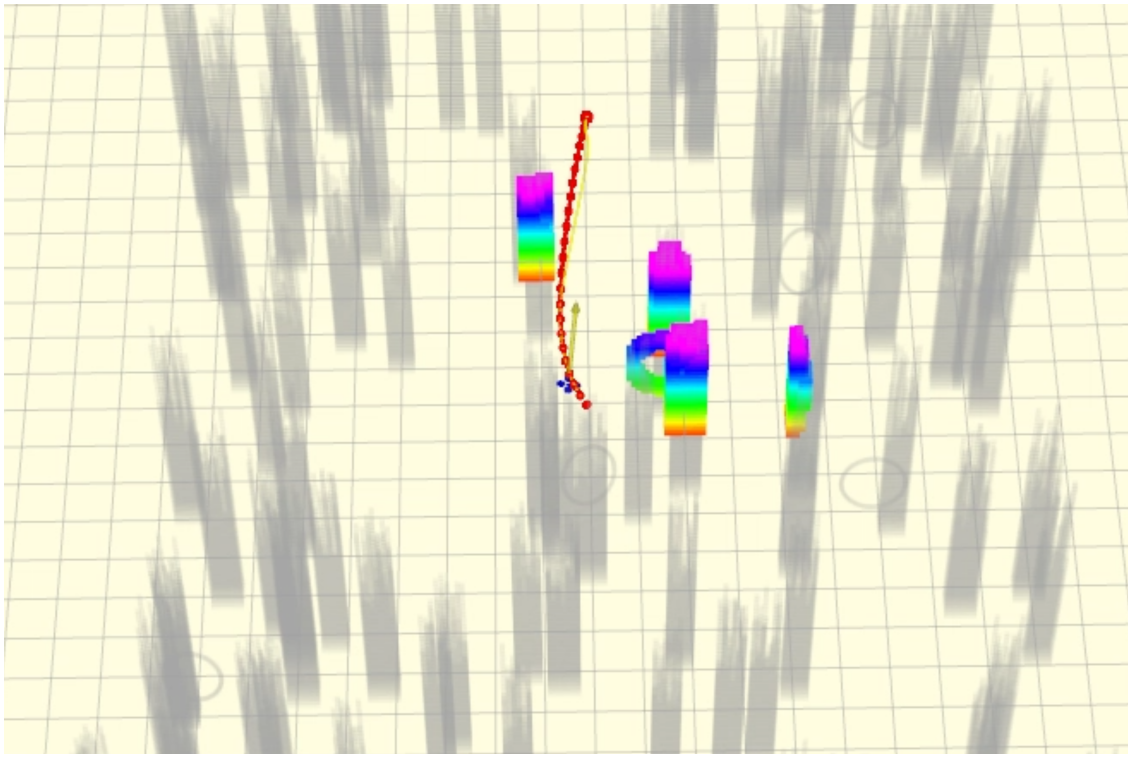

Navigation and Localization for Railway Inspection Drone in GPS-denied Environments

An Investigative Study of Modular SLAM Baselines and End-to-End Learning-based Approaches

Master's thesis in Complex Adaptive Systems

GUANFEI WANG

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's Thesis in Complex Adaptive Systems

Navigation and Localization for Railway Inspection Drone in GPS-denied Environments

An Investigative Study of Modular SLAM Baselines and End-to-End Learning-based Approaches

GUANFEI WANG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2026

Navigation and Localization for Railway Inspection Drone in GPS-denied Environments
A Comparative Study of Modular SLAM-based and End-to-End Learning Approaches
GUANFEI WANG

Supervisor: Krister Wolff, Department of Mechanics and Maritime Sciences
Examiner: Krister Wolff, Department of Mechanics and Maritime Sciences

Master's Thesis 2026
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Fast-Planner trajectory generation demonstration showing the planned path (red) navigating through a a simulated random forest environment. The colorful voxel structures represent detected obstacles, and the algorithm successfully generates a collision-free trajectory while maintaining dynamic feasibility.

Typeset in LaTeX
Gothenburg, Sweden 2026

Navigation and Localization for Railway Inspection Drone in GPS-denied Environments
A Comparative Study of Modular SLAM-based and End-to-End Learning Approaches
GUANFEI WANG
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

Abstract

Autonomous navigation in GPS-denied environments remains a critical challenge for unmanned aerial vehicles performing infrastructure inspection. This thesis investigates the feasibility of learning-based navigation for railway-inspection drones by first constructing and analyzing a state-of-the-art modular baseline and then evaluating emerging end-to-end paradigms.

A high-performance navigation system combining FasterLIO SLAM with Fast-Planner trajectory generation is implemented in high-fidelity simulation and used as an analytical baseline. While the system successfully navigates dense forest environments, controlled experiments reveal three structural failure modes—SLAM localization drift, flight-controller tracking limitations, and planner-induced trajectory constraints—highlighting deeper challenges such as cumulative error propagation and real-time sensor–compute bottlenecks.

Building on these insights, the thesis conducts an experimentally grounded feasibility study of three dominant end-to-end learning directions: predictive world-model architectures, self-supervised representation learning pipelines, and vision-reinforcement-learning approaches. By implementing prototype models and stress-testing their stability and data requirements, the study identifies several infeasible or unstable directions—such as feature-forecasting models and self-distillation objectives—and reveals simulator limitations that currently block scalable vision-RL for UAVs.

Rather than delivering a complete end-to-end navigation system, this work provides a systematic evaluation of the landscape, clarifies the fundamental obstacles facing learning-based navigation in GPS-denied environments, and establishes concrete design requirements and a research roadmap for future PhD-level research.

Keywords: autonomous navigation, UAV, GPS-denied, railway inspection, modular systems, LiDAR, SLAM, end-to-end learning, self-supervised learning, reinforcement learning, world models.

Preface

This report presents the outcome of our master's thesis project carried out at the Department of Mechanics and Maritime Sciences at Chalmers University of Technology. The research investigates autonomous navigation systems for UAVs in GPS-denied environments, comparing traditional modular SLAM-based approaches with emerging end-to-end learning methods.

Gothenburg, December 2025

GUANFEI WANG

Acknowledgements

I would like to thank our supervisor for guidance throughout this project and acknowledge the robotics and ai community for providing the foundational tools that enabled this research.
The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

Guanfei Wang, Gothenburg, December 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AI	Artificial Intelligence
EKF2	Extended Kalman Filter 2
ENU	East-North-Up
FCU	Flight Control Unit
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LOAM	LiDAR Odometry and Mapping
MAV	Micro Aerial Vehicle
MEMS	Micro-Electro-Mechanical System
NED	North-East-Down
ROS	Robot Operating System
SDF	Simulation Description Format
SITL	Software In The Loop
SLAM	Simultaneous Localization and Mapping
UDP	User Datagram Protocol
UAV	Unmanned Aerial Vehicle
VIO	Visual-Inertial Odometry
GRU	Gated Recurrent Unit
PPO	Proximal Policy Optimization

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background and Motivation

The rapid advancement of unmanned aerial vehicles (UAVs) has fundamentally transformed numerous sectors, establishing autonomous navigation as a cornerstone technology for modern robotics applications. Among the many UAV applications, the automated inspection and maintenance of railway infrastructure has emerged as a mission-critical and highly demanding field. Traditional railway inspection is labor-intensive, time-consuming, and often exposes workers to hazardous conditions. UAV-based automated inspection offers the promise of greatly enhanced safety, reduced operational costs, and the ability to perform frequent, high-resolution monitoring of rail assets such as tracks, bridges, tunnels, and stations.

However, the railway environment introduces unique and formidable technical challenges for autonomous UAV navigation. Railway corridors commonly traverse GPS-denied environments—such as long tunnels, urban canyons, and densely vegetated areas—where conventional positioning systems become unreliable or unavailable. The UAV must navigate long, often monotonous routes with complex geometry, rapidly detect and respond to obstacles (such as maintenance personnel, animals, or debris on the tracks), and operate under variable weather and lighting conditions. These real-world constraints demand advanced onboard perception and decision-making capabilities that are robust, adaptive, and do not rely on external infrastructure [4].

To address these challenges, the state-of-the-art in autonomous UAV navigation for railway inspection has largely adopted modular architectures. These approaches decompose navigation into sequential processing stages: UAVs equipped with LiDAR and Inertial Measurement Units (IMUs) utilize Simultaneous Localization and Mapping (SLAM) frameworks to generate precise pose estimates and environmental reconstructions, which are then used by path planning modules to compute safe, efficient trajectories [12, 2]. In the context of railway inspection, such systems have achieved robust performance in structured, predictable track sections and provide a solid foundation for practical deployment.

While end-to-end learning paradigms have recently gained traction, their suitability for safety-critical railway inspection remains unclear. Instead of building a full end-to-end system, this thesis analyzes several potential technical pathways—such as predictive world models, self-supervised representation learning, and vision-based re-

inforcement learning—and evaluates them through targeted prototype experiments. The results reveal fundamental limitations that render several directions currently infeasible, providing a clear understanding of the challenges ahead. These insights form a solid technical foundation for future PhD-level research into learning-based autonomous navigation in GPS-denied railway environments.

1.2 Research Objectives

The primary objective of this thesis is to develop and rigorously evaluate a modular, SLAM-based autonomous navigation framework for UAV railway inspection in complex, GPS-denied environments. The work aims to:

- Design, implement, and optimize a state-of-the-art modular UAV navigation system tailored for railway inspection, integrating LiDAR-based SLAM with robust trajectory planning.
- Systematically assess the performance, robustness, and failure modes of modular navigation pipelines through high-fidelity simulations and controlled experiments.
- Analyze potential technical pathways for end-to-end learning-based navigation, supported by targeted prototype experiments to identify infeasible directions and clarify future research requirements.

1.3 Research Contributions

The main contributions of this thesis are as follows:

- **Comprehensive Modular Baseline Implementation:** Development and optimization of a state-of-the-art modular UAV navigation system, integrating advanced LiDAR-based SLAM (e.g., FasterLIO) with high-performance trajectory planning, and adapting the pipeline to the specific operational demands of railway inspection in GPS-denied environments.
- **Systematic Experimental Evaluation:** Rigorous evaluation of the modular navigation pipeline through high-fidelity simulations and controlled experiments, providing a detailed analysis of navigation accuracy, robustness, computational constraints, and characteristic failure modes in representative railway scenarios.
- **Analysis of End-to-End Technical Pathways:** A systematic examination of multiple candidate technical routes for end-to-end learning-based navigation—including predictive world models, self-supervised representation learning, and vision-based reinforcement learning—supported by targeted prototype experiments that help rule out several infeasible directions.

1.4 Scope and Limitations

This thesis operates within clearly defined boundaries to ensure a focused investigation and to highlight opportunities for future research:

1.4.1 Sensor Modality Constraints

The study centers on navigation systems built upon LiDAR and IMU sensing, reflecting current industry standards for UAV-based railway inspection. Although this configuration provides robust geometric perception, performance may degrade in environments with limited structural features or in adverse weather conditions (e.g., heavy rain, fog, snow). Although vision-based sensing holds significant potential, its practical deployment in railway UAV navigation remains highly challenging. Therefore, this thesis limits itself to a conceptual and technical-route analysis of vision-based sensing modalities rather than their implementation.

1.4.2 Experimental Environment Limitations

Experimental evaluation is conducted primarily in simulation environment. While simulation enables safety, repeatability, and systematic parameter exploration, real-world deployment involves additional complexities—such as regulatory limitations, safety considerations, and operational overhead—that are not comprehensively addressed in this thesis.

1.4.3 Methodological Limitations

This thesis focuses exclusively on designing and evaluating a modular, SLAM-based navigation framework. Although multiple end-to-end learning-based navigation paradigms are analyzed through literature review and small-scale prototype experiments, they are not implemented or benchmarked as complete systems. Their feasibility, challenges, and long-term potential are discussed primarily to establish a foundation for future research rather than as contributions of this study.

1.4.4 Cross-Platform Analysis Scope

Generalization to other robotic platforms—such as ground vehicles or legged robots—is outside the scope of this thesis and is mentioned only briefly as a potential future direction.

1.5 Thesis Organization

This thesis is structured to provide a comprehensive investigation of modular, SLAM-based autonomous navigation for UAV railway inspection, progressing from foundational methods to implementation, evaluation, and forward-looking analysis.

Chapter 2 presents the methodological framework, detailing the architecture, integration, and implementation of the LiDAR-based SLAM and trajectory planning modules.

Chapter 3 describes the experimental environments and protocols, followed by an extensive evaluation of system performance, robustness, and failure modes in representative railway scenarios.

Chapter 4 synthesizes these results, discussing the strengths and limitations of the modular approach and providing an analytical overview of potential pathways for end-to-end learning-based navigation.

Chapter 5 outlines future research directions. It expands on the conceptual and exploratory discussion of end-to-end learning-based navigation, comparing different technical routes—including predictive world models, self-supervised representation learning, and vision-based reinforcement learning—while identifying the key challenges and requirements for their eventual realization in safety-critical railway inspection.

2

Theory

2.1 UAV Kinematic Modeling

The mathematical representation of unmanned aerial vehicle motion forms the foundation for developing autonomous navigation algorithms. This section establishes the kinematic framework that abstracts the complex dynamics of quadrotor flight into a tractable mathematical model suitable for high-level path planning and control applications.

2.1.1 Simplified Kinematic Representation

For the purposes of autonomous navigation research, the UAV is modeled as a point mass operating within a three-dimensional Euclidean coordinate system. This abstraction is justified by the assumption that a robust low-level flight control system—such as PX4 Autopilot or ArduPilot—maintains attitude stability and accurately executes high-level trajectory commands [6, 22]. Consequently, the complex quadrotor dynamics including motor thrust allocation, aerodynamic forces, and torque generation are considered to be handled by the underlying flight controller and are not explicitly modeled in this analysis.

This modeling approach focuses on the kinematic behavior of the UAV, describing its motion in terms of position, velocity, and orientation without explicit consideration of the forces and moments that generate the motion [14]. Such abstraction is appropriate for research emphasizing high-level autonomous navigation and trajectory planning algorithms, where the primary concern is the generation of feasible reference trajectories rather than their detailed execution.

2.1.2 State Space Representation

The UAV's kinematic state is characterized by its position vector $\mathbf{p} = [x, y, z]^T$ and linear velocity vector $\mathbf{v} = [\dot{x}, \dot{y}, \dot{z}]^T$ expressed in an inertial navigation frame. The vehicle's orientation, while internally managed by the flight control system, can be represented by its yaw angle ψ in the horizontal plane for trajectory planning purposes, with the assumption that roll and pitch angles are maintained near zero by the low-level stabilization controller.

The governing kinematic equations are expressed as:

$$\dot{x} = v_x \tag{2.1}$$
$$\dot{y} = v_y \tag{2.2}$$
$$\dot{z} = v_z \tag{2.3}$$

where v_x, v_y, v_z represent the commanded velocity components along the respective coordinate axes.

2.1.3 Dynamic Constraints

The UAV's motion is constrained by physical limitations and flight controller capabilities that must be respected during trajectory generation. These constraints include maximum linear velocity (v_{max}), maximum acceleration (a_{max}), and maximum angular rates, which define the feasible control space for autonomous navigation [26]. The trajectory planning algorithms developed in this research assume that the flight controller can accurately track velocity commands within these predefined operational limits.

This simplified kinematic representation provides a computationally efficient framework for designing and evaluating autonomous navigation strategies while abstracting the complexities of the underlying control architecture.

2.2 Sensor Technologies and Data Structures

Autonomous navigation systems depend critically on accurate environmental perception and vehicle state estimation through multi-modal sensor integration. This section examines the fundamental principles and data characteristics of the primary sensing modalities employed in this research: Light Detection and Ranging (LiDAR) systems and Inertial Measurement Units (IMUs).

2.2.1 LiDAR Systems

LiDAR technology provides high-precision three-dimensional environmental perception through time-of-flight measurement of laser pulses, enabling direct distance measurement to surrounding objects with millimeter-level accuracy [12].

2.2.1.1 Point Cloud Data Structure

The fundamental output of LiDAR sensors is a point cloud—a discrete sampling of the three-dimensional environment represented as a collection of spatial coordinates. Each measurement point within the cloud contains several key attributes:

- **Spatial Coordinates**: Three-dimensional position (x, y, z) representing the location of laser reflection, typically expressed relative to the sensor's coordinate frame.

- **Intensity Information**: Reflectance strength of the returned laser pulse, providing material property information that can enhance object classification and feature extraction.
- **Temporal Synchronization**: Precise timestamps enabling temporal alignment with other sensor measurements for effective multi-modal fusion [18].

For continuous scanning LiDAR systems, individual measurements are aggregated over complete rotation cycles to generate comprehensive environmental snapshots. The spatial density and range characteristics of the resulting point clouds depend on sensor specifications including beam count, angular resolution, and maximum detection range [20].

2.2.2 Inertial Measurement Units

Inertial Measurement Units provide high-frequency motion sensing through microelectromechanical systems (MEMS) that measure linear acceleration and angular velocity. These sensors are fundamental for vehicle state estimation and motion prediction in navigation systems [16].

2.2.2.1 Multi-Axis Sensor Data

IMU systems typically integrate accelerometers and gyroscopes to provide comprehensive motion information:

- **Linear Acceleration**: Three-axis measurements (a_x, a_y, a_z) capturing both vehicle dynamics and gravitational effects, typically expressed in meters per second squared.
- **Angular Velocity**: Three-axis rotation rates $(\omega_x, \omega_y, \omega_z)$ indicating instantaneous rotational motion, commonly measured in radians per second.
- **High-Frequency Sampling**: Measurements at rates typically ranging from 100 Hz to 1000 Hz, providing excellent temporal resolution for motion tracking [15].

While IMU data provides excellent short-term motion tracking capabilities, measurements are susceptible to integration drift over extended periods, necessitating fusion with complementary sensors for robust long-term state estimation [9].

2.3 Trajectory Planning and Optimization

Trajectory planning constitutes a central component of autonomous navigation systems, responsible for generating collision-free, dynamically feasible paths that optimize specified performance criteria. This section establishes the mathematical foundation for smooth trajectory representation using B-spline curves, which provide the flexibility and continuity properties essential for UAV navigation.

2.3.1 B-Spline Mathematical Framework

B-splines (Basis Splines) represent a class of piecewise polynomial curves that offer superior properties for trajectory generation compared to alternative parametric representations. Unlike Bézier curves, B-splines provide local control characteristics, where modifications to individual control points affect only localized curve segments—a property particularly valuable for iterative optimization and real-time trajectory adaptation [26].

A B-spline curve $\mathbf{C}(u)$ of degree p is mathematically defined through a set of $n+1$ control points $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$ and a non-decreasing knot vector $U = [u_0, u_1, \ldots, u_m]$, where $m = n + p + 1$. The curve is expressed as:

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i$$

where $N_{i,p}(u)$ represent the p-th degree B-spline basis functions, recursively defined through the Cox-de Boor recursion formula:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

with the mathematical convention that zero denominators result in zero-valued fractions.

2.3.2 Advantageous Properties for UAV Navigation

B-splines possess several mathematical properties that make them particularly suitable for autonomous vehicle trajectory planning [25]:

- **Localized Control**: Control point modifications influence only curve segments within the basis function support, enabling efficient local optimization without global trajectory disruption.
- **Differential Continuity**: B-splines of degree p exhibit C^{p-1} continuity, ensuring smooth velocity, acceleration, and jerk profiles essential for stable flight dynamics [22].
- **Convex Hull Property**: The curve remains entirely within the convex hull of its control points, facilitating collision detection and ensuring trajectory containment within specified bounds.
- **Parametric Flexibility**: Adjustable degree, control point count, and knot vector distribution enable representation of diverse trajectory shapes and complexity levels.
- **Analytical Derivatives**: Closed-form derivative computation provides direct access to velocity and acceleration profiles along the trajectory [2].

2.3.3 Application in Autonomous Navigation

In autonomous navigation systems, B-splines serve as the mathematical foundation for trajectory representation, where optimization algorithms manipulate control point positions to satisfy multiple objectives including obstacle avoidance, dynamic feasibility, and performance criteria [23]. The combination of local control properties and smooth continuity makes B-splines particularly well-suited for real-time trajectory planning applications where computational efficiency and motion quality are both critical requirements [24].

2.3.4 Transformer Architectures

Transformer architectures form the foundation of many modern deep learning models, originally introduced for sequence-to-sequence tasks [27] and later adapted to computer vision. Their core computational primitive is the *self-attention* mechanism, which enables the model to capture global dependencies among inputs. Given a sequence of input token embeddings $X \in \mathbb{R}^{N\times d}$, the Transformer computes *queries*, *keys*, and *values*:

$$Q = XW_Q, \qquad K = XW_K, \qquad V = XW_V,$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d\times d_h}$ are learned projection matrices and d_h is the head dimension. The scaled dot-product attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V.$$

To increase representation capacity, multi-head attention (MHA) runs h attention heads in parallel:

$$\text{MHA}(X) = \text{Concat}\Big(\text{head}_1, \ldots, \text{head}_h\Big)W_O,$$

where $W_O \in \mathbb{R}^{hd_h\times d}$. In the Vision Transformer (ViT) [28], an image is divided into $P \times P$ patches, each linearly projected into a token embedding:

$$z_0 = \Big[x^1E;\ x^2E;\ \cdots;\ x^NE\Big] + E_{\text{pos}},$$

where E is the patch embedding matrix and E_{pos} provides positional encodings. A Transformer encoder layer then applies:

$$z' = \text{MHA}(\text{LN}(z)) + z,$$

$$z_{\text{out}} = \text{MLP}(\text{LN}(z')) + z'.$$

These residual connections and normalization layers stabilize optimization, enabling Transformers to scale effectively with model size and dataset size. In UAV navigation, Transformer-based visual encoders can learn high-capacity representations from raw images or video, making them promising for future end-to-end navigation systems. However, their computation and data requirements pose challenges for real-time onboard deployment, especially in GPS-denied railway environments.

2.3.5 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) [29] is a widely used reinforcement learning algorithm designed to stabilize policy gradient updates while maintaining sample efficiency. PPO achieves this by introducing a clipped surrogate objective that constrains the deviation between the updated policy and the previous policy, preventing destructive updates that can destabilize learning. Given a policy $\pi_\theta(a \mid s)$ with parameters θ, PPO maximizes the clipped objective:

$$L^{\text{PPO}}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right],$$

where $r_t(\theta)$ is the probability ratio between the new and old policies and A_t is the advantage estimate. The clipping mechanism ensures that updates remain within a safe trust region, providing a practical balance between stability and learning speed. In the context of UAV navigation, PPO provides a simple yet effective framework for training control policies from high-dimensional observations. However, vision-based PPO requires realistic, diverse simulation environments to generalize effectively—an obstacle highlighted in this thesis and identified as a core limitation of current end-to-end learning approaches.

3

Methodology for the Modular Navigation System

3.1 Overall System Architecture

The modular autonomous navigation system for unmanned aerial vehicles (UAVs) is built upon a sophisticated layered architecture that seamlessly integrates state-of-the-art open-source platforms with custom-designed communication interfaces. This comprehensive framework addresses the complex challenges of real-time autonomous flight in cluttered environments while maintaining computational efficiency and system reliability [3, 4].

The foundation of our system rests on Gazebo Harmonic [5], which provides a high-fidelity physics-based simulation environment for UAV dynamics and sensor modeling. This simulation platform interfaces directly with PX4 Autopilot [6], serving as the primary flight control unit responsible for low-level stability control and precise command execution. The navigation intelligence is powered by two specialized modules: FasterLIO [1] for LiDAR-based simultaneous localization and mapping (SLAM), and Fast-Planner [2] for dynamic path planning and trajectory generation.

A critical design consideration involves the integration of modern and legacy robotics frameworks. While sensor data acquisition utilizes the contemporary ROS 2 Jazzy environment [7] to publish IMU and LiDAR measurements as standardized topics, the core navigation algorithms—developed primarily within the mature ROS 1 ecosystem [8]—operate on ROS 1 Noetic. To bridge this technological gap, a custom UDP-based communication bridge has been implemented to efficiently translate data between these environments. This architectural decision not only leverages the strengths of both frameworks but also facilitates future migration to a unified ROS 2 ecosystem by encapsulating the bridging logic in a modular component.

The data flow within the system follows a carefully orchestrated sequence: sensor data from Gazebo is initially processed through ROS 2 topics, then transmitted via our custom bridge to the ROS 1-based navigation stack. FasterLIO processes the converted LiDAR and IMU streams to generate both low-frequency odometry estimates and registered point clouds. The odometry data undergoes sensor fusion within PX4's Extended Kalman Filter 2 (EKF2) [9], which combines high-rate IMU measurements with the SLAM-derived pose estimates to produce robust, high-

frequency state estimation. Finally, Fast-Planner utilizes both the fused odometry and registered point clouds to generate collision-free trajectories, which are executed by PX4 to achieve autonomous flight. An overview of the complete system architecture and the corresponding data flow is illustrated in Figure 3.1.

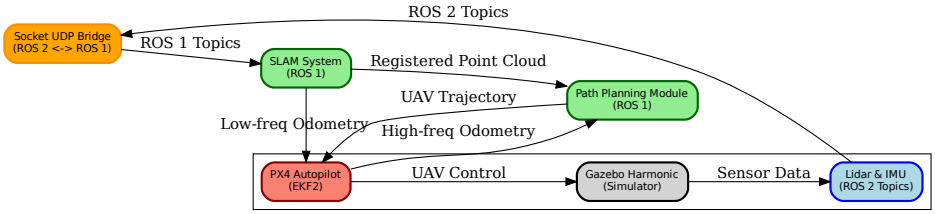

Figure 3.1: Overall System Architecture for UAV Autonomous Navigation.

3.2 Simulation Environment Setup

The development and validation of autonomous navigation systems necessitate high-fidelity simulation environments that accurately represent real-world operational conditions. Our simulation framework has been meticulously designed to provide both realistic physics modeling and comprehensive sensor simulation while maintaining computational efficiency for iterative development and testing.

3.2.1 Simulation Platform Selection

Gazebo Harmonic serves as the primary simulation engine, chosen specifically for its native compatibility with PX4 Autopilot as the officially supported Software-in-the-Loop (SITL) simulator [10]. This platform offers significant advantages beyond basic physics simulation: its internal topic-based communication architecture closely mirrors ROS 2's messaging paradigm, enabling seamless data integration and reducing the complexity of sensor data acquisition pipelines. The inherent compatibility between Gazebo Harmonic and ROS 2 eliminates many common integration challenges encountered with alternative simulation platforms [11].

3.2.2 UAV Platform and Sensor Configuration

The simulated UAV is modeled after the Holybro X500 quadrotor, a widely-used research platform that provides realistic flight dynamics and payload capacity constraints. To enable robust environmental perception in cluttered scenarios, the vehicle is equipped with a high-resolution 128-line LiDAR sensor strategically mounted atop the UAV frame. This sensor operates at 20 Hz to provide dense temporal sampling of the environment, with a vertical field of view spanning from 90 degrees upward to 15 degrees downward. This configuration ensures comprehensive spatial awareness while maintaining computational feasibility for real-time processing [12].

The sensor placement, illustrated in Figure 3.2, maximizes the observable volume while minimizing occlusions from the UAV's own structure. This configuration is critical for effective obstacle detection and mapping in complex three-dimensional environments.

Figure 3.2: UAV with Lidar.

Computational Hardware Specifications

The simulation and navigation system operates on a high-performance workstation equipped with specific hardware components selected to ensure real-time processing capabilities for the demanding computational requirements of autonomous navigation. The system specifications include:

- **CPU:** 13th Generation Intel Core i7-13700H processor featuring 14 cores (6 performance cores + 8 efficiency cores) with a base frequency of 2.4 GHz and boost frequency up to 5.0 GHz. This multi-core architecture provides sufficient computational resources for parallel processing of SLAM algorithms, path planning computations, and system integration tasks.
- **GPU:** NVIDIA GeForce RTX 4060 Laptop GPU with 8 GB GDDR6 VRAM, operating under CUDA 12.8 framework. The GPU acceleration is utilized primarily for possible point cloud processing operations within the FasterLIO SLAM pipeline and visualization rendering of the 3D environmental maps.
- **Memory:** 16 GB DDR5 RAM ensures adequate memory bandwidth for handling large point cloud datasets and maintaining multiple concurrent processes across the ROS 1/ROS 2 bridge architecture.
- **Operating System:** Ubuntu 24.04 LTS provides the stable foundation for ROS 2 Jazzy and ROS 1 Noetic integration.

This hardware configuration comfortably handles the computational demands of the system, as the 20 Hz update rate represents the maximum frequency achievable by current state-of-the-art LiDAR sensors in the market. Commercial high-end LiDAR systems, such as Ouster OS1-128, typically operate at 20 Hz in frequency, making our simulated 20 Hz configuration representative of the best available technology, and the computational load associated with processing 128-line point clouds at this frequency is well within the capabilities of the selected hardware.

For UAV flight speeds of 2-5 m/s typical in cluttered forest environments, the 20 Hz update rate provides spatial resolution of 10-25 cm between consecutive scans. This resolution is sufficient for reliable obstacle detection given the typical safety clearance margins (minimum 1 meter) required for safe navigation in dense vegetation.

3.2.3 Test Environment Design

The evaluation environment consists of a dense forest scenario specifically designed to challenge the navigation system's obstacle avoidance and path planning capabilities. This virtual forest spans 80 meters in length and 20 meters in width, featuring irregular tree placement and varying canopy densities that create a complex three-dimensional obstacle field. Strategic clear zones are positioned at the environment's extremities to facilitate safe takeoff and landing operations.

The navigation task requires the UAV to autonomously traverse from a designated starting position to a target endpoint while avoiding collisions with the densely packed vegetation and maintaining flight stability. This scenario represents one of the most challenging environments for autonomous navigation [13], requiring robust perception, accurate state estimation, and dynamic trajectory planning. The complete environmental layout, including dimensional specifications and waypoint locations, is depicted in Figure 3.3.

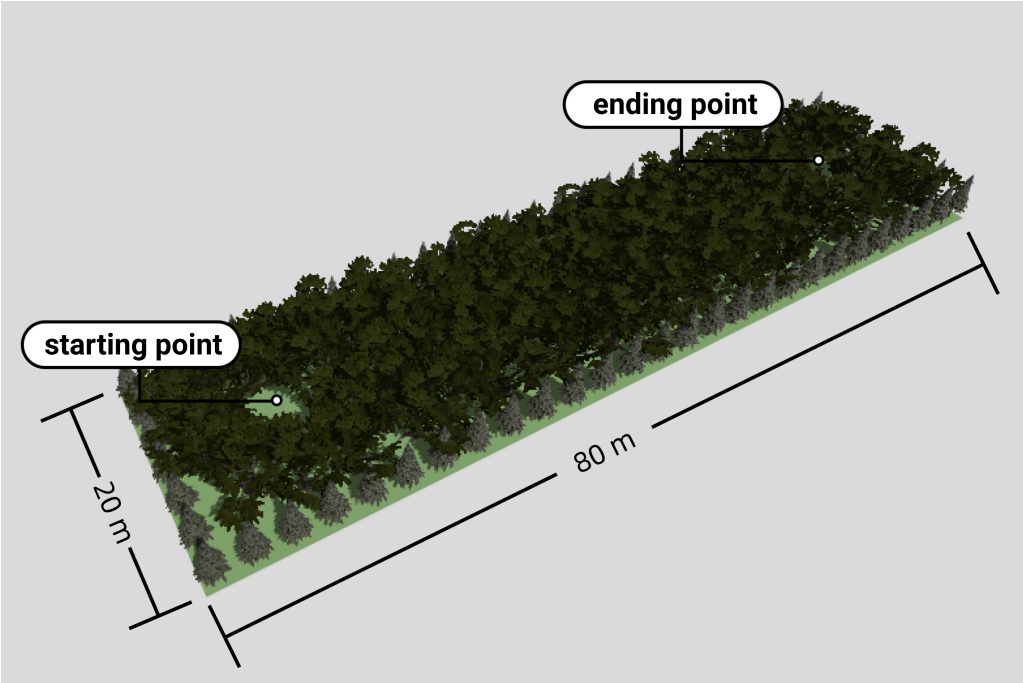

Figure 3.3: Annotated overview of the simulated dense forest environment, showing its dimensions (80 meters in length and 20 meters in width) and the designated starting and ending points for UAV navigation.

3.3 LiDAR-Based Simultaneous Localization and Mapping

Accurate pose estimation and environmental mapping form the cornerstone of reliable autonomous navigation. This section details the SLAM implementation that provides both vehicle localization and obstacle mapping capabilities necessary for safe path planning.

3.3.1 FasterLIO Algorithm Selection

FasterLIO was selected as the primary SLAM solution based on its superior performance characteristics in computational efficiency, accuracy, and practical deployment considerations. As an advanced evolution of the well-established FastLIO2 framework [18], FasterLIO offers several critical advantages for real-time autonomous navigation:

- **Enhanced Computational Performance:** FasterLIO demonstrates 1.5-2x speed improvements over its predecessor, enabling real-time operation on embedded computing platforms typical of UAV applications [12].
- **Improved Practical Accuracy:** Beyond computational gains, the algorithm exhibits enhanced robustness in challenging scenarios including dynamic lighting conditions, sparse features, and rapid motion [19].

- **Reduced Configuration Complexity:** The algorithm requires minimal parameter tuning compared to alternative SLAM solutions [20], facilitating rapid deployment and reducing system complexity.
- **Robust Sensor Fusion:** The tight coupling between LiDAR and IMU data provides superior performance in scenarios where individual sensors might fail or provide degraded measurements [21].
- **Integrated Safety Mechanisms:** The system incorporates fundamental collision avoidance capabilities through real-time distance monitoring, automatically triggering emergency stop procedures when obstacles are detected within a critical 0.5-meter safety zone. This safety feature provides an essential fail-safe mechanism that operates independently of higher-level path planning algorithms.

3.3.2 Accuracy Validation Methodology

The validation of SLAM odometry accuracy presents unique challenges that differ significantly between simulated and real-world deployment scenarios. In practical autonomous vehicle applications, the SLAM system serves as the primary and often sole source of accurate odometric information, making direct accuracy assessment inherently difficult without external reference systems.

For real-world applications, the SLAM odometry represents the ground truth navigation reference, as alternative positioning systems such as GPS may be unavailable, unreliable, or insufficiently precise for autonomous navigation requirements. This fundamental limitation necessitates alternative validation approaches, including consistency checks, loop closure validation, and comparative analysis with supplementary sensor data when available.

However, simulation environments provide controlled conditions that enable comprehensive accuracy assessment through direct comparison with known ground truth data. In the Gazebo simulation framework, precise vehicle pose information can be directly extracted from the physics engine, providing an absolute reference for odometry validation. This capability allows for quantitative error analysis by computing the deviation between SLAM-estimated trajectories and the simulator's ground truth odometry data. Such comparative analysis yields valuable metrics including translational drift, rotational error accumulation, and trajectory consistency over extended operational periods.

3.3.3 System Configuration and Calibration

Precise system configuration is essential for achieving the accuracy levels required for collision avoidance in cluttered environments. The LiDAR sensor parameters are configured to match the simulated 128-line, 20 Hz specifications, ensuring optimal point cloud processing and feature extraction.

A critical configuration aspect involves the extrinsic calibration between the LiDAR sensor and IMU. Rather than relying on online estimation, which can introduce drift

and compromise navigation accuracy, the precise spatial relationship is extracted from the UAV's SDF model file. The `extrinsic_est_en` parameter is explicitly disabled to prevent runtime calibration drift, prioritizing odometry accuracy over mapping aesthetics. This design choice reflects the system's primary objective of safe navigation rather than optimal map visualization.

3.3.4 Output Generation and Frequency Considerations

FasterLIO generates two primary outputs essential for autonomous navigation: pose estimates and registered point clouds, both operating at the LiDAR's native 20 Hz frequency. The pose estimates provide vehicle position, orientation, and velocity information with accuracy suitable for navigation but at frequencies insufficient for high-bandwidth control applications.

The registered point clouds undergo coordinate transformation from the LiDAR's local reference frame to the UAV's body frame, ensuring consistency with other system components. While the 20 Hz update rate is adequate for mapping and strategic path planning, it represents a fundamental limitation for reactive collision avoidance, particularly during high-speed flight or in highly dynamic environments. This constraint is partially mitigated through the sensor fusion approach described in Section 3.4, which elevates the effective odometry rate through IMU integration.

Figure 3.4 illustrates the real-time performance of FasterLIO in the dense forest environment, showing the generated point cloud map (in green) along with the planned UAV trajectory (in cyan). The visualization demonstrates the system's capability to accurately reconstruct the complex three-dimensional structure of the forest environment while simultaneously maintaining precise vehicle localization throughout the navigation process.

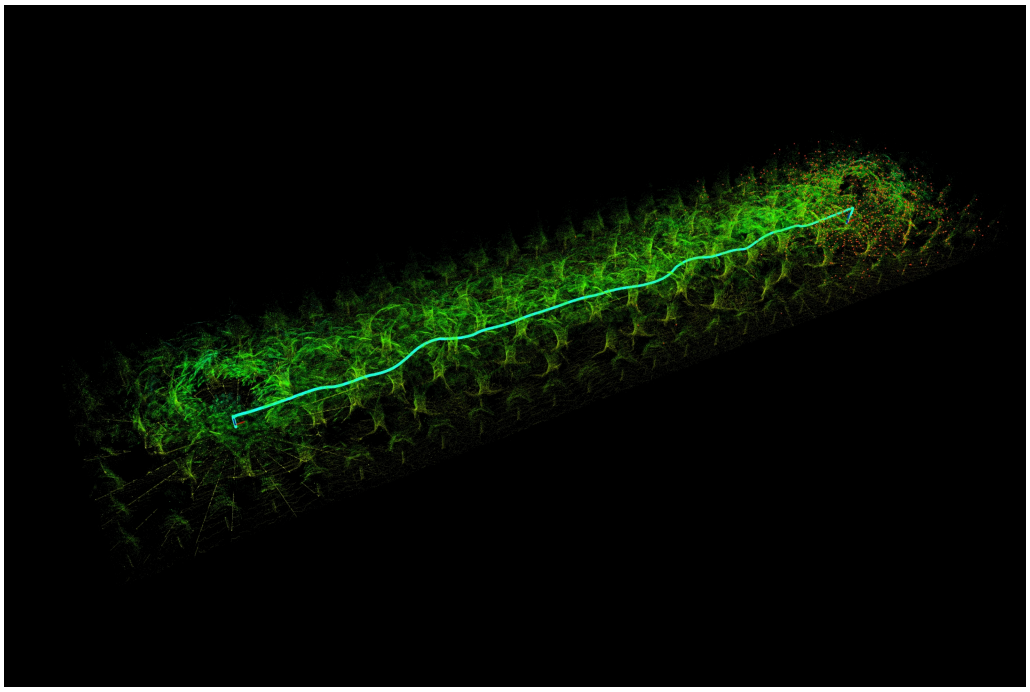

Figure 3.4: FasterLIO SLAM system demonstration showing the real-time point cloud mapping (green) and UAV trajectory (cyan) in the dense forest environment. The visualization illustrates the system's ability to simultaneously localize the vehicle and map the complex environmental structure.

3.4 Flight Control and State Estimation

The Flight Control Unit (FCU) represents the critical real-time control layer that translates high-level navigation commands into precise actuator inputs. This section details the control architecture and state estimation algorithms that enable robust autonomous flight performance.

3.4.1 PX4 Autopilot Integration

PX4 Autopilot version 1.15 serves as the primary flight controller, providing comprehensive flight control capabilities ranging from basic stabilization to complex trajectory tracking [6]. The system's architecture enables seamless integration with high-level path planners through standardized interfaces that accept position, velocity, and attitude commands. PX4's role extends beyond simple command execution; it encompasses attitude control, motor mixing, safety monitoring, and failsafe management—all critical components for safe autonomous operation [14].

The controller's sophisticated trajectory tracking capabilities enable precise execution of complex flight paths generated by the planning layer. Through its cascaded control architecture, PX4 translates desired trajectories into appropriate thrust and torque commands while accounting for vehicle dynamics, actuator constraints, and environmental disturbances.

3.4.2 Extended Kalman Filter State Estimation

Central to the system's navigation performance is the Extended Kalman Filter 2 (EKF2) module, which provides optimal sensor fusion and state estimation. The EKF2 addresses a fundamental challenge in autonomous navigation: combining high-frequency but drift-prone inertial measurements with accurate but low-frequency external positioning data [15].

The sensor fusion process leverages the complementary characteristics of different sensor modalities. High-rate IMU data (typically 200-1000 Hz) provides excellent short-term motion tracking but suffers from integration drift over time [16]. Conversely, the SLAM-derived odometry offers superior long-term accuracy but operates at significantly lower frequencies (20 Hz). The EKF2 optimally combines these data streams using a probabilistic framework that accounts for sensor noise characteristics and measurement uncertainties [17].

This fusion approach yields a high-frequency, drift-corrected state estimate that maintains both the responsiveness necessary for agile flight control and the accuracy required for precise navigation. The resulting odometry solution serves as the foundation for both control feedback and path planning, ensuring consistency across the entire navigation pipeline.

3.4.3 IMU Gravity Compensation and Uncertainty Modeling

A critical consideration in UAV navigation systems involves the proper handling of gravitational effects in IMU measurements and the subsequent propagation of measurement uncertainties to downstream planning algorithms. The accelerometer measurements from the IMU inherently include gravitational acceleration, which must be estimated and compensated to obtain the vehicle's true dynamic acceleration.

The gravity compensation process introduces additional uncertainty, particularly in the vertical direction where gravitational effects are most pronounced. Unlike horizontal accelerations that primarily reflect vehicle dynamics, the vertical acceleration measurement requires subtraction of the estimated gravitational component:

$$\mathbf{a}_{dynamic} = \mathbf{a}_{measured} - \mathbf{g}_{estimated}$$

where $\mathbf{a}_{dynamic}$ represents the true vehicle acceleration, $\mathbf{a}_{measured}$ is the raw IMU measurement, and $\mathbf{g}_{estimated}$ is the estimated gravitational acceleration vector.

This compensation process inherently introduces larger uncertainties in the vertical direction compared to horizontal directions, as any errors in gravity estimation directly affect the computed vertical acceleration. Additionally, the coupling between attitude estimation errors and gravity vector projection creates further uncertainty amplification in the Z-axis direction.

To account for these physical limitations, the odometry data transmitted to the path planning module includes appropriately scaled covariance matrices that reflect the increased uncertainty in vertical state estimates. Specifically, the Z-axis position

and velocity covariance values are increased by a factor of 10 compared to horizontal components:

$$\mathbf{\Sigma}_{odometry} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \alpha\sigma_z^2 \end{bmatrix}$$

where $\alpha \geq 2$ represents the uncertainty scaling factor for the vertical direction.

This uncertainty modeling serves a dual purpose: first, it provides the path planning algorithm with realistic estimates of state estimation reliability, and second, it encourages the generation of trajectories with increased vertical clearance margins. By explicitly communicating the higher uncertainty in Z-axis measurements, the planning system can make informed decisions about trajectory feasibility and safety margins, particularly when navigating in vertically constrained environments.

3.5 Dynamic Path Planning and Trajectory Generation

The path planning subsystem represents the cognitive layer of the autonomous navigation architecture, responsible for generating safe, feasible, and optimal trajectories that guide the UAV from its current position to designated waypoints while avoiding environmental obstacles.

3.5.1 Fast-Planner Framework

Fast-Planner serves as the primary trajectory generation engine, chosen for its robust performance in complex three-dimensional environments and its proven track record in autonomous navigation applications [2]. The planner operates on a continuous cycle, receiving high-frequency odometry data from the sensor fusion pipeline and registered point cloud data from the SLAM module to maintain current environmental awareness.

The system outputs comprehensive trajectory specifications including position, velocity, acceleration, and yaw angle commands at each time step. This detailed trajectory information enables the flight controller to execute smooth, dynamically feasible paths while maintaining optimal energy efficiency and flight stability [22]. The integration between planner and controller through standardized trajectory interfaces ensures consistent performance across different flight scenarios.

3.5.2 Topological Path Planning Strategy

Among Fast-Planner's available planning modes, topological path searching was selected for its superior robustness and comprehensive solution space exploration. This approach excels in highly cluttered environments where naive direct-path strategies would inevitably encounter obstacles [23].

The topological planning algorithm initiates with a direct line-of-sight trajectory

toward the target destination. Upon detecting obstacle interference, the system systematically explores alternative topological structures within the environment, generating candidate paths that navigate around or over obstacles while maintaining dynamic feasibility constraints [24]. This approach ensures that even in severely constrained environments, the planner can identify viable navigation solutions that might be missed by more conventional search strategies.

The iterative refinement process continues until a collision-free, dynamically feasible trajectory is identified, or until the search space is exhaustively explored. This methodology provides both computational efficiency for straightforward scenarios and robust solution discovery for complex navigation challenges, making it well-suited for the dense forest environment employed in this study.

Figure 3.5 demonstrates the Fast-Planner's topological path planning capabilities in a complex obstacle field. The visualization shows the generated trajectory (red path) navigating through the environment while avoiding obstacles represented by the colorful voxel structures. The planner successfully identifies a feasible path that maintains safe clearance from all detected obstacles while adhering to the UAV's dynamic constraints.

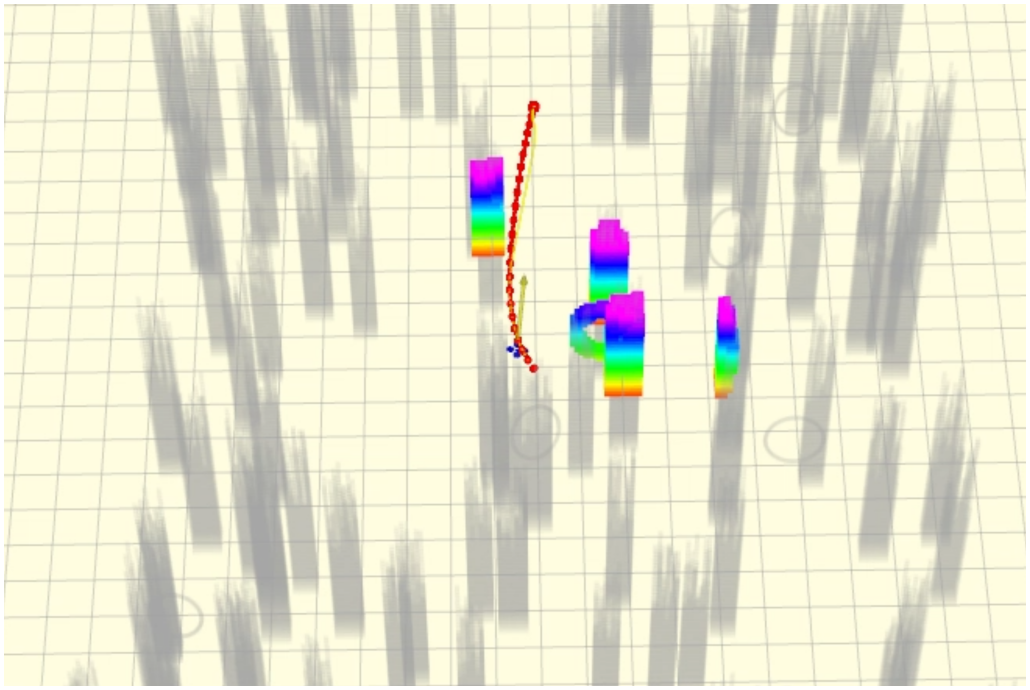

Figure 3.5: Fast-Planner trajectory generation demonstration showing the planned path (red) navigating through a simulated random forest environment. The colorful voxel structures represent detected obstacles, and the algorithm successfully generates a collision-free trajectory while maintaining dynamic feasibility.

3.5.3 Real-Time Performance Considerations

The path planning system operates under strict real-time constraints to ensure responsive navigation behavior. The 20 Hz point cloud update rate from the SLAM

system establishes the fundamental replanning frequency, while the high-frequency odometry enables continuous trajectory refinement between major replanning cycles [25].

The computational architecture balances planning horizon length with update frequency to maintain real-time performance while providing sufficient lookahead for safe navigation. Trajectory feasibility constraints account for vehicle dynamic limitations, ensuring that generated paths remain within the UAV's operational envelope while optimizing for factors such as flight time, energy consumption, and passenger comfort where applicable [26].

3.6 Inter-Process Communication Architecture

Effective data exchange between heterogeneous software components is paramount for maintaining real-time performance and system reliability. This section describes the communication infrastructure that enables seamless integration across different ROS versions and processing modules, following the data pipeline established by the SLAM and path planning systems.

3.6.1 ROS 2 Sensor Data Pipeline

The simulation environment operates primarily within the ROS 2 Jazzy framework, which handles all sensor data acquisition and initial processing. IMU measurements are published on the `/velodyne/imu` topic using the standardized `sensor_msgs/msg/Imu` message format, providing high-frequency inertial data at rates suitable for control applications. Simultaneously, the 128-line LiDAR generates dense point cloud data published on `/velodyne/lidar` using the `sensor_msgs/msg/PointCloud2` format, delivering comprehensive environmental observations at 20 Hz.

3.6.2 Custom UDP Bridge Implementation

The integration of ROS 1-based navigation algorithms with the ROS 2 simulation environment necessitated the development of a custom communication bridge. After extensive testing of existing solutions, including official ROS bridge packages, it was determined that a purpose-built UDP-based bridge offered superior performance and reliability for our specific data streams and timing requirements.

The custom bridge implements bidirectional data translation with optimized serialization protocols for each data type. Critical data flows managed by the bridge include:

- **Sensor Data Forward Path:** High-frequency IMU measurements and dense LiDAR point clouds are efficiently converted from ROS 2 to ROS 1 formats with minimal latency.
- **State Estimation Feedback:** SLAM-generated odometry data flows from ROS 1 back to ROS 2 for integration with PX4's state estimation pipeline.

- **Control Command Path:** Planned trajectories generated by ROS 1-based algorithms are transmitted to ROS 2 for execution by the flight controller.
- **System Monitoring:** Bidirectional status and diagnostic information ensures system health monitoring across all components.

3.6.2.1 Latency Analysis and Performance Characteristics

A critical consideration in the bridge design is the introduction of communication latency between ROS 1 and ROS 2 components. Since both ROS distributions operate on the same computational platform, the UDP bridge serves purely as an interprocess communication mechanism rather than a network transport layer. Comprehensive latency measurements demonstrate that the bridge introduces minimal temporal overhead, typically ranging from 2-4 milliseconds for standard message types.

The bridge implementation employs timestamp preservation strategies to maintain temporal consistency across the system. Specifically, all sensor data packets include the original acquisition timestamp from the data source, ensuring that downstream algorithms can accurately account for any introduced delays. This approach effectively decouples the bridge-induced latency from the temporal accuracy requirements of the navigation algorithms.

For high-frequency data streams such as IMU measurements (typically 200-400 Hz) and LiDAR point clouds (10-20 Hz), the sub-5ms bridge latency represents a negligible fraction of the sensor sampling periods. Moreover, the deterministic nature of the UDP communication on localhost ensures consistent latency characteristics, avoiding the temporal jitter that could degrade filter performance in the SLAM and state estimation algorithms.

The UDP-based implementation provides several advantages over alternative approaches: reduced computational overhead compared to serialization-heavy bridges, configurable quality-of-service parameters for different data types, and simplified debugging and monitoring capabilities. Most importantly, the modular design facilitates future migration to pure ROS 2 implementations without requiring architectural modifications to other system components.

3.6.3 Coordinate System Transformation

A critical consideration in the system integration involves the fundamental difference in coordinate system conventions between ROS 2 and PX4 Autopilot. This discrepancy necessitates careful coordinate transformations to ensure consistent data interpretation across all system components.

ROS 2 employs the East-North-Up (ENU) coordinate system convention, where:

- X-axis points towards East
- Y-axis points towards North
- Z-axis points towards the sky (Up)

Conversely, PX4 Autopilot utilizes the North-East-Down (NED) coordinate system convention, where:

- X-axis points towards North
- Y-axis points towards East
- Z-axis points towards the ground (Down)

Furthermore, to facilitate conversions between quaternion and axis-angle representations—particularly relevant for interpreting and constructing rotations—the following relationship can be used:

$$q = \left[\cos\left(\frac{\theta}{2}\right),\ \sin\left(\frac{\theta}{2}\right)\mathbf{u}\right]$$

where:

- θ is the rotation angle (in radians),
- $\mathbf{u} = (x, y, z)$ is the unit vector representing the axis of rotation.

Conversely, given a unit quaternion $q = [w,\ \vec{v}] = [w,\ x,\ y,\ z]$, the axis-angle representation can be recovered as:

$$\theta = 2 \cdot \arccos(w), \quad \mathbf{u} = \frac{\vec{v}}{\sin\left(\frac{\theta}{2}\right)} = \frac{(x,\ y,\ z)}{\sqrt{1 - w^2}}$$

This formulation assumes q is a unit quaternion (i.e., $\|q\| = 1$) and is widely used for interpreting small rotational deltas or composing larger attitude transformations.

A quaternion, when used to represent attitude, can be interpreted as the result of rotating the vehicle from an initial horizontal orientation aligned with the X-axis, about a specific axis by a certain angle. The axis and angle can be extracted directly from the quaternion using the method above.

As previously defined, the ENU and NED coordinate systems exhibit opposing conventions in their horizontal axes. Specifically, the X- and Y-axes of these frames point in reversed directions. This discrepancy implies that the quaternion representing a vehicle's orientation—when interpreted from one coordinate system—must be appropriately transformed to maintain consistency.

A common scenario arises in transforming a quaternion expressed in the ENU frame into the equivalent quaternion in the NED frame. In such cases, the orientation quaternion often describes the rotation from the **Y-axis** to the desired attitude in the ENU convention. However, the goal in the NED system is to determine the rotation from the **X-axis** to the same desired attitude. Since the vehicle's physical attitude remains unchanged, the transformation can be decomposed into two sequential rotations:

1. Apply a **90-degree rotation about the Z-axis** to reorient the vehicle from facing along the X-axis to the Y-axis in ENU.

2. Multiply this rotation with the provided quaternion to yield the desired rotation from the X-axis in the NED frame.

This process is visualized in Figure 3.6. Conversely, converting from NED to ENU follows an analogous process:

1. Apply a **-90-degree rotation about the Z-axis** to reorient from the X-axis to the Y-axis (in NED coordinates).
2. Multiply this rotation with the provided quaternion to recover the equivalent ENU-frame representation.

This inverse operation is illustrated in Figure 3.7. In both conversions, the underlying principle remains the same: maintain the vehicle's physical orientation while changing the reference axis used for interpreting the quaternion.

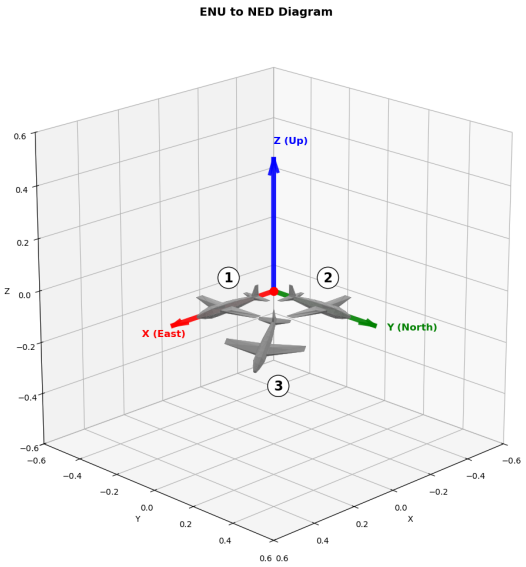

Figure 3.6: Transforming a quaternion from ENU to NED coordinates via a +90° rotation about the Z-axis.

These transformations ensure consistency when interpreting vehicle attitudes across systems that adopt different coordinate conventions, such as ROS 2 and PX4. The use of fixed-axis pre-rotations allows seamless quaternion adaptation without altering the physical orientation of the vehicle.

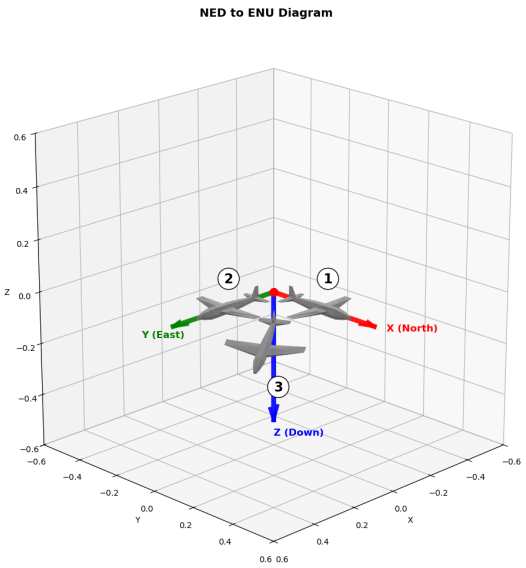

Figure 3.7: Transforming a quaternion from NED to ENU coordinates via a $-90°$ rotation about the Z-axis.

4

Evaluation of the Modular Navigation System

4.1 Quantitative Performance Metrics

The autonomous navigation system was evaluated through 100 independent trials in the dense forest simulation environment. The following results summarize system reliability, accuracy, and computational performance.

Table 4.1: Overall Mission Performance

Outcome	Count	Percentage
Successful Completions	77	77%
Mission Failures	19	19%
Timeout Events	4	4%
Total Trials	100	100%

Table 4.2: Pose Estimation Error Statistics

Metric	Position RMSE (m)	Rotation RMSE (°)
Mean	2.11	8.46
Std. Dev.	0.55	5.13
Min	0.11	1.12
Max	2.96	27.42
Variance	0.31	26.34

Table 4.3: Trajectory Tracking Error

Statistic	Tracking RMSE (m)
Mean	0.21
Std. Dev.	0.05
Min	0.17
Max	0.48
Variance	0.002

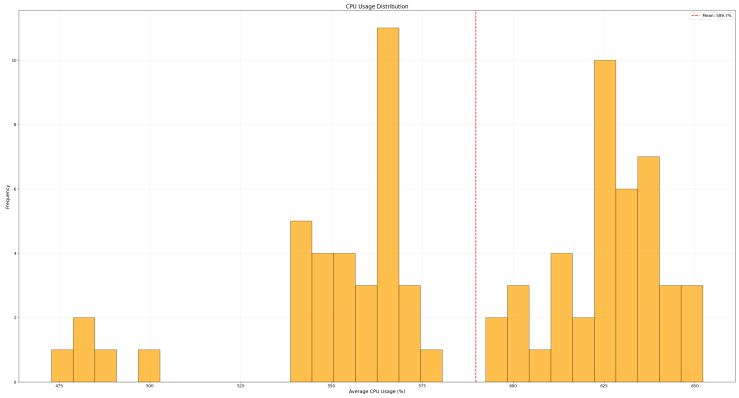

Figure 4.1: Distribution of CPU Usage During Navigation Trials

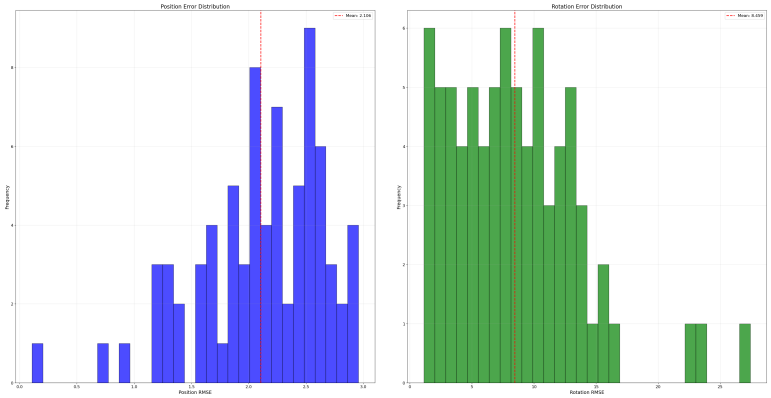

Figure 4.2: Pose Estimation Error Distribution (Position and Rotation)

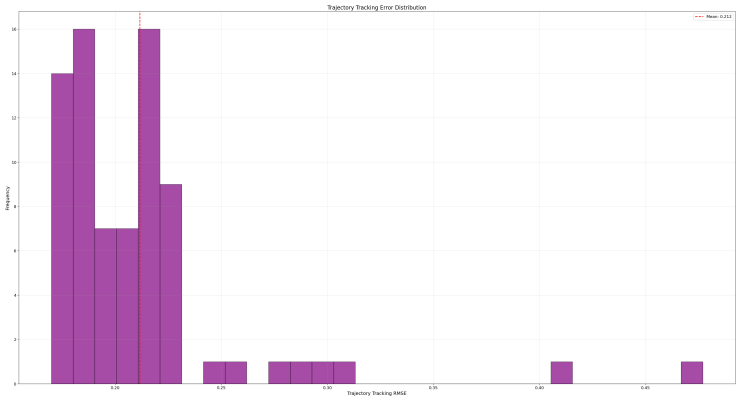

Figure 4.3: Trajectory Tracking Error Distribution

4.1.1 Performance Distribution Analysis

During the navigation experiments, the average CPU usage was 589.7%, corresponding to approximately 49% of the total 12-core system capacity. The distribution analysis reveals that CPU load remained consistently high throughout the trials, with most measurements clustered closely around the mean value (Figure 4.1).

Position errors exhibited an approximately normal distribution centered around 2.1 meters, while rotation errors were right-skewed, with the majority of values below 10 degrees (Figure 4.2). The trajectory tracking errors were notably consistent, with most values concentrated below 0.25 meters (Figure 4.3), demonstrating reliable execution of planned paths by the flight controller.

These quantitative results establish a comprehensive baseline for evaluating the modular navigation system's performance and provide the foundation for the detailed failure mode analysis presented in subsequent sections.

4.2 Overall System Performance

The autonomous navigation system demonstrated substantial success across multiple experimental trials, validating the fundamental feasibility of the integrated architecture. The majority of navigation attempts successfully achieved the objective of traversing the dense forest environment from the designated starting point to the target destination without collision incidents. These successful missions confirm that the combination of FasterLIO SLAM, Fast-Planner trajectory generation, and PX4 flight control can effectively operate in complex three-dimensional environments with dense obstacle distributions [13].

However, despite the predominantly positive outcomes, occasional collision events were observed during the experimental evaluation. These incidents provide valuable insights into the system's operational limitations and highlight areas requiring further development. The following analysis examines the primary failure modes identified through systematic evaluation of both successful and unsuccessful navigation attempts, following established methodologies for autonomous navigation assessment [3].

4.3 Failure Mode Analysis

While the autonomous navigation system achieved satisfactory performance in most scenarios, understanding the failure mechanisms is crucial for improving system reliability and safety margins. Through detailed analysis of collision incidents and near-miss events, three primary failure modes have been identified, each contributing to navigation failures under different operational conditions. This systematic failure analysis approach aligns with established practices in autonomous robotics evaluation [4, 17].

4.3.1 SLAM System Localization Errors

The most significant contributor to navigation failures stems from inherent limitations in the SLAM system's pose estimation accuracy. While FasterLIO provides robust odometry estimates under normal operating conditions, the system cannot achieve perfect localization precision due to sensor noise, computational constraints, and environmental factors [1, 18].

Under most circumstances, moderate localization errors remain within acceptable tolerance bounds and do not compromise navigation safety. When the UAV maintains substantial clearance from obstacles, small pose estimation deviations are easily compensated by the path planning system's safety margins. However, critical failures occur when localization errors manifest during close-proximity obstacle encounters, particularly during active trajectory replanning phases.

In dense forest scenarios, the UAV frequently encounters situations requiring immediate trajectory adjustments to avoid newly detected or approaching obstacles. During these critical moments, even modest localization errors can result in insufficient avoidance maneuvers [12]. The cumulative effect of position uncertainty and velocity estimation errors can cause the planning system to underestimate the required avoidance amplitude, leading to trajectory solutions that appear collision-free based on the estimated state but prove inadequate given the actual vehicle position.

4.3.2 Flight Controller Trajectory Tracking Errors

The second major failure mode arises from the inherent limitations in trajectory tracking performance exhibited by the PX4 flight control system. Even when the path planning module generates theoretically optimal collision-free trajectories, the physical UAV cannot achieve perfect trajectory following due to actuator constraints, aerodynamic disturbances, and control system bandwidth limitations [6, 22].

Under nominal conditions with sparse obstacle distributions, these tracking errors remain within acceptable bounds and do not compromise mission safety. The generous clearance margins typically maintained in open environments can accommodate the modest deviations between planned and executed trajectories. However, the dense forest environment presents a fundamentally different challenge where clearance margins are severely constrained by the proximity of vegetation and obstacles.

In such scenarios, trajectory tracking errors that would be inconsequential in open space can result in collision incidents [14]. The control system's inability to precisely follow rapid trajectory changes—particularly during aggressive avoidance maneuvers—becomes a limiting factor for safe navigation in cluttered environments. This limitation is further exacerbated during high-speed flight segments where control response delays and actuator saturation effects become more pronounced.

4.3.3 Path Planning Algorithm Limitations

Perhaps most concerning from a system design perspective is the identification of inherent limitations within the path planning algorithm itself. Even under ideal-

ized simulation conditions where perfect trajectory tracking is assumed—eliminating both SLAM localization errors and flight controller tracking deviations—occasional collision events still occurred during experimental trials.

These incidents reveal fundamental limitations in the topological path planning approach employed by Fast-Planner [2]. While the algorithm demonstrates robust performance in most scenarios, certain environmental configurations can challenge its obstacle avoidance capabilities. The discrete nature of the planning algorithm's environmental representation, combined with finite computational resources and real-time constraints, can occasionally result in trajectory solutions that appear collision-free during planning but prove inadequate during execution [23, 24].

The 20 Hz update frequency of the planning system, while suitable for most navigation scenarios, may prove insufficient for handling rapidly changing environmental conditions or high-speed flight through extremely dense obstacle fields [25]. Additionally, the algorithm's reliance on discrete voxel-based environmental representations can introduce artifacts where obstacle boundaries are not perfectly captured, potentially leading to trajectories that pass closer to obstacles than intended.

4.3.4 LiDAR Sensor Coverage Limitations

A particularly critical failure mode stems from the inherent limitations in LiDAR sensor coverage, which creates blind spots in environmental perception. In the implemented system, the 128-line LiDAR sensor configuration provides comprehensive coverage spanning from 90 degrees upward to 15 degrees downward relative to the horizontal plane. While this configuration ensures excellent forward and lateral obstacle detection, it creates a significant blind zone directly beneath the UAV.

This sensor blind spot becomes problematic when the path planning algorithm generates trajectories that involve significant altitude changes, particularly rapid descents. Since the planning module relies exclusively on the registered point cloud data for obstacle detection, areas not covered by the LiDAR sensor appear as free space in the environmental representation. Consequently, the algorithm may generate descent trajectories that bring the UAV dangerously close to or into collision with obstacles located directly below the vehicle.

The blind zone issue is exacerbated in dense forest environments where tree canopies, fallen logs, or ground-level vegetation may not be detected until the UAV descends into the sensor's detection range. By this time, the proximity to obstacles may be insufficient for the planning system to generate effective avoidance maneuvers, particularly given the limited vertical clearance margins available in cluttered environments.

This limitation highlights a fundamental challenge in sensor-based navigation systems: the trade-off between sensor coverage completeness and practical implementation constraints. While omnidirectional sensing would theoretically eliminate blind spots, such configurations are often impractical due to weight, power, and computational constraints typical of UAV platforms.

As illustrated in Figure 4.4, the 128-line LiDAR sensor provides excellent coverage

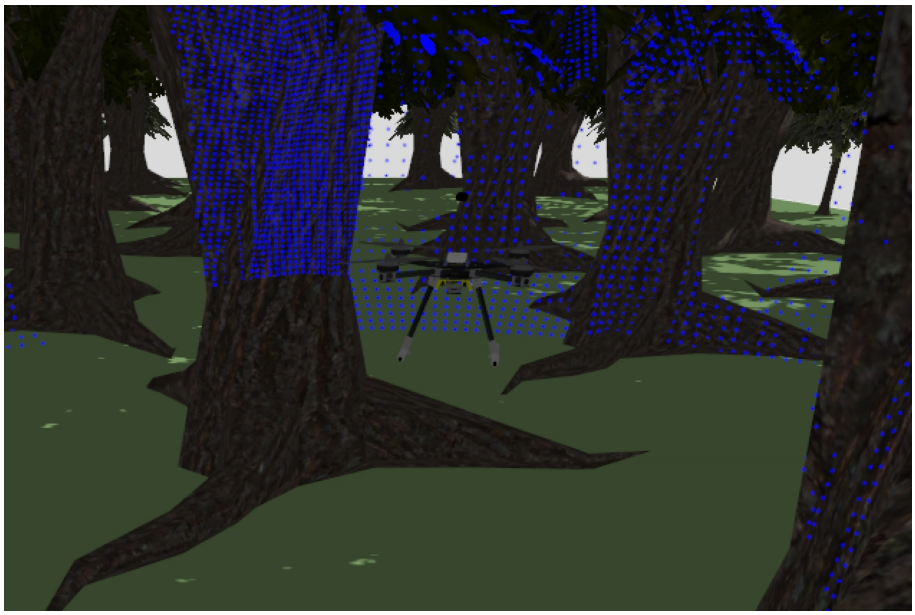

Figure 4.4: Example of a UAV navigating in a dense forest environment using a 128-line LiDAR sensor. The blue points indicate the LiDAR point cloud; coverage is comprehensive laterally and forward but leaves a significant blind zone directly beneath the UAV. As a result, obstacles under the UAV—such as tree roots, fallen logs, or uneven ground—may remain undetected until the vehicle descends, limiting reaction time for safe avoidance.

in forward and lateral directions, while a distinct blind spot exists directly below the UAV. In cluttered environments such as forests, obstacles within this blind zone may remain undetected until the UAV is in close proximity, increasing the risk of collision during rapid descent or low-altitude maneuvers.

4.4 Performance Implications and System Limitations

The identified failure modes collectively highlight the challenges inherent in autonomous navigation systems operating in complex real-world environments. While each subsystem—SLAM, path planning, and flight control—performs adequately within its design parameters, the integration of these components introduces systemic challenges that can compromise overall reliability [4].

The cumulative effect of individual subsystem errors represents a fundamental challenge for autonomous navigation in constrained environments. Small errors in each component can combine constructively, leading to system-level failures even when individual components operate within their specified tolerances [17]. This phenomenon underscores the importance of robust error propagation analysis and the implementation of appropriate safety margins throughout the system architecture.

Furthermore, the real-time constraints imposed by autonomous flight applications

limit the computational resources available for error detection and correction. While more sophisticated algorithms might theoretically provide improved performance, the practical requirement for real-time operation constrains the complexity of feasible solutions, necessitating trade-offs between computational efficiency and navigation precision [26].

5

Discussion and Conclusions on the Modular Approach

5.1 Overall System Performance

The evaluation of the autonomous navigation system has yielded valuable insights into the practical challenges of deploying robust UAV navigation in complex environments. The developed modular architecture—integrating LiDAR-based SLAM, trajectory planning, and PX4 flight control—demonstrated reliable performance in many test scenarios and validated the feasibility of modular systems for UAV navigation in cluttered or GPS-denied settings. At the same time, the experiments revealed several important limitations that shape the broader understanding of system behavior and potential failure modes.

5.2 Identified System Limitations

5.2.1 Error Accumulation Effects

The experiments demonstrate that small inaccuracies in individual subsystems can compound over time, ultimately leading to system-level failures. This highlights the need for careful error budgeting, tighter system integration, and improved robustness at the architecture level.

5.2.2 Real-Time Computational Constraints

Meeting strict real-time requirements imposes fundamental trade-offs between algorithmic sophistication and computational feasibility. This limits the complexity of perception, mapping, and planning algorithms that can be deployed reliably on embedded platforms.

5.2.3 Sensor Update Rate Limitations

The 20 Hz update rate of the LiDAR-based SLAM system forms a performance bottleneck in scenarios requiring fast reaction times. Limited temporal resolution constrains navigation responsiveness and reduces safety margins in highly dynamic or densely cluttered environments.

5.2.4 Environmental Complexity Challenges

Dense forest environments expose the boundaries of current geometric mapping and planning methods, revealing situations where traditional assumptions about obstacle structure or free-space connectivity no longer hold.

These observations collectively underscore the difficulty of ensuring reliable autonomous navigation in real-world conditions and emphasize the need for further research into improving both system robustness and environmental adaptability.

5.3 Architectural Insights and Lessons Learned

5.3.1 Hierarchical Analysis of System Bottlenecks

A deeper examination of the failure cases reveals that SLAM accuracy, while important, is not the dominant bottleneck in the navigation pipeline. As long as relative pose estimates remain locally consistent, the system can typically maintain reasonable navigation performance. More fundamental constraints arise from limited sensor update rates and from planned trajectories that may be infeasible to execute in densely cluttered environments.

5.3.2 Information Flow and Abstraction Limitations

An inherent characteristic of modular navigation architectures is the staged processing of sensor data into increasingly abstract representations. Raw sensory information is progressively distilled through predefined interfaces—from sensor measurements to geometric maps, from maps to planned paths, and from paths to low-level control inputs. Each abstraction step necessarily discards certain information that might otherwise support more informed decision-making.

This structured information flow, while valuable for modularity, interpretability, and system maintainability, also restricts the system's ability to utilize the full richness of the available sensory data. The SLAM subsystem may omit perceptual cues not relevant to geometric reconstruction; the planner reasons over simplified spatial abstractions that do not account for uncertainty or scene context; and the controller operates without direct access to the original sensory evidence that informed upstream decisions.

This progressive information reduction constitutes a fundamental limitation of modular designs. Regardless of how well individual modules are tuned, the architecture inherently constrains the amount and type of information that can propagate through the system. These findings point to important directions for improving future navigation pipelines, particularly regarding tighter integration between perception, planning, and control.

6

Future Work and Exploratory Studies on Vision-Based End-to-End Navigation

This chapter presents future research directions for vision-based end-to-end navigation and summarizes the preliminary exploratory studies conducted during this thesis. While the primary focus of the work has been the development and evaluation of a LiDAR–SLAM-based modular navigation system, several vision-driven end-to-end approaches were examined at a conceptual and prototype level. These exploratory investigations provide insight into how learning-based methods may leverage raw visual data and highlight both the potential and the current limitations of vision-based navigation in GPS-denied railway environments.

6.1 Motivation for End-to-End Navigation

The findings from the modular system highlight several structural limitations, particularly the fact that information flow between perception, mapping, planning, and control is predetermined by human-designed interfaces. Because each module receives only a predefined abstraction of upstream sensor data, the system cannot fully exploit the richness of the original measurements. While LiDAR provides geometrically precise and stable input for modular pipelines, its representational capacity is fundamentally limited to spatial structure. In contrast, visual sensors capture far more information about the environment, including appearance, texture, context, and semantic cues that are discarded in conventional SLAM–planning workflows.

Recent rapid advances in computer vision and representation learning suggest that vision-based models may increasingly overcome the traditional drawbacks of visual sensing—such as sensitivity to illumination or motion blur—and may eventually surpass LiDAR in tasks requiring high-level scene understanding. Task-driven end-to-end learning approaches, which map raw visual observations directly to control actions, allow the navigation strategy and information extraction process to be learned jointly rather than imposed by predefined interfaces. This creates the possibility of leveraging the full expressive power of visual data and achieving performance beyond the limits of modular architectures.

Although a complete vision-based end-to-end navigation system is beyond the scope

of this thesis, these trends provide a strong motivation for exploring such methods in future research.

6.2 Technical Pathways for End-to-End Learning

Based on current literature and the exploratory studies conducted in this thesis, three representative technical pathways for end-to-end learning-based navigation were examined and analyzed:

6.2.1 Next-Token Prediction Models

Self-supervised learning (SSL) has become a central paradigm in modern computer vision, enabling models to acquire visual representations without human-annotated labels. Unlike supervised approaches that rely on explicit ground truth, SSL exploits intrinsic structure in data—such as spatial redundancy, temporal coherence, or masked content—to formulate learning objectives. For sequential visual data such as video, a natural SSL objective is to predict future observations from past frames. This formulation is conceptually analogous to the next-token prediction objective used in large language models (LLMs), where predicting the next element in a sequence encourages the model to internalize latent structure.

DINOv3 Patch Embeddings. To construct a tokenized visual representation suitable for sequential modeling, this work employs DINOv3 patch embeddings [30] as the underlying visual tokens. DINOv3 provides rich, semantically structured features [30] and has demonstrated strong performance across a range of downstream computer vision tasks. Understanding the spatial organization of these pretrained embeddings is important for motivating their use as the foundation of our prediction model.

To qualitatively analyze the structure of the patch tokens, a 3D PCA projection was applied to the high-dimensional embeddings. As shown in Figure 6.1, the resulting pseudo-color maps exhibit coherent alignment with object boundaries, semantic regions, and scene geometry. This visualization highlights that the pretrained encoder already captures meaningful spatial structure, making it a strong candidate for use as the tokenization backbone for sequential prediction.

Architecture for Next-Token Prediction. Motivated by the success of next-token prediction in LLMs, this thesis investigates whether a similar paradigm can serve as a foundation for end-to-end visual navigation. The initial idea was to pretrain a model on large-scale online video data so that it could develop a generic understanding of visual dynamics, and subsequently fine-tune this representation via few-shot supervised learning or reinforcement learning for navigation-specific tasks.

Departing from decoder-only architectures used in language modeling, this work adopts an encoder–decoder design to better separate representation learning from future prediction. The encoder processes the first half of a video clip to produce a latent representation of the past, whereas the decoder autoregressively predicts the

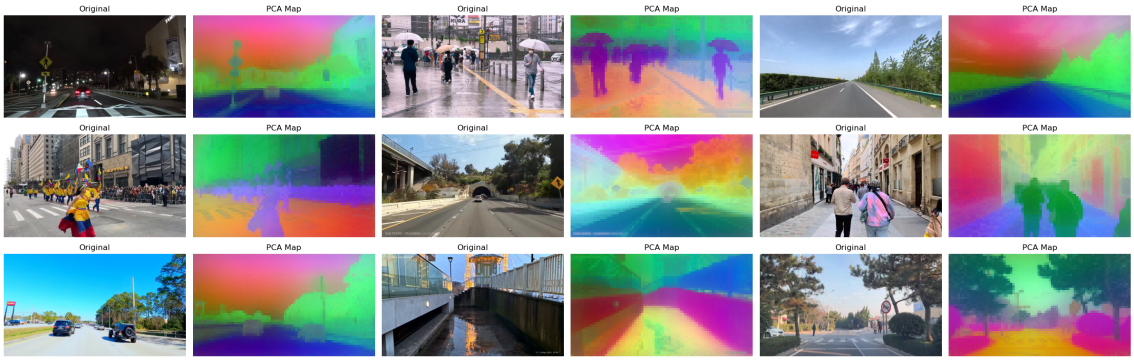

Figure 6.1: Qualitative visualization of DINOv3 patch-token embeddings. For each sampled frame, the left image shows the original RGB input, and the right image shows a 3D PCA projection of the corresponding patch tokens.

patch embeddings corresponding to the second half. This design aims to mitigate the short-horizon bias commonly observed in purely decoder-only models.

To support real-time robotic inference, the encoder uses a decoder-style masked attention mechanism, enabling the use of key–value (KV) caching. This allows the model to incrementally integrate new frames while reusing cached representations from previous steps, significantly reducing computation. During inference, the decoder anticipates future observations, while the encoder continuously incorporates both historical and predicted frames, forming a coherent and evolving representation of the scene. A schematic illustration of the architecture is shown in Figure 6.2.

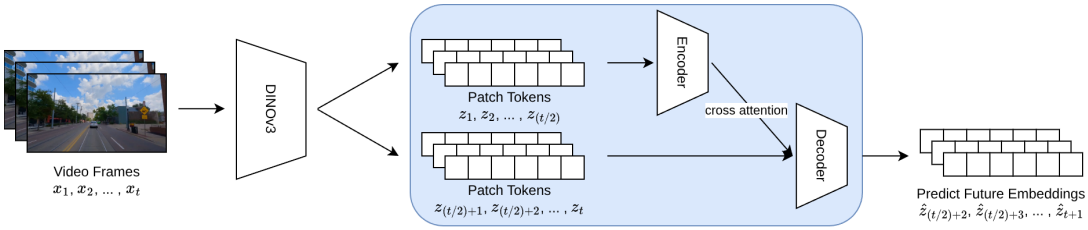

Figure 6.2: Overview of the proposed next-token prediction architecture. The encoder processes patch embeddings from the first half of a video clip, while the decoder predicts patch embeddings of the future frames. Cross-attention allows predictions to be conditioned on the encoded past, forming a lightweight world-modeling structure suitable for robotic navigation.

Dataset and Experiments. To support this prototype, a mixed video dataset was assembled to expose the model to diverse motion patterns. Two publicly available datasets were selected to represent distinct motion regimes: Sekai-Real-Walking-HQ [32] for low-speed egocentric motion, and OpenDV [33] for higher-speed driving dynamics. For each dataset, 100 videos were sampled to ensure balanced coverage. An automated YouTube collection pipeline was also implemented to harvest additional video clips based on keyword and resolution constraints, though the prototype experiments primarily used the curated mixed dataset.

The model consumes DINOv3 patch embeddings as input tokens. The encoder receives embeddings from earlier frames, and the decoder autoregressively predicts embeddings for future frames. The working hypothesis was that by learning to forecast future visual features, the model would internalize motion dynamics and environmental structure, effectively functioning as a world model.

However, the experiments revealed fundamental limitations. The model quickly converged to a trivial solution—copying input tokens—because consecutive video frames are often highly similar, allowing loss minimization without learning meaningful temporal structure. Further, real-world video future prediction is inherently ambiguous due to occlusions, sudden motion, and the appearance of new objects. These ambiguities make future prediction substantially more difficult than language modeling, where the next token is more tightly constrained by context.

Conclusion. Overall, these findings indicate that pure next-token prediction on visual features is fundamentally unsuitable as the core learning objective for world modeling in navigation. The trivial copy solution prevents meaningful temporal abstraction, while the inherent uncertainty and multimodality of future video frames make prediction ill-posed without additional structural constraints. Consequently, this approach cannot form a stable foundation for downstream control or planning. Therefore, next-token prediction is not a feasible pathway for robust end-to-end navigation within the scope of this thesis.

6.2.2 Self-Supervised Learning Based on Self-Distillation

Self-supervised learning (SSL) provides a general framework for learning visual representations without labeled data by exploiting the inherent structure of images or video. Among the many SSL paradigms, self-distillation has emerged as a particularly effective approach in recent vision models such as DINO [31] and related teacher–student architectures. In self-distillation, a network is trained to produce consistent feature representations across different augmentations or views of the same input, encouraging invariance to appearance, lighting, and geometric transformations.

In this work, the motivation for exploring self-distillation was twofold. First, the same mixed video dataset used for the next-token prediction experiments—consisting of Sekai-Real-Walking-HQ, OpenDV, and additional YouTube clips—offers diverse egocentric motion patterns that could support general-purpose representation learning. Second, the structure of video naturally provides multiple spatial "views" of the same scene: different regions of a single frame are captured by the same camera under identical environmental conditions. As illustrated in Figure 6.3, a single frame can be partitioned into several spatial tiles, each presenting a localized view that shares global illumination, camera motion, and scene geometry with the full frame. Intuitively, enforcing consistency across these tiles could encourage the model to extract higher-level information common to the entire scene, rather than overfitting to local textures.

To explore this idea, a prototype tile-consistency SSL objective was implemented. Each video frame was divided into a grid of spatial tiles, and the model was trained to

Figure 6.3: Illustration of the tile decomposition strategy used in the self-distillation prototype. A full video frame (left) is uniformly partitioned into spatial tiles (right), each providing a localized view captured by the same camera at the same time step.

produce consistent feature embeddings across tiles originating from the same frame. The intention was that, by sharing the same underlying camera pose and scene dynamics, the tiles would provide a natural source of positive pairs for self-distillation. In addition, the loss function incorporated a regularization term inspired by Meta's recent LeJEPA framework [34], which observes that well-behaved pretrained representations tend to follow approximately Gaussian feature distributions. A simplified SigReg-style regularization term was therefore included [34] to penalize collapsed feature statistics and stabilize training.The overall training pipeline is illustrated in Figure 6.4, which shows how spatial tiles are extracted from video frames, encoded into patch embeddings using DINOv3, passed through a lightweight temporal module, and finally optimized using a consistency loss augmented with SigReg regularization.

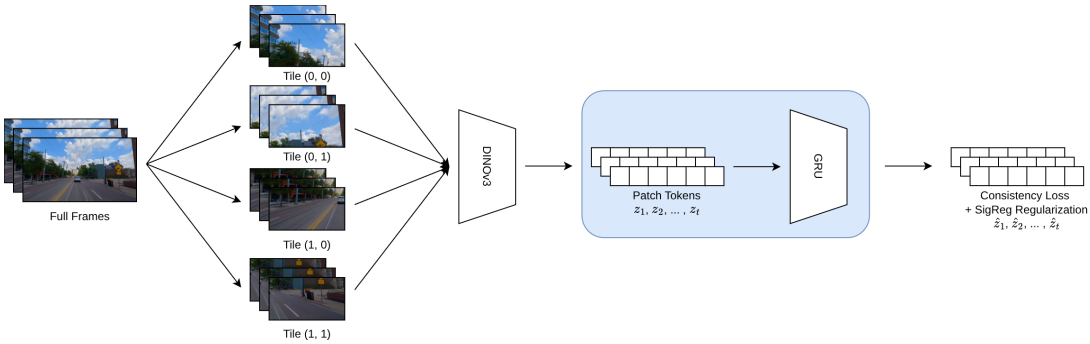

Figure 6.4: Overview of the tile-based self-distillation pipeline. Spatial tiles extracted from each video frame are processed by a pretrained DINOv3 encoder to obtain patch embeddings, which are then passed through a lightweight GRU module. The model is trained using a tile-consistency loss combined with SigReg regularization to stabilize representation learning and prevent collapse.

Despite these enhancements, the experiments revealed significant challenges. Unlike multi-view camera setups used in more sophisticated self-distillation pipelines, tiles extracted from a single image lack true viewpoint diversity and often break important global structures such as perspective geometry. As a result, the model tended to focus on highly localized texture patterns rather than extracting scene-level information relevant to camera motion or layout. Furthermore, even with SigReg-style regularization, the simplified training setup remained susceptible to collapse, with the model drifting toward representations that preserved local differences but failed

to encode meaningful global structure. In practice, stable self-distillation requires additional mechanisms such as momentum encoders, multi-scale prediction tasks, large-batch normalization, or explicit teacher–student separation—components that exceed the scope of this prototype.

Conclusion. In summary, although self-distillation is a powerful paradigm in large-scale vision models, the tile-based formulation explored here does not provide the geometric or viewpoint diversity needed for stable representation learning. The model gravitates toward local textures, fails to capture scene-level structure, and remains prone to collapse without sophisticated stabilization mechanisms such as momentum encoders, teacher–student separation, or large-batch normalization. These requirements exceed the computational and methodological scope of this project. As a result, tile-based self-distillation is not a viable route toward learning navigation-relevant visual representations within a Master's thesis setting.

6.2.3 Vision-Based Reinforcement Learning

Reinforcement learning (RL) offers a direct framework for training control policies through trial-and-error interaction, making it an appealing candidate for end-to-end navigation. When paired with modern visual encoders, vision-based RL enables policies to learn navigational behaviors directly from raw images, bypassing the need for manually engineered perception and planning modules. In principle, this approach allows the agent to extract task-relevant features automatically and to learn behaviors that integrate perception, prediction, and control within a unified model.

To evaluate the feasibility of this direction, a prototype pipeline was implemented using a pretrained DINOv3 encoder in combination with a Proximal Policy Optimization (PPO) policy network [35]. For each observation, DINOv3 first produced high-dimensional patch tokens, typically on the order of several hundred to a thousand dimensions. Such high-dimensional embeddings tend to hinder RL convergence due to the increased sample complexity and the instability of policy-gradient updates. To mitigate this issue, a 1×1 convolution was applied to downsample the patch-token dimension to a more compact latent space. A lightweight CNN then aggregated the spatial structure of the downsampled tokens into a single feature vector representing the visual observation. This compact representation was subsequently passed to the policy network, which output action distributions and value estimates. An overview of the resulting architecture is shown in Figure 6.5, illustrating the transformation from raw visual input to control actions.

However, several practical limitations emerged during experimentation. The most fundamental challenge lies in the simulation environment itself. Existing UAV or railway simulators lack the visual richness—lighting variation, clutter, material diversity, and complex geometry—present in real railway environments. As a result, the learned policy tended to overfit to specific textures and color patterns in the simulator and failed to generalize even under small appearance shifts. This highlights a significant domain gap: without realistic photometric and structural variation, the RL agent cannot associate visual cues with transferable navigational semantics.

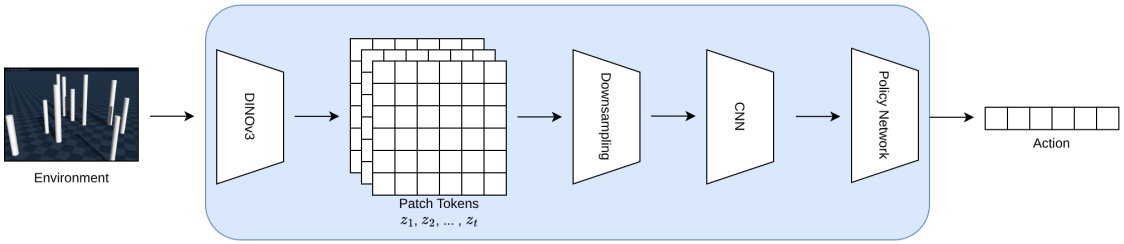

Figure 6.5: Overview of the vision-based reinforcement learning pipeline used in this thesis. Raw observations from the simulation environment are passed through a pretrained DINOv3 encoder to obtain patch-token embeddings. A 1×1 convolution downsampling layer reduces the token dimensionality, after which a lightweight CNN aggregates spatial information into a compact feature vector. The policy network, trained with PPO, outputs control actions based on this representation.

A second challenge concerns sample efficiency. Vision-based PPO requires a very large number of interactions to converge, especially when observations are high-dimensional. Each training step involves both a forward pass through DINOv3 and a policy update, making the process computationally expensive. Even though the encoder was frozen during training, the overall throughput remained limited, and the agent did not experience enough diverse trajectories to learn stable behavior. In long-horizon tasks such as UAV navigation, this issue becomes particularly pronounced, as meaningful rewards are often delayed and rare.

Finally, preliminary experiments indicated difficulties in achieving stable learning dynamics. Because the visual encoder is fixed and the simulator lacks diversity, the policy gradients often collapsed toward highly reactive or locally optimal behaviors—such as drifting toward visually distinctive textures—rather than learning navigational structure. These findings are consistent with known challenges in vision-based RL, where insufficiently diverse training environments and sparse reward structures can prevent policies from exploiting high-quality visual embeddings.

Additionally, Recent advances in generative world models have also explored the idea of training control policies inside learned, interactable video environments. Models such as Genie series [36] and Dreamer series [37] aim to capture environment dynamics through video prediction or latent imagination, allowing an agent to learn by acting within the generated visual world rather than relying on an external simulator. While these approaches offer an appealing alternative to traditional RL pipelines, they currently face substantial limitations. Maintaining temporal consistency, physical plausibility, and interactive fidelity in long-horizon imagined rollouts remains challenging, and errors in the generative model can quickly accumulate and mislead policy learning. Moreover, training such models requires computational resources far beyond what is commonly available in academic settings, often involving large-scale TPU clusters or multi-node GPU pipelines. For these reasons, generative world-model-based RL was not adopted as a primary direction in this thesis, though it remains an intriguing avenue for future research as the underlying technology matures.

Conclusion. The experiments demonstrate that the primary barrier to vision-based

reinforcement learning is not the RL algorithm or the policy architecture, but the simulation environment itself. The lack of photometric realism, geometric complexity, and appearance diversity in existing UAV simulators creates a severe reality gap that prevents policies from learning transferable visual semantics. Combined with the poor sample efficiency of vision-based PPO and the high computational cost of processing DINOv3 embeddings, the approach becomes impractical for long-horizon UAV tasks. Thus, vision-based RL is currently infeasible as a standalone path toward end-to-end navigation for this thesis.

7

Conclusion and Future Work

This thesis developed a high-performance modular navigation baseline for UAV railway inspection in GPS-denied environments and conducted targeted exploratory experiments on several emerging end-to-end learning approaches. Rather than delivering an end-to-end navigation system, the thesis clarifies the structural constraints inherent in current methods and identifies the core obstacles that must be addressed by future research.

The evaluation of the modular FasterLIO + Fast-Planner pipeline revealed that a state-of-the-art SLAM-based system can achieve robust performance in simulated dense-forest railway environments, while still exhibiting characteristic failure modes including localization drift, trajectory tracking deviations, LiDAR coverage gaps, and planner-induced geometric constraints. These observations establish a rigorous and practical baseline for benchmarking future learning-based approaches.

On the learning side, prototype experiments with predictive world models, self-supervised representation learning, and vision-based reinforcement learning revealed several structural bottlenecks: (1) predictive feature copying in temporal models, (2) representation collapse in self-distillation pipelines, and (3) simulator limitations that currently prevent robust Vision–RL. Importantly, while simulation platforms can and should improve substantially, *a perfectly realistic simulator is fundamentally impossible*. Real-world environments remain open-ended, stochastic, and influenced by long-tail phenomena that no simulator can fully capture. Therefore, long-term progress in embodied AI cannot rely solely on scaling simulation fidelity.

These findings motivate a shift in perspective toward more structured alternatives. Among the possible directions, neuro-symbolic methods offer a promising and conceptually aligned pathway for embodied intelligence. The remainder of this chapter discusses this direction in detail.

7.1 Future Directions: Neuro-Symbolic Approaches for Embodied Intelligence

Beyond the technical routes explored in this thesis, an emerging and potentially transformative direction for embodied AI lies in neuro-symbolic approaches—methods that combine neural representation learning with explicit rule-based reasoning. In this paradigm, the system first constructs a symbolic or rule-based abstraction of the

environment, and then uses these rules to constrain or guide decision-making. Such an approach differs fundamentally from current end-to-end neural methods, which treat perception, prediction, and control as continuous function-approximation problems.

Looking back at the history of AI, major breakthroughs have often come from models that respect the underlying structure of the data domain. Convolutional neural networks surpassed multilayer perceptrons on image recognition by exploiting the two-dimensional locality of images, avoiding unnecessary interactions between distant pixels. Transformers surpassed recurrent neural networks in sequence modeling by recognizing that long-range dependencies can be more effectively captured through pairwise attention rather than hidden-state recurrence. These examples highlight a recurring theme: progress emerges when model architectures are aligned with the structural properties of the domain.

In embodied intelligence, the central structural element is *action*. Unlike perception tasks—where models excel at pattern recognition without explicitly encoding rules—actions inherently carry intent, causality, and constraint. Humans, for instance, may struggle to articulate why a certain image depicts a dog rather than a cat, but can easily provide precise reasons for their own actions: braking to avoid a collision, lifting a cup to drink water, or turning away from an obstacle. Even habitual or reflexive actions can typically be traced back to meaningful causes. This suggests that, unlike pattern recognition, action-generation naturally benefits from having an intermediate layer of structured rules or symbolic abstractions.

Traditional neuro-symbolic AI has been limited by the non-differentiable nature of symbolic rules, which makes gradient-based optimization difficult. However, in embodied settings, the environment itself is already non-differentiable—robots must act in unknown, discontinuous, and often stochastic worlds. This removes a major barrier: the rule layer can be treated as part of the environment and optimized indirectly through reinforcement learning. In this formulation, neural networks provide powerful perceptual representations, while the symbolic or rule-based layer restricts the action space according to interpretable constraints. Effectively, the agent searches over a discrete or structured rule space rather than a high-dimensional continuous action space, which may significantly accelerate convergence, enhance generalization, and improve transfer from simulation to the real world.

This perspective also opens the possibility of achieving a degree of interpretability. If the rule layer can be inspected or analyzed, one might gain insight into the agent's decision-making logic—although this remains an open challenge, as defining the rule space and designing mechanisms for rule-driven action generation are far from solved problems.

Overall, neuro-symbolic approaches represent a promising future direction for robust embodied intelligence. By leveraging the strengths of pretrained neural models for perception while incorporating rule-based abstractions for reasoning and control, such methods may overcome several limitations observed in pure end-to-end systems, including sample inefficiency, poor generalization, and a lack of structural understanding. While the development of practical neuro-symbolic architectures

for real-world robotics is still in its early stages, the conceptual advantages suggest that this line of research may hold considerable potential for advancing autonomous navigation in complex environments.

Bibliography

[1] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao, "Faster-LIO: Lightweight Tightly Coupled Lidar-Inertial Odometry Using Parallel Sparse Incremental Voxels," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4861–4868, 2022.

[2] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[3] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. H. Lee, S. Lynen, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, S. Weiss, and L. Meier, "Vision-controlled Micro Flying Robots: From System Design to Autonomous Navigation and Mapping in GPS-denied Environments," *IEEE Robotics and Automation Magazine*, vol. 21, no. 3, pp. 26–40, 2014.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[5] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, an Open-source Multi-robot Simulator," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2149–2154, 2004.

[6] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A Micro Aerial Vehicle Design for Autonomous Flight Using Onboard Computer Vision," *Autonomous Robots*, vol. 33, no. 1–2, pp. 21–39, 2012.

[7] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, Architecture, and Uses in the Wild," *Science Robotics*, vol. 7, no. 66, 2022.

[8] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An Open-source Robot Operating System," in *Proceedings of the ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, p. 5, 2009.

[9] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based Visual–Inertial Odometry Using Nonlinear Optimization," *The Interna-*

tional Journal of Robotics Research, vol. 34, no. 3, pp. 314–334, 2015.

[10] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS—A Modular Gazebo MAV Simulator Framework," in *Robot Operating System (ROS): The Complete Reference*, pp. 595–625, Springer, 2016.

[11] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity Visual and Physical Simulation for Autonomous Vehicles," in *Field and Service Robotics*, pp. 621–635, Springer, 2018.

[12] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9135–9142, 2020.

[13] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar, "Fast, Autonomous Flight in GPS-denied and Cluttered Environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[14] B. T. Lopez and J. P. How, "Aggressive 3-D Collision Avoidance for High-speed Navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5759–5765, 2016.

[15] A. I. Mourikis and S. I. Roumeliotis, "A Multi-state Constraint Kalman Filter for Vision-aided Inertial Navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3565–3572, 2007.

[16] O. J. Woodman, "An Introduction to Inertial Navigation," Technical Report UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, 2007.

[17] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005.

[18] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast Direct Lidar-inertial Odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[19] T. Qin, J. Pan, S. Cao, and S. Shen, "A General Optimization-based Framework for Local Odometry Estimation with Multiple Sensors," *arXiv preprint arXiv:1901.03638*, 2019.

[20] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Proceedings of Robotics: Science and Systems*, vol. 2, pp. 1–9, 2014.

[21] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-optimized Lidar Odometry and Mapping on Variable Terrain," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, 2018.

[22] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control for Quadrotors," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2520–2525, 2011.

[23] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion

Planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[24] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.

[25] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust Real-time UAV Replanning Using Guided Gradient-based Optimization and Topological Paths," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1208–1214, 2019.

[26] C. Richter, A. Bry, and N. Roy, "Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments," in *Robotics Research*, pp. 649–666, Springer, 2016.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[28] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[30] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. Jégou, P. Labatut, and P. Bojanowski, "DINOv3," *arXiv preprint arXiv:2508.10104*, 2025.

[31] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging Properties in Self-Supervised Vision Transformers," *arXiv preprint arXiv:2104.14294*, 2021.

[32] Z. Li, C. Li, X. Mao, S. Lin, M. Li, S. Zhao, Z. Xu, X. Li, Y. Feng, J. Sun, Z. Li, F. Zhang, J. Ai, Z. Wang, Y. Wu, T. He, J. Pang, Y. Qiao, Y. Jia, and K. Zhang, "Sekai: A Video Dataset towards World Exploration," *arXiv preprint arXiv:2506.15675*, 2025.

[33] J. Yang, S. Gao, Y. Qiu, L. Chen, T. Li, B. Dai, K. Chitta, P. Wu, J. Zeng, P. Luo, J. Zhang, A. Geiger, Y. Qiao, and H. Li, "GenAD: Generalized Predictive Model for Autonomous Driving," *arXiv preprint arXiv:2403.09630*, 2024.

[34] R. Balestriero and Y. LeCun, "LeJEPA: Provable and Scalable Self-Supervised Learning Without the Heuristics," *arXiv preprint arXiv:2511.08544*, 2025.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] J. Bruce, M. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps, Y. Aytar, S. Bechtle, F. Behbahani,

S. Chan, N. Heess, L. Gonzalez, S. Osindero, S. Ozair, S. Reed, J. Zhang, K. Zolna, J. Clune, N. de Freitas, S. Singh, and T. Rocktäschel, "Genie: Generative Interactive Environments," *arXiv preprint arXiv:2402.15391*, 2024.

[37] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to Control: Learning Behaviors by Latent Imagination," *arXiv preprint arXiv:1912.01603*, 2020.

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY