predicted ego vehicle trajectory (blue) with the ground truth (green) for track 1416

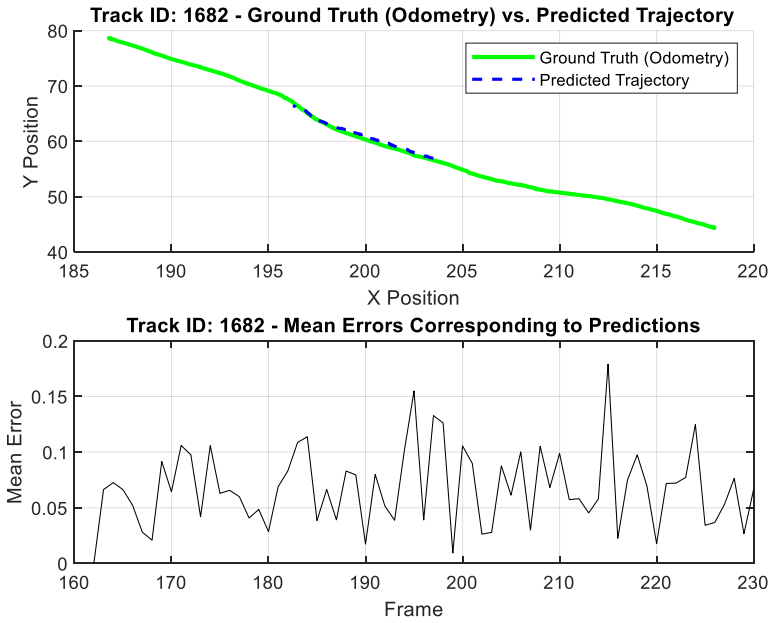


Fig. 6.3 Localization results comparing the predicted ego vehicle trajectory (blue) with the ground truth (green) for track 1682

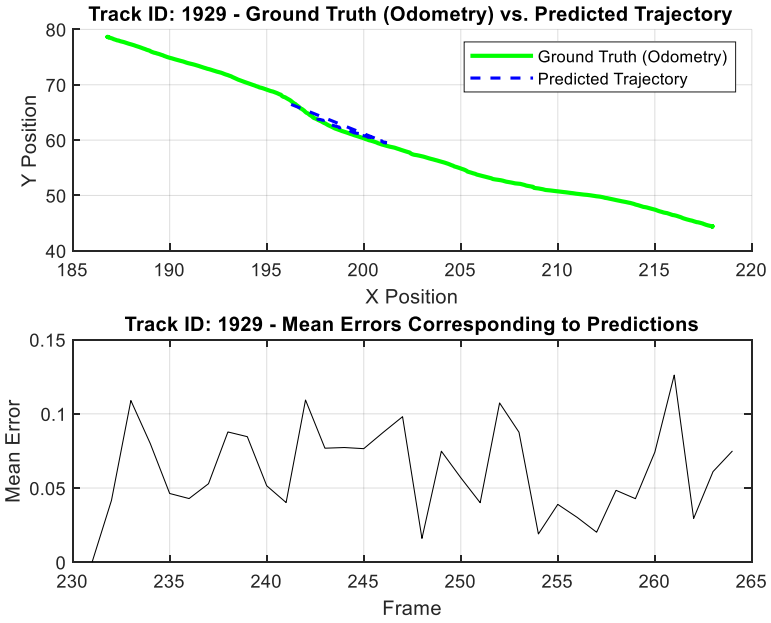


Fig. 6.4 Localization results comparing the predicted ego vehicle trajectory (blue) with the ground truth (green) for track 1929

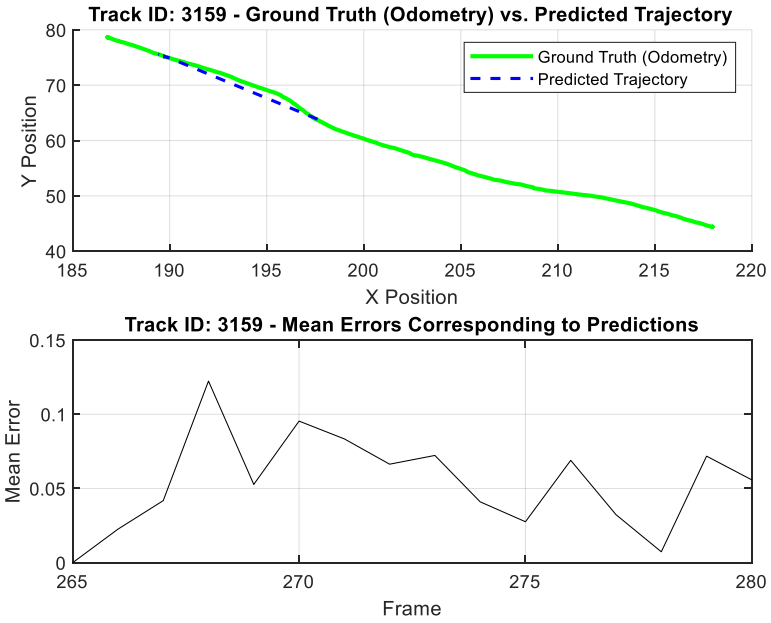


Fig. 6.5 Localization results comparing the predicted ego vehicle trajectory (blue) with the ground truth (green) for track 3159

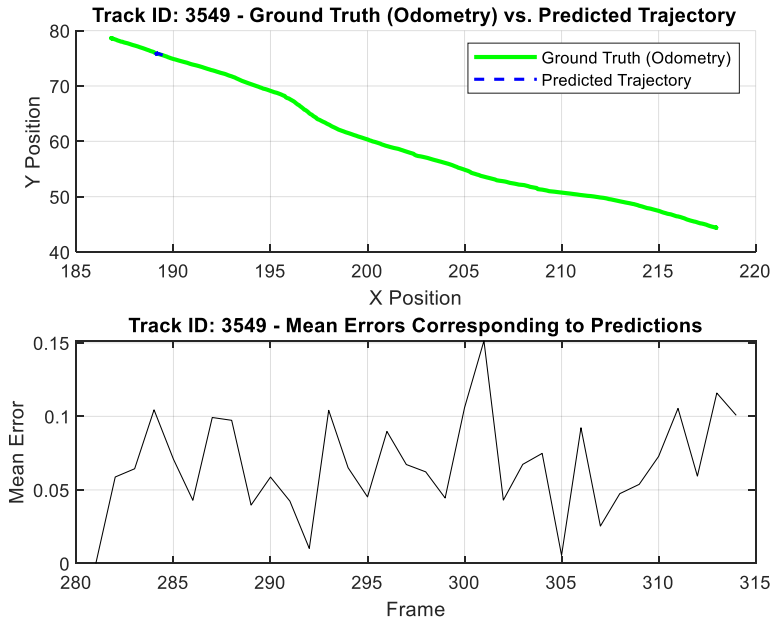


Fig. 6.6 Localization results comparing the predicted ego vehicle trajectory (blue) with the ground truth (green) for track 3549

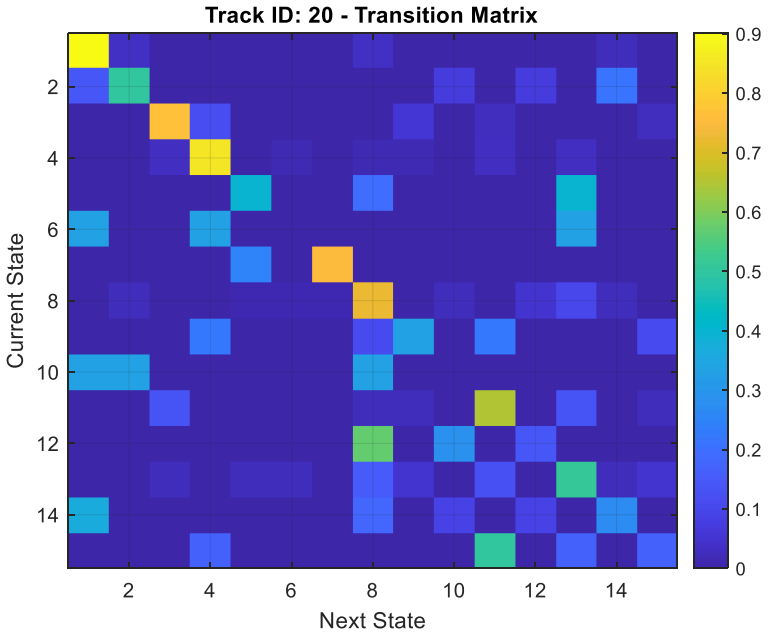


Fig. 6.7 Transition matrix reflecting state predictions achieved using static landmark as building

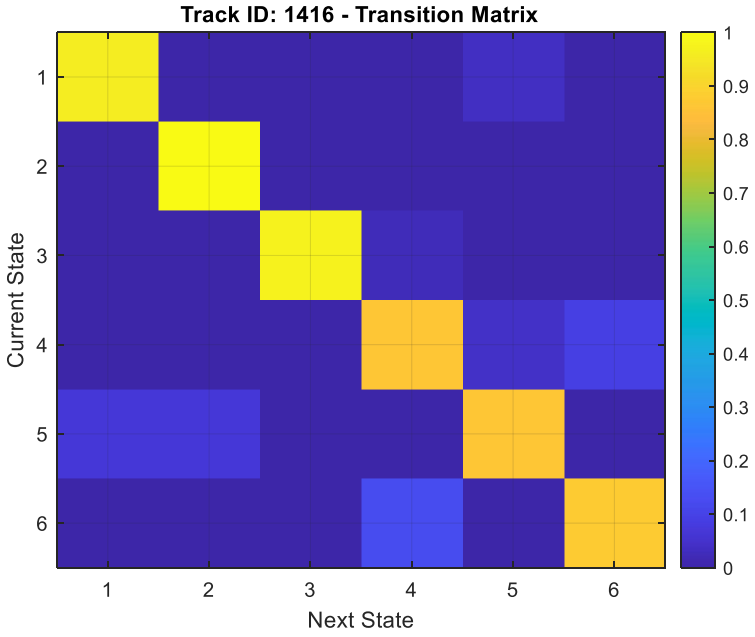


Fig. 6.8 Transition matrix reflecting state predictions achieved using static landmark as pole

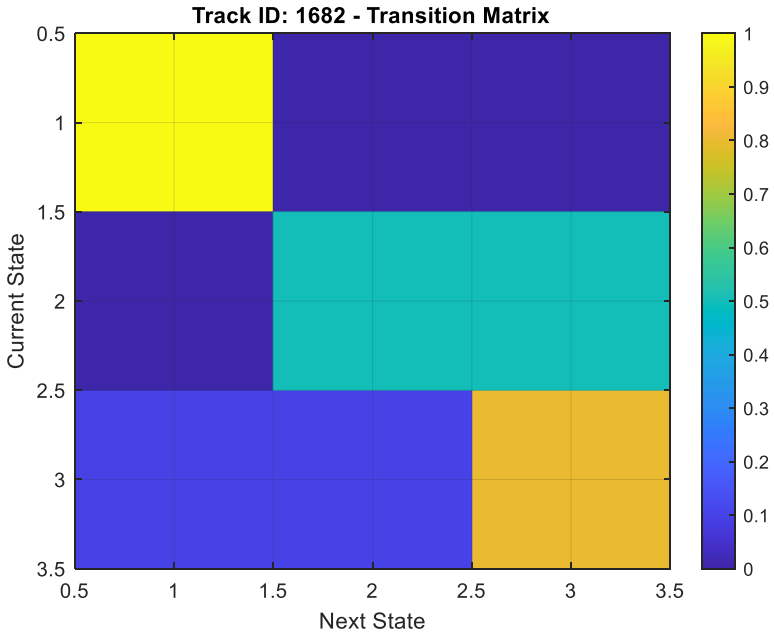


Fig. 6.9 Transition matrix reflecting state predictions achieved using static landmark as tree

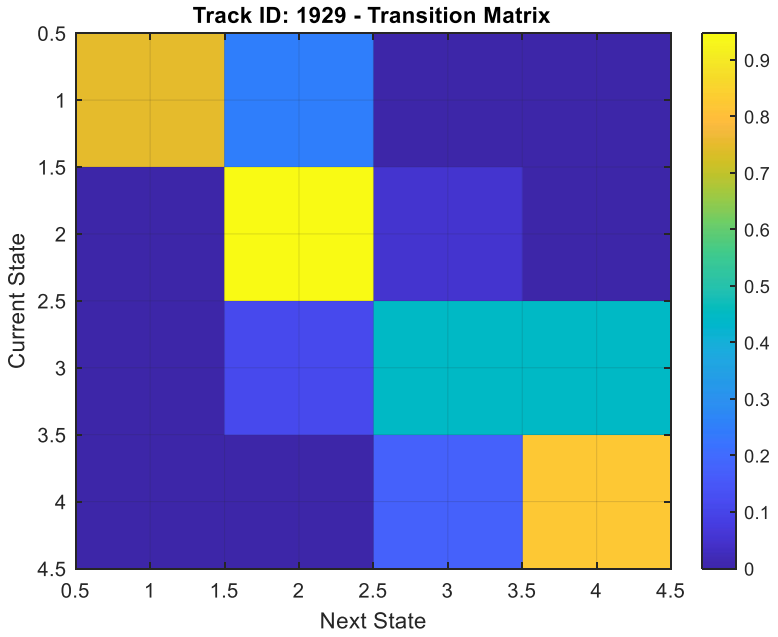


Fig. 6.10 Transition matrix reflecting state predictions achieved using static landmark as pole

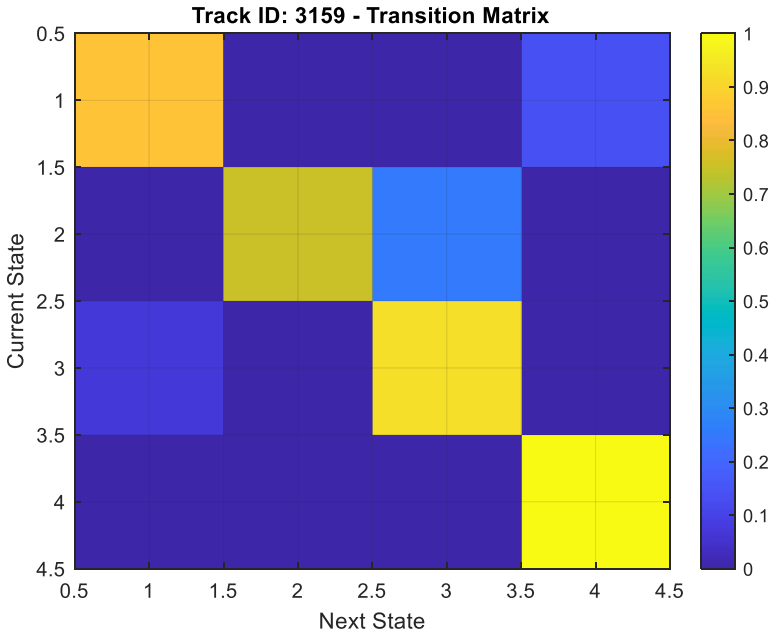


Fig. 6.11 Transition matrix reflecting state predictions achieved using static landmark as tree

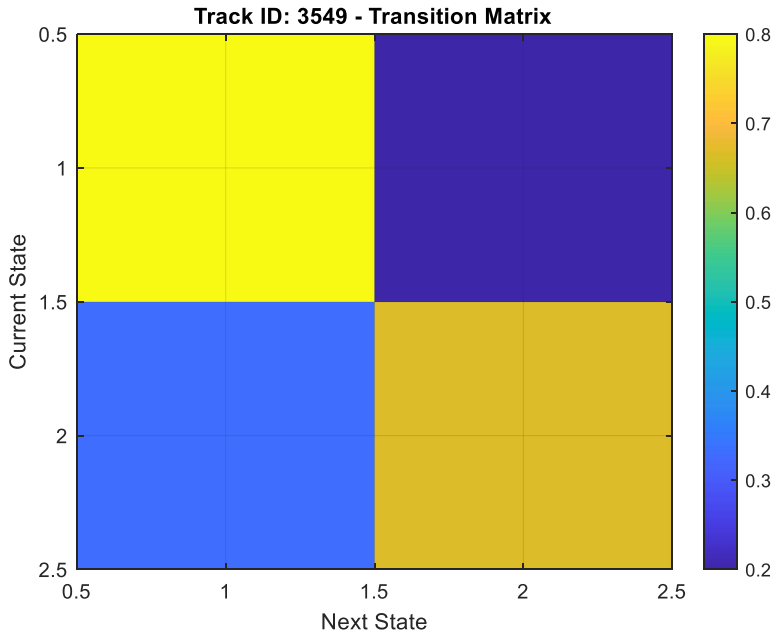


Fig. 6.12 Transition matrix reflecting state predictions achieved using static landmark as tree

6.2 Results and Analysis based on Combined Dictionary Predictions

In this section, we present the results of our extended MJPF with the *combined dictionary* and analyze the performance under the following scenarios:

- **Simultaneous Observations** of multiple static tracks,
- **No Observation** periods,
- **Isolated Track** updates,
- **Continuous And Discrete Level Anomalies** over time.

Simultaneous Observations

Figure 6.13 illustrates the ego-vehicle trajectory estimates (red dashed line) versus ground truth (blue line) when the AV observes up to six static tracks: one building, two poles, and three trees at different time intervals. Morevoer, in Figure 6.14 different predicted states scenarios are shown in different colors. Notice that:

- The *combined* MJPF trajectory (red dashed) closely aligns with ground truth, showing smaller lateral drift compared to no observation periods.
- Off-diagonal components in the *combined* transition matrix shown in Figure 6.15 remain minimal, indicating the dictionary is well-specified and synergy among tracks is consistent.

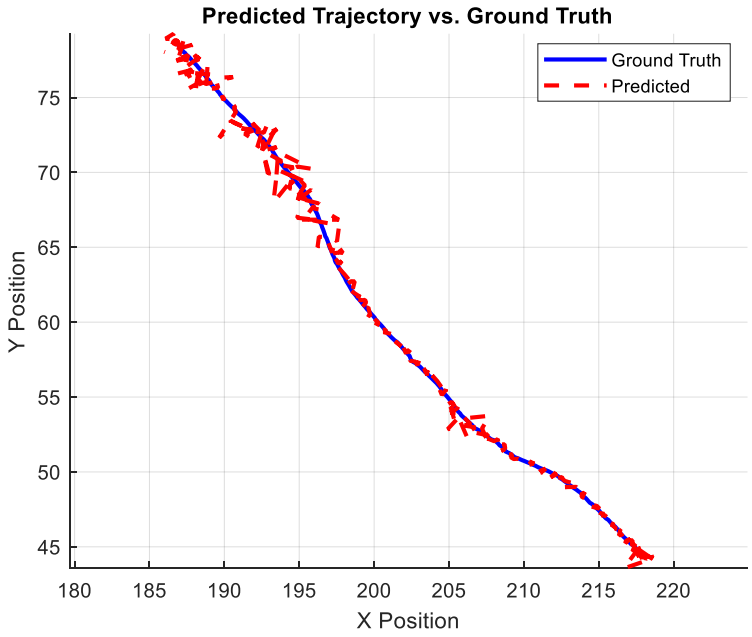


Fig. 6.13 Localization result of comparing the predicted ego vehicle trajectory (red) with the ground truth (blue)

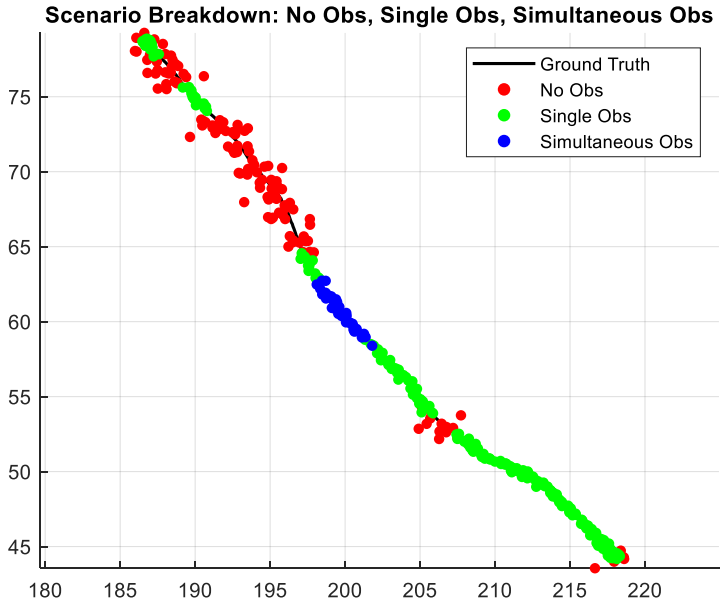


Fig. 6.14 Localization result of showing various scenarios in different colors

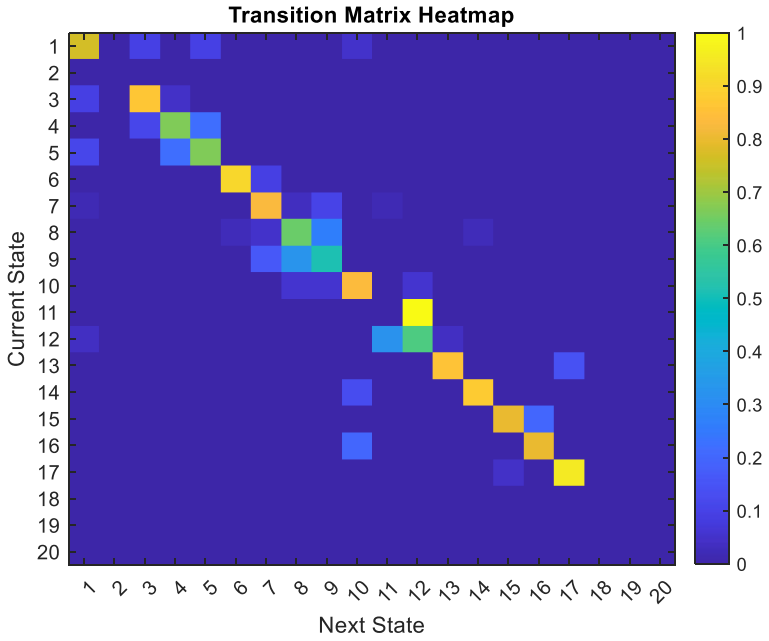


Fig. 6.15 Transition matrix with different states

No-Observation Periods

During certain intervals, no new LiDAR updates are received due to sensor noise. The AV relies on predicted states and the prior dictionary knowledge. We observe:

- A slight growth in localization error, but it remains bounded due to the learned transition matrices from multiple tracks (i.e., having more dictionary references helps maintain stable predictions).
- Once new observations become available, the ego-vehicle state quickly snaps back to align with ground truth.

Isolated Track Interactions

Even if only *one* track is visible, the combined dictionary approach is consistent with the single-dictionary approach. However, subtle differences arise because the combined dictionary can cross-validate the state transitions gleaned from *all* tracks, slightly reducing the risk of outlier updates from a single, possibly ambiguous track.

Continuous Level Anomalies

Looking at the continuous-level anomaly plot (Figure 6.16) over 41.5 seconds of testing data, we can observe:

(a) Initial High Peak (near 0–2,s): Right at the start, the anomaly rises above 2.0. At this stage, the MJPF has just begun predicting track states from limited initialization, and the LiDAR observations of those tracks (our only available sensor

data) do not yet align with the filter's estimates. As soon as enough measurements accumulate, the filter corrects its internal states, causing the anomaly to drop below 0.5 by around 2–9,s.

(b) Major Spike at 15,s (Frame 150): During the initial frames, both the ego vehicle and the front car stay nearly motionless, causing only minor deviations from the ground truth and thus minimal continuous-state anomalies. Although no new LiDAR returns arrive for a short period, the MJPF still predicts the front car's position using its previously learned motion model and the six static-track dictionaries obtained during training. In many cases, these learned static references help constrain the filter's state-space enough to prevent dramatic drift, even without direct observations.

However, around frame 150 (approximately 15,s), both vehicles start moving, and the ego vehicle attempts a close overtake of the front car. At this point, newly received LiDAR returns show the front car at a location that diverges significantly from the MJPF's predicted state. The sudden reintroduction of observations—contrasting with the filter's prior expectation—produces a pronounced mismatch, causing a sharp peak in the continuous-state anomaly measure. Essentially, the abrupt transition from a no-observation or low-motion scenario into a fast, near-overtaking maneuver causes the filter's prediction to be rapidly "corrected" by the sensor data, pushing the anomaly above 2.

(c) Subsequent Peaks at 28,s and 33,s: These coincide with additional no-observation intervals in the testing dataset. The filter again drifts while predicting states in the absence of LiDAR updates. Then, upon receiving new measurements of the track positions, a large discrepancy emerges, causing notable spikes in the continuous-level anomaly. Each time the MJPF must "snap back" to actual measurements after drifting, the anomaly measures briefly jump.

Overall, these continuous-level anomalies reflect how the MJPF's predicted positions for each LiDAR-derived track (our "generalized states") differ from the actual measurements. Large spikes typically mark transitions from no-observation to observation or our ego vehicle nearly overtakes closely another car.

Discrete Level Anomalies

Figure 6.17 shows discrete-level anomalies (the y-axis is typically below 0.3–0.4 but spikes whenever the predicted superstate distribution differs sharply from observations), we see:

(a) Low Baseline (near 0–14,s): Early in testing, KLDA hovers around 0.02–0.03, indicating that the filter's internal mapping of the observed track states to the combined dictionary's clusters remains fairly consistent. Even though the testing dataset

only provides track positions (not cluster labels), the filter internally assigns these measurements to the most likely cluster. Minimal divergence suggests good alignment in this initial phase.

(b) Significant Jump at 15,s: When our ego vehicle overtakes closely another car, the MJPF's discrete predictions (i.e., which cluster or mode each track belongs to) must adjust rapidly. The incoming LiDAR data shows a track in a state that does not match the filter's prior cluster assignment, causing KLDA to jump above 0.2. This spike reveals that not only are the continuous states off, but also the filter's discrete labeling (cluster choice) needs a sudden reallocation.

(c) Spikes at 28,s and 33,s: As with the continuous-level measure, these occur after no-observation gaps. Once observations resume, the filter discovers that its predicted discrete state for the track is incorrect (or far less likely) compared to the newly observed positions. It must then transition abruptly to a different cluster assignment, producing discrete-level anomaly peaks.

Throughout the 41.5 seconds test sequence, KLDA remains low whenever the filter's discrete assignments match the observed track states, but it spikes sharply at moments of reintroduction or dynamic-track interference—indicating a high mismatch between predicted and actual superstate distributions.

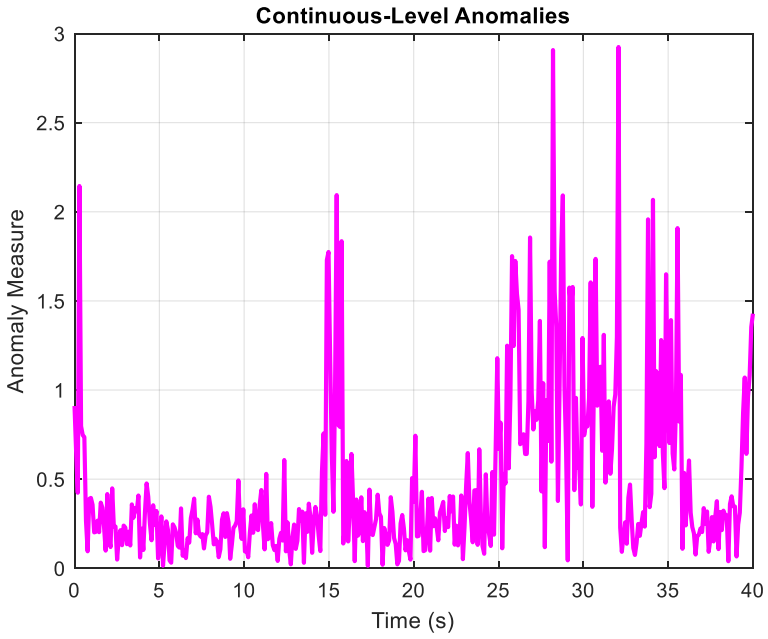


Fig. 6.16 Continuous states anomalies over time

The final two figures i.e. Figure 6.18 and 6.19, present distributions of continuous and discrete anomalies across the three scenarios (NoObs, SingleObs, and MultiObs). As a result of these final plots, it is clear how each scenario affects the overall deviation from ground truth. There is a large spread and a high median value in the first box

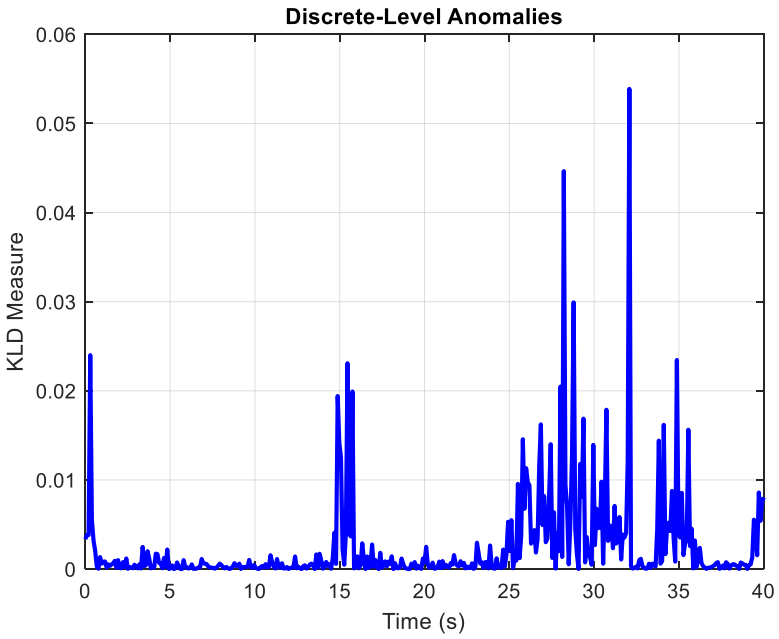


Fig. 6.17 Discrete states anomalies over time

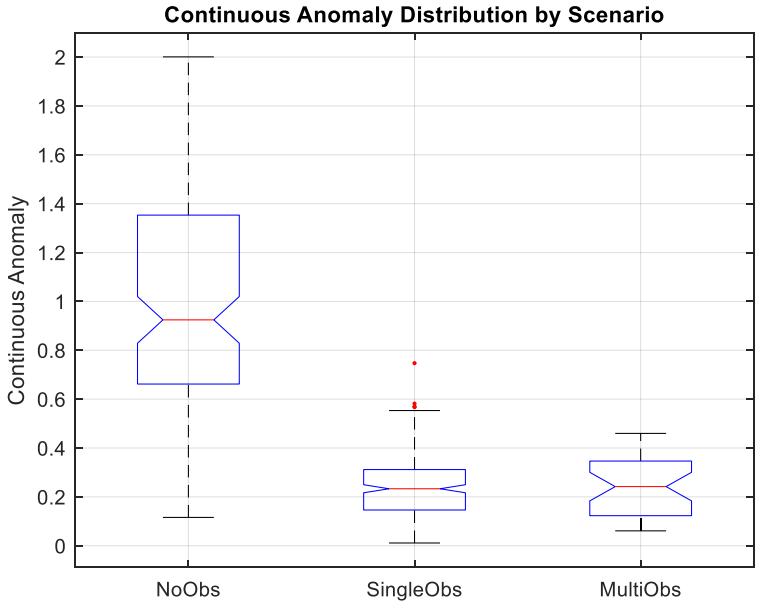


Fig. 6.18 Box plot of continuous anomalies

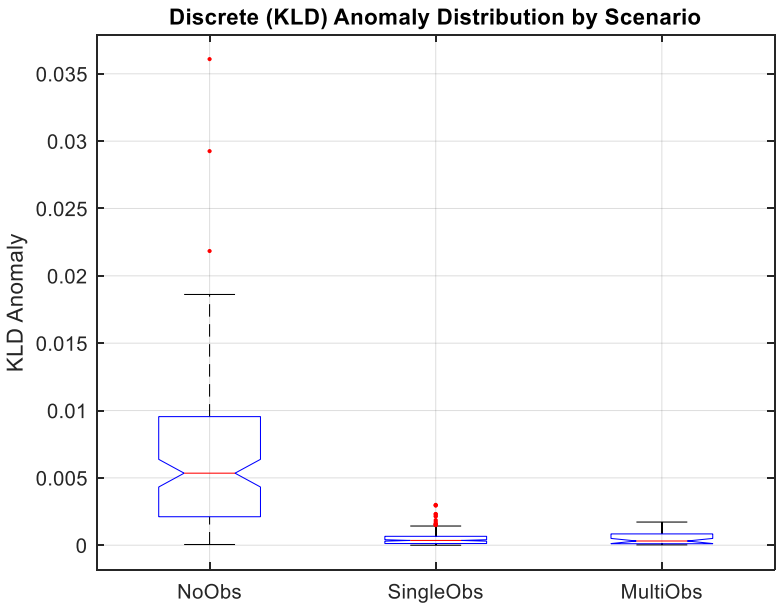


Fig. 6.19 Box plot of discrete anomalies

plot (Continuous Anomaly), indicating that times without any track observations have a considerable deviation from the ground truth. As a result of these extended gaps, the filter is forced to propagate its estimates without new sensor feedback, often leading to increased drift. Alternatively, the categories "SingleObs" and "MultiObs" remain more tightly clustered near lower anomaly values, indicating that even a single track (or multiple tracks) can constrain the filter's predictions and keep ego vehicle estimates more accurate.

A similar pattern can be observed in the second box plot (Discrete or KLD Anomaly): "NoObs" intervals produce larger dispersion of anomaly values, including several outliers well above the typical value. Without any direct updates to the track, the probabilistic distribution of the ego-vehicle state and track assignments drifts more noticeably, resulting in a drastic mismatch between predicted and actual states when measurements resume. Moreover, both "SingleObs" and "MultiObs" scenarios show comparatively smaller box widths and lower median anomalies, suggesting that even intermittent track information contributes to the model's ability to maintain more accurate discrete-state assignments over time.

6.3 Computational Efficiency Analysis

To assess the practical feasibility of the proposed localization framework, we conducted a detailed evaluation of its runtime performance on a standard non-GPU-enabled system. The implementation was executed on a laptop equipped with an Intel Core i7-7500U CPU @ 1.80 GHz, 16 GB RAM, and running Windows 11 Home (64-bit).

Unlike image-based localization approaches, our system operates on LiDAR point cloud frames, where each frame corresponds to a 3D scan obtained from a Velodyne VLP-16 sensor. The average computation time for processing a single point cloud frame during the online testing phase was measured to be approximately 41 milliseconds, enabling a processing rate of around 24 frames per second (FPS). This confirms the potential of our framework for near real-time autonomous vehicle localization tasks.

To ensure computational efficiency, the following optimizations were applied:

- Cluster Matching Optimization: Only the most relevant clusters are matched using Bhattacharyya distance, significantly reducing the number of comparisons during track association.
- Selective Kalman Updates: Kalman filter prediction and update steps are performed exclusively for those clusters actively involved in interactions, avoiding unnecessary computations.
- Efficient MJPF Design: The Markov Jump Particle Filter marginalizes linear components within the state space, which reduces the number of particles and accelerates inference.

In terms of memory scalability, the interaction dictionaries grow linearly with the number of observed tracks and frames in a given environment. Each dictionary entry encodes one frame-level interaction, making the size of the dictionary proportional to the temporal span and the number of interacting static objects. For example, in our current setup, combining six track dictionaries results in a total memory footprint of less than 15 MB. During testing, only the relevant time slices are queried from these pre-indexed dictionaries, ensuring low-latency access. Thus, even when scaled to more complex scenes with dozens of static landmarks, the structure remains computationally efficient. This is due to the selective querying and use of spatial hashing for dictionary access, supporting real-time performance in extended urban driving scenarios.

These design strategies ensure that the proposed approach remains computationally feasible even on resource-constrained hardware, and can be potentially extended to embedded systems with comparable performance capabilities.

6.4 Ablation Study of Core Components

To evaluate the importance of individual components within the proposed localization framework, we conducted an ablation study by progressively removing or altering key modules. Each ablation scenario was tested under identical conditions using the LiDAR-based point cloud testing dataset described earlier (415 frames). The average localization error was recorded for each setting and compared to the full system.

The results in Table 6.3 show that both the MJPF and the combined dictionary substantially enhance localization accuracy (the best result is highlighted in bold). Removing the MJPF leads to increased error due to the lack of stochastic reasoning across cluster transitions, while omitting the combined dictionary weakens the system's ability to handle simultaneous track observations and sensor dropout scenarios. The highest error occurs when both components are disabled, confirming their combined necessity.

Table 6.3 Ablation study results showing mean localization error (in meters) for different configurations of the framework.

Method	**Mean Error (m)**
Full Framework (MJPF + Combined Dictionary)	**0.17**
Without Combined Dictionary (Single Track Only)	0.23
Without MJPF (Kalman Only)	0.29
Without Dictionary (Odometry Only, No LiDAR)	0.36

These findings reinforce the robustness and necessity of our proposed system architecture in ensuring reliable localization across diverse real-world scenarios.

6.5 Summary

This chapter demonstrated the accuracy and robustness of the proposed MJPF-based localization framework using only LiDAR-derived static track interactions. Both individual and combined dictionary models showed strong performance across varied scenarios, including sparse observations and simultaneous track interactions. The results confirmed the system's resilience and adaptability, supported by anomaly analysis. The final chapter reflects on the key contributions of this work and outlines potential directions for future research.

Chapter 7

Conclusion and Future Works

7.1 Conclusion

AVs represent a transformative technological advancement in artificial intelligence (AI) and intelligent transportation systems. Ensuring the precise localization of an AV in dynamic environments remains a critical challenge, particularly in GPS-denied areas [99]. This thesis has presented a LiDAR-based classification and localization framework that integrates interaction dictionaries, probabilistic graphical models, and clustering techniques to provide accurate, real-time state estimation for AVs.

By leveraging MJPF and DBNs, the proposed framework effectively captures the interactions between the ego vehicle and static or dynamic tracks, modeling their transition states probabilistically. This research provides a novel approach to track classification and trajectory prediction, optimizing localization accuracy through interaction-aware state estimation.

This work makes several key contributions:

- Track Classification and Clustering: The implementation of GNG clustering has enabled the precise classification of static and dynamic tracks. This classification process enhances the ego vehicle's ability to utilize stable environmental features for improved localization.
- Interaction Dictionaries for Localization: This thesis has introduced a methodology for encoding and leveraging track-ego vehicle interaction data. These interaction dictionaries enhance transition modeling, capturing spatiotemporal correlations between different tracks and improving localization robustness.
- Integration of Probabilistic Graphical Models: DBNs were utilized to model the dependencies between track observations and vehicle state transitions, allowing for accurate inference of the ego vehicle's position [100].
- MJPF-Based Localization: The use of MJPF enables the system to switch dynamically between multiple interaction states while maintaining probabilistic

state estimations. This allows for robust localization even in the presence of missing observations.

- Anomaly Detection and State Validation: The application of Bhattacharyya Distance for track matching, along with the use of continuous level and discrete level anomalies, has improved the framework's ability in refining ego vehicle state predictions.
- Comparative Analysis of Separate and Combined Track Predictions: The framework was tested with both separate track-based models and a combined dictionary approach. The combined approach demonstrated superior localization accuracy by leveraging multiple track interactions simultaneously.

The experimental results, obtained using the UC3M dataset, validated the effectiveness of this methodology. The integration of LiDAR-based perception with probabilistic state estimation techniques has significantly enhanced the localization capabilities of AVs, providing a scalable and adaptable solution for autonomous navigation.

7.2 Limitations and Deployment Considerations

While the proposed framework demonstrates strong performance in controlled conditions, certain limitations should be acknowledged with respect to real-world deployment. The approach assumes the availability of reliable static tracks for localization, which may not always be guaranteed in dense urban settings or highly dynamic environments. Moreover, the current experimental setup does not explicitly address scenarios involving partial or full occlusion of key landmarks, nor does it account for sensor degradation due to adverse weather conditions such as rain, fog, or snow. These factors, although outside the scope of the present study, could impact the robustness and scalability of the framework in practical applications and thus represent important areas for future investigation.

7.3 Future Works

While this thesis has introduced a robust LiDAR-based localization methodology, several key research directions remain open for exploration:

- Validation with Open-Source Datasets: Future work will involve evaluating the framework using additional publicly available datasets (e.g., KITTI, Waymo, NuScenes) to benchmark its performance and generalizability under different driving conditions [101].

- Adaptive Transition Matrices: Incorporating reinforcement learning techniques to enable real-time adaptation of transition matrices can enhance the system's ability to dynamically model track interactions [102].
- Multi-Agent Localization: Extending the proposed methodology to multi-vehicle scenarios will be explored, enabling collaborative localization through vehicle-to-vehicle (V2V) communication for improved state estimation in dense traffic environments [103].
- Deep Learning for Enhanced Feature Extraction: Investigating convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to enhance feature extraction from LiDAR data could improve classification accuracy and anomaly detection [16, 104, 105].
- Sensor Fusion for Robust Localization: Combining LiDAR data with radar, cameras, and global navigation satellite system (GNSS) to develop a multi-modal localization framework can improve accuracy in challenging environmental conditions, such as fog or rain [106].
- Real-Time Deployment and Computational Optimization: Implementing the proposed model in a real-time embedded system and optimizing the computational efficiency of MJPF-based localization will be essential for its practical deployment in AVs.

By addressing these future research challenges, the proposed framework can be extended to enhance its applicability in real-world autonomous driving scenarios. These improvements will contribute towards the broader goal of developing reliable, scalable, and safe AV localization systems for next-generation intelligent transportation networks.

References

[1] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[2] Yong Shean Chong and Yong Haur Tay. Abnormal event detection in videos using spatiotemporal autoencoder. In *Advances in Neural Networks-ISNN 2017: 14th International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan, June 21–26, 2017, Proceedings, Part II 14*, pages 189–196. Springer, 2017.

[3] Sheida Nozari, Ali Krayani, Pablo Marin, Lucio Marcenaro, David Martin Gomez, and Carlo Regazzoni. Modeling autonomous vehicle responses to novel observations using hierarchical cognitive representations inspired active inference. *Computers*, 13(7):161, 2024.

[4] Iuliia Kotseruba and John K Tsotsos. 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, 53(1):17–94, 2020.

[5] Alain Morin. Levels of consciousness and self-awareness: A comparison and integration of various neurocognitive views. *Consciousness and cognition*, 15(2):358–371, 2006.

[6] Nikil Dutt, Carlo S Regazzoni, Bernhard Rinner, and Xin Yao. Self-awareness for autonomous systems. *Proceedings of the IEEE*, 108(7):971–975, 2020.

[7] Debasis Kumar and Naveed Muhammad. A survey on localization for autonomous vehicles. *IEEE Access*, 11:115865–115883, 2023.

[8] Ronny Blum, Dominik Dötterböck, and Thomas Pany. Investigation of the vulnerability of mobile networks against spoofing attacks on their gnss timing-receiver and developing a meaconing protection. In *Proceedings of the 2019 International Technical Meeting of The Institute of Navigation*, pages 345–362, 2019.

[9] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of field robotics*, 37(3):362–386, 2020.

[10] Le Chang, Xiaoji Niu, Tianyi Liu, Jian Tang, and Chuang Qian. Gnss/ins/lidar-slam integrated navigation system based on graph optimization. *Remote Sensing*, 11(9):1009, 2019.

[11] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

[12] Jessica Van Brummelen, Marie O'brien, Dominique Gruyer, and Homayoun Najjaran. Autonomous vehicle perception: The technology of today and tomorrow. *Transportation research part C: emerging technologies*, 89:384–406, 2018.

[13] Andreas Schütz, Daniela E Sánchez-Morales, and Thomas Pany. Precise positioning through a loosely-coupled sensor fusion of gnss-rtk, ins and lidar for autonomous driving. In *2020 IEEE/ION position, location and navigation symposium (PLANS)*, pages 219–225. IEEE, 2020.

[14] Li Tang, Yunpeng Shi, Qing He, Adel W Sadek, and Chunming Qiao. Performance test of autonomous vehicle lidar sensors under different weather conditions. *Transportation research record*, 2674(1):319–329, 2020.

[15] Xieyuanli Chen. *LiDAR-Based Semantic Perception for Autonomous Vehicles*. PhD thesis, Universitäts-und Landesbibliothek Bonn, 2022.

[16] Feiya Li, Chunyun Fu, Dongye Sun, Jian Li, and Jianwen Wang. Sd-slam: A semantic slam approach for dynamic scenes based on lidar point clouds. *Big Data Research*, 36:100463, 2024.

[17] Xuebo Tian, Zhongyang Zhu, Junqiao Zhao, Gengxuan Tian, and Chen Ye. Dl-slot: Dynamic lidar slam and object tracking based on collaborative graph optimization. *arXiv preprint arXiv:2212.02077*, 2022.

[18] Muhammad Adnan, Pamela Zontone, Lucio Marcenaro, David Martín Gómez, and Carlo Regazzoni. Classifying static and dynamic tracks for lidar-based navigation of autonomous vehicle systems. In *2024 9th International Conference on Frontiers of Signal Processing (ICFSP)*, pages 111–117. IEEE, 2024.

[19] Tianrui Liu, Changxin Xu, Yuxin Qiao, Chufeng Jiang, and Jiqiang Yu. Particle filter slam for vehicle localization. *arXiv preprint arXiv:2402.07429*, 2024.

[20] Wael Farag. Real-time autonomous vehicle localization based on particle and unscented kalman filters. *Journal of Control, Automation and Electrical Systems*, 32(2):309–325, 2021.

[21] Zhongmou Dai, Zhiwei Guan, Qiang Chen, Yi Xu, and Fengyi Sun. Enhanced object detection in autonomous vehicles through lidar—camera sensor fusion. *World Electric Vehicle Journal*, 15(7), 2024.

[22] Ba-ngu Vo, Mahendra Mallick, Yaakov Bar-Shalom, Stefano Coraluppi, Richard Osborne, Ronald Mahler, and Ba-tuong Vo. Multitarget tracking. *Wiley encyclopedia of electrical and electronics engineering*, (2015), 2015.

[23] Yiming Zeng, Yu Hu, Shice Liu, Jing Ye, Yinhe Han, Xiaowei Li, and Ninghui Sun. Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving. *IEEE Robotics and Automation Letters*, 3(4):3434–3440, 2018.

[24] Biruk Habtemariam, Ratnasingham Tharmarasa, Thayaparan Thayaparan, Mahendra Mallick, and Thia Kirubarajan. A multiple-detection joint probabilistic data association filter. *IEEE Journal of Selected Topics in Signal Processing*, 7(3):461–471, 2013.

[25] Su Pang and Hayder Radha. Multi-object tracking using poisson multi-bernoulli mixture filtering for autonomous vehicles. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7963–7967. IEEE, 2021.

[26] Abhijit Sinha, Zhen Ding, Thia Kirubarajan, and Mohamad Farooq. Track quality based multitarget tracking approach for global nearest-neighbor association. *IEEE Transactions on Aerospace and Electronic Systems*, 48(2):1179–1191, 2012.

[27] Ángel F García-Fernández, Lennart Svensson, Jason L Williams, Yuxuan Xia, and Karl Granström. Trajectory multi-bernoulli filters for multi-target tracking based on sets of trajectories. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2020.

[28] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A review of machine learning techniques for processing multimedia content*, 1(2004):9–16, 2004.

[29] Yiming Zhao, Xiao Zhang, and Xinming Huang. A technical survey and evaluation of traditional point cloud clustering methods for lidar panoptic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2464–2473, 2021.

[30] Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.

[31] Wenjie Ouyang, Yiwen Jiao, Yang Liu, Yang Li, Manjiang Hu, and Hongmao Qin. Hierarchical clustering-based state grouping reinforcement learning for switching decision of autonomous vehicles. In *2023 IEEE International Conference on Unmanned Systems (ICUS)*, pages 1375–1380. IEEE, 2023.

[32] Abrham Shiferaw Alemaw, Giulia Slavic, Hafsa Iqbal, Lucio Marcenaro, David Martin Gomez, and Carlo Regazzoni. A data-driven approach for the localization of interacting agents via a multi-modal dynamic bayesian network framework. In *2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8. IEEE, 2022.

[33] Ambati Pravallika, Mohammad Farukh Hashmi, and Aditya Gupta. Deep learning frontiers in 3d object detection: A comprehensive review for autonomous driving. *IEEE Access*, 2024.

[34] Morteza Saadati, Sayyed Majid Mazinani, Ali Akbar Khazaei, and Seyyed Javad Seyyed Mahdavi Chabok. Energy efficient clustering for dense wireless sensor network by applying graph neural networks with coverage metrics. *Ad Hoc Networks*, 156:103432, 2024.

[35] Jianing Shen and Yang Zhou. Accurate and real-time object detection in crowded indoor spaces based on the fusion of dbscan algorithm and improved yolov4-tiny network. *Journal of Intelligent Systems*, 32(1):20220268, 2023.

[36] Kevin P Murphy. *Probabilistic machine learning: an introduction*. MIT press, 2022.

[37] Giulia Slavic, Mohamad Baydoun, Damian Campo, Lucio Marcenaro, and Carlo Regazzoni. Multilevel anomaly detection through variational autoencoders and bayesian models for self-aware embodied agents. *IEEE Transactions on Multimedia*, 24:1399–1414, 2021.

[38] Zoubin Ghahramani. Learning dynamic bayesian networks. *International School on Neural Networks, Initiated by IIASS and EMFCSC*, pages 168–197, 1997.

[39] Joshua Robinson and Alexander Hartemink. Non-stationary dynamic bayesian networks. *Advances in neural information processing systems*, 21, 2008.

[40] Tiago Leao, Sara C Madeira, Marta Gromicho, Mamede de Carvalho, and Alexandra M Carvalho. Learning dynamic bayesian networks from time-dependent and time-independent data: Unraveling disease progression in amyotrophic lateral sclerosis. *Journal of Biomedical Informatics*, 117:103730, 2021.

[41] Arman Asgharpoor Golroudbari and Mohammad Hossein Sabour. Recent advancements in deep learning applications and methods for autonomous navigation: A comprehensive review. *arXiv preprint arXiv:2302.11089*, 2023.

[42] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[43] Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

[44] Sungwoo Park, Frank Pfenning, and Sebastian Thrun. A probabilistic language based upon sampling functions. *ACM SIGPLAN Notices*, 40(1):171–182, 2005.

[45] Youngjoo Kim and Hyochoong Bang. Introduction to kalman filter and its applications. In *Introduction and implementations of the Kalman filter*. IntechOpen, 2018.

[46] Shishan Yang and Marcus Baum. Extended kalman filter for extended object tracking. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4386–4390. IEEE, 2017.

[47] Himali Singh, Kumar Vijay Mishra, and Arpan Chattopadhyay. Inverse unscented kalman filter. *IEEE Transactions on Signal Processing*, 2024.

[48] Sanat Biswas. *Computationally Efficient Non-linear Kalman Filters for On-board Space Vehicle Navigation*. PhD thesis, University of New South Wales, Sydney, Australia, 2017.

[49] Hafsa Iqbal, Damian Campo, Pablo Marin-Plaza, Lucio Marcenaro, David Martín Gómez, and Carlo Regazzoni. Modeling perception in autonomous vehicles via 3d convolutional representations on lidar. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):14608–14619, 2021.

[50] Mark Lloyd Pupilli. *Particle filtering for real-time camera localisation*. PhD thesis, Citeseer, 2006.

[51] Mohammad El Yabroudi, Khalfalla Awedat, Rakan C Chabaan, Osama Abudayyeh, and Ikhlas Abdel-Qader. Adaptive dbscan lidar point cloud clustering for autonomous driving applications. In *2022 IEEE International Conference on Electro Information Technology (eIT)*, pages 221–224. IEEE, 2022.

[52] Caihong Li, Feng Gao, Xiangyu Han, and Bowen Zhang. A new density-based clustering method considering spatial distribution of lidar point cloud for object detection of autonomous driving. *Electronics*, 10(16):2005, 2021.

[53] Feng Gao, Caihong Li, and Bowen Zhang. A dynamic clustering algorithm for lidar obstacle detection of autonomous driving system. *IEEE Sensors Journal*, 21(22):25922–25930, 2021.

[54] You Li, Clément Le Bihan, Txomin Pourtau, and Thomas Ristorcelli. Insclustering: Instantly clustering lidar range measures for autonomous vehicle. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[55] Xuebin Sun, Han Ma, Yuxiang Sun, and Ming Liu. A novel point cloud compression algorithm based on clustering. *IEEE Robotics and Automation Letters*, 4(2):2132–2139, 2019.

[56] Caihong Zhang, Shaoping Wang, Biao Yu, Bichun Li, and Hui Zhu. A two-stage adaptive clustering approach for 3d point clouds. In *2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 11–16. IEEE, 2019.

[57] Haozhe Yang, Zhiling Wang, Linglong Lin, Huawei Liang, Weixin Huang, and Fengyu Xu. Two-layer-graph clustering for real-time 3d lidar point cloud segmentation. *Applied Sciences*, 10(23):8534, 2020.

[58] Sukai Wang, Yuxiang Sun, Chengju Liu, and Ming Liu. Pointtracknet: An end-to-end network for 3-d object detection and tracking from point clouds. *IEEE Robotics and Automation Letters*, 5(2):3206–3212, 2020.

[59] Achim Kampker, Mohsen Sefati, Arya S Abdul Rachman, Kai Kreisköther, and Pascual Campoy. Towards multi-object detection and tracking in urban scenario under uncertainties. In *VEHITS*, pages 156–167, 2018.

[60] Ge Guo and Shijie Zhao. 3d multi-object tracking with adaptive cubature kalman filter for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 8(1):512–519, 2022.

[61] Chenxu Luo, Xiaodong Yang, and Alan Yuille. Exploring simple 3d multi-object tracking for autonomous driving. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10488–10497, 2021.

[62] Martin Simon, Karl Amende, Andrea Kraus, Jens Honer, Timo Samann, Hauke Kaulbersch, Stefan Milz, and Horst Michael Gross. Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[63] Hai Wu, Wenkai Han, Chenglu Wen, Xin Li, and Cheng Wang. 3d multi-object tracking in point clouds based on prediction confidence-guided data association. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5668–5677, 2021.

[64] Eren Erdal Aksoy, Saimir Baci, and Selcuk Cavdar. Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving. In *2020 IEEE intelligent vehicles symposium (IV)*, pages 926–932. IEEE, 2020.

[65] Yingjie Wang, Qiuyu Mao, Hanqi Zhu, Jiajun Deng, Yu Zhang, Jianmin Ji, Houqiang Li, and Yanyong Zhang. Multi-modal 3d object detection in autonomous driving: a survey. *International Journal of Computer Vision*, 131(8):2122–2152, 2023.

[66] KS Arikumar, A Deepak Kumar, Thippa Reddy Gadekallu, Sahaya Beni Prathiba, and K Tamilarasi. Real-time 3d object detection and classification in autonomous driving environment using 3d lidar and camera sensors. *Electronics*, 11(24):4203, 2022.

[67] Masaru Yoshioka, Naoki Suganuma, Keisuke Yoneda, and Mohammad Aldibaja. Real-time object classification for autonomous vehicle using lidar. In *2017 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, pages 210–211. IEEE, 2017.

[68] Wei Song, Shuanghui Zou, Yifei Tian, Simon Fong, and Kyungeun Cho. Classifying 3d objects in lidar point clouds with a back-propagation neural network. *Human-centric Computing and Information Sciences*, 8(1):29, 2018.

[69] Hongbo Gao, Bo Cheng, Jianqiang Wang, Keqiang Li, Jianhui Zhao, and Deyi Li. Object classification using cnn-based fusion of vision and lidar in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics*, 14(9):4224–4231, 2018.

[70] Edouard Capellier, Franck Davoine, Véronique Cherfaoui, and You Li. Evidential deep learning for arbitrary lidar object classification in the context of autonomous driving. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1304–1311. IEEE, 2019.

[71] Valentin Magnier, Dominique Gruyer, and Jerome Godelle. Automotive lidar objects detection and classification algorithm using the belief theory. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 746–751. IEEE, 2017.

[72] Lina Fang, Zhilong You, Guixi Shen, Yiping Chen, and Jianrong Li. A joint deep learning network of point clouds and multiple views for roadside object classification from lidar point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 193:115–136, 2022.

[73] Abderrahim Kasmi, Johann Laconte, Romuald Aufrere, Dieumet Denis, and Roland Chapuis. End-to-end probabilistic ego-vehicle localization framework. *IEEE Transactions on Intelligent Vehicles*, 6(1):146–158, 2020.

[74] Mattia Bersani, Simone Mentasti, Pragyan Dahal, Stefano Arrigoni, Michele Vignati, Federico Cheli, and Matteo Matteucci. An integrated algorithm for ego-vehicle and obstacles state estimation for autonomous driving. *Robotics and Autonomous Systems*, 139:103662, 2021.

[75] Naoki Akai, Takatsugu Hirayama, and Hiroshi Murase. Persistent homology in lidar-based ego-vehicle localization. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 889–896. IEEE, 2021.

[76] Yuzhe He, Shuang Liang, Xiaofei Rui, Chengying Cai, and Guowei Wan. Egovm: Achieving precise ego-localization using lightweight vectorized maps. *arXiv preprint arXiv:2307.08991*, 2023.

[77] Susu Fang, Hao Li, and Ming Yang. Lidar slam based multivehicle cooperative localization using iterated split cif. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):21137–21147, 2022.

[78] Qi Wang, Jian Chen, Jianqiang Deng, Xinfang Zhang, and Kaixiang Zhang. Simultaneous pose estimation and velocity estimation of an ego vehicle and moving obstacles using lidar information only. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):12121–12132, 2021.

[79] Shuran Zheng, Jinling Wang, Chris Rizos, Weidong Ding, and Ahmed El-Mowafy. Simultaneous localization and mapping (slam) for autonomous driving: Concept and analysis. *Remote Sensing*, 15(4):1156, 2023.

[80] B Ravi Kiran, Luis Roldao, Benat Irastorza, Renzo Verastegui, Sebastian Suss, Senthil Yogamani, Victor Talpaert, Alexandre Lepoutre, and Guillaume Trehard. Real-time dynamic object detection for autonomous driving using prior 3d-maps. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

[81] Liang Chen, Jian Yang, and Hui Kong. Lidar-histogram for fast road and obstacle detection. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1343–1348. IEEE, 2017.

[82] Tim Schreier, Katrin Renz, Andreas Geiger, and Kashyap Chitta. On offline evaluation of 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4084–4089, 2023.

[83] Pablo Marín-Plaza, Jorge Beltrán, Ahmed Hussein, Basam Musleh, David Martín, Arturo de la Escalera, and José María Armingol. Stereo vision-based local occupancy grid map for autonomous navigation in ros. In *International Conference on Computer Vision Theory and Applications*, volume 4, pages 701–706. SciTePress, 2016.

[84] Michael Himmelsbach, Felix V Hundelshausen, and H-J Wuensche. Fast segmentation of 3d point clouds for ground vehicles. In *2010 IEEE Intelligent Vehicles Symposium*, pages 560–565. IEEE, 2010.

[85] AA Rachman. 3d-lidar multi object tracking for autonomous driving. *Faculty of Mechanical, Maritime and Materials Engineering (3mE) Delft University of Technology, Master of Science Thesis*, 2017.

[86] Yaakov Bar-Shalom, Thomas E Fortmann, and Peter G Cable. Tracking and data association, 1990.

[87] Samuel S Blackman. Multiple-target tracking with radar applications. *Dedham*, 1986.

[88] Giulia Slavic, Pablo Marin, David Martin, Lucio Marcenaro, and Carlo Regazzoni. Interpretable anomaly detection using a generalized markov jump particle filter. In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pages 1–5. IEEE, 2021.

[89] Giulia Slavic, Pablo Marín Plaza, Lucio Marcenaro, David Martín Gómez, and Carlo Regazzoni. Simultaneous localization and anomaly detection from first-person video data through a coupled dynamic bayesian network model. In *2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8. IEEE, 2022.

[90] Lakis Christodoulou, Takis Kasparis, and Oge Marques. Advanced statistical and adaptive threshold techniques for moving object detection and segmentation. In *2011 17th International conference on digital signal processing (DSP)*, pages 1–6. IEEE, 2011.

[91] Fahmy Amin and M Mahmoud. Confusion matrix in binary classification problems: A step-by-step tutorial. *Journal of Engineering Research*, 6(5):0–0, 2022.

[92] Johann Laconte, Abderrahim Kasmi, Romuald Aufrère, Maxime Vaidis, and Roland Chapuis. A survey of localization methods for autonomous vehicles in highway scenarios. *Sensors*, 22(1):247, 2021.

[93] Jonathon Luiten, Tobias Fischer, and Bastian Leibe. Track to reconstruct and reconstruct to track. *IEEE Robotics and Automation Letters*, 5(2):1803–1810, 2020.

[94] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. Robust multi-modality multi-object tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2365–2374, 2019.

[95] Yun Zhu, Mahendra Mallick, Shuang Liang, and Junkun Yan. Generalized labeled multi-bernoulli multi-target tracking with doppler-only measurements. *Remote Sensing*, 14(13):3131, 2022.

[96] Lionel Rakai, Huansheng Song, ShiJie Sun, Wentao Zhang, and Yanni Yang. Data association in multiple object tracking: A survey of recent techniques. *Expert systems with applications*, 192:116300, 2022.

[97] Lichuan Ren and Zhimin Xi. Bias-learning-based model predictive controller design for reliable path tracking of autonomous vehicles under model and environmental uncertainty. *Journal of Mechanical Design*, 144(9):091706, 2022.

[98] Huan Yin, Xuecheng Xu, Sha Lu, Xieyuanli Chen, Rong Xiong, Shaojie Shen, Cyrill Stachniss, and Yue Wang. A survey on global lidar localization: Challenges, advances and open problems. *International Journal of Computer Vision*, 132(8):3139–3171, 2024.

[99] Minjun Choi, Junhyeong Ryu, Yongseok Son, Sungrae Cho, and Jeongyeup Paek. Lidar-based localization for autonomous vehicles-survey and recent trends. In *2024 15th International Conference on Information and Communication Technology Convergence (ICTC)*, pages 456–460. IEEE, 2024.

[100] Minoo Dolatabadi, Fardin Ayar, Ehsan Javanmardi, Manabu Tsukada, and Mahdi Javanmardi. Neural error covariance estimation for precise lidar localization. *arXiv preprint arXiv:2501.02558*, 2025.

[101] Hongyang Li, Yang Li, Huijie Wang, Jia Zeng, Huilin Xu, Pinlong Cai, Li Chen, Junchi Yan, Feng Xu, Lu Xiong, et al. Open-sourced data ecosystem in autonomous driving: the present and future. *arXiv preprint arXiv:2312.03408*, 2023.

[102] Novel Certad, Enrico Del Re, Helena Korndörfer, Gregory Schröder, Walter Morales-Alvarez, Sebastian Tschernuth, Delgermaa Gankhuyag, Luigi del Re, and Cristina Olaverri-Monreal. Interaction of autonomous and manually controlled vehicles multiscenario vehicle interaction dataset. *IEEE Intelligent Transportation Systems Magazine*, 2024.

[103] Syed Adnan Yusuf, Arshad Khan, and Riad Souissi. Vehicle-to-everything (v2x) in the autonomous vehicles domain–a technical review of communication, sensor, and ai technologies for road user safety. *Transportation Research Interdisciplinary Perspectives*, 23:100980, 2024.

[104] Luca Barbieri, Bernardo Camajori Tedeschini, Mattia Brambilla, and Monica Nicoli. Deep learning-based cooperative lidar sensing for improved vehicle positioning. *IEEE Transactions on Signal Processing*, 2024.

[105] Anas Charroud, Karim El Moutaouakil, Vasile Palade, and Ali Yahyaouy. Enhanced autoencoder-based lidar localization in self-driving vehicles. *Applied Soft Computing*, 152:111225, 2024.

[106] Florian Sauerbeck, Dominik Kulmer, Markus Pielmeier, Maximilian Leitenstern, Christoph Weiß, and Johannes Betz. Multi-lidar localization and mapping pipeline for urban autonomous driving. In *2023 IEEE SENSORS*, pages 1–4. IEEE, 2023.