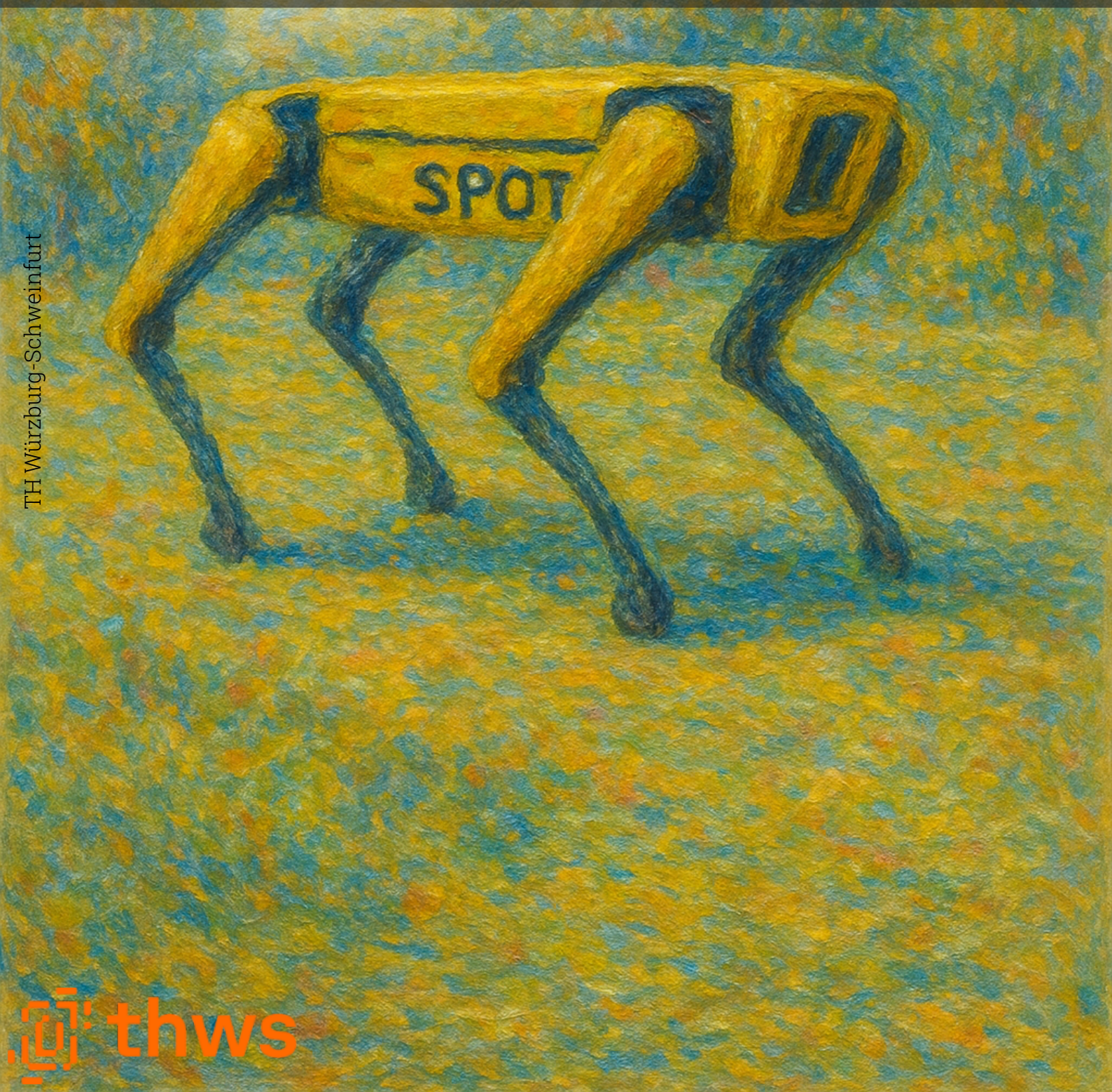
Autonomous Exploration of Unstructured Environments using a Quadruped Robot

Master's Thesis



TH Würzburg-Schweinfurt

thws

Master's Thesis

Autonomous Exploration of Unstructured Environments using a Quadruped Robot

by

Chirag Anand Reddy

Supervisors:	Dorit Borrmann and Tobias Kaupp
Degree Program:	Robotics (M.Sc.)
Faculty:	Faculty of Electrical Engineering, THWS
Submission Date:	Thursday 1st October, 2026

thws

Abstract

Acknowledgements

Contents

Abstract i

Acknowledgements ii

Nomenclature xii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem Statement and Research Questions . . . 2
  - 1.2.1 Scope and Limitations . . . 3
- 1.3 Contributions . . . 3
- 1.4 Thesis Outline . . . 4

2 Background and Related Work 6
- 2.1 Autonomous Exploration and Next-Best-View Selection . . . 6
- 2.2 Map Representations for Exploration . . . 9
- 2.3 Odometry and Simultaneous Localization and Mapping . . . 11
- 2.4 Navigation and Integrated Exploration . . . 14
- 2.5 Legged Locomotion and Reinforcement Learning . . . 17
- 2.6 Simulation Platforms for Robotics Validation . . . 18
- 2.7 Synthesis and Research Gap . . . 20

3 System Architecture 21
- 3.1 Hardware Platform: Boston Dynamics Spot . . . 21
- 3.2 Sensor Suite: Velodyne VLP-16 and Depth Cameras . . . 23
- 3.3 Onboard Compute and Robot Communication . . . 26
- 3.4 Software Stack and ROS 2 Integration . . . 28
- 3.5 The Autonomy Loop: Perception, Exploration, Navigation . . . 31
- 3.6 Operational Deployment and Data Collection . . . 33

4 Simulation and Validation 35
- 4.1 Simulation Strategy and Validation Roles . . . 35
- 4.2 Webots: Maze-World ROS 2 Integration . . . 37
- 4.3 Transition to Isaac Sim and the ROS 2 Boundary . . . 38
- 4.4 Benchmark Worlds in Isaac Sim . . . 39
- 4.5 Isaac Lab and Uneven-Terrain Locomotion . . . 43
- 4.6 Hardware-to-Simulation Evidence Boundary . . . 44

5 Odometry and Mapping **47**
5.1 Deployed Odometry and Frame Ownership 47
5.1.1 Frame ownership and RTAB-Map configuration 47
5.1.2 Closure evidence and map coexistence 49
5.2 Mapping Configuration and Custom Changes 50
5.2.1 nvblox configuration and sensor responsibilities 50
5.2.2 Mapping-corruption investigation 51
5.2.3 Floor-relative reduction and the Isaac Sim crest correction . . . 52
5.3 Attempted Methods and Comparison 55
5.3.1 Odometry and SLAM alternatives 55
5.3.2 Mapping alternatives and final selection 58

6 Frontier-Based 3D Exploration **61**
6.1 Frontier Detection in the TSDF–ESDF Volume 61
6.2 Goal Scoring and Selection Strategies 62
6.2.1 Nearest, Count-Gain and Info-Gain Strategies 63
6.2.2 Gain Shaping and Repetition Penalties 63
6.2.3 Volume-Flood Ground-Frontier Scoring 64
6.3 Exploration Behaviors and Robustness 64
6.3.1 Hysteresis and Stuck-Blacklisting 64
6.3.2 Wedge Prevention and Near-Field Map Safety 65
6.3.3 Holonomic Follower and Transform Hardening 65
6.3.4 Completion Latch and Termination 66
6.4 Walkable-Goal Filtering 66
6.4.1 Filter Design and Bag-Replay Validation 67
6.4.2 Cold-Start Exploration from Depth Cameras 69
6.5 Frontier-Seal Termination 69
6.6 Alternative Exploration Frameworks Evaluated 70
6.6.1 GBPlanner Ground Branch and Elevation Mapping 70
6.6.2 TARE for Goal Generation 71
6.6.3 Hydra Scene Graphs 71

7 Navigation **72**
7.1 Navigation Requirements and Boundaries 72
7.2 Early Navigation Methods 73
7.2.1 Planar Navigation Baseline and Early Physical Findings 73
7.2.2 Comparison of Three Simulated Navigation Mechanisms 73
7.3 The Terrain-Aware Ground-Autonomy Navigator 74
7.3.1 Architecture of the Ground-Autonomy Stack 74
7.3.2 Navigator Defects and Integration Fixes 74
7.3.3 Comparative Simulated Evaluation and Route Variance 74
7.4 Physical Deployment of Ground Autonomy 75
7.5 The Nav2 Migration and Predictive Control 76

7.5.1 Model Predictive Path Integral Control and the Mini-Global Relay 76
7.5.2 Sensor-Band Geometry and Hardware Phantom Obstacles . . . 76
7.6 Terrain-Preserving Two-Dimensional Projection 77
7.6.1 Floor-Relative Obstacle Slicing and Step-Gated Crest Recovery . 77
7.6.2 Preserved Features, Discarded Properties, and Safety Limits . . 78
7.7 Final Configuration and Unresolved Limits 78
7.7.1 Final Configuration and Verification 80
7.7.2 Representational Limits and the Ramp-Preference Problem . . . 80

8 Reinforcement-Learning Terrain Locomotion 81
8.1 Locomotion Limits on Non-Planar Terrain 81
8.2 Training Environment and Policy Interface 82
8.3 Initial Policy and Reward-Threshold Exploit 82
8.4 Correction of Station Keeping 84
8.5 Recovery of Lateral Tracking and Motion Smoothness 85
8.6 Support-Relative Alignment and Symmetry 85
8.7 Stack Integration and Sim-to-Real Boundaries 86

9 Experiments and Evaluation 88
9.1 Evaluation Methodology and Metrics 88
9.1.1 Evidence Domains and Evaluation Units 88
9.1.2 Metrics 89
9.1.3 Stopping Conditions and Comparison Rules 90
9.2 Isaac Sim Mapping Results 90
9.2.1 Crest Correction in Isaac Sim 91
9.3 Isaac Sim Exploration and Navigation Results 94
9.3.1 Crest Observation and Goal Placement 96
9.3.2 Walkable Goals and Route Execution 96
9.3.3 Dune Episodes 98
9.3.4 Frontier Sealing and Termination 99
9.3.5 Ramp Preference and Evidence Boundary 101
9.4 Locomotion Evaluation on Uneven Terrain 101
9.4.1 Instruments 101
9.4.2 Correction Sequence 102
9.4.3 Integration, Slope Results, and Boundaries 103
9.5 Webots Maze Evaluation 105
9.5.1 Timing Isolation in Webots 105
9.5.2 Sensing Configuration Comparison 106
9.5.3 Protocol and Reference Episodes 108
9.5.4 Route Execution Comparison 109
9.5.5 Route Variance 111
9.5.6 Nav2 Migration Evidence 111
9.5.7 Evidence Boundary 112

9.6 Physical-Robot Evidence . . . 115
9.6.1 Graph Closure and Correction Magnitude . . . 115
9.6.2 Odometry on a Multi-Level Recording . . . 115
9.6.3 Staged Perception and Goal-Only Validation . . . 115
9.6.4 Continuous Supervised Navigation . . . 117
9.6.5 Hardware-Specific Integration Findings . . . 118
9.6.6 Evidence Boundary . . . 118
9.7 Cross-Environment Synthesis . . . 118

10 Discussion and Conclusion **122**
10.1 Thesis Summary . . . 122
10.2 Contributions and Findings . . . 123
10.3 Answers to the Research Questions . . . 125
10.4 Limitations . . . 126
10.5 Future Work . . . 127

References **129**

A Launch Files and System Configuration **138**

B Rosbag Datasets and Topic Tables **139**

C Benchmark Scoreboards and Episode Logs **140**

D RL Training Configurations **141**

E Docker Images and Software Versions **142**

List of Figures

2.1 Spatial meaning of four map representations that section 2.2 reviews. . 10

3.1 Instrumented Boston Dynamics Spot platform. 22
3.2 Compute and communication architecture. 28
3.3 ROS 2 nodes, topics, and transforms. 30
3.4 Autonomy-loop interfaces and update rates. 31

4.1 Validation roles in this thesis. 36
4.2 Webots maze integration environment. 38
4.3 Isaac Sim benchmark worlds. 43

5.1 Frame ownership and coexistence of corrected odometry with the live nvblox volume. The left column is the transform chain and the owner of each edge; the upper path is the RTAB-Map pose correction, and the lower path is the live volume that the frontier layer reads. 49
5.2 RViz views of one crest pair in the large dune field before and after the known-floor rule with the step-gated crest guard; the views show the probe route near the spawn, region A of the top-down reference in Figure 9.1(a). Mesh colour encodes surface height; in the grid views, light grey is free, black is occupied, and dark grey is unknown. The scored results and the curb control appear in Figure 9.3. 54

6.1 Frontier-to-waypoint pipeline and process boundaries. 63
6.2 Walkable-goal gate sequence and terrain quantities used by the filter. . 68

7.1 Terrain-preserving navigation contract. 79

8.1 Simulation-only terrain-locomotion training progression. The stages address drift, station keeping, lateral tracking, and support-relative alignment in sequence; stage metrics are not a common cross-stage score. 83

9.1 Top-down reference views of the simulated worlds. 92
9.2 Complete three-dimensional maps of the simulated worlds. 93

9.3 Top-down planning grids from the Isaac Sim crest experiment in the large dune field, rendered from the recorded runs: panels (a) and (b) show region A and panels (c) and (d) region B of Figure 9.1(a). (a) and (b) show the dune route before and after the known-floor rule with the step-gated crest guard; crest cells inside the common evaluation mask are coloured by their classification. (c) and (d) show the curb route; the scored cells of the 0.40 m boundary curb are green when occupied and red when the ungated fill frees them. Grey is free space, black is occupied, white is unknown, and the dashed brown line outlines the authored dunes. 95
9.4 Goal candidates around the robot in region C of the large dune field (Figure 9.1(a)) in a representative episode. Points show the mapped support surface coloured by height, red squares are frontier voxels, green triangles are candidates that passed the walkable-goal gates, and the dashed brown line outlines the authored dunes. The background grid uses the colours of Figure 9.3. 97
9.5 Maps built by representative episodes in the large dune field (Figure 9.1(a)) after about twenty simulated minutes: (a) the retained explorer and (b) the ground-frontier variant. Colour encodes the mapped support height on the scale of the reference view, black cells are occupied, white is not yet observed, the line is the robot path, circles are published goals, and the dashed brown line outlines the authored dunes. 99
9.6 Planning grids and frontier voxels in representative episodes in the small walled arena (Figure 9.1(b)): (a) sealing enabled during exploration, (b) sealing enabled after the completion latch, and (c) sealing disabled at a similar elapsed time. Red squares are frontier voxels and the dashed brown line outlines the authored dunes. The quantitative results are those of Table 9.3. 100
9.7 Simulator frames of the selected locomotion policy walking towards goal D of Figure 9.1(a) in the large dune field (perspective view in Figure 4.3(b)): (a) at the spawn, (b) walking out across the plaza, (c) stepping onto the dune flank, and (d) standing on the crest. The green sphere marks the spawn point. 104
9.8 Webots sensing comparison in the start room of the maze (box E in Figure 9.1(c); perspective view in Figure 4.2): dense map meshes after the 60 s warm-up and after the 180 s scripted drive for each sensor configuration. Mesh colour encodes height. 107
9.9 Teleoperated coverage reference in the Webots maze (Figure 4.2; floor plan in Figure 9.1(c)). The panels show the common scored map region; in (b), color denotes support height from 0 m to 0.5 m, and in (c), green marks the 13 970 walkable columns while red marks the 1818 rejected columns. 108

9.10 Mesh-navigation episode in the Webots maze (floor plan in Figure 9.1(c), perspective view in Figure 4.2). Blue points are the vertices of the navigation mesh exported at an early, an intermediate, and the final stage of the episode; the dark line is the whole robot path, and the walls come from the maze world file. 110
9.11 Peak support columns of representative episodes in the Webots maze (Figure 9.1(c)). The dashed line is the teleoperated coverage reference. The episodes use different stopping rules and route protocols, so the chart does not establish a universal ranking. 112
9.12 Robot paths of three ground-autonomy episodes in the Webots maze (floor plan in Figure 9.1(c), perspective view in Figure 4.2). Episode 1 is the strongest episode, Episode 2 the predecessor decay configuration, and Episode 3 the byte-identical repetition of Episode 1; they reach 15 478, 15 296, and 14 239 peak support columns. Walls come from the maze world file. 113
9.13 Top-down density of the rotating-sensor points accumulated over a 236 m multi-level physical recording, (a) with leg odometry only and (b) with leg-assisted GenZ-ICP. Darker pixels hold more points. Offline processing of a physical recording in a real indoor building; unlike the simulated worlds of Figure 9.1, no authored layout exists for it, so the straight walls of panel (b) are the only layout reference. 116

List of Tables

2.1 Comparison of representative exploration methods by shared mechanism, candidate generation, gain, and path cost. 8
2.2 Comparison of map representations by stored data, geometric query, and evidence for supporting ground. 12
2.3 Comparison of odometry and SLAM families by sensing and global correction. 15
2.4 Simulation platforms, distinguishing properties, and roles in this thesis. 19

3.1 Sensor interface contracts. 24

4.1 Webots and Isaac Sim integration choices in this thesis. 40
4.2 Specification and validation purpose of the Isaac Sim benchmark worlds. 42
4.3 Evidence boundary between physical-robot work and simulation work. 45

5.1 Production RTAB-Map configuration. 48
5.2 Production nvblox and costmap parameters. 50
5.3 Odometry and SLAM methods evaluated in this work. 56
5.4 Mapping configurations and representations considered in this work. . 59

6.1 Goal-generation frameworks evaluated, by representation and integration fit. 70

9.1 Research questions with their evidence domains and metrics; the RQ column follows the question numbering in Chapter 1, and the metrics named here are those defined in Section 9.1.2. 91
9.2 Isaac Sim autonomous episodes with the integrated stack. The goal columns count distinct goals issued and arrived, while the validator column counts publication events rechecked against map snapshots; its entries sum to the 12 396 checks reported in Section 9.3.2. 96
9.3 Isaac Sim frontier-sealing results. Support percentages divide by the control support, the value reached by the sealing-disabled episode in the same arena, except in that row, which is relative to its own final support. A dash marks a row without a validator entry in this comparison, so it denotes a missing record rather than zero failures. 101

9.4 Representative per-axis metrics before and after each locomotion correction. The first group reports seven per-axis gates for each terrain condition under that stage's harness, and the later groups report the stage totals cited in the text under the common evaluation task, so passed fractions from different groups are not comparable. 102
9.5 Mapping outcome by sensing configuration. 106
9.6 Representative Webots maze episodes under the cameras-only mapping configuration. Support and walkable report the counts reached by each episode, percentages of the teleoperated reference use its 15 788 support columns as the denominator, and the validator column reports accepted and checked publication events (Section 9.1.2), where *no validator* marks a series in which the validator was not active and which therefore carries no admissibility record rather than zero failures. Termination names the stopping condition that ended the run, and unobservable tiles are explorer candidate units counted by the completion rule in Section 6.3.4. 114
9.7 Physical evidence and permitted inference. 119
9.8 Supported conclusions and evidence boundaries per research question; the RQ column follows the question numbering in Chapter 1. 121

Nomenclature

Abbreviations

Abbreviation	Definition
API	Application Programming Interface
CPU	Central Processing Unit
CUDA	NVIDIA parallel-computing platform and programming model
FAST	Features from Accelerated Segment Test
BRIEF	Binary Robust Independent Elementary Features
DARPA	Defense Advanced Research Projects Agency
DBoW2	Bag-of-binary-words place-recognition library
DDS	Data Distribution Service
DOF	Degree of Freedom
DWB	Dynamic Window Based
EAP	Enhanced Autonomy Payload
ESDF	Euclidean Signed Distance Field
FAR	Fast, Attemptable Route
FUEL	Fast UAV Exploration using Incremental Frontier Structure and Hierarchical Planning
GICP	Generalized Iterative Closest Point
gRPC	Google Remote Procedure Call
GPU	Graphics Processing Unit
GTSAM	Georgia Tech Smoothing and Mapping
ICP	Iterative Closest Point
I/O	Input/Output
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LIO-SAM	LiDAR-Inertial Odometry via Smoothing and Mapping
LOAM	LiDAR Odometry and Mapping
LTM	Long-Term Memory

Abbreviation	Definition
MPPI	Model Predictive Path Integral
Nav2	Navigation 2, the ROS 2 navigation stack
ORB	Oriented FAST and Rotated BRIEF
PD	Proportional-Derivative
POSIX	Portable Operating System Interface
PPO	Proximal Policy Optimization
PRM	Probabilistic Roadmap
RANSAC	Random Sample Consensus
REP	ROS Enhancement Proposal
RGB-D	Red, green, blue, and depth
RH-NBVP	Receding-Horizon Next-Best-View Planner
RL	Reinforcement Learning
RMS	Root Mean Square
ROS	Robot Operating System
ROS 2	Robot Operating System 2
RRT	Rapidly Exploring Random Tree
SDK	Software Development Kit
SLAM	Simultaneous Localization and Mapping
SSH	Secure Shell
STM	Short-Term Memory
SURF	Speeded-Up Robust Features
TCP	Transmission Control Protocol
TF	Transform, the ROS coordinate-frame system
TSDF	Truncated Signed Distance Field
UDP	User Datagram Protocol
UAV	Unmanned Aerial Vehicle
URDF	Unified Robot Description Format
USD	Universal Scene Description
VILENS	Visual, Inertial, Lidar, and Leg Odometry for All-Terrain Legged Robots
WM	Working Memory
XYZI	Cartesian x, y, and z coordinates with intensity

1 Introduction

This thesis develops and evaluates an autonomous exploration system for a Boston Dynamics Spot quadruped on uneven, partially observed terrain. This chapter motivates the problem, states the research questions, sets the scope of the work, lists its contributions, and outlines the remaining chapters.

1.1. Motivation

Autonomous exploration enables a robot to build a representation of an environment while deciding where to sense next. It is useful where an unknown environment is dangerous, costly, or slow for people to enter. Search and rescue, underground and tunnel inspection, and inspection of damaged industrial sites require mapping under poor visibility, debris, stairs, uneven ground, and unreliable communication [Del+19; Mil+20; Bou+20]. The Defense Advanced Research Projects Agency (DARPA) Subterranean Challenge combines these conditions in tunnel, urban, and cave courses with limited operator contact [Kul+22].

In an initially unknown environment, exploration must distinguish between space that sensors have observed and space that may still contain useful information. A frontier is the boundary between mapped free space and adjacent unobserved space [Yam97]. Frontier detection is therefore necessary. It supplies candidate regions for further observation. A frontier does not automatically provide a valid goal: it may lack support (a valid mapped floor estimate in the column beneath it), lie beyond reach, face an obstruction, or reveal no useful new geometry. Exploration connects sensing, mapping, goal selection, navigation, and locomotion through this decision.

This decision becomes harder on uneven terrain. Wheeled and tracked robots are reliable on continuous ground, but their need for continuous surface contact becomes a constraint on stairs, rubble, gaps, and steep slopes. A legged platform selects footholds and adjusts body height and orientation to the terrain. Systems such as

ANYmal [Hut+16] and Spot [Bos24] establish quadrupedal locomotion over rough terrain. This makes a quadruped suitable for environments in which a traversable route is discontinuous rather than planar.

A planner cannot treat the ground as a two-dimensional plane. The body pitches and rolls while walking, and the sensors move with it. The terrain that determines whether a goal is reachable is the same terrain the robot must reconstruct from noisy, possibly incomplete sensor data. A robot that builds an accurate map but cannot climb a ramp has mapped part of the space rather than explored it. The relevant failures occur at the interfaces between components. Frontier candidates, terrain support, route planning, and locomotion must agree before the navigation stack can execute an observation goal.

Three integration conditions motivate this work. Frontier detection can return no candidates before exploration is complete. Near-field sensing can leave the support surface unobserved. A route can also be geometrically valid while exceeding the gait's capability. These conditions motivate recovery-aware frontier handling, support tests that require a valid floor estimate beneath a candidate goal (Section 6.4), and terrain-capable locomotion. The thesis examines these conditions on one Boston Dynamics Spot platform using physical-robot trials and task-specific simulation.

1.2. Problem Statement and Research Questions

The thesis develops and evaluates an autonomous three-dimensional exploration stack for uneven, partially observed terrain. The stack uses a Boston Dynamics Spot quadruped with a Velodyne VLP-16 light detection and ranging (LiDAR) sensor and body-mounted depth cameras. Its software separates pose estimation, volumetric mapping, frontier selection, route execution, and locomotion; explicit interfaces support evaluation of each layer and the complete loop. The design integrates established components rather than introducing a new exploration theory.

The implemented mapping path combines loop-closed odometry with nvblox, a volumetric mapper that a graphics processing unit (GPU) executes and that maintains the distance-field and occupancy layers the frontier layer reads. A custom three-dimensional frontier layer selects observation goals and applies support and reachability checks before handing them to a separate navigator.

The evaluation assesses navigation and locomotion under the physical and simulated configurations available for this work. Chapter 3 defines these interfaces and the platform; Chapters 4–8 describe their implementation and evidence.

The following questions organize the evaluation:

1. Can the integrated odometry and volumetric mapping path maintain a usable representation for frontier-based exploration [LM19; Mil+24; Yam97] under the tested physical and simulated configurations?

2. How do the available depth and LiDAR sensing configurations affect the suitability of the resulting map for exploration?
3. Can a true-three-dimensional frontier layer produce supported, reachable, collision-free goals on ramps and dunes, including regions observed only above a crest?
4. How do the tested navigation configurations execute exploration goals on uneven terrain, and which map, controller, and interface constraints determine their suitability?
5. Which parts of the integrated stack can simulation evaluate, and which claims require evidence from the physical robot?

1.2.1. Scope and Limitations

The evaluation examines one configured Spot stack, its documented sensors and software, the selected terrain and worlds, and the interfaces between its subsystems. Physical-robot trials provide evidence about sensor behavior, mapping, timing, and navigation under the available hardware constraints. Webots and Isaac Sim provide task-specific environments for Robot Operating System 2 (ROS 2) integration, map consistency, frontier goals, route execution, and terrain experiments; Isaac Lab provides the repeated policy-training environment for rough-terrain locomotion. Chapters 3 and 4 define the roles and boundaries of these environments.

These measurements do not rank method families or establish performance beyond the tested configurations. Joint actuator faults end hardware access before the final ramp, dune, frontier-seal, and learned-locomotion evaluations, which therefore remain confined to simulation. Simulation can evaluate interfaces, map consistency, frontier goals, route execution, and policy behavior under declared models; it cannot certify real contact forces, actuator response, sensor timing, or transfer to hardware. Run-to-run variation remains within the reported scope.

1.3. Contributions

This thesis makes five engineering contributions:

1. a sensing comparison that selects cameras-only depth integration for nvblox mapping under the tested conditions;
2. a mapper-side correction that retains frontier visibility above terrain crests;
3. a walkable-goal filter that moves three-dimensional frontier candidates onto supported terrain and rejects invalid slope, step, roughness, reachability, and headroom conditions;
4. a frontier-seal rule that removes persistent interior air frontiers, which are frontier voxels above an interior mapped floor, while retaining at the sensor horizon the

forward frontier ring that represents genuinely unobserved ground; and

5. a blind rough-terrain policy for Spot that receives no terrain geometry in its observation and keeps the existing proprioceptive observation and action interface.

Together, these contributions define the system evaluated in the later chapters. The next section explains how the thesis moves from background and architecture through simulation, subsystem implementation, and evaluation.

1.4. Thesis Outline

Chapter 1 introduces the exploration problem, defines the system and evidence domains, states the research questions, and describes the contributions and scope.

Chapter 2 distinguishes the roles of exploration, navigation, and locomotion before describing the choices specific to this thesis.

Chapter 3 sets out the architecture of the stack. It identifies the physical Spot platform, its sensors, the Docker-based runtime, and the interfaces between the subsystems. It also establishes the physical and simulated sensor, timing, and transport contracts that later chapters use.

Chapter 4 covers simulation and validation. Webots [Mic04] supports system-level ROS 2 integration, while Isaac Sim supports the later terrain and full-stack experiments. Isaac Lab supports rough-terrain policy learning. The chapter explains the roles of these platforms and the boundary between simulated evidence and evidence from the physical Spot.

Chapter 5 covers odometry and mapping. It describes Real-Time Appearance-Based Mapping (RTAB-Map) loop closure and the nvblox GPU signed distance field mapper. It reports the sensing-configuration comparison that supports the cameras-only configuration and the mapper-side fix for crest visibility.

Chapter 6 presents the custom three-dimensional frontier exploration layer. It details frontier detection, goal generation, the walkable-goal filter, and the frontier-seal rule (Section 6.5) that removes persistent interior air frontiers while retaining the forward frontier at the sensor horizon.

Chapter 7 covers two navigation phases: terrain-aware ground-autonomy navigation in simulation, which combines the CMU Autonomous Exploration Development Environment (ADEV) [Cao+22] with the Fast, Attemptable Route (FAR) planner [Yan+22], and Navigation 2 (Nav2) on the physical robot. It examines the map, controller, and interface constraints in each phase. Neither phase is a universal winner; suitability depends on those constraints.

Chapter 8 describes terrain locomotion with Isaac Lab. It covers the training of reinforcement-learning policies for rough terrain and their deployment in simulation.

Chapter 9 presents the experiments and evaluation. It reports coverage as the number of horizontal map columns with a valid support surface, goal validity as the share of published goals that an independent validator accepts, a comparison with a teleoperated reference episode that bounds the coverage available from the same sensing and mapping path under operator goal selection and route execution, and the final simulation experiments.

Chapter 10 concludes the thesis, answers the research questions, and identifies future work.

2

Background and Related Work

Autonomous exploration combines five functions: selecting where to observe next, representing known and unknown space, estimating the robot's pose, planning and executing a route to the selected goal, and traversing the terrain. These functions correspond to exploration, mapping, localization, navigation, and locomotion.

This chapter reviews existing technologies for each function and compares their assumptions, outputs, and constraints. The comparison identifies the methods that suit each part of the autonomous exploration stack and the simulation platforms that support their evaluation.

2.1. Autonomous Exploration and Next-Best-View Selection

Autonomous exploration determines which part of an unknown environment the robot observes next. Next-best-view selection assigns each sensor pose a value from the coverage or information it should add under a stated cost. Each cycle trades gain against cost, updates the map, and repeats [LLA21]. Navigation computes a route and locomotion executes it, so a selected viewpoint may lie beyond reach and a route may exceed the gait's capabilities [Mil+20; Kim+20]. Three dimensions separate methods: candidate generation, gain, and path cost.

Candidate generation. Frontier methods read candidates from the map boundary. Yamauchi defines a frontier as the boundary between known open and unexplored space, then selects the nearest accessible one [Yam97]. Later work changes implementation, not that rule, as in wavefront detection with a group median rather than a centroid [KSK12; TIK18]. Sampling ignores the boundary: the receding-horizon next-best-view planner (RH-NBVP) grows a random tree of collision-free viewpoints [Bir+16], GBPlanner1 (this thesis's label for the 2020 graph-based planner) samples a

local random graph with a sparse global graph [Dan+20], GBPlanner2 uses an adaptive principal-axis box [Kul+22], and Tabib et al. expand state-lattice primitives [Tab+16]. Hybrids cache boundary structure: FUEL caches frontier clusters, viewpoints, and connection costs, so updates invalidate only affected entries [Zho+21]; TARE is a hierarchical framework whose local layer samples viewpoints in a local horizon to cover mapped surfaces while a global layer holds outer space as sparse subspaces [Cao+21].

Information gain and coverage. Three definitions of expected return appear, and they are not interchangeable. Volumetric gain counts space: GBPlanner1 and GBPlanner2 ray cast along the path to a graph vertex [Dan+20; Kul+22], and RH-NBVP scores a branch by the unmapped volume it reveals [Bir+16]. Surface coverage counts observed geometry: TARE terminates when no uncovered local surface and no exploring subspace remain [Cao+21], and FUEL asks whether a viewpoint covers its cluster frontier cells [Zho+21]. In a micro aerial vehicle simulation, Heng et al. report an increase in observed voxel faces from 74.1 % at 98.4 m to 89.9 % at 136.1 m when coverage-maximizing paths replace cost-weighted frontier selection [Hen+15]. Information-theoretic gain scores uncertainty reduction, as in the Cauchy-Schwarz quadratic mutual information Tabib et al. evaluate over an occupancy belief [Tab+16]; it represents map quality rather than extent. Learned variants instead reward traversal experience [PBF22].

Path cost. One phrase covers four quantities. The weakest is Euclidean distance in closest-frontier selection [Yam97; Zho+21] and in the wavefront implementation [TIK18]. The second is route length through mapped free space, which prices all traversable space alike, as in the weighting of Heng et al. and the RH-NBVP branch-length discount [Hen+15; Bir+16]. The third is time or energy: Tabib et al. normalize gain by expected traversal time [Tab+16], TARE requires a kinodynamically feasible local path [Cao+21], and FUEL adds minimum-time trajectory duration to a cluster tour [Zho+21]. The fourth admits terrain, as constraint or as weight. GBPlanner2 requires mapped support under projected edge samples and bounded inclination; its support filter attenuates gain for hanging vertices and prevents them from serving as commands [Kul+22]. Azpúrua et al. instead weight expected gain by terrain traversability [ACM21]. On legs the difference is decisive: a constraint removes a candidate, a weight only discounts it.

Aerial, ground, and legged assumptions. Aerial vehicles occupy any collision-free point, so their candidates need no support: Heng et al. assume a known pose and a forward-looking depth or stereo sensor on a micro aerial vehicle [Hen+15], Tabib et al. assume perfect state estimation for a quadrotor in simulated pit exploration [Tab+16], and FUEL uses quadrotors in its benchmark worlds [Zho+21]. A legged platform changes three assumptions. Candidates must lie above mapped, sufficiently

Table 2.1: Comparison of representative exploration methods by shared mechanism, candidate generation, gain, and path cost.

System	Candidate generation	Gain definition	Path cost	Platform
Frontier [Yam97; TIK18]	Grouped frontier cells on an occupancy grid	Implicit; any accessible frontier qualifies	Distance to nearest accessible frontier	Wheeled
RH-NBVP [Bir+16]; Tabib et al. [Tab+16]	Random viewpoint tree; state-lattice primitives	Unmapped volume; quadratic mutual information	Branch length; expected time or energy	Aerial
GBPlanner1 [Dan+20]; GBPlanner2 [Kul+22]	Local sampled graph with sparse global graph; adaptive principal-axis box	Ray-cast volumetric gain, attenuated for hanging vertices	Graph cost; mapped support and inclination as constraints	Aerial and legged
FUEL [Zho+21]	Cached frontier clusters with sampled viewpoints	Coverage of cluster frontier cells	Cluster tour plus minimum-time duration	Quadrotor
TARE [Cao+21]	Local viewpoints for uncovered surfaces; global subspaces	Surface coverage in the local horizon	Kinodynamically feasible local path; global roadmap	Ground and aerial
Azpúrua et al. [ACM21]	Frontier candidates in confined space	Expected gain	Gain weighted by terrain traversability	Wheeled and tracked

level support with body headroom, since a free voxel may be open air. Vertical cost is asymmetric: slope, step height, and roughness make edges inadmissible. Sensing is not static: a body-mounted sensor moves with the gait, so the realized viewpoint depends on body pitch, roll, and height. TARE ground experiments use flat built surfaces [Cao+21], and TravExplorer extracts frontiers from headroom-checked support surfaces for a quadruped [Zhe+26].

Comparison of published systems. Table 2.1 groups representative systems on the three dimensions. By column, generation moves from boundary reading through cached boundary structures to sampling, gain from counted volume to covered surface to uncertainty reduction, and cost from distance through time to terrain as constraint or weight. These published designs fuse selection with routing and expose no replaceable map, graph, or terrain interface [Cao+21; Dan+20; Kul+22].

From frontier evidence to a commandable goal. One step lacks a definition, and it matters most for a walking robot. A frontier indicates where further sensing may help, not where the robot can stand. The boundary object may be an unknown cell, a free voxel, a mesh face, or a traversable support cell, and that choice fixes the cluster representative, the gain estimate, and whether the navigator can command the point. Faster detection does not settle it: reachability follows from the route and surrounding terrain, and map updates can invalidate a component. GBPlanner2 takes the support decision inside its graph [Kul+22]; a frontier layer that publishes one target to an external navigator must take it before publication.

The exploration layer of this thesis takes one position per dimension that fits these constraints. It reads candidates from the frontier boundary of a three-dimensional volumetric map and projects them onto mapped supporting ground, keeping generation and support filtering separate. It scores candidates with a counted unknown-volume

proxy, which sums unknown voxels near a candidate and discounts them with distance, as geometric rather than information-theoretic gain (section 2.1). Path cost stays outside the layer, which publishes one supported waypoint and delegates routing and feasibility to a navigator. The design requires stable goals under map updates, rejection of unsupported or unreachable candidates, and termination only after the robot exhausts reachable new space. This choice is not a superiority claim. Chapter 3 defines the interface and chapter 6 the implementation and validation.

2.2. Map Representations for Exploration

The map beneath the decisions of section 2.1 determines their expressive power. Four questions separate the representations: how many spatial dimensions they store, whether they distinguish unobserved space from observed free space, whether they support direct distance queries, and what evidence they provide for supporting ground. No representation answers all four, and none states walkability by itself. Three terms require separation here: support is the mapped surface beneath a goal, a body pose, or a foot; walkable ground is support that also satisfies the slope, step, and roughness limits the gait imposes; and traversable ground is the route-level property that admits, removes, or weights terrain along a planned path.

What a representation stores. A two-dimensional map stores one value per horizontal cell, describing a single plane; a two-and-a-half-dimensional (2.5D) map stores one terrain height per column, so the surface may be non-planar while staying single-valued; a three-dimensional map stores a value per volume element and can hold several surfaces above one horizontal position. A ramp, a step, an overhang, or a second floor puts several relevant heights in one column, so a planar map cannot serve a walking robot.

An occupancy value stores the probability that a cell is occupied; OctoMap implements this as a probabilistic octree with multi-resolution access, where ray integration marks traversed cells free and endpoints occupied while cells no ray has traversed stay unknown [Hor+13]. The explicit unknown state makes frontier detection possible, a frontier being a known-free cell adjacent to an unknown one. Occupancy offers no signed clearance and no gradient, so slope and traversability follow only from the occupied geometry.

The truncated signed distance field. Each voxel within a truncation band around an observed surface holds a signed distance along the sensor ray and an integration weight, and the surface is the zero level set [CL96; Ole+17]. The sign separates the free from the occupied side inside the band, fusion over many frames suppresses noise, and marching cubes extracts a mesh from the zero crossing [LC87; New+11]. The stored distance is projective and truncated, hence not Euclidean clearance and undefined outside the band. Sparse spatially hashed allocation addresses the memory growth

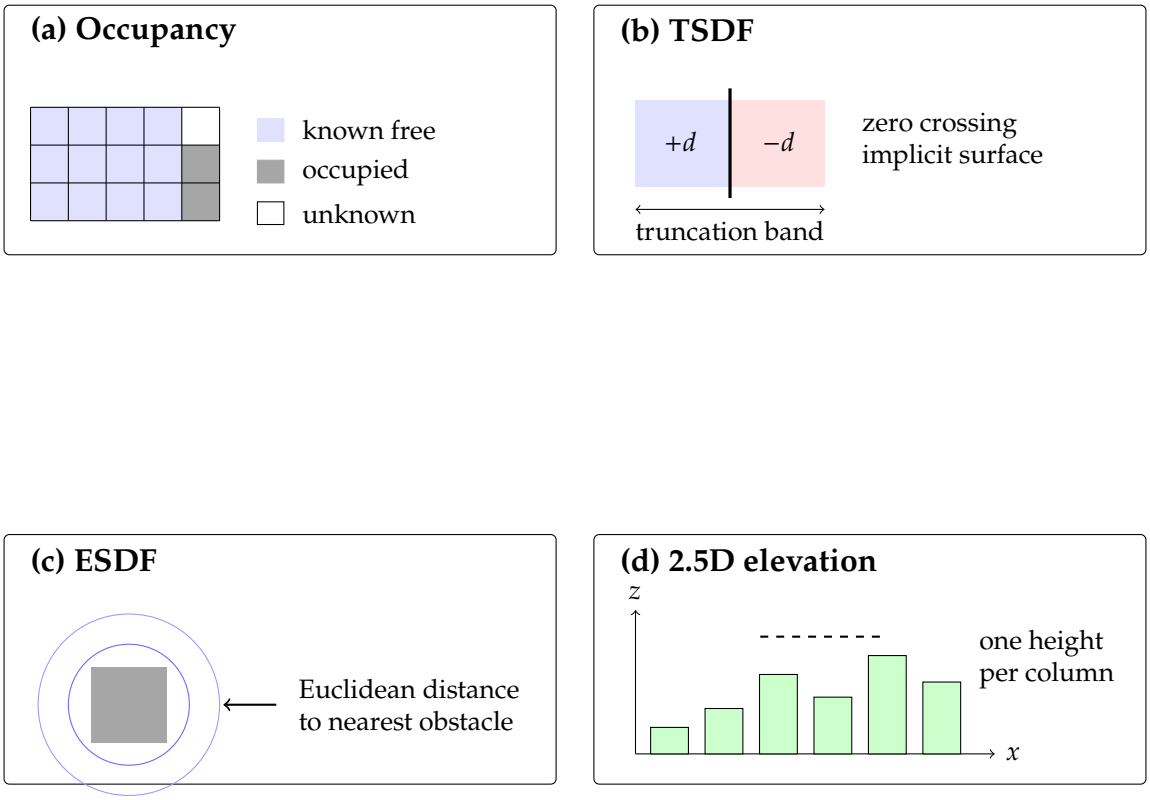


Figure 2.1: Spatial meaning of four map representations that section 2.2 reviews.

with resolution [Nie+13].

The Euclidean signed distance field. An ESDF stores, per voxel, the Euclidean distance to the nearest occupied boundary, positive in free space and negative inside obstacles [Ole+17]. A planner needs a second field because a TSDF cannot answer its clearance query: on an ESDF, whether a body of radius r fits is one lookup and one comparison, instead of a scan of every cell within r of each path sample, and the gradient points toward increasing clearance. Voxblox derives it incrementally from a hashed TSDF by propagating lower and raise wavefronts, leaving unknown voxels unupdated [Ole+17]; the costs are extra computation, a quasi-Euclidean approximation, and stale values.

Explicit surfaces: surfels and meshes. Surfel and mesh maps store the surface explicitly: a TSDF samples a volume and retains which side of the surface a point is on, whereas these keep only surface primitives and leave no volumetric state to query. ElasticFusion holds unordered oriented surface elements whose connectivity requires a separate meshing step [Whe+15], and a mesh that marching cubes extracts from a TSDF records neither the free rays towards surfaces nor the unknown volume behind occlusions [Mil+24]. A planner can route geodesically along the surface [Püt+21], but floor, wall, and ceiling faces form one connected surface, so support requires a separate per-point traversability cost [Cao+22].

Terrain: elevation maps. Elevation mapping compresses terrain into a 2.5D grid with one height and one variance per cell, with a robot-centric window carrying accumulated pose uncertainty [FBH18]. The grid-map library supplies layers for variance, normals, slope, and roughness [FH16], and GPU implementations raise the update rate [Mik+22a]. Height variation, local slope, and per-cell uncertainty are the support evidence a legged planner reads, but the single value per column excludes overhangs, tunnels, vertical walls, and multi-surface columns and holds no free or unknown volume, so a three-dimensional frontier explorer needs a volumetric representation underneath any terrain layer.

Aligned layers and the choice in this thesis. nvblox maintains aligned sparse voxel layers for TSDF, ESDF, occupancy, and mesh data on a GPU, allocating blocks only where observations arrive, meshing changed blocks, updating the ESDF incrementally for those blocks, and maintaining a fully Euclidean rather than quasi-Euclidean distance field [Mil+24]. Voxgraph instead lets pose-graph corrections re-pose submaps to reach global consistency [Rei+20].

Figure 2.1 illustrates these spatial distinctions, which the preceding paragraphs state one representation at a time: occupancy separates free, occupied, and unknown cells; a TSDF stores projective signed distance only within a band near a surface; an ESDF stores Euclidean clearance throughout updated free space; and a 2.5D elevation map stores one height per column and therefore cannot hold the dashed overhang in panel (d). Table 2.2 compares stored data, planning read-out, and support evidence. The mapping layer of this thesis uses nvblox because aligned occupancy, TSDF, and ESDF layers supply unknown volume, a reconstructed surface, and Euclidean clearance within one incrementally updated GPU map [Mil+24]; walkability remains a separate judgement above those layers. Chapter 5 reports the configuration and sensor comparison, and chapter 6 the support filtering and goal selection.

2.3. Odometry and Simultaneous Localization and Mapping

Every representation in section 2.2 receives each measurement at its acquisition pose, so pose error enters the map as geometry and corrupts the frontier, clearance, and support queries above it.

Odometry, SLAM, and pose dimensionality. Odometry estimates incremental motion between consecutive measurements, so its error accumulates without bound. Simultaneous localization and mapping (SLAM) recognizes a revisited place and corrects that accumulation: optimization over a pose graph of poses and relative constraints spreads the correction over the trajectory rather than as a jump at the current pose [Cad+16]. A local odometry frame stays continuous for control while a parent map frame absorbs the global correction, so mapping consumes corrected poses

Table 2.2: Comparison of map representations by stored data, geometric query, and evidence for supporting ground.

Map	Stored data	Planning read-out	Supporting ground	Principal limit
2D occupancy grid [Yam97]	Free, occupied, unknown per planar cell	Planar collision test	None	Discards vertical structure
2.5D elevation map [FBH18; FH16]	Height and variance per cell; no unknown state	Height, slope, roughness, uncertainty	Input to a separate traversability test	One surface per column; no free or unknown volume
3D occupancy octree [Hor+13]	Free, occupied, unknown per voxel	Multi-resolution collision and frontier queries	None without surface classification	No signed clearance or gradient
TSDF [CL96; New+11; Ole+17]	Truncated projective distance and weight in a surface band	Zero crossing and mesh	Surface needs ground classification	Not Euclidean clearance; undefined outside band
ESDF [Ole+17; Mil+24]	Euclidean distance to nearest occupied boundary, signed	Single-lookup clearance and gradient	Clearance only	Extra update cost; unknown voxels unupdated; unsafe if stale
Surfel set or triangle mesh [Whe+15; LC87; Püt+21]	Observed surface primitives only; no unknown state	Surface geometry, normals, geodesic routes	Faces need ground and wall separation	No free or unknown volume; stale or fragmented topology
nvblox aligned layers [Mil+24]	Sparse hashed TSDF, ESDF, occupancy, mesh; free, occupied, unknown in 3D	Clearance, unknown volume, and surface in one map	Aligned surface evidence; walkability external	GPU required; voxels not re-posed after graph correction

without replacing the local estimate [LM19]. Pose dimensionality separates the families: a planar estimator reports three degrees of freedom (DOF), two translations and one heading, complete only for planar motion, whereas a walking platform requires six DOF because the body rolls, pitches, and changes height, and a three-DOF pose cannot correctly place a measurement taken from a rolled and pitched sensor.

Planar LiDAR SLAM. GMapping is the planar particle-filter baseline and reports a three-DOF pose [GSB07]. Hector SLAM needs no external odometry and tracks the full sensor motion during matching, so a rolled or pitched scan registers correctly, while the official documentation states that the published map and pose remain two-dimensional[1] and lists no loop closure. Cartographer moves the family onto a pose graph, and official documentation also records a volumetric six-DOF configuration[2]. Either leaves the interface to a separate volumetric exploration map open.

Visual odometry and visual SLAM. Visual odometry estimates camera motion between frames without a globally consistent map, visual-inertial odometry adds an inertial measurement unit (IMU) to make the metric scale observable, and visual SLAM adds place recognition and correction. ORB-SLAM closes loops in six DOF over a sparse keyframe graph, with scale ambiguous in monocular mode [MMT15]; stereo or colour-and-depth input in ORB-SLAM2 resolves it [MT17]. A sparse feature map is neither an occupancy map nor a terrain model, so camera appearance fits place recognition rather than replacing the volumetric map.

Three-dimensional LiDAR odometry and registration. LOAM is the reference for direct six-DOF LiDAR estimation, where within-scan acquisition-time compensation addresses the dominant error source for a rotating sensor on a walking body [ZS14]. Registration itself is the iterative closest point (ICP) algorithm, alternating correspondence assignment with the rigid transform minimizing the residual and converging to an initialization-dependent local minimum [BM92]. Point-to-point ICP minimizes distance between corresponding points and stays accurate with careful deskewing [Viz+23], point-to-plane ICP minimizes distance along the target surface normal, which exploits planar structure but requires normals, and GenZ-ICP weights the two residuals adaptively [LLH25]. These scan-to-map front ends supply no loop detector, and registration degrades on sparse or repetitive geometry.

LiDAR-inertial estimation. An IMU measures angular rate and acceleration far above the LiDAR frame rate, so an estimator can reconstruct within-scan motion and deskew the scan. LIO-SAM places IMU preintegration, LiDAR odometry, and optional loop-closure factors in one six-DOF factor graph; its documentation describes the loop closure as a proof of concept, and the official implementation disables it by default

[1] `https://wiki.ros.org/hector_mapping`
[2] `https://google-cartographer.readthedocs.io/en/latest/`

[Sha+20]. The implementation requires a nine-axis IMU at 200 Hz or more, per-point time and ring fields, and mechanical lidars[3]. FAST-LIO uses a tightly coupled iterated extended Kalman filter [XZ21], FAST-LIO2 registers raw points against an incremental ikd-Tree map [Xu+22], and both require synchronized sensors and per-point timing[4]; both remain local without a global back end.

Estimation on legged platforms. Legged platforms change the problem: slip, attitude oscillation, and terrain deformation make a rigid-body motion model incomplete, while joint encoders supply a motion source wheeled platforms lack. VILENS fuses inertial, visual, LiDAR, and leg measurements in one factor graph, and that tight fusion makes the linear-velocity-bias state observable [WCF23]; the paper documents no global loop-closure stage. Ramezani et al. instead register inertial-kinematic clouds with ICP, propose loops with a learned segment descriptor, and distribute corrections through a pose graph [Ram+20]. Both precedents differ in sensors, compute, and robot dynamics.

The graph back end and the choice in this thesis. Table 2.3 compares these families by sensing, global closure, and the property that matters for this stack. Except for Cartographer and the legged system of Ramezani et al., these are front ends; a graph back end accepts any local motion estimate and supplies global consistency. RTAB-Map is such a back end: external odometry gives the motion prior, neighbour links join consecutive locations, loop-closure links connect locations that appearance recognition matches under an incremental bag-of-binary-words model [GT12; LM13], proximity links join geometrically close locations, and LiDAR registration supplies the geometric constraints [LM19]. This thesis selects RTAB-Map for replaceable odometry in three or six DOF, bounded online locations, so that per-cycle processing cost does not grow with the graph, and a continuous map-to-odometry correction. Chapter 5 reports the deployed configuration: leg-kinematic odometry from the platform driver, computed from joint and body-state kinematics, supplies the prior; point-to-plane ICP on clouds from the platform's 16-beam Velodyne VLP-16 rotating LiDAR [Vel19] refines neighbour links; geometric proximity closes the graph; and a separate test on recorded data validates body-camera appearance closure, the visual place-recognition closure that the live deployment does not exercise.

2.4. Navigation and Integrated Exploration

Navigation converts a selected observation goal into motion. The section proceeds in the order these responsibilities compose: route execution, map and path families, then integrated systems.

[3] https://github.com/TixiaoShan/LIO-SAM
[4] https://github.com/hku-mars/FAST_LIO

Table 2.3: Comparison of odometry and SLAM families by sensing and global correction.

Family and representatives	Sensing	Global closure	Property that matters here
Planar configurations: GMapping, Hector SLAM, Cartographer [GSB07; Koh+11; Hes+16]	Planar laser, odometry optional	Particle set, none, or branch-and-bound scan-to-submap	Planar output; Cartographer also documents volumetric six-DOF mapping; graph closure only with its own map product
Visual: ORB-SLAM, ORB-SLAM2 [MMT15; MT17]	Monocular, stereo, colour-and-depth	Appearance-based	Sparse feature map; scale from stereo; texture dependent
LiDAR registration: LOAM, KISS-ICP, GenZ-ICP [ZS14; Viz+23; LLH25]	Rotating LiDAR only	None	Feature or raw-point scan-to-map registration; within-scan timing compensation
LiDAR-inertial: LIO-SAM, FAST-LIO, FAST-LIO2 [Sha+20; XZ21; Xu+22]	LiDAR and IMU	Optional graph factor or none	Preintegrated deskewing; local estimator unless a back end is added
Legged: VILENS, Ramezani et al. [WCF23; Ram+20]	Inertial, visual, LiDAR, leg	None documented, or learned segment descriptors	Legged-platform precedents; leg measurements inside the estimator
RTAB-Map [LM19]	External odometry with LiDAR and camera links	Appearance and proximity links	Replaceable odometry, bounded memory, published map-to-odometry correction

Route execution. Route execution uses two horizons. A global planner searches a map or graph beyond the sensor horizon and answers connectivity and route length toward a distant waypoint. A local planner evaluates short trajectories against recent sensor data, the local map, the robot footprint, and velocity limits. The two levels disagree whenever a newly observed obstacle invalidates the global plan, so a stack also needs replanning, path pruning, recovery behaviors, or a relay that shortens the commanded route. Nav2, the navigation stack of the second-generation Robot Operating System (ROS 2), implements that split explicitly: a planner server searches a global costmap, a controller server executes on a local one, and replaceable plugins fill both roles [Mac+20; Mac+23]. Independent layers assemble the costmaps, so static geometry, live observations, and inflation remain separately maintainable [LHS14]. The Autonomous Exploration Development Environment (ADEV), the ground-autonomy pipeline used as one route backend, divides the work into route planning, terrain analysis, and a local planner over terrain-analysis point clouds [Cao+22]. Plugin behavior and defaults vary across ROS 2 distributions, so a configuration statement names its distribution.

Map and path families. Grid and costmap search operates on a two-dimensional grid of free, occupied, unknown, or inflated cells [Mac+23]. Grid search uses Dijkstra or A* [HNR68]. Nav2's model predictive path integral (MPPI) controller applies sampling-based predictive control [Wil+16] to the local costmap at control rate [Mac+23]. The planar grid discards the vertical structure of section 2.2, so a free cell does not establish support.

Sampling-based planning searches a continuous configuration space instead. A rapidly exploring random tree grows toward random samples [LaV98], and a probabilistic roadmap builds a reusable graph from collision-free local connections. The asymptoti-

cally optimal variants RRT* and PRM* attain optimality only under stated sampling and connection assumptions [KF11] that do not cover support-aware execution on a changing map. GBPlanner1 adds exploration gain under traversability and dynamic limits [Dan+20]; GBPlanner2 adds the support and inclination admissibility of section 2.1 [Kul+22].

Field-based and surface-based methods change local behavior. Artificial potential fields combine goal attraction with obstacle repulsion at control rate, yet they stall in local minima and narrow passages [Kha86]. Euclidean distance fields replace the repulsive term with metric clearance and its gradient, as established in section 2.2, and remain silent on support, slope, and step height [Ole+17; Mil+24]. Surface-geodesic routes measure distance along the walked surface and preserve terrain connectivity [Püt+21], subject to section 2.2; holes and disconnected components can hide valid paths.

Terrain-aware legged planning states constraints that free-space tests omit: ground under the path, slope inside the gait envelope, bounded steps, body clearance, and reachable footholds. Elevation maps hold these quantities compactly [FH16; FBH18], and legged navigation scores slope, roughness, and step cost against robot capability during path search [Wer+16]. Reachability planning can restrict body poses to those a learned foothold model accepts, with those decisions remaining conditional on the robot and the training distribution [WH21]. Calibrated transforms, pose estimates, and time-consistent measurements belong to the route-planning contract rather than to operational detail.

Integrated exploration systems. Integrated systems couple goal selection to route execution. TARE couples the two inside one hierarchy, as section 2.1 records. GBPlanner2 adds global relocation toward observed frontiers, and COHORT coordinates legged and aerial teams under intermittent communication [Kul+22]. The Fast, Attemptable Route (FAR) Planner maintains a global visibility graph over planar polygonal obstacles and reconnects edges as visibility changes, supplying route guidance above terrain analysis and collision avoidance [Yan+22; Cao+22]. Autonomous Spot combines traversability mapping, perception-aware local planning, and mission behaviors on a legged platform [Bou+20], and Miller et al. route through a topological tunnel graph [Mil+20]. Each carries its own exploration objective, map format, and sensor contract, so substituting one replaces goal selector and navigator together and removes the interface between their evaluations.

Choice in this thesis. This thesis retains the waypoint boundary of section 2.1 and evaluates two route-execution candidates across it. ADEV provides terrain-aware local execution over non-planar support [Cao+22]; Nav2 provides ROS 2 planner and controller interfaces over costmaps [Mac+20]. Chapter 7 reports the evaluated configurations and results.

2.5. Legged Locomotion and Reinforcement Learning

The route from section 2.4 still requires locomotion. Quadruped mobility separates into layers with distinct inputs and control authority, and this thesis uses two gait terms throughout. A swing leg remains off the ground while moving toward its next foothold; a stance leg contacts the ground and carries part of the body load. A locomotion controller maps velocity, body, or foothold commands onto joint actions while regulating the contacts that the stance legs establish.

Model-based control. Model-based whole-body control and model predictive control (MPC) regulate body motion through dynamics, contact, and force constraints, assuming actuators that track commanded torques. The ANYmal platform supplies a hardware reference, with compliant torque-controlled joints supporting walking, trotting, running, and stair manoeuvres [Hut+16]. A route plan does not stabilize contacts; the controller beneath it must. The principal advantage is explicit assumptions: friction cones, force limits, and body clearance remain inspectable. The same explicitness becomes a burden on deformable or poorly observed ground, where model mismatch, contact uncertainty, and optimization latency matter more, and runtime tuning grows with gait and terrain complexity. The controller must also receive contact schedules and foothold candidates. Terrain-aware planning inherits a related weakness from its map, which may become stale or wrong under occlusion or vegetation, and foothold feasibility alone does not guarantee contact stability.

Simulation-trained control and blind policies. Simulation-trained control replaces the explicit model with a learned policy. Hwangbo et al. transfer a model-free policy to ANYmal without on-robot fine-tuning and report velocity tracking, fall recovery, and running [Hwa+19]. That transfer rests on a matched robot model, a learned actuator network, and a stochastic model set, and it does not establish transfer to another morphology. A blind policy receives no terrain geometry before contact. Lee et al. train a proprioceptive student policy on body state, commands, and previous actions under a privileged teacher that receives terrain and contact information; the student transfers to ANYmal in mud, snow, rubble, vegetation, and moving water [Lee+20]. That outcome belongs to the trained policy and its randomization rather than to blindness as a method family. Blind control gives up preview. It cannot inspect an unseen gap before contact and adapts only after the foot lands. Isaac Gym makes such training practical by running simulation and policy learning on one GPU [Mak+21]. Rudin et al. train ANYmal C on thousands of parallel robots with proximal policy optimization (PPO) [Rud+22; Sch+17], and their observation includes 108 terrain-height samples, so the definition here does not classify that policy as blind.

Perceptive locomotion. Perceptive locomotion adds exteroception to restore terrain preview. Miki et al. fuse proprioception with height samples from a robot-centric elevation map, train the policy by imitating a privileged teacher against noisy and

incomplete maps, and deploy it across seasonal natural and urban terrain [Mik+22b]. The same paper documents exteroception failures: stereo depth needs texture, and depth degrades under difficult lighting, dust, fog, transparent or reflective surfaces, and sensor occlusion. Yu et al. keep model-based control underneath learned perception; a vision policy outputs footholds and base motion above swing-leg position control and MPC stance control [Yu+22]. Both designs require calibrated sensors, reliable terrain reconstruction, and training coverage of the perception variation they will meet.

Choice in this thesis. This thesis uses a proprioceptive policy without terrain-height observations, because it fits the available command and observation interface, which accepts a velocity command with a fixed proprioceptive state vector and emits joint-position targets, without a second live terrain map inside the controller. Perceptive locomotion remains an open alternative, not a settled rejection. Chapter 8 presents the training of this policy and reports its transfer assumptions and failure cases from simulation evidence.

2.6. Simulation Platforms for Robotics Validation

The evaluation first uses simulation before physical deployment and then relies on it after joint actuator faults end physical operation and remove robot access. Platform choice depends on ROS 2 integration and on physics and sensor fidelity for the evaluation question. A comparative preprint benchmark of Gazebo, MuJoCo, PyBullet, and Webots finds that larger timesteps increase speed while reducing physical validity, and that simulator ranking changes with the metric and the scenario [Kör+21]. The benchmark does not cover Isaac Sim or Isaac Lab, so this thesis does not transfer its rankings to those platforms. Role rather than quality therefore organizes the platforms below.

Webots places world files, robot models, controllers, sensors, actuators, and a rigid-body engine in one desktop application [Mic04; Cyb24], and ROS 2 connects through the `webots_ros2_driver` package. The official modelling guide documents sensor and actuator noise alongside physics-engine noise, which supports interface testing under controlled perturbation[5], while contact behavior still depends on the model and the engine configuration. Gazebo Sim instead separates scene description, transport, plugins, and selectable physics engines, and its official physics documentation states that no engine is best for every task[6]. Its ROS 2 path runs through the `ros_gz_bridge` package[7], which is an additional integration boundary rather than a quality difference.

MuJoCo occupies a narrower role. It addresses articulated-body dynamics and contact through an explicit stepping interface that runs headless[8], which suits policy

[5] `https://cyberbotics.com/doc/guide/modeling`
[6] `https://gazebosim.org/libs/physics/`
[7] `https://gazebosim.org/docs/latest/ros2_integration/`
[8] `https://mujoco.readthedocs.io/en/stable/overview.html`

Table 2.4: Simulation platforms, distinguishing properties, and roles in this thesis.

Platform	Distinguishing property	Role and evidence boundary
Webots [Mic04; Cyb24]	Integrated world, robot, sensor, and controller application with a direct ROS 2 driver	Middleware, sensing, mapping, and navigation checks; interface behavior under the configured models, no statement about physical contact
Gazebo Sim	Modular physics engines and plugins with bridge-mediated ROS 2 exchange	Not used; retained as a middleware-centric alternative, since the bridge adds an interface boundary without resolving sensor or contact fidelity
MuJoCo	Articulated-body dynamics and contact with explicit headless stepping	Not used; retained for policy and solver studies, since a full exploration run needs separate world, sensor, and ROS 2 layers
Isaac Sim	USD scenes, GPU physics, ray-traced sensors, and a ROS 2 bridge in one scene	Terrain worlds and full-stack exploration experiments; pipeline behavior in the declared synthetic worlds under the simulated robot model
Isaac Lab [Mit+25]	Vectorized GPU environments with domain randomization and learning integrations	Rough-terrain locomotion policy training and evaluation; policy behavior under the declared terrain, observation, reward, and physics settings

prototypes, system identification, and solver studies, while a full exploration experiment requires additional world, sensor, and middleware components. Isaac Sim combines Universal Scene Description (USD) scenes, GPU physics, ray-traced rendering, physics-based sensors, and a ROS 2 bridge, so articulated robots, uneven terrain, and sensor publication coexist in one scene. Its ray-traced sensor models produce material-dependent returns under documented configuration settings[9], and version, rendering, collision, actuator, friction, and contact settings remain part of the experiment definition. Isaac Lab adds vectorized environments, domain randomization, tiled rendering, seeded workflows, and reinforcement-learning integrations on top of Isaac Sim [Mit+25], continuing the line of GPU simulation for robot learning [Mak+21] and parallel locomotion training [Rud+22]. Identical results require matching software, hardware, and determinism settings, and may still differ across hardware[10].

Sensor and contact fidelity bounds simulation claims. A geometric ray or depth sensor tests field of view, coordinate conventions, topics, and map integration, and omits lens artifacts, multipath returns, firmware filtering, and device delays. Dynamics randomization reduces the dependence of a learned controller on one parameter set [Pen+18], and simulation-trained legged skills have transferred to hardware under controlled procedures [Hwa+19], but real actuators, contacts, sensors, and safety still require physical validation. The absence of robot access prevents physical validation of the final configuration. Table 2.4 states each platform's role and the evidence it can carry; chapter 4 reports the configurations, and chapter 8 reports the policy training.

[9] `https://docs.isaacsim.omniverse.nvidia.com/latest/sensors/isaacsim_sensors_rtx.html`

[10] `https://isaac-sim.github.io/IsaacLab/main/source/features/reproducibility.html`

2.7. Synthesis and Research Gap

The literature reviewed here supplies each function that this thesis requires, and exploration systems differ mainly in candidate generation, gain, and path cost.

What that work leaves open is a single assignment of those functions that holds across systems. Each system fixes its own map format, timing requirements, sensor contract, and controller interface, and the integrated planners couple goal selection to route execution. The interfaces therefore carry the risk. A corrected pose has to reach a volumetric map that keeps unknown space explicit, that map has to yield Euclidean clearance and supported three-dimensional goals, a terrain-aware navigator has to execute those goals, and each route has to lie inside the envelope of the locomotion controller. Each publication validates its method inside its own stack, so the compatibility of these contracts on one quadruped platform and sensor payload remains open.

That question scopes the remainder. Chapter 3 defines the architecture, platform, and interface contracts, chapters 4 to 8 report the implementation of each layer, chapter 9 reports the evaluation, and chapter 10 states which interface conditions the evidence settles.

3

System Architecture

This chapter documents the architecture of the autonomous exploration stack that runs on the Boston Dynamics Spot quadruped. The design separates mechanical actuation, sensing, computation, software middleware, planning, and deployment into distinct layers, and it places each computational component explicitly on the robot or on an offboard workstation. Explicit interface contracts connect the layers.

Section 3.1 covers the quadruped morphology, payload mounting, the low-level motor interface, and the lease and emergency-stop behaviour that governs every commanded motion. Section 3.2 presents the complementary sensor suite and states the interface contract of each stream. Section 3.3 defines the compute hosts, the network paths, the container and middleware boundaries, and the clock. Sections 3.4 and 3.5 respectively document package ownership, topic schemas, the transform tree, navigation backends, and the closed-loop autonomy pipeline with its velocity conditioning and safety arbitration. The final section covers deployment and data collection. Physical and simulated sensor sources enter through the same interfaces, so their message, timing, and transform contracts are introduced here alongside the robot. Chapter 4 then explains the roles of Webots, Isaac Sim, and Isaac Lab and the evidence each environment can provide.

3.1. Hardware Platform: Boston Dynamics Spot

The experimental platform is the Boston Dynamics Spot quadruped [Bos24]. Four symmetric limbs carry twelve active revolute joints, with hip abduction, hip flexion, and knee flexion on each leg; comparable research quadrupeds use the same joint arrangement [Hut+16]. Manufacturer dimensions give a nominal standing footprint of 1.10 m by 0.50 m and a shoulder height between 0.61 m and 0.70 m [Bos24]. Commercial specifications state an operational mass of 32.7 kg, including the 4.2 kg battery pack, and a payload rating of 14.0 kg on the dorsal rails [Bos24]. The Unified Robot Description Format (URDF) model used for simulation assigns 32.86 kg to the base link and, for



Figure 3.1: Instrumented Boston Dynamics Spot platform.

each of the four legs, 1.68 kg to the hip, 2.34 kg to the thigh, and 0.35 kg to the shank. The base link already carries the complete chassis and battery package, so the model totals 50.34 kg and exceeds the manufacturer operational mass by the summed limb masses. That total is an inertial-modelling convention rather than a measured mass.

Standard dorsal T-slot rails carry the payload [Bos24]. An elevated mast and roll-cage hold a Velodyne Puck VLP-16 rotating LiDAR on the central longitudinal axis above the rear chassis. Five factory stereo camera pairs sit flush in the chassis perimeter. A Spot CORE input/output (I/O) unit occupies the rear payload bay and connects by a wired Ethernet umbilical to the operator workstation that Section 3.3 defines. Figure 3.1 shows the assembled platform, with the sensor mast visible above the chassis and the body otherwise unmodified. Comparable elevated sensor payloads appear in published subterranean and mine exploration deployments with the same platform [Bou+20; Mil+20].

The software control boundary is the ROS 2 driver, which reaches the robot base through the Google Remote Procedure Call (gRPC) interface [BDA24]. Velocity commands stream on `/spot/cmd_vel` and expire after a 0.25 s lease timeout. Dedicated ROS 2 services govern motor power and stance transitions [BDA24]. Base control uses single-client lease arbitration, so exactly one client holds the motor lease at any moment. During physical trials, touching the handheld tablet silently preempts the lease: the driver continues to accept velocity commands while the firmware discards them without raising a ROS error. Autonomous runs therefore keep the tablet aside and release its lease while the operator retains control of the physical emergency stop, which opens the motor contactors and removes motor power outside the ROS command path.

The kinematic tree follows the conventions of ROS Enhancement Proposal (REP) 105, from `map` through `spot/odom` and `spot/body` to the mounted sensor links, which matches the arrangement in published legged state-estimation systems [WCF23]. Driver inspection identifies one deviation from the message contract: the twist field of the published odometry message carries world-frame rather than body-frame linear velocity. Replay of recorded physical data resolves the interpretation. The world-frame interpretation matches motion with $0.019\,\mathrm{m\,s^{-1}}$ root-mean-square error, compared with $0.506\,\mathrm{m\,s^{-1}}$ for a body-relative interpretation. The downstream monitors account for this distinction before closed-loop tracking begins.

Two further hardware limits shape the stack. Across 653 measured samples, the low-level gait controller silently suppresses linear velocity commands below approximately $0.13\,\mathrm{m\,s^{-1}}$ to $0.15\,\mathrm{m\,s^{-1}}$, so the command path conditions candidates before they reach the driver, as Section 3.5 specifies. Joint actuator faults end physical operation entirely and move the final validation into simulation while the kinematic model and the interface contracts of this chapter remain unchanged. Chapter 4 records the resulting evidence boundary.

3.2. Sensor Suite: Velodyne VLP-16 and Depth Cameras

Autonomous exploration requires perception at two spatial scales, and the platform assigns them to different sensors. The elevated Velodyne Puck VLP-16 supplies panoramic horizontal measurements for state estimation and far-field obstacle detection [Vel19]. Five stereo depth camera pairs embedded in the chassis perimeter, at the front-left and front-right heads, the two lateral flanks, and the rear, resolve terrain beneath and around the body [Bos24]. The dorsal payload rails mount the rotating sensor behind the body origin and orient it to match the payload plate; Table 3.1 gives the exact transform. The chassis mounts each camera pair with a downward pitch, and the same table lists the pose ranges. This combination assigns long-range structure to the rotating sensor and near-field support geometry to the depth cameras.

Table 3.1 records the sensor interfaces that the stack uses. Velodyne manufacturer specifications come from the Puck VLP-16 datasheet [Vel19]; separate entries identify this work's configurations and measured rates, and "not published" marks information absent from the available manufacturer documentation. Three entries have direct consequences. First, the factory payload service decimates the rotating-sensor cloud and publishes it at approximately 3 Hz; the bypass in Section 3.3 restores approximately 9.5 Hz. Second, the physical depth streams carry 16-bit unsigned integers in millimetres with a scale of 999 counts per metre, whereas both simulators publish 32-bit floating-point metres, so the ingestion node detects the encoding and applies the matching scale rather than assuming either one. Third, each hardware depth stream delivers only 1.0 Hz to 1.5 Hz against an aggregate service ceiling of seven to ten images per second, so volumetric integration must accept asynchronous cameras rather than synchronize all five. Message types, reliability policy, and role in the stack are fixed entries of

the table; rates, ranges, encodings, and sensor names are environment-specific, and the angle-bracketed element of each depth topic names the camera at one of the five perimeter positions, with one stream per stereo pair. The driver and the simulators spell those names differently, and the launch configuration reconciles the two so that downstream nodes read five depth topics with matching camera-info topics.

Table 3.1: Sensor interface contracts.

Contract element	Velodyne Puck VLP-16, physical	Five body depth cameras, physical	Five body depth cameras, simulated
Channels and angular resolution	16 channels; 2.0° vertical spacing; 0.2° azimuth increment at 10 Hz rotation	Five factory stereo pairs; the driver rectifies each stream to a pinhole model with zero distortion; pixel format and sensor internals are not published	Ideal pinhole projection with zero distortion
Field of view	360° azimuth; 30° elevation, spanning ±15.0°	Approximately 90° horizontal and 59° vertical per camera	Modelled on the physical mounting geometry
Manufacturer range	Up to 100 m	Not published	Not applicable
Configured range	Minimum 0.50 m; cutoff 20 m; the derived planar slice covers 0.50 m to 8.0 m	0.3 m to 2.5 m	0.3 m to 3.0 m
Native measurement rate	10 Hz rotation; approximately 300 000 points per second in single-return mode	Requested at 15 Hz per camera	Bounded by the simulator step rate
Delivered publication rate	Through the bypass: approximately 9.5 Hz at 29 184 points per cloud, about 627 KiB. Through the factory payload service: approximately 3 Hz at 5752 points. At the recorded low-battery packet rate: about 6.2 Hz	1.0 Hz to 1.5 Hz per camera, bounded by an image service ceiling of seven to ten images per second across all channels	3.0 Hz in Webots; up to 10.0 Hz in Isaac Sim

Continued on next page

Table 3.1: Sensor interface contracts (continued).

Contract element	Velodyne Puck VLP-16, physical	Five body depth cameras, physical	Five body depth cameras, simulated
Topic and message type	`/velodyne_points` as `sensor_msgs/msg/PointCloud2`; derived `/scan` as `sensor_msgs/msg/LaserScan`	`/spot/depth_registered/<cam>/image` and the matching `camera_info`, as `sensor_msgs/msg/Image` and `sensor_msgs/msg/CameraInfo`, best-effort reliability	The same message types and the same reliability policy
Encoding and scale	Fields `x`, `y`, `z`, `intensity`, `ring`, and `time`, in metres	`16UC1` in millimetres, 999 counts per metre	`32FC1` in metres
Transport	Unicast User Datagram Protocol (UDP) on port 2368 to the payload computer, link-layer tap to local Transmission Control Protocol (TCP) port 2369, tunnel to the workstation, and local re-injection at about 750 packets per second	Spot software development kit over gRPC into the driver nodelets	Simulator ROS 2 bridge inside the simulation container
Frame	`spot/velodyne`, at $x = -0.2375$ m, $y = 0$ m, $z = 0.1914$ m and 36.0° yaw relative to `spot/body`	Five optical frames below `spot/body`, pitched down 23° to 27° at 0.63 m to 0.67 m elevation	Five corresponding simulator camera optical frames
Role in the stack	Odometry refinement and loop closure, planar safety slice, and far-field obstacle detection; excluded from volumetric mapping	Sole input to volumetric mapping and to the costmap derived from it	The same roles as in the physical configuration

The simulated rotating sensor retains 16 layers. Webots publishes it at 5.0 Hz with a 1.00 m minimum range, while the Isaac Sim ray-traced sensor delivers approximately 2.5 Hz to 4.0 Hz. The simulator names its link `Velodyne_Puck`, a simulator asset name that its ROS 2 bridge republishes on `/velodyne_points` under the frame contract of the physical configuration. Depth integration uses two motion gates: it drops frames

above 0.25 rad s^{-1} yaw rate and integrates a frame after the camera optical centre moves 0.05 m, subject to a 10 s maximum skip.

Measurements, rather than preference, fix the division of labour between the modalities. The comparison shows that fusing both streams into one volume corrupts the reconstruction because grazing floor returns integrate across elevation bins, carve false holes in the floor, and deposit phantom surfaces above it. A controlled comparison on the physical robot over one indoor floor reports 88.9 % floor coverage for the fused configuration and 99.6 % for the cameras alone. The stack therefore uses the five depth cameras as the sole input to volumetric mapping, which supplies the signed distance fields and the floor-relative two-dimensional costmap [Mil+24]. It excludes the rotating sensor from volumetric integration and dedicates it to odometry refinement and loop closure in RTAB-Map [LM19], the planar safety slice on `/scan`, and far-field obstacle detection. Chapter 5 reports the measurements behind this decision; the later sections of this chapter reuse the conclusion without restating it.

3.3. Onboard Compute and Robot Communication

Computation divides across three hosts. The Spot base computer runs joint control, balance, and leg odometry [Bos24]. The Spot CORE I/O payload unit, reachable at Internet Protocol address `192.168.80.3`, acts as the payload gateway and hosts the packet forwarder described below; later paragraphs call this unit the payload computer. Volumetric mapping and state estimation run offboard on the operator workstation. That workstation is a laptop with an Intel Core Ultra 9 285H processor and an NVIDIA GeForce RTX 5070 Ti Mobile graphics processor under Linux. Driver version `590.48.01` supports a working Isaac Sim renderer on the Blackwell laptop GPU. The workstation executes volumetric mapping, pose-graph optimisation, exploration, route planning, monitoring, and the simulation environments.

Telemetry and camera streams share the payload Ethernet subnet, which the workstation reaches over the wired umbilical of Section 3.1, and the wireless transport is a separate off-board link that does not carry the full-rate rotating-sensor stream. The `spot_driver` package connects to the base over gRPC and delivers proprioceptive telemetry, leg odometry, and the five depth streams [BDA24]. The rotating sensor uses a separate path. Its packets arrive by unicast UDP on port 2368 of the payload computer, where the factory payload service already holds that port, so a conventional driver cannot bind it, and the service itself publishes only a decimated cloud. A forwarder on the payload computer therefore reads raw link-layer frames through an `AF_PACKET` socket without binding the port and streams them to local TCP port 2369. A Secure Shell (SSH) tunnel carries that stream to the workstation, where a receiver re-emits the original datagrams on local UDP port 2368. The stock `velodyne_driver` and `velodyne_pointcloud` nodes then unpack them unchanged, which preserves compatibility with the standard driver while leaving the factory service untouched.

Virtualisation separates incompatible dependency sets. The production autonomy stack runs in a Docker container named `spot_container` on Ubuntu 22.04 long-term support with ROS 2 Humble, using host networking so that large point clouds and image streams avoid a virtual network hop. Separate containers host Webots (`webots_sim`) [Mic04] and Isaac Sim (`isaac-sim`); Chapter 4 defines the additional bridge and planning containers on the Isaac path. Communication inside ROS 2 uses the eProsima Fast implementation of the Data Distribution Service (DDS). Tests show that default socket buffers drop more than 80 % of the 627 KiB point clouds on the loopback interface, so the production profile `fastdds_largemsg.xml` raises socket buffers to 16 MB and enables shared-memory transport between processes in the same container. Cross-container tests use a UDP-only profile to avoid stale shared-memory segments. Every launcher creates a new process group, records its identifier, and tears down the group explicitly.

Three evaluations require middleware outside ROS 2 Humble, and their evidence boundaries are strict. An isolated ROS 1 Noetic container runs the `gbplanner3` branch of GBPlanner from the OmniPlanner line [Zac+26] with its own voxblox map [Ole+17]. The graph-based planning literature describes the earlier published versions [Dan+20; Kul+22]; this thesis instead evaluates the July 2026 build of the branch and keeps its result separate from the published results. Two other sandboxes run in ROS 2 Jazzy: Elevation Mapping CuPy [Mik+22a] and the MIT-SPARK Hydra scene-graph mapper [HCC22]. Elevation Mapping CuPy receives its input from the Humble stack on port 5602 and emits two products to the ROS 1 container: a reachability grid converted to an occupancy grid on port 5600, and the raw elevation layer on port 5603. The ROS 1 planner returns its waypoint on port 5599, while a service relay travels in the opposite direction on port 5601. Hydra consumes recorded data only. No relay connects Hydra because its place and generalised Voronoi graph layers do not form during offline evaluation: surface-band voxels never reach the place extractor on that build, so the evaluation stops before a live interface becomes meaningful.

Humble and Jazzy never exchange DDS traffic. The compatibility test freezes the workstation twice when a Jazzy participant runs on the same domain and host network as the Humble stack. The discovery message type in `rmw_dds_common` changes its binary representation between the two distributions, a receiving participant misreads a field as an array length, and the resulting allocation exhausts host memory outside the container limits. Standard bridging does not solve the fault. A domain bridge that links both distribution schemas inside one process reproduces the same failure, and unstable endpoint behaviour rules out the Zenoh bridges. The stack therefore assigns Jazzy containers their own domain identifiers on bridge networks and moves live data across distribution boundaries through application-level TCP relays that frame each message with a length prefix. Figure 3.2 shows the arrangement: the three robot-side computers above, the workstation containers inside the dotted boundary with their ROS 2 distributions and domain identifiers, and the isolated sandboxes in the right

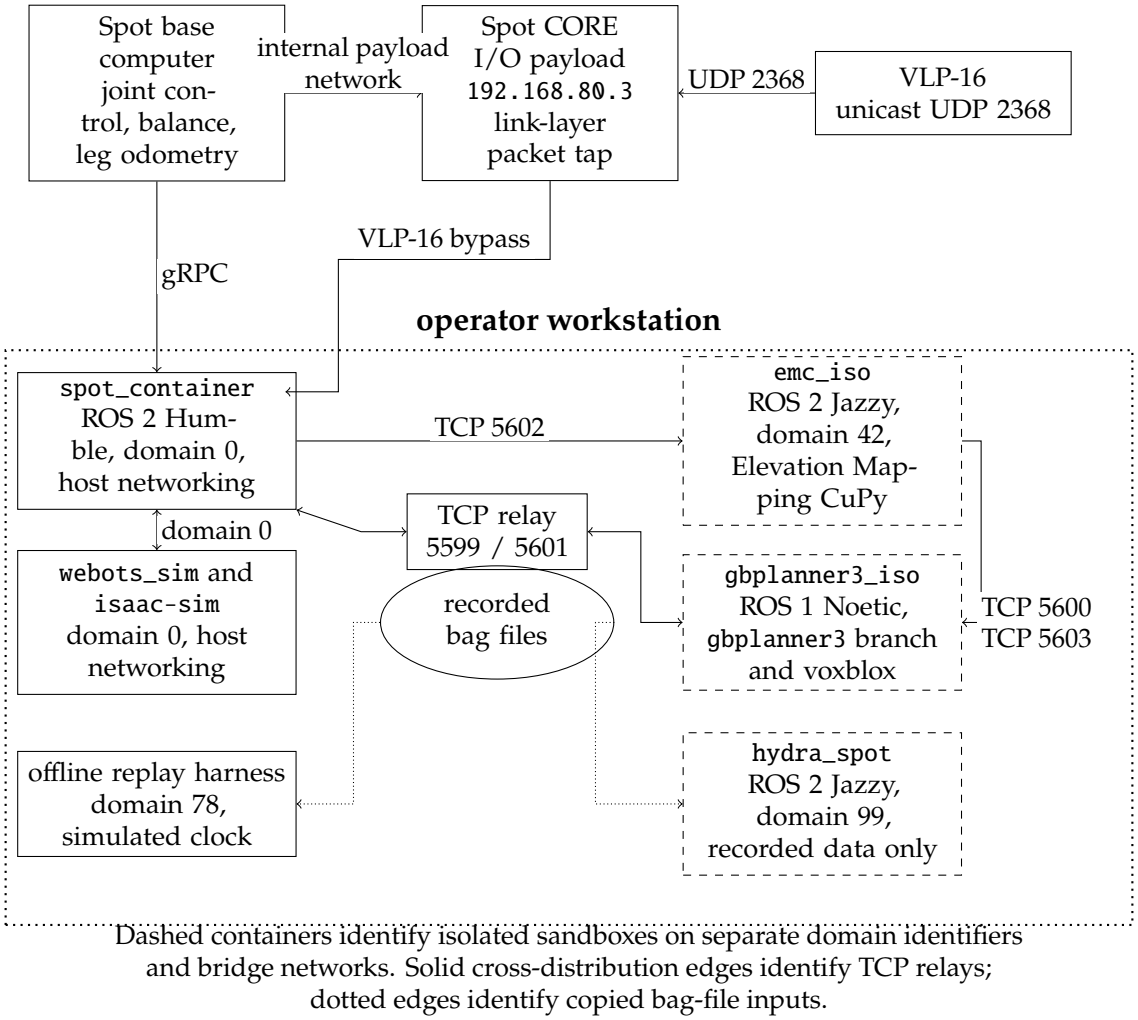


Figure 3.2: Compute and communication architecture.

column, with every edge labelled by the transport it carries and the legend below the boundary naming the edge styles.

Clock handling differs by environment. Physical trials use host wall-clock time, while simulation containers and offline replay follow a published clock topic. In Webots, the default sensor plugins timestamp messages at publication rather than acquisition. This choice places rotating-sensor timestamps 56 ms ahead of acquisition and one front depth stream 40 ms ahead, which distorts reconstructed surfaces while the body walks. Tying each timestamp to the corresponding simulator buffer update removes the offset. Simulated time also dilates under heavy rendering load, so no node assumes a fixed ratio between simulated and wall time.

3.4. Software Stack and ROS 2 Integration

The stack partitions into modular packages with a single owner for each product; its runtime and transport follow Section 3.3. The `spot_driver` package wraps the robot

software development kit over gRPC, publishes joint states, battery telemetry, and base odometry, and exposes trigger services for motor power, standing, and seated shutdown [BDA24; Bos24]. For state estimation, `rtabmap_slam` refines odometry by scan-to-scan matching and maintains pose-graph loop closures [LM19]. Real-time reconstruction runs in `nvblox_live`, which maintains truncated and Euclidean signed distance fields on the graphics processor and emits the products that it derives from them [Mil+24]. The exploration node `nvblox_explorer.py` clusters frontier voxels and publishes goals. Two arbitration nodes, `nav2r_cap_floor.py` and `nav_safety.py`, close the command path, and Section 3.5 defines their behaviour.

The transform tree keeps a single parent for every frame and a single publisher for every edge. Here `rtabmap_slam` publishes the correction from `map` to `spot/odom`, and `spot_driver` alone broadcasts `spot/odom` to `spot/body`; static transforms attach the sensor links below the body frame. This separation keeps local trajectory tracking continuous while confining discrete loop-closure jumps to the global frame. The same rule governs map topics: `nvblox_live` alone publishes the authoritative costmap on `/map` from a horizontal slice of the Euclidean signed distance field, and the stack remaps the RTAB-Map occupancy grid to `/rtabmap/map` so that two publishers never contend for one topic. Figure 3.3 shows the node graph with the topic that connects each pair of nodes, together with the transform tree and its owners.

Message types rather than conventions declare the interfaces. The rotating sensor publishes `/velodyne_points`. Its message type is `sensor_msgs/msg/PointCloud2`; `pointcloud_to_laserscan` derives the planar `/scan` slice from it. The exploration nodes publish goals strictly as `geometry_msgs/msg/PointStamped` in the `map` frame on `/way_point` or `/nav2x/explorer_way_point`; neither topic accepts a pose message, so a goal implies no orientation. The ground-autonomy bridge consumes `/way_point` and the Nav2 relay consumes `/nav2x/explorer_way_point`, so the active backend determines which name carries the goal. Candidate velocities travel on intermediate topics as `geometry_msgs/msg/Twist`; Section 3.5 defines the output owner and safety arbitration.

This thesis evaluates alternative backends behind these interfaces in the isolated containers of Section 3.3 and reports their outcomes as project results. The `gbplanner3` ground branch returns one vertex and no edges on the harness that produces complete graphs for its aerial configuration; the finding applies only to the July 2026 build of that branch. An offline Elevation Mapping CuPy comparison shows that a terrain reachability gate can change the order in which regions receive visits, although three concurrent distributions exceed the compute available for live operation. The Hydra evaluation stops for the reason that Section 3.3 gives. Chapter 6 compares these goal-generation systems as such.

Several iterations produce the present navigation layer. The first baseline pairs `slam_toolbox` with `explore_lite` and Nav2 [Mac+20; MJ21]. It fails because two-

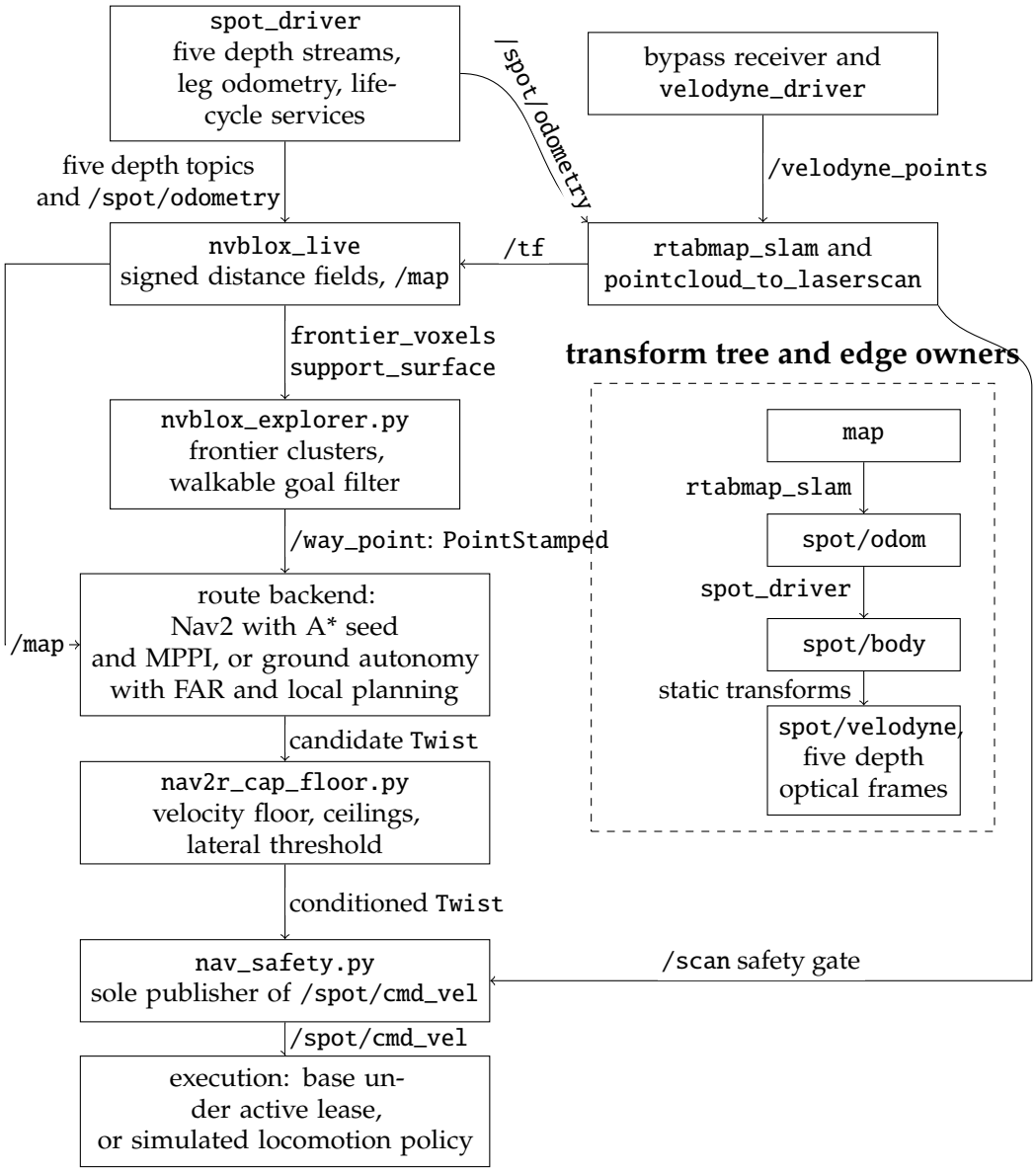


Figure 3.3: ROS 2 nodes, topics, and transforms.

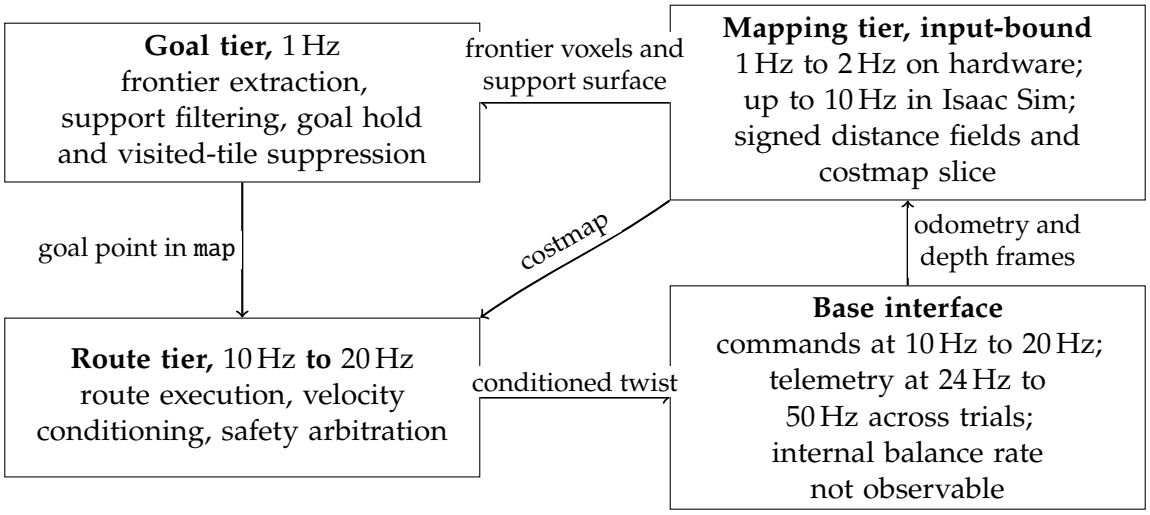


Figure 3.4: Autonomy-loop interfaces and update rates.

dimensional occupancy projection collapses overhanging geometry and because `explore_lite` terminates permanently the first time the candidate set becomes momentarily empty. Unbounded memory growth rules out point-cloud accumulation. The architecture now admits two route-execution backends behind one waypoint contract. The Carnegie Mellon University ground-autonomy stack combines the Fast, Attemptable Route (FAR) visibility-graph planner [Yan+22] with terrain-aware local planning [Cao+22] for multi-level and overhanging geometry. Nav2 pairs an A* global seed path [HNR68] with a model predictive path integral (MPPI) controller for continuous execution on a single surface [Mac+20]. Neither backend is generally superior, and Chapter 7 reports the comparison together with the conditions that govern each use.

3.5. The Autonomy Loop: Perception, Exploration, Navigation

The autonomy loop couples perception, mapping, frontier extraction, route execution, and locomotion in a feedback cycle whose stages run at different rates. Frontier extraction and goal selection run at 1 Hz. Volumetric integration and the costmap slice are input-bound, reaching approximately 1 Hz to 2 Hz on hardware and up to 10 Hz in Isaac Sim. Route execution, velocity conditioning, and safety arbitration run at 10 Hz to 20 Hz. The stack sends commands to the physical base in that band; trial records place incoming driver telemetry between approximately 24 Hz in mapping trials and 50 Hz in interface diagnostics. The driver does not expose the proprietary balance-controller rate. Figure 3.4 separates these rates from the internal controller and shows the four tiers with the signals exchanged between them.

Perception converts the exteroceptive streams into metric representations on the workstation graphics processor [Mil+24]. The five body cameras supply volumetric

mapping, while the rotating sensor supplies loop-closed state estimation [LM19] and the planar `/scan` slice for obstacle clearing and protective gating. Section 3.2 documents that assignment and the evidence for it.

At the 1 Hz tier, frontier extraction operates on boundary voxels between mapped free space and unobserved space, which applies the classical frontier criterion to a volume rather than a plane [Yam97]. The walkable goal filter groups boundary voxels into spatial clusters before applying any test, then admits a cluster only if it passes four gates: support snapping with a height match, footprint walkability from slope, step, and roughness, geodesic reachability across the support surface, and body headroom. The filter may project a boundary goal that fails on geometry alone inward onto traversable ground. A 60 s minimum-hold timer for physical deployment bounds target churn, and permanent marking of visited tiles suppresses cycling around occluded pockets. Chapter 6 derives the gates and the scoring.

Both route backends attach at the same place. Each consumes the goal coordinate from the waypoint topic, resolves the transforms it needs, and emits a candidate twist on an intermediate topic. Neither addresses the driver, and the selection between them depends on whether the environment presents overlapping vertical levels or one continuous surface.

Velocity conditioning and safety arbitration occupy two nodes with a single owner for the output. The node `nav2r_cap_floor.py` shapes the candidate twist. It applies a symmetric linear floor of $0.15\,\mathrm{m\,s^{-1}}$ on hardware, at the top of the measured $0.13\,\mathrm{m\,s^{-1}}$ to $0.15\,\mathrm{m\,s^{-1}}$ deadband, and $0.40\,\mathrm{m\,s^{-1}}$ for the simulated locomotion policy, above the gait-initiation deadband that Chapter 8 measures. The ceilings are environment-specific as well: $0.30\,\mathrm{m\,s^{-1}}$ linear and $0.40\,\mathrm{rad\,s^{-1}}$ yaw on hardware, chosen to preserve stopping distance indoors, while the simulated configurations apply the values recorded with their results in Chapter 7 and Chapter 9. It also holds a lateral threshold of $0.05\,\mathrm{m\,s^{-1}}$ so that the floor does not amplify small lateral noise. The node `nav_safety.py` alone publishes `/spot/cmd_vel`. Hardware requires single ownership because concurrent publishers produce command chatter and can cost the motor lease that Section 3.1 describes. That node blocks forward motion whenever the planar scan reports an obstacle inside the near-field clearance zone, a configured radius of the node evaluated on the derived `/scan` slice, and it also blocks motion when the scan falls silent, so a failed sensor stops the robot instead of leaving commands unchecked. It enforces silence while no goal is active, attenuates gait-induced body rocking through a yaw low-pass filter with slew limiting, and gives immediate precedence to recovery commands, so one node resolves command priority.

Execution differs by environment while the command contract does not. On the physical platform, `spot_driver` streams the twist to the base under an active lease and the internal controller resolves joint kinematics [Bos24; BDA24]. In Isaac Sim, a graph bridge routes the same twist to a blind reinforcement learning locomotion

policy [Rud+22] that outputs position targets for twelve proportional-derivative joint controllers. Software supervision closes the loop. Progress checks compare Euclidean distance to the active waypoint and release the temporal hold on arrival. A stall monitor issues a recovery twist on the dedicated escape topic `/cmu_escape_cmd_vel` when measured displacement stays below 0.15 m for 15 s despite nonzero commanded motion. The mission terminates once eight consecutive planning cycles yield no reachable frontier, after which the stack commands the robot to park.

3.6. Operational Deployment and Data Collection

The stack operates in four modes that share one set of topic and transform contracts: the physical quadruped, staged hardware validation, offline replay of recorded data, and simulation in Webots and Isaac Sim [Mic04]. Section 3.3 defines the compute division. Staged hardware validation follows three stages in order. Stage A runs mapping and the frontier planner while an operator teleoperates the robot and suppresses actuation commands, which exposes goal quality without motion risk. Stage B enables actuation but decouples the explorer; manual goal dispatch isolates path tracking and deceleration. Stage C closes the loop and lets the explorer drive the robot. Offline replay runs the same nodes on a separate domain under a simulated clock.

Physical bring-up follows a fixed sequence. First, checksum comparison checks the protected nodes inside the container against their host mirrors, and a process audit confirms that no node from a previous run survives. The packet path for the rotating sensor then starts in the order forwarder, tunnel, receiver, because only one receiver may hold the local port. Next, the launcher starts the driver, powers the robot, and commands it to stand through ROS 2 services alone under the lease discipline in Section 3.1. It verifies the transform chain from `map` through `spot/odom` and `spot/body` to the sensor links before dispatching any goal. Autonomous exploration starts only after five manual gates pass under click-to-point control: three consecutive goal arrivals, zero lease losses or ignored commands, one path bending around a visible wall without contact or planner failure, correct execution of commanded forward and lateral velocities without crabbing or stalling, and a deliberate obstruction that confirms the safety gate halts the robot.

Run supervision combines a human safety layer with a diagnostic layer. An operator maintains line-of-sight supervision with the emergency stop of Section 3.1 in hand, and the operating rules restrict autonomy to bounded flat arenas and exclude staircases. Instrumentation makes each stop explanatory rather than ambiguous. Whenever motion ceases, runtime monitors emit one verdict from a fixed set: arrival and settling, goal superseded, safety gate, planner unable to produce a trajectory, path follower idle, or command ignored by the base. A goal ledger records every dispatched waypoint with its timestamp, coordinate, duration, and outcome. Rate monitors compare each interface against an expected band, and an additional check confirms silence on the commanded velocity topic while no goal is active. Section 3.5 specifies the safety node

that produces a halt.

Two failure modes motivate explicit process lifecycle control. The launchers spawn node subtrees in separate Portable Operating System Interface (POSIX) sessions and record their process group identifiers, so teardown scripts signal exactly those groups. This design removes an earlier pattern in which a shell command that matches process names by string terminates itself before reaching its targets. Teardown also stops supervisory watchdogs first and then allows a bounded grace period for socket release and state flushing before signalling the remaining groups, because signalling a supervised process first only causes its watchdog to restart it.

Data collection supports offline evaluation. The payload computer captures raw rotating-sensor packets through the same link-layer socket as the forwarder rather than over the wireless link that the earlier laptop-side recordings used, which preserves the full packet rate and removes wireless dropouts. A separate recording of the ROS interfaces provides the evaluation corpus. It covers single-floor indoor traversal, a multi-level traversal with stair ascents, dedicated stair sequences, and a visual sequence for appearance-based loop-closure evaluation [LM19]. Replay publishes a simulated clock and injects the static transforms that the recording engine does not latch for nodes that start after playback begins, so every geometric frame remains resolvable throughout. These recordings provide physical-robot evidence and remain distinct from simulation results; Chapter 4 states that boundary and Chapter 9 reports which measurements come from each domain. When joint actuator faults end physical availability, the shared interface contracts in this chapter allow the evaluation to continue in simulation without redefining them.

4

Simulation and Validation

Simulation provides a set of task-specific instruments. The software stack, sensor timing, terrain representation, contact response, and learned locomotion policy impose different requirements, so no single simulator provides equally strong evidence for every layer. This chapter records the division of responsibilities, the interfaces between the simulation systems, and the boundary between simulated evidence and physical Spot evidence. Chapter 9 reports quantitative run scores; Chapter 5 develops the mapping and timestamping details.

4.1. Simulation Strategy and Validation Roles

Layers organize the validation plan. Software-in-the-loop validation checks ROS 2 topics, transforms, clock handling, goal publication, and process coordination. Sensor and perception validation checks the geometry, rates, coordinate conventions, and timing assumptions that RTAB-Map and nvblox receive. World and pipeline validation checks whether terrain geometry, mapping, frontier selection, and terrain-aware execution operate together. Locomotion validation addresses contact and policy behavior. Physical-robot trials supply a separate evidence domain for sensor calibration, hardware scheduling, actuator behavior, and navigation under the real robot's constraints. These layers overlap in the running system, but a passing result at one layer does not certify the others.

Simulator comparisons establish a methodological point behind this division: each target task requires its own measurements of timestep, hardware load, real-time factor, and physical validity rather than an inference from the simulator name [Kör+21]. A simulator can provide a complete world, robot, controller, sensor, actuator, and physics workflow, as Webots does in the general mobile-robot setting that Michel describes [Mic04]. That capability makes it useful for interface and integration work, but it does not establish Boston Dynamics actuator fidelity or the timing of a physical VLP-16. The evaluation question therefore determines the validation role.

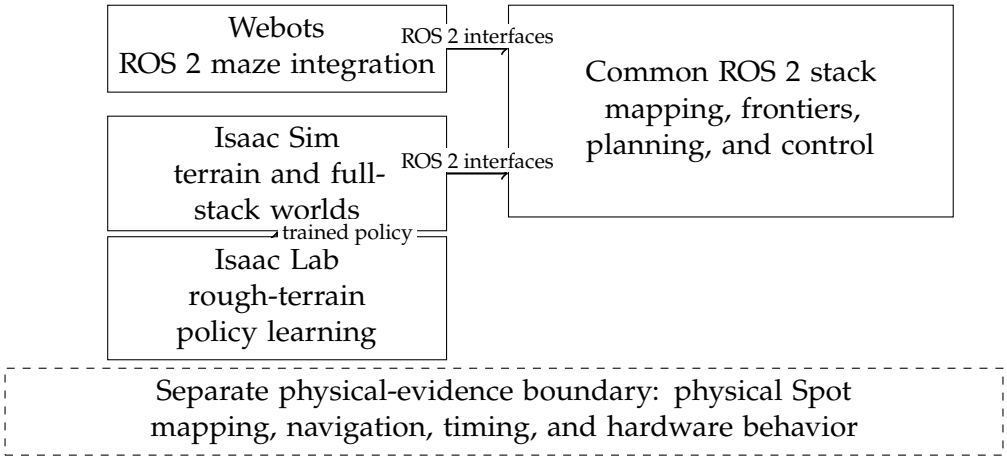


Figure 4.1: Validation roles in this thesis.

Figure 4.1 summarizes the arrangement. Webots supplies the first ROS-connected maze testbed, Isaac Sim supplies articulated terrain worlds and rendered sensor streams, and Isaac Lab supplies repeated policy rollouts. The common ROS 2 stack therefore runs against more than one simulated source, while the physical boundary prevents a software result from becoming a hardware-validation claim.

Webots supports the maze and ROS 2 integration. Isaac Sim supports uneven-terrain worlds, the first-party articulated Spot asset, GPU physics, rendered sensors, and the connection to the nvblox and Isaac Lab workflows. Isaac Lab covers GPU-vectorized rough-terrain learning, where the number of contact-rich rollouts matters more than a complete ROS graph. GPU-resident physics and training make this distinction practical in the Isaac Gym lineage [Mak+21], and Rudin et al. demonstrate massively parallel policy learning for rough locomotion on a single workstation GPU [Rud+22].

The validation split excludes Gazebo Sim and MuJoCo after consideration. This selection does not provide a universal ranking. The older cross-simulator comparison that includes Gazebo, MuJoCo, and Webots also shows why results depend on the selected task, model, timestep, and hardware rather than on a platform label alone [Kör+21]. The implemented split answers the separate integration, terrain-pipeline, and policy-learning questions without introducing another model-conversion and clock boundary.

The physical robot supports a substantial part of mapping and navigation development. Physical trials and replay tune sensor selection, loop closure, floor-relative mapping, and navigation behavior before the later simulation refinement of terrain and locomotion. Simulation therefore does not substitute for the preceding physical work. The subsequent loss of physical Spot access confines the final ramp and dune evaluation to the Isaac Sim domain, as Section 4.6 details.

4.2. Webots: Maze-World ROS 2 Integration

Webots comes first because integration poses the early risk rather than contact-rich locomotion. The integration brings up the ROS 2 graph, publishes simulated depth and LiDAR data, establishes the transform tree, supplies nvblox with data, displays the map in RViz, and connects frontier goals to navigation. Webots provides an integrated world and robot workflow suitable for this form of rapid prototyping [Mic04]. The setup uses the R2025a container and the stock maze world from the `webots_ros2_spot` package, published by the Mobile Autonomous Systems and Cognitive Robotics Institute through its MASKOR repository [Mob22]. The approximately 18 m by 28 m maze provides a repeatable environment for exploration and ROS 2 behavior rather than a general terrain benchmark.

The package's Spot model exposes the VLP-16 as `Velodyne_Puck`. Its walking controller uses a scripted, open-loop gait that the Webots package's Spot model provides. A conversion plugin publishes the simulated LiDAR path to ROS 2, and the Spot model contains five body-mounted depth range finders. The configuration publishes the five streams as `32FC1` depth images in metres; the surrounding URDF and robot-state-publisher path own their ROS topics and frames. This arrangement lets the same nvblox and explorer interfaces consume body depth data while preserving the LiDAR input to RTAB-Map and the planar scan path, the two-dimensional scan that the near-field map safety layer reads from the rotating LiDAR (Section 6.3.2).

Webots supplies supervisor ground-truth odometry to the ROS 2 graph. This simulation oracle removes pose drift from the integration and timing test; it does not represent an odometry source on the physical robot. RTAB-Map consumes the simulated scan and odometry interfaces, while nvblox produces the volumetric map, mesh, support surface, and planning costmap. RViz exposes the map, mesh, frontier markers, costmap, and goal markers in one view. Chapter 5 describes the mapping representation, reduction to a planning grid, and loop-closure behavior.

Figure 4.2 shows an elevated view of the Spot model, maze partitions, obstacles, and overlaid route. The image documents the simulated integration environment rather than physical-robot validation.

Capture timing provides the main Webots finding. The range-finder and LiDAR plugins refresh an internal sample buffer at a device-specific rate but timestamp messages at publication. During walking, measurements show a mean lag of approximately 56 ms for the LiDAR and 40 ms for one front depth stream, with variation under simulator load and callback scheduling. A capture-time timestamp from the buffer update replaces the publish-time stamp and removes the principal pose mismatch in the stationary-floor diagnostic. Webots supports this test because the supervisor trajectory provides an independent reference; Chapter 5 contains the full projection, timestamp, and cameras-only mapping analysis, in which the nvblox volume is built from the five body depth streams alone (Section 5.2.1).

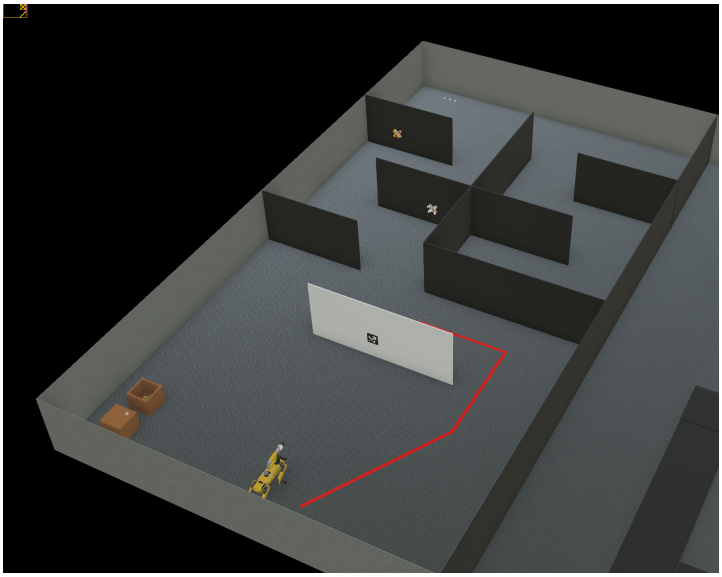


Figure 4.2: Webots maze integration environment.

The ROS 2 work also exposes a limit of the Webots Spot model. In a ramp probe specific to this setup, the scripted gait climbs a 10° incline but rolls laterally at 15°. This observation concerns the selected model, controller, contact parameters, and probe geometry. It does not support a general claim about Webots performance. The result places the terrain locomotion requirement outside the Webots role and motivates the Isaac Sim and Isaac Lab work in the following sections.

Webots remains useful after the Isaac transition. Walkable-goal filtering (Section 6.4), cold-start behavior, the five-camera graph, RTAB-Map integration, nvblox publication, and the maze exploration loop provide integration evidence. Chapter 9 reports the human teleoperation reference and autonomous maze scores; this chapter records why those runs use Webots and what their evidence domain supports.

4.3. Transition to Isaac Sim and the ROS 2 Boundary

In the implemented stack, Isaac Sim hosts the first-party Boston Dynamics Spot asset, USD terrain, PhysX GPU simulation, rendered depth and LiDAR sensors, the Isaac ROS/nvblox path, and the interface to Isaac Lab policy training. These mechanisms connect the existing ROS 2 mapping and planning stack to uneven-terrain tests; they do not make Isaac Sim a replacement for the physical robot.

The stack uses Isaac Sim 5.1. Driver version 590.48.01 gives the Blackwell laptop GPU a working renderer. The deployed topology separates Isaac Sim, the ROS 2 mapping and exploration processes in `spot_container`, and a third container for supporting bridge and planner processes. Private shared-memory namespaces isolate the containers. Fast DDS shared memory consequently creates a transport trap, so every participant

on the Isaac path uses the UDP-only profile. The ROS 2 side remains on Humble with `rmw_fastrtps_cpp`, and the bridge environment that links the simulator to the ROS 2 graph matches that distribution and middleware choice (Section 3.3).

The simulator publishes time on `/clock`. A startup gate holds TARE, the surface-coverage goal-selection component that the stack retains as a fallback (Section 6.6.2), and dependent nodes until the clock becomes nonzero, because a zero simulation timestamp causes immediate time and message-age failures. Workload determines the wall-clock rate: recorded early terrain sessions run near 0.26 times real time and the smaller arena near 0.36 times real time. These values characterize the recorded runs rather than simulator-wide performance. Isaac odometry also needs a frame correction: the initial computation treats a local origin as the world pose, whereas the ROS 2 graph requires the absolute USD world transform. Publishing the corrected world transform removes the offset for open-spawn terrain worlds, in which the robot starts away from the world origin.

The setup constructs the Spot sensor configuration explicitly. Five body depth cameras publish `32FC1` images at approximately 320×240 pixels. Isaac provides no ready-made VLP-16 profile, so a conversion maps the measured JavaScript Object Notation (JSON) beam layout into a USD LiDAR asset. The single-return profile publishes 28,800 points per scan. The default dense configuration produces roughly 456,000 points per scan and starves the CPU: bridge and planner processes accumulate high load, and goals abort before the map completes. The reduced profile restores a usable balance between sensor publication and the ROS 2 stack.

The Webots and Isaac implementations share more software than their visual interfaces suggest. The ROS 2 harness already uses the `/Spot/Velodyne_Puck/point_cloud`, `map`, `odom`, and `base_link` conventions. Isaac therefore needs a boundary adaptation for sensors, time, transforms, and transport rather than a new exploration stack. Table 4.1 compares the two integrations and does not rank them beyond their roles in this thesis.

4.4. Benchmark Worlds in Isaac Sim

Each Isaac world serves a separate question. The warehouse scene provides the baseline for sensor and stack bring-up. Ramp World tests height changes, slopes, turning space, and upper-platform access. The large dune field subjects the system to repeated uneven surfaces and long-range frontier behavior. The small walled arena shortens mapper and frontier-seal iterations, and frontier sealing is the termination rule that stops residual frontier voxels above an interior mapped floor or a supported crest from counting as unexplored while preserving the forward frontier ring at the sensor horizon (Section 6.5). Table 4.2 summarizes their geometry and purpose; it does not rank traversal difficulty.

The warehouse provides the initial Isaac baseline for the Spot sensor and stack scripts.

Table 4.1: Webots and Isaac Sim integration choices in this thesis.

Integration aspect	Webots maze path	Isaac Sim terrain path
World and robot	Text-based maze world with the MASKOR Webots Spot model and scripted/open-loop gait	USD worlds with the first-party articulated Spot asset; policy-controlled actuation in the recorded runs
Physics and terrain	Model derived from the Open Dynamics Engine (ODE), used for ROS 2 and maze integration; terrain result depends on the selected gait and world parameters	PhysX GPU simulation with authored ramps, dunes, stairs, and walls; contact and actuator parameters remain simulated values
LiDAR	Built-in VLP-16-style `Velodyne_Puck` path, with a timestamp correction in the conversion plugin	JSON beam description converted to USD; the single-return profile adopted here publishes 28,800 points per scan
Depth streams	Five body range finders configured as ROS 2 `32FC1` streams	Five body depth cameras added to the Spot sensor script, publishing `32FC1` images at 320×240
Odometry and clock	Supervisor ground-truth odometry provides a simulation oracle; the ROS 2 graph uses simulated time	Simulated odometry publishes the corrected absolute USD world transform; dependent nodes require `/clock` gating before startup
DDS and containers	ROS 2 graph connected through the Webots container and the large-message Fast DDS profile	Three-container graph with UDP-only Fast DDS because private shared memory caused cross-container transport failures
Primary question	ROS 2 interfaces, sensor conventions, timestamp behavior, mapping, and maze exploration	Articulated terrain scenes, rendered sensors, terrain-aware pipeline behavior, and connection to the Isaac Lab policy path

Its known scene supports checks of rendered depth, LiDAR publication, transforms, odometry, RViz, and nvblox before custom terrain enters the experiment. It remains a reference environment because this separation distinguishes an error at the ROS 2 boundary from a terrain or locomotion error.

Ramp World contains two levels and three diagnostic ramp surfaces. Ramp A has a 12° deck, Ramp B a 20° deck, and Ramp C a 12° solid-earthwork surface. Ramp C addresses a 2.5D representation failure. The preceding representation reduces an overhang to one height per horizontal column and cannot distinguish the deck from the lower ground. It consequently creates false connections and false obstacles. Solid earthwork makes the intended traversable surface explicit for the terrain-aware pipeline.

The ramp world blends the foot of each slope and the crest into the adjacent flat surface. The joins use approximately 1.0 m foot and crest blends, with a 0.30 m landing overlap. A wider turning bay at the Ramp C foot permits the planned approach, and a 15-step staircase provides a separate height-change structure. The world uses metre-scale Z-up USD geometry with static colliders. Collider-count and stable-canary-traversal checks, where a canary traversal is a short precondition traversal that must pass before a full run, establish world construction and pipeline preconditions rather than physical Spot acceptance.

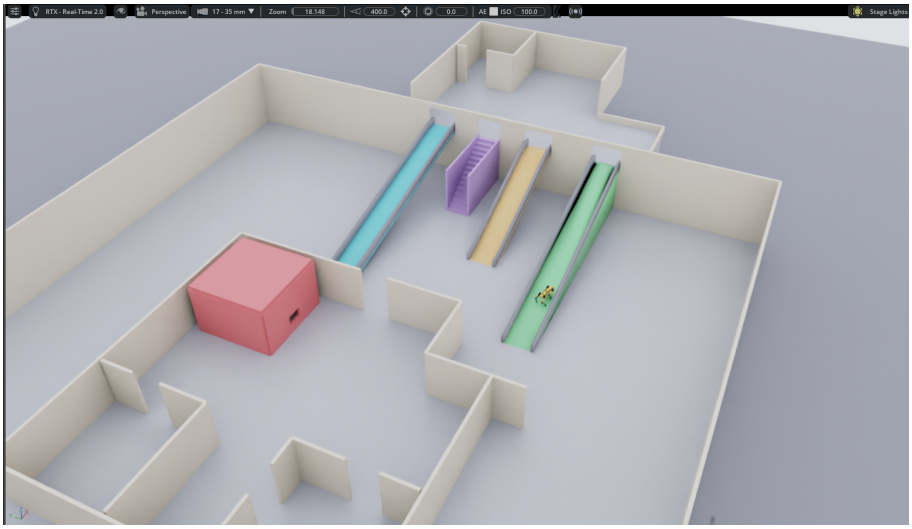
The large dune world is a deterministic 24 m by 24 m usable field inside a larger slab. A 0.40 m high, 0.20 m thick boundary curb encloses the field on all four sides, and it forms the only vertical obstacle in the world. The field contains 133 raised-cosine mounds that a soft-max height function merges, five flatter lanes, and a spawn plaza. The construction uses seed 20260724 and one continuous triangle mesh rather than separate mound meshes. Measurements give a maximum slope of 15.92° and a seam step of 0.0064 m; raycast checks find no difference between the visual and collision surfaces at sampled vertices. These checks separate genuine terrain interaction from mesh seams or collision mismatch.

The small arena is 14 m by 10 m and uses seed 20260863. Seven connected dunes form one compact massif. A flat 4 m by 10 m spawn pad gives the policy a controlled starting area, and walls on all four sides are 0.40 m high and 0.20 m thick. The configuration retains this boundary height because depth and occupancy checks register it consistently. The arena shortens mapper and frontier-seal iterations without implying that a smaller world forms an easier version of the large field. Its shorter routes reduce the time needed to observe an interface change and make controlled before-and-after runs practical.

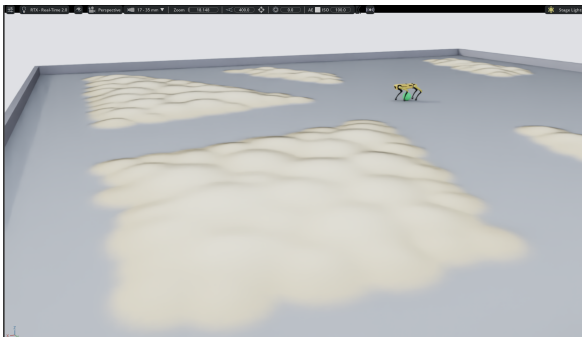
The maze, ramps, dunes, and walled arena answer different questions. The maze isolates ROS 2 integration and exploration in a structured world. Ramps expose height transitions and upper-platform interfaces. Dunes exercise continuous uneven geometry and repeated crest interactions. The small arena reduces elapsed iteration time while retaining connected dunes and visible boundaries. A result in one world is

Table 4.2: Specification and validation purpose of the Isaac Sim benchmark worlds.

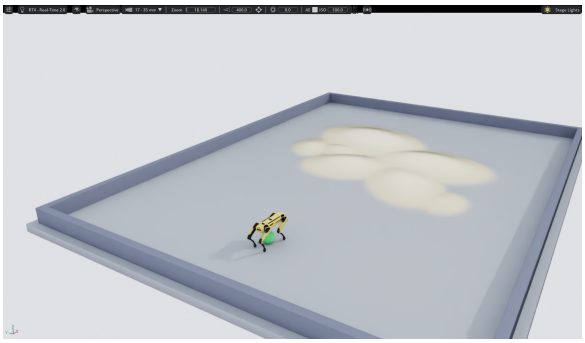
World	Geometry and fixed parameters	Validation purpose	Evidence boundary
Simple Warehouse	Stock warehouse scene used by the Isaac Spot sensor scripts	Sensor, transform, clock, and ROS 2 stack bring-up	Software and sensor-interface behavior in a known scene
Ramp World	Ramp A: 12°; Ramp B: 20°; Ramp C: 12° solid earthwork; 15-step stairs; turning bay; blended foot and crest joins	Terrain-aware goal generation, upper-platform access, 2.5D failure isolation, and transition geometry	World and pipeline checks; no claim of physical Spot climbability
Large dune field	24 m usable field; 0.40 m boundary curb; 133 raised-cosine mounds; five lanes; seed 20260724; one continuous mesh; maximum slope 15.92°; seam step 0.0064 m	Uneven-surface policy deployment, mapping, exploration, and long-route interaction	Declared Isaac Sim model and mesh; run scores belong to Chapter 9
Small walled arena	14 m by 10 m; seven connected dunes; 4 m×10 m spawn pad; four walls, 0.40 m high and 0.20 m thick; seed 20260863	Fast mapper, frontier-seal, and controlled before-and-after iteration	Shorter simulation loop; not a traversal difficulty ranking



Isaac Sim (a) Ramp World



Isaac Sim (b) Large dune field



Isaac Sim (c) Small walled arena

Figure 4.3: Isaac Sim benchmark worlds.

not interpreted as an ordering of the other worlds.

Figure 4.3 compares the three Isaac Sim environments. Ramp World provides ramp approaches and a staircase for height-transition checks, the large dune field supports longer uneven-terrain episodes, and the small walled arena shortens mapper and frontier-seal iterations. These views document simulation infrastructure rather than physical Spot performance.

4.5. Isaac Lab and Uneven-Terrain Locomotion

The Isaac Sim `SpotFlatTerrainPolicy` that the simulator provides lacks terrain-height observations and cannot satisfy the 12° Ramp C diagnostic. A command-only test exposes the failure as a locomotion blocker: the policy cannot enter and hold the ramp even though the terrain and command pipeline remain available. This result characterizes the selected policy and training distribution rather than Isaac Sim or quadrupedal locomotion in general.

The replacement policy uses Isaac Lab v2.3.2 because it provides the GPU-vectorized environment workflow that repeated contact-rich rollouts require. Isaac Gym establishes the GPU-resident physics and learning pattern [Mak+21], and Rudin et al. demonstrate parallel policy training on a single workstation GPU for rough locomotion tasks [Rud+22]. Isaac Lab extends that family with task configuration, sensor and actuator components, terrain generation, and policy-training integration [Mit+25]. These features provide learning throughput; Isaac Lab does not replace the ROS 2 mapping graph.

A custom rough-terrain Spot task trains the selected policy from a warm start with a terrain curriculum and simulated actuator and contact settings. Chapter 8 reports the task configuration, curriculum, reward-gating correction, checkpoint analysis, and policy-speed constraint. These results remain observations under the declared Isaac configuration; they do not calibrate hardware mass, compliance, foot friction, actuator current limits, or a physical gait threshold.

Learned locomotion literature provides context for treating the policy as its own validation layer. Simulation-based motor-skill transfer, terrain locomotion, and perceptive locomotion studies show that observations, actuation, terrain distribution, and transfer procedures form part of the method [Hwa+19; Lee+20; Mik+22b; Yu+22]. This work does not transfer the Isaac Lab policy into the Webots open-loop gait because the actuator and contact models differ. The scope here covers the reason for the policy layer and the boundary of its evidence.

4.6. Hardware-to-Simulation Evidence Boundary

The evidence domains remain separate. Physical mapping and navigation work precede much of the Isaac terrain refinement. The physical Spot supplies evidence for RTAB-Map loop closure, nvblox modality selection, floor-relative mapping, navigation timing, and hardware-specific failure handling. Webots isolates the ROS 2 maze and timestamp behavior. Isaac Sim and Isaac Lab address terrain worlds and rough-terrain policy behavior after the Webots gait probe exposes its limit.

An actuator fault removes access to the physical robot toward the end of the thesis. Hardware work therefore stops, and simulation bounds the final evaluation. The remaining objective evaluates three-dimensional exploration on ramps and dunes with obstacles along the route using the articulated Isaac Spot model, its selected policy, the cameras-only nvblox configuration, and the declared ROS 2 stack.

The boundary in Table 4.3 states what each domain can support. No corresponding experiment means that outcomes specific to this thesis stay within their rows. In particular, simulated mass, actuator, contact, sensor, and timing parameters still govern a simulated Spot model, even when a first-party articulated asset represents the robot.

Under the recorded Isaac Sim version, model, world, sensor configuration, and

Table 4.3: Evidence boundary between physical-robot work and simulation work.

Evidence domain	Work performed	Can support	Cannot support
Physical Spot	Mapping, odometry, sensor-modality comparisons, navigation bring-up, and hardware-specific operation before the robot became unavailable	Evidence about the tested robot, payload, sensor timing, compute load, leases, and physical navigation behavior	Results for an unavailable future robot configuration, or broad claims about all Spot hardware
Webots maze	R2025a ROS 2 maze, MASKOR scripted/open-loop gait, five depth streams, simulated LiDAR, supervisor odometry oracle, RTAB-Map, nvblox, and RViz	ROS 2 graph behavior, simulated sensor conventions, timestamp mechanisms, explorer integration, and repeatability in the recorded maze	Boston Dynamics actuator behavior, real sensor timing, physical gait stability, or hardware safety
Isaac Sim terrain	Isaac Sim 5.1, articulated Spot asset, PhysX GPU, rendered sensors, ramps, dunes, walls, ROS 2 bridge, and nvblox pipeline	Pipeline behavior under declared USD worlds, simulated Spot model, sensor profiles, transport settings, and policy deployment	Real foot placement, actuator limits, contact forces, hardware map quality, or live physical frontier sealing
Isaac Lab policy	Custom rough-terrain Spot task, warm start, terrain curriculum, simulated actuators and contacts, and GPU-vectorized rollouts	Policy behavior under the declared observations, rewards, terrain, randomization, and physics model; same-simulator deployment checks	Transfer to the physical Spot, hardware stability, or ROS 2 exploration validity when the full graph is omitted
Final evaluation state	Simulation-only ramp and dune work after the physical robot became unavailable; frontier-seal live measurement and a live demonstration were not performed	Repeatable software and policy observations in the stated Isaac configuration; run scores reported in Chapter 9	A claim that final exploration was sealed or demonstrated live, or that simulated results certify real-robot performance

ROS 2 boundary, the mapping, planning, frontier, and policy components produce the observed behavior. The results also support comparisons between declared configurations in the same simulator when records preserve the evaluation masks, that is, the declared cell sets over which map-integrity metrics are computed, together with the seeds and stopping conditions (Section 9.1.2). They cannot support claims about physical Spot reliability, real sensor calibration or timestamp latency, actuator transfer, physical safety, or live frontier sealing and demonstration. The work does not complete a live frontier-seal measurement or a live demonstration, so neither counts as completed validation.

The evaluation chapter reports the numerical outcomes and their stopping conditions in Chapter 9. The discussion chapter records the resulting limitations and interpretation in Chapter 10. This separation keeps simulation evidence useful without assigning it a physical meaning that measurements do not support.

5

Odometry and Mapping

This chapter reports the deployed pose-estimation and mapping pipeline, changes that a walking platform requires, and the alternatives that the evaluation considers before final selection. Chapter 2 introduces general SLAM, TSDF, ESDF, and elevation-map principles; the focus here is their implementation and measured behavior. Labels identify results as physical-robot evidence, Webots simulation, Isaac Sim, or offline replay of physical recordings and mark staged and build-only attempts explicitly. Configuration labels separate the deployed production values from the earlier diagnostic values used during the corruption investigation, and distinguish the floor-relative `v2` variant of Section 5.2.3 from the fixed and body-relative bands. Metric definitions and the rules that permit numerical comparison come from Section 9.1.2 and Section 9.1.3; every comparison below names the recordings or drives that it uses.

5.1. Deployed Odometry and Frame Ownership

This section establishes the pose source and frame contract used by every downstream map update. It first defines transform ownership and the deployed RTAB-Map configuration, then presents the closure evidence and explains how the optimized pose graph coexists with the live nvblox volume. This separation distinguishes global pose correction from continuous volumetric integration.

5.1.1. Frame ownership and RTAB-Map configuration

The Boston Dynamics driver supplies leg-kinematic odometry at approximately 24 Hz, and although its output is locally continuous, foot slip, compliance, and integration error accumulate over time. Physical trials record approximately 2 m of translational drift during more than 100 s of walking. Open-loop LiDAR odometry reduces some scan-to-scan errors but provides no global correction after a revisit. The deployed graph SLAM back end therefore uses RTAB-Map [LM19].

Each transform edge has one owner. RTAB-Map owns `map -> spot/odom`; the Boston

Table 5.1: Production RTAB-Map configuration.

Setting	Deployed value and role
Frames	`map`, `spot/odom`, and `spot/body`; the driver owns the local body edge
Input and registration	VLP-16 cloud; `Reg/Strategy=1`; point-to-plane ICP with `Icp/Strategy=1` (libpointmatcher); `RGBD/NeighborLinkRefining=true` and `RGBD/CreateOccupancyGrid=false`
ICP parameters	0.10 m voxel grid, 20-neighbor normals, 10 iterations, `Epsilon=0.001`, `MaxTranslation=2.0`, `MaxCorrespondenceDistance=0.5`, `OutlierRatio=0.7`, and `CorrespondenceRatio=0.2`
Keyframe cadence	1 s window, node updates at 0.1 m or 0.1 rad, and about 0.06 s processing in the physical deployment
Graph optimization	GTSAM strategy 2 and optimize-max-error 3.0
Closure search	Geometric proximity by space, with at most 10 path neighbors
Transform publication	Driver odometry about 24 Hz; latest global correction refreshed at 50 Hz

Dynamics driver owns `spot/odom -> spot/body`; and static transforms define the LiDAR and camera links below `spot/body`. Composition of the optimized RTAB-Map body pose with the inverse of the simultaneous driver body pose computes the `map -> spot/odom` correction. Driver odometry thus remains continuous for local control while graph optimization can change the parent map frame. Single-parent ownership also keeps tf2 lookups deterministic across mapping and navigation consumers [BDA24; Bos24].

RTAB-Map receives the VLP-16 cloud and driver odometry in one SLAM node. Enabling `RGBD/NeighborLinkRefining` refines sequential links with point-to-plane ICP on a 0.10 m voxel grid and surface normals from 20 neighbors. A 1 s window processes keyframes; physical deployment needs about 0.06 s per window. A transform freshener republishes the latest valid `map -> spot/odom` correction at 50 Hz, improving transform availability without creating pose estimates.

The graph contains sequential neighbor links, visual appearance closures, and geometric proximity closures. `RGBD/ProximityBySpace` enables geometric closure and considers at most 10 path neighbors for a candidate. `Mem/STMSize=30` limits short-term memory. `Optimizer/Strategy=2` selects the Georgia Tech Smoothing and Mapping (GTSAM) optimizer, and `RGBD/OptimizeMaxError=3.0` rejects constraints that conflict with the graph. Table 5.1 summarizes the deployed parameters.

The ownership contract and the two map paths are summarized in Figure 5.1. RTAB-Map corrects the global parent frame, while nvblox continues to integrate a separate

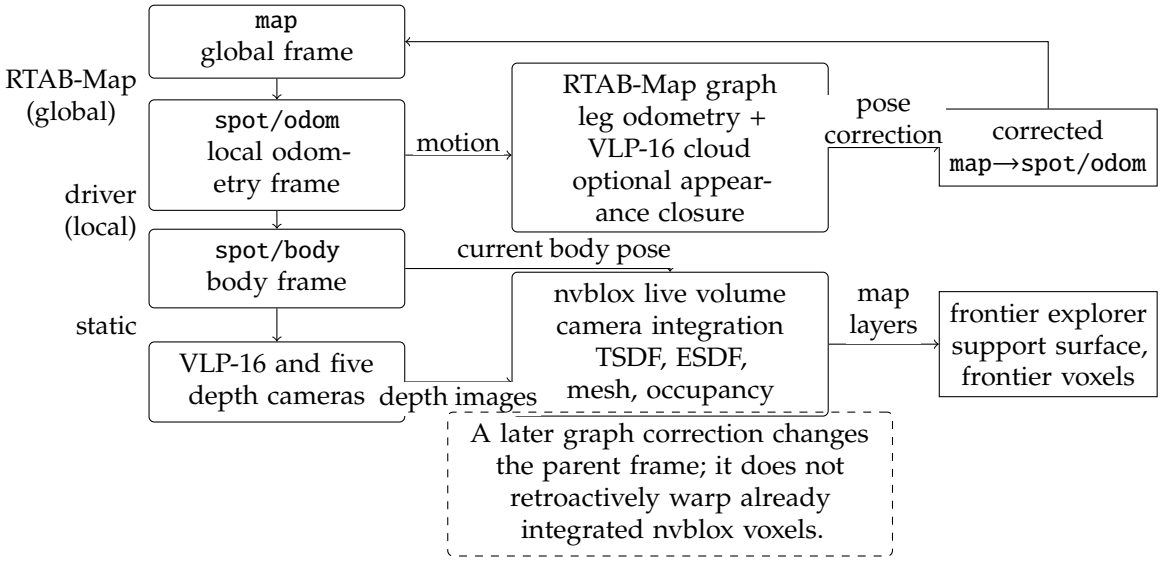


Figure 5.1: Frame ownership and coexistence of corrected odometry with the live nvblox volume. The left column is the transform chain and the owner of each edge; the upper path is the RTAB-Map pose correction, and the lower path is the live volume that the frontier layer reads.

live volume from the current body pose. The arrangement preserves local continuity for control but does not retroactively deform voxels that were integrated before a graph correction.

5.1.2. Closure evidence and map coexistence

The live physical deployment uses LiDAR-only SLAM. Its record contains 35 `LocalSpace Closure` links (geometric proximity closures) and 0 visual `GlobalClosure` links (appearance closures), together with a final 68 MB RTAB-Map database. These counts remain distinct from the later appearance experiment. In an offline replay of a physical multi-sensor recording that covers 46.5 m over 132 s, the 102-node graph contains 4 visual closures and 31 geometric local closures and stores 73,850 visual features, about 724 per node. Approximate-time colour-and-depth (RGB-D) synchronization handles a camera-clock offset of about 67 ms. The result shows operational appearance closure offline; it does not show active appearance closure in the live LiDAR-only deployment. The DBoW2 bag-of-binary-words library provides the visual place-recognition stage when this mode runs [GT12; LM19].

RTAB-Map and nvblox apply pose corrections differently. RTAB-Map stores scans at graph nodes, updates the node poses, and re-assembles the global cloud from rigid keyframe scans. It does not non-rigidly deform those scans. nvblox integrates measurements into a spatially hashed voxel volume and has no historical pose attached to each voxel. It therefore cannot retrospectively re-pose already integrated geometry after RTAB-Map changes the parent transform [Mil+24].

The deployed arrangement uses RTAB-Map as the corrected global pose source and nvblox as a continuously updated local volume. The design rejects replacement of

Table 5.2: Production nvblox and costmap parameters.

Parameter	Working value
Voxel and block size	0.05 m; $8 \times 8 \times 8$ voxels per block
Truncation	0.20 m production; 0.15 m diagnostic
Camera integration	0.30 m to 3.00 m, weight 1.0, accumulated cap 100
GPU target	Blackwell `sm_120`, 12 GB video memory
Update cycles	Camera 1.0 Hz to 1.5 Hz, mesh 1 Hz, costmap 0.5 Hz
Grid and obstacle threshold	0.05 m cells; threshold reduced from 0.15 m to 0.10 m

nvblox with RTAB-Map's planar occupancy grid because such a replacement removes the three-dimensional distance and unknown-space representation that downstream layers need. A pose graph of rigid TSDF submaps, as Voxgraph uses, would address large corrections but would require a different volume-management architecture [Rei+20].

5.2. Mapping Configuration and Custom Changes

This section describes how the dense map is built and how it is reduced to the grid that the planner uses. It first fixes which sensor feeds which part of the stack, then traces the investigation that led to that split, and finally describes the floor-relative reduction with the crest correction that Isaac Sim evaluates.

5.2.1. nvblox configuration and sensor responsibilities

nvblox provides the sparse TSDF, ESDF, mesh, and occupancy layers for this work [Mil+24]. The source build targets the NVIDIA GeForce RTX 5070 Ti Laptop GPU with 12 GB of video memory and Blackwell `sm_120`. The deployed volume uses 0.05 m voxels. Each block contains $8 \times 8 \times 8$ voxels and spans 0.40 m per side. The production truncation distance is 0.20 m; the earlier diagnostic configuration uses 0.15 m. Camera samples between 0.30 m and 3.00 m enter with weight 1.0, and the mapper caps accumulated voxel weight at 100.

Table 5.2 lists the working mapper and costmap values.

The final sensor split follows the measured sensor geometry. The five downward-facing depth cameras build the near-field nvblox volume. The VLP-16 remains available to RTAB-Map for neighbor refinement and proximity closure, to the planar safety scan, and to long-range obstacle sensing. Connecting the mapper to the dormant `/nolidar_disabled` cloud topic excludes it from dense floor fusion, so scanner returns never enter the volume while the driver continues to publish them for the consumers named above. An optional structural-LiDAR mode retains only points above `spot/body` height minus 0.30 m and thereby removes floor-grazing returns; the production camera-only mode disables this path and admits depth frames only.

Four protections surround volume integration and grid publication. A yaw gate suppresses camera frames above 0.25 rad s^{-1}. Physical spin measurements show 68 floating voxels, stray occupied voxels above the floor, without the gate and zero with it. A translation gate requires 0.05 m of horizontal motion before another integration, with a 10 s backstop that eventually admits a stationary view. The optional structural-LiDAR lower cut described above removes floor-grazing returns, and connected-component filtering removes isolated derived components of at most 6 voxels in three dimensions or 6 cells in the final two-dimensional grid.

5.2.2. Mapping-corruption investigation

The investigation separates sensor timing, spherical projection, and projective integration. In Webots, the estimated trajectory first matches supervisor ground truth to 2.7 mm horizontal root-mean-square (RMS) and 0.155° attitude RMS over 90 s. Simulated odometry drift therefore does not explain the remaining map damage.

Capture-to-publish latency varies between −32 and +160 ms. Assigning the timestamp when a sensor buffer changes reduces raw floor motion to 0.97 mm peak to peak for LiDAR and 0.88 mm for depth images. This correction provides a mapping result because it removes one source of pose mismatch, but Chapter 4 covers the Webots capture and publish implementation. Live TSDF corruption remains after the timestamp correction, which separates it from the later integration failure.

The VLP-16 emits 3,600 azimuth samples per revolution while the initial spherical projection uses 1,800 columns. Increasing the projection to 3,600 columns removes 92.7 % of the offline Webots phantom vertices, mesh vertices that correspond to no observed surface, from 20,194 to 1,464, and reduces simulated wall thickness from 40.97 cm to 0.83 cm. Moving-floor corruption, floor damage that appears during motion and is absent from the stationary warm-up, still remains: the corrected fused Webots drive reaches only a 30.7 % walkable-cell ratio, the share of mapped support columns that also pass the slope, step, and roughness gates (Section 9.1.2).

Far, grazing LiDAR rays provide the final cause. A 0.5° attitude change displaces a floor return by about 8.7 cm at 10 m, whereas the same perturbation gives about 1.3 cm for a downward camera observation at 1.5 m. The long ray can carve or add floor structure across several 0.05 m voxels. This range-dependent effect remains after corrections to timing and angular discretization.

The cameras-only Webots comparison uses the same 180 s drive for each sensor configuration, so the three configurations differ only in the sensor split. The defect counts cover reconstructed geometry below the floor surface inside the declared evaluation bands (Section 9.1.2). Cameras-only mapping changes from a stationary 97.3 % walkable ratio to 94.5 % after motion and produces zero sub-floor crossings and zero sub-floor mesh vertices in the evaluated bands. LiDAR-only mapping changes from 27.5 % to 51.1 % and ends with 1,118 sub-floor crossings and 2,146 sub-floor mesh

vertices. The fused configuration falls from 79.2 % to 30.7 %.

The physical A/B trial uses identical walking loops for the two sensor splits and evaluates a 2.5 m-radius disk around the trajectory. Disk coverage is the share of that disk for which the mesh supplies a floor surface, and a hole is a connected region of the disk that the mesh leaves uncovered. Cameras-only mapping covers 99.6 % of the disk with 3 holes and a maximum hole area of 4 cells. Fused LiDAR and camera mapping covers 88.9 %, with 46 holes, a maximum hole area of 52 cells, and a visibly doubled floor. These physical results and the Webots comparison establish the production sensor split.

The Webots runs use the same maze world as Figure 4.2; after the 180 s drive, each map covers only the start room around the drive route, which is why the maps look smaller than the full maze. Chapter 9 reports these runs as results and shows the matched start and after-drive meshes of the three configurations in Figure 9.8.

5.2.3. Floor-relative reduction and the Isaac Sim crest correction

The evaluation considers three vertical bands for reducing the three-dimensional field to a two-dimensional planning grid. The fixed band uses the map-frame interval from 0.30 m to 0.80 m and fails when the map datum or terrain height changes; in one physical replay, the map origin places the body at map $z = 2.72$ m while the floor remains at zero. The body-relative band moves with `spot/body`. Its fallback offsets are −0.38 m to 0.12 m, but approximately 0.10 m of gait-height change forces a choice between slicing into the floor and missing low obstacles. The latter choice produces an effective blind region of about 0.38 m.

Floor-relative reduction estimates a support height in each horizontal column, the local floor surface $H(x, y)$ defined in Section 7.6.1, and then checks the region from 0.30 m to 0.80 m above that local floor; a column counts as occupied when any observed voxel in that region has an obstacle distance at or below the obstacle threshold. The deployed variant sets `cm_floor_relative=true` (`NVBLOX_VARIANT=v2`). The implementation computes the support grid at 0.10 m resolution and resamples it by nearest neighbor to the 0.05 m costmap. A column without a valid floor estimate uses the body-relative offsets of −0.38 m and 0.12 m. A 3×3 median filter suppresses isolated support-height errors. Reducing the obstacle threshold from 0.15 m to 0.10 m lowers raw speckle, isolated occupied cells, by 27 %, and an 8-connected filter removes occupied components of at most 6 cells. In physical evidence, a 0.35 m box at 1.0 m to 1.5 m in front of Spot produces an occupied component of about 49 cells.

Only Isaac Sim evaluates the remaining crest defect and its correction. Downward cameras observe the terrain surface but sometimes leave the air directly above a convex crest unknown. The support test of Section 7.6.1 then rejects that column, and the fallback body-relative band may classify the crest itself as an obstacle. Known-Floor-Implies-Free changes this ordering: an observed obstacle remains occupied; otherwise,

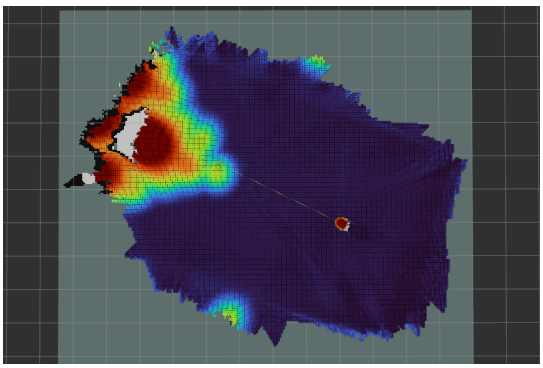
the mapper may mark a column with known supporting floor as free; all other unobserved columns remain unknown. Obstacle evidence always has precedence.

The crest guard propagates support height through a 3×3 neighborhood only where the local elevation span is at most 0.15 m. The gate prevents a support height from crossing a structural step. In the Isaac Sim curb control, ungated propagation incorrectly frees 140 of 360 curb cells, while the gated version preserves 360 of 360. On the trajectory-independent common mask of 21,534 cells, including 1,001 crest cells, the deployed camera-only configuration marks 100 % of crest cells free with no crest cells occupied or unknown and no flat-ground false occupied cells, meaning no cells marked occupied where the authored ground is flat. A separate 46,516-cell corridor mask reports 75.6 % crest-free because simulated locomotion stops after 15.4 m and leaves distant terrain unvisited. The corridor result therefore measures both mapper output and trajectory coverage; it does not contradict the mapper score.

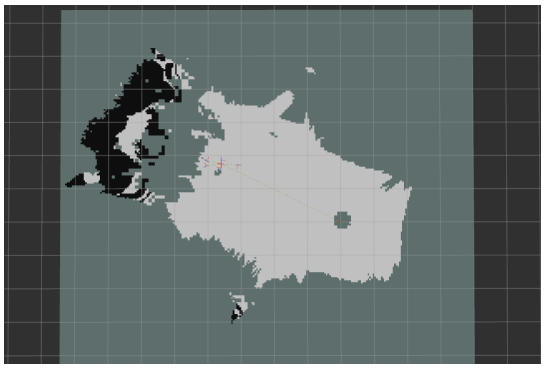
Figure 5.2 shows the effect on one crest pair in RViz. The views come from the large dune field of Figure 4.3(b); they show only the region mapped along the probe route near the spawn, not the whole world, so the field appears as a flat area with two dune massifs at its edge. The mesh views on the left colour the reconstructed surface by height and are identical in both runs, which confirms that the terrain did not change. The grid views on the right show the planning decision: before the correction, the crest columns remain unknown or occupied, and after it, they read free. The comparison is therefore about the mapper's grid decision, not a change to the simulated terrain.

The curb control, which uses the 0.40 m boundary curb of the same world, shows why the step gate remains necessary: without the gate, support height from the narrow curb top spreads into neighbouring columns and frees part of a real obstacle. Figure 9.3(c) and (d) show this control as rendered planning grids. The control supports the obstacle-precedence claim, but it remains Isaac Sim evidence rather than physical validation.

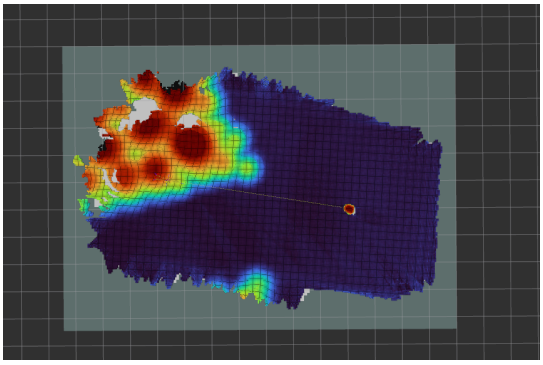
The LiDAR range comparison also provides Isaac Sim evidence rather than physical validation, and it scores both sensor configurations against the authored terrain of the same world (Section 9.2.1). Support-height RMS is the root-mean-square difference between the reconstructed support surface and that authored terrain. Below 1.5 m, LiDAR support-height RMS is 0.062 m, close to the camera value of 0.059 m. At ranges of at least 6 m, LiDAR RMS rises to 0.394 m, and flat-ground false occupied cells increase from 55 to 278. A 5 m range cap does not remove the effect. The result agrees with the cameras-only mapping choice but does not imply unchanged transfer of LiDAR motion smear from Webots to Isaac Sim. Frontier sealing remains a separate exploration-layer change in Chapter 6.



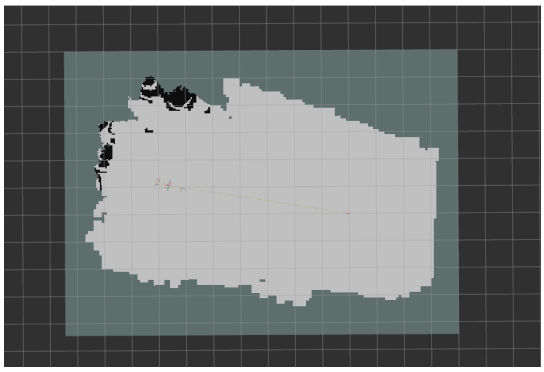
(a) Mesh before the correction



(b) Grid before the correction



(c) Mesh after the correction



(d) Grid after the correction

Figure 5.2: RViz views of one crest pair in the large dune field before and after the known-floor rule with the step-gated crest guard; the views show the probe route near the spawn, region A of the top-down reference in Figure 9.1(a). Mesh colour encodes surface height; in the grid views, light grey is free, black is occupied, and dark grey is unknown. The scored results and the curb control appear in Figure 9.3.

5.3. Attempted Methods and Comparison

The adopted configuration emerged from a sequence of tested alternatives. This section records them in two tables, first for odometry and localization and then for mapping representations, so that each rejection or replacement can be traced to the evidence that caused it.

5.3.1. Odometry and SLAM alternatives

Table 5.3 records every odometry or SLAM approach that this work evaluates, and the status column names the action taken on each approach, from retained through replaced, superseded, and staged to blocked and failed. The offline comparison uses physical VLP-16 recordings: a 550° in-place rotation, a 95 m single-floor traversal, and a 236 m multi-level traversal. Wall RMS measures the root-mean-square spread of reconstructed wall points about their wall plane, so a lower value means a thinner wall, and a value is compared only against methods run on the same recording; the `lab_only` entry reports a value that no matched recording supports, so the table does not rank it. A slip frame is a frame during fast in-place yaw in which foot slip degrades the leg-kinematic estimate, and the slip-frame rate gives the share of such frames. The results compare implementations on sparse 16-beam indoor quadruped data and do not establish an ordering for denser LiDARs, outdoor scenes, or other compute budgets. Mean Map Entropy (MME) serves only as a local surface-dispersion score; a lower value does not by itself establish trajectory accuracy.

The early `slam_toolbox` and assembler pipeline corrects planar position and yaw but leaves body roll, pitch, and vertical motion uncorrected [MJ21]; the assembler component is the step that concatenates the registered scans into a single three-dimensional cloud. A 1° pitch change displaces points by about 8.7 cm at 5 m; gait-induced pitch and roll reach roughly 1° to 2°. Accumulation in `spot/odom` moves the complete cloud when the parent correction changes, while accumulation directly in `map` retains duplicated walls. OctoMap bounds storage, but ray updates still thicken surfaces during gait motion, and the 0.025 m filtering path cannot keep pace on the available CPU [Hor+13].

Raw KISS-ICP loses yaw during in-place rotation because the 16-beam floor and ceiling rings provide weak rotational information [Viz+23]. Leg-assisted initialization reduces that failure. Retaining planar points and setting the initial correspondence threshold to 0.3 m reduces the largest observed pitch error from 35.7° to 6.6°. Kinematic twist deskew reduces high-yaw-rate slip frames from 69 % to 17 %. A later offline pose graph assigns yaw to the kinematic estimate, roll, pitch, and height to generalized ICP (GICP), and horizontal translation to GICP only while driving. On the controlled rotation recording, this degree-of-freedom split reduces wall RMS from 21.9 cm for leg odometry and 17.7 cm for rigid yaw locking to 9.3 cm. It does not enter the live deployment.

Table 5.3: Odometry and SLAM methods evaluated in this work.

Approach and status	Inputs or configuration	Evidence scope	Outcome
Spot leg odometry; retained	Joint and body-state kinematics	Physical	Continuous local prior; approximately 2 m drift after more than 100 s; 21.9 cm wall RMS on the rotation comparison.
`slam_toolbox` plus assembler; tested, replaced	Planar scan matching plus accumulated 3D cloud	Physical	Uncorrected roll, pitch, and height produced layered surfaces; historical points could not be re-posed consistently.
Raw KISS-ICP; tested, failed	VLP-16 point-to-point ICP	Physical	Rotational degeneracy during in-place spins produced duplicated room orientations.
Leg-assisted KISS-ICP; tested, superseded	Kinematic prior and twist deskew	Physical and offline	Slip-frame rate fell from 69 % to 17 %; wall RMS was 9.9 cm on the rotation recording and 19.7 cm on the single-floor recording.
Vanilla GenZ-ICP; offline-only, not adopted	VLP-16 registration without leg prior	Offline physical recordings	7.8 cm wall RMS on the rotation recording; diverged to 21.2 cm on the multi-level recording.
Leg-assisted GenZ-ICP; tested and staged, not selected	GenZ-ICP with kinematic initialization; live-selectable as `odom_source:=genz`	Offline physical recordings; live source staged	MME −2.625 on the rotation recording and the only coherent multi-level reconstruction in that comparison, with 11.9 cm wall RMS. Open-loop use still lacked global closure [LLH25].
KISS-SLAM; offline-only, not selected	KISS-ICP with bird's-eye-view ORB closure	Offline physical recordings	6.2 cm rotation and 9.4 cm single-floor wall RMS; six false cross-floor closures produced eight wall segments in the multi-level recording.

Approach and status	Inputs or configuration	Evidence scope	Outcome
PIN-SLAM; offline-only, staged	Point-based implicit representation on CUDA 12.8	Offline physical recordings	6.1 cm rotation wall RMS, but processing required several seconds per scan.
MOLA-LO; offline-only, not selected	LiDAR odometry package	Offline physical recording (`lab_only`)	The 7.0 cm wall RMS belongs to the `lab_only` recording and lacks a consistent dataset comparison, so it is not used for ranking.
Degree-of-freedom-split graph; offline-only, superseded	Leg sequential edges and split GICP closure edges	Offline physical recording	Removed the doubled-room profile and reached 9.3 cm wall RMS in the controlled rotation trial; never entered the live control loop.
FAST-LIO2; build and licence blocked	VLP-16 plus internal high-rate IMU	Compiled and configured only	Physical execution required the 333 Hz internal IMU stream behind an unavailable commercial joint-control licence. This was an access constraint, not an algorithmic failure.
Point-LIO with BNO055; offline-only, not selected	VLP-16 plus external Universal Serial Bus (USB) IMU	Offline physical data (`spin_imu`)	The evaluation obtains no Point-LIO measurement. On the `spin_imu` recording, external-gyro deskew in the same sensor path changes GenZ-ICP wall RMS from 9.63 cm to 13.19 cm; the record does not characterize USB timing jitter.
GraphNav; staged, not adopted	Boston Dynamics graph localization	Physical integration staged	Available through the software development kit (SDK), but proprietary and tied to pre-recorded maps; no measurement in this work supports a quantitative comparison [Bos24].

Approach and status	Inputs or configuration	Evidence scope	Outcome
cuVSLAM stereo; tested, failed	Front fisheyes passed to a pinhole stereo interface	Physical	The unrectified cameras had about 62° relative roll and produced zero landmarks, so tracking did not initialize.
cuVSLAM RGB-D; tested, operational but not selected	Single depth camera, scale divisor 999, capture timestamps	Physical and public-data check	The public-data check completed 800 frames over 8.29 m with no track losses. After the software fixes, a 14.9 m physical walk had no tracking loss but accumulated about 2 m drift over 15 m; observed packet rate was about 2–3 Hz.
RTAB-Map; tested and adopted	Leg prior, VLP-16 neighbor refinement, graph closures; production `odom_source:= rtabmap`	Physical, offline replay, and Webots simulation	Live: 35 `LocalSpaceClosure` and 0 `GlobalClosure`; appearance replay: 4+31 closures; extended LiDAR-only replay: 322 nodes and 47 local closures; Webots ground-truth-odometry post-processing: 9 closures, 25–73 ms per 1.0 s cycle, and 551 MB resident set size.

The cuVSLAM stereo result reflects a camera-model failure rather than GPU incompatibility. Two implementation corrections enable the RGB-D path on Blackwell: Spot depth values require a divisor of 999, and sensor acquisition timestamps replace host-arrival timestamps. Runtime compilation converts embedded `sm_90` Parallel Thread Execution code for `sm_120`. Tracking then remains active, but metric drift prevents use as the primary pose source. FAST-LIO2 and Point-LIO address motion with inertial sensing, but the former lacks access to the internal IMU and the latter depends on an external USB IMU whose timing degrades the tested registration. These constraints leave RTAB-Map as the deployed graph-based correction layer.

5.3.2. Mapping alternatives and final selection

The mapping comparison in Table 5.4 separates the configurations that this work evaluates from literature references, and each status label names the action taken on that configuration, from adopted and retained through superseded, rejected, and abandoned to literature only. Its figures are comparable only within one environment:

the Webots rows use the 180 s drive of Section 5.2.2, and the physical rows use the walking loops of the A/B trial. In particular, Voxblox provides published context for incremental TSDF-to-ESDF mapping, but this work does not run it [Ole+17]. The Elevation Mapping CuPy (EMC) path provides a separate planner-side experiment for terrain-aware goal processing related to published elevation-mapping work [FBH18]. It does not replace the nvblox global mapping input of the adopted exploration stack.

Table 5.4: Mapping configurations and representations considered in this work.

Configuration and status	Map product	Evidence scope	Outcome
Planar SLAM plus cloud accumulation; tested, replaced	2D occupancy plus appended 3D points	Physical	Planar corrections did not resolve body attitude or height, and the cloud retained ghost surfaces after corrections.
OctoMap; tested, abandoned	Probabilistic occupancy octree	Physical	Gait-dependent rays thickened surfaces; 0.025 m filtering lagged the available CPU [Hor+13].
Prebuilt Isaac ROS nvblox; attempted, abandoned	Packaged volumetric mapper	Physical integration attempt	The Humble integration does not proceed. Available records establish no specific technical cause, so this work assigns none.
Fused LiDAR and camera nvblox; tested, rejected	TSDF, mesh, and reduced costmap	Webots and physical	Webots walkable ratio fell from 79.2 % to 30.7 %; physical disk coverage was 88.9 % with 46 holes and doubled floor structure.
LiDAR-only nvblox; tested, rejected	LiDAR projective TSDF	Webots	Walkable ratio changed from 27.5 % to 51.1 %, with 1,118 sub-floor crossings and 2,146 sub-floor vertices after motion.
Cameras-only nvblox; tested, adopted	Near-field TSDF, ESDF, mesh, and grid	Webots and physical	Webots ended at 94.5 % walkable with no phantom vertices in the evaluated bands; physical disk coverage reached 99.6 % with 3 small holes.

Configuration and status	Map product	Evidence scope	Outcome
Fixed global costmap band; tested, superseded	Constant-height 2D reduction	Physical	Map-height offsets and changing terrain caused the band to intersect floor or miss obstacles.
Body-relative costmap band; tested, retained as fallback	Chassis-relative 2D reduction	Physical and simulation	Offsets of −0.38 m to 0.12 m tolerated map datum changes but produced a gait-floor versus low-obstacle trade-off.
Floor-relative v2; tested, adopted	Support-relative 0.30 m to 0.80 m obstacle band	Physical and Isaac Sim	Physical box detection and speckle filtering validated the reducer; the later crest extension has Isaac-Sim-only evidence.
EMC elevation map; separate planner experiment	Single-height terrain grid with camera densification	Offline and planner-side trials	Used for terrain-aware planning experiments; it was not the adopted global mapper or source of nvblox frontiers.
Voxblox; literature only	Incremental TSDF and ESDF	Published literature only	Reviewed as a related representation; not implemented or tested in this work [Ole+17].

The final selection follows the resulting interface constraints. Leg odometry supplies local continuity. RTAB-Map supplies global correction. The VLP-16 supplies scan refinement, geometric proximity closures, and long-range obstacle sensing. The five depth cameras supply the local nvblox volume, and floor-relative reduction converts that volume into the planning grid. Each component serves the measurement or state representation that the evidence in this work supports.

6

Frontier-Based 3D Exploration

This chapter reports the exploration layer that selects where the robot observes next. Chapter 2 introduces frontier methods and volumetric representations. Labels identify results as physical-robot evidence, Webots simulation, Isaac Sim, offline replay, or dependency-light synthetic tests; no comparison combines measurements from different environments directly.

Exploration uses the existing volumetric map, and every goal must identify a point that the legged platform can occupy. The GPU mapper therefore publishes frontier voxels as an x-y-z-intensity (XYZI) `PointCloud2`, with per-voxel gain in intensity. A separate ROS node clusters, scores, gates, commits, and publishes one `geometry_msgs/PointStamped` on `/way_point`. This boundary keeps the goal executor replaceable: the navigation backend plans a route to that waypoint and tracks it with a follower that emits body-frame velocity commands. The mapper already classifies voxels in three dimensions for its costmap slice [Mil+24]; frontier extraction reuses that tensor instead of maintaining a second map [Cao+21; Dan+20].

6.1. Frontier Detection in the TSDF–ESDF Volume

A dedicated 0.10 m grid samples the ESDF rather than the 0.05 m map voxel, over the mapped horizontal bounding box and only across the navigable vertical band. A physical-robot limit forces the coarser resolution: the first live extraction queries a $305 \times 337 \times 97$ grid, exceeds the eight-million-cell query cap, and skips every cycle, whereas the coarse clipped grid queries roughly 310 000 cells. The mapper emits a voxel when observations mark it free, its ESDF distance reaches at least the 0.4 m standoff, and at least one neighbor remains unknown, so candidates carry clearance by construction.

Neighbor counting considers only the horizontal plane, which bounds per-voxel gain to one through four. Physical evidence also motivates this rule: the early implementation

repeatedly publishes waypoints on top of the robot because the narrow vertical field of view leaves a permanently unobserved cone that travels with the body. Free voxels at body height therefore border unknown space that no motion can clear and score maximum gain indefinitely.

The vertical band combines an always-on body-relative interval, from 0.5 m below to 0.8 m above the body through a transform lookup, with a per-voxel floor-relative term that admits a voxel when an observed surface lies 0.2 m to 1.2 m directly below it. Both terms matter because the floor-relative term alone requires an observed floor, which the LiDAR cannot see within roughly 2.5 m. In offline replay, the robot-anchored height gate keeps 7757 of 8416 voxels with the body at map $z = 2.71$ m.

That datum establishes the transferable rule: the robot transform anchors height decisions, never an absolute map height, because the map origin is an artifact of the odometry back end, as the 2.72 m body height in Chapter 5 shows. Two interface properties enable every later frontier diagnosis. The mapper publishes an empty cloud when no frontiers exist, so the selection node can distinguish no frontiers from no data, and a throttled funnel line reports the resolved band and the count that survives each stage.

6.2. Goal Scoring and Selection Strategies

A dependency-free 26-neighbor breadth-first connected-component search groups frontier voxels on a coarse tolerance grid. The search uses the horizontal projection of the voxel set and sums per-voxel gain over each column. The implementation then tiles clusters at a 1.5 m extent because, without tiling, the maze boundary forms a single connected component on every Webots replan and collapses to one central target. FUEL splits large clusters recursively along principal components; fixed-extent tiling provides a cheaper form of that idea [Zho+21].

The pipeline first clusters, then gates each cluster, and finally projects inward when a cluster fails entirely. An earlier Webots implementation gates frontier voxels before clustering, so it discards boundary voxels before any cluster exists and starves on the boundary it should pursue. The adopted rule moves invalid boundary goals inward instead of discarding them: when no cluster member passes the gates, the selection node projects the cluster onto the nearest fully valid support column within 1.5 m that lies beyond the reached radius and inherits its utility. GBPlanner2 records the same separation of informative from commandable candidates [Kul+22].

Figure 6.1 summarizes the implemented path from volumetric evidence to a commandable goal, from the ESDF query through clustering and the walkability gates to the published waypoint. The process boundary matters: the mapper reports informative free–unknown boundaries, while the explorer decides whether any corresponding ground pose is admissible.

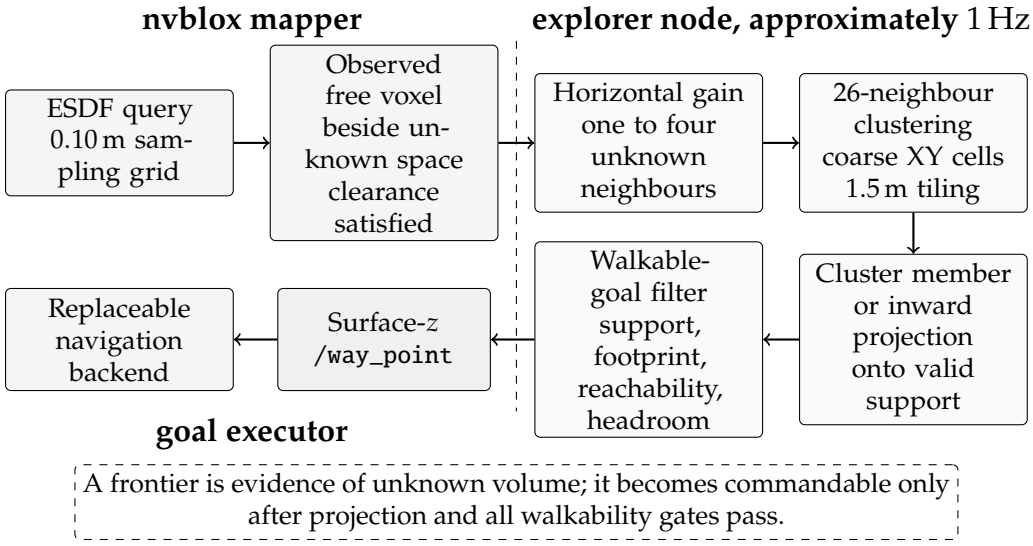


Figure 6.1: Frontier-to-waypoint pipeline and process boundaries.

6.2.1. Nearest, Count-Gain and Info-Gain Strategies

Three selectable strategies share that candidate set. The `nearest` strategy takes the closest cluster and reproduces the classical closest-frontier rule [Yam97; Hol+10; LLA21]. The default, `count_gain`, discounts summed unknown-neighbor gain exponentially by distance,

$$U = g \exp\left(-\frac{d}{\lambda}\right), \tag{6.1}$$

with g the cluster's summed per-voxel gain, d the distance to the candidate and λ a decay length; every scored run uses this strategy. The `info_gain` strategy sums range-decayed per-voxel gain and provides a cheap proxy rather than a posterior information measure from a belief and sensor model [Tab+16].

A dependency-light synthetic test of (6.1) fixes the intended behavior before robot involvement: with a near cluster of gain 8 at 2 m and a far cluster of gain 48 at 7 m under a 4 m decay, `count_gain` selects the far cluster, $48e^{-7/4} \approx 8.3$ against $8e^{-1/2} \approx 4.9$, while `nearest` selects the near one. A short decay recovers the nearest rule, so λ alone sets that balance; tuning flattens it from an effective 4.0 m to 12.0 m in the maze and raises it to 20 m in the Isaac Sim dune configuration.

6.2.2. Gain Shaping and Repetition Penalties

The final design contains two multiplicative modifiers. The first downweights air frontiers by a factor of 0.15. The presence of a valid support surface strictly below the voxel classifies each column; columns without one count as air, and columns with valid ground keep full weight, so ramps and slopes retain full utility without a global height plane. Raw unweighted gain remains the cluster-retention criterion, so weighting can never delete a candidate. Physical evidence motivates the rule: after exploratory rotation, voxels appear above the body where the cameras cannot see, and 1579 of 1620

remaining frontiers, about 97 %, receive the air classification and stall progress.

The second modifier penalizes already-traversed tiles by a factor of 0.03. It answers a Webots episode in which the robot remains mobile while the scorer alternates between two frontier members about 0.9 m apart for 44 events in which the target is reached and immediately reselected, without blacklisting either member. Arrival counts as success, so it never trips a progress-based blacklist. An initial 300 s tile suppression expires and lets the robot return to drained regions. The final form instead marks a tile permanently once the body stays in it for 1.0 s while traveling at least 0.30 m in one segment, and it lifts a mark only when that tile's frontier count rises by 40 % and 20 voxels above the count at marking. The scorer penalizes marked candidates rather than deleting them, leaving the termination logic to decide when to stop.

6.2.3. Volume-Flood Ground-Frontier Scoring

A ground-frontier variant replaces per-voxel gain with the unknown area reachable beyond a walkable support edge. An edge is a valid walkable cell beside an unknown support column. A pit guard removes seen-but-floorless holes, and dilation before component labeling prevents one doorway from fragmenting into many candidates. Each retained edge carries an outward normal and a 0.55 m wall standoff.

The value metric floods unknown cells in the two-dimensional `/map` grid. Occupied and free cells stop the flood, so a box top scores near zero while a doorway reaches the connected unknown room. A position cooldown suppresses revisits within 1.0 m until it expires or the map changes. The evaluation treats this variant separately from the final ESDF frontier scorer but does not complete the comparison.

6.3. Exploration Behaviors and Robustness

Frontier scoring decides which region deserves a visit, not how long that decision stands. Classical closest-frontier and utility-ranked selection treat re-evaluation as free [Yam97; Hol+10], yet selection runs at 1 Hz while a physical transit takes 20 s to 50 s. A measured failure motivates each mechanism below.

6.3.1. Hysteresis and Stuck-Blacklisting

Commitment uses a hysteresis margin, a held target that matches against both cluster centroid and members, a minimum hold timer, and progress-based blacklisting. The margin keeps a marginally better competitor from displacing the committed point, the held target carries it across replans, and the hold timer bounds its lifetime. Blacklisting is the failure path: a stuck detector retires the region around a goal once the body stops approaching it for the stuck timeout (25 s in the deployed configuration), keys the entry on the tile that contains the goal, and retains it for the remainder of the run, releasing it only through the single endgame retry of Section 6.3.4. The physical deployment uses the hold timer and raises it from 8 s to 60 s after shorter values expire

during transit; it remains an interim setting. One state machine carries a candidate through these states: a gate rejection drops it, a traversed tile suppresses it, and the stuck detector blacklists it; no scoring step deletes a candidate, and the termination rule in Section 6.3.4 requires a terminal outcome for every tile.

Arrival follows executor completion. In Webots, an earlier 0.8 m explorer radius declares a target reached while the body remains outside the executor's 0.35 m radius, and the executor then discards the replacement under its duplicate filter, which ignores a goal within 0.25 m of the one just completed. The final explorer test uses 0.36 m held for 2.5 s, ordered behind the executor's own completion test.

Successful short goals can still oscillate without triggering a progress blacklist. The visited-tile penalty in Section 6.2.2 handles this case without removing candidates from the termination state machine.

6.3.2. Wedge Prevention and Near-Field Map Safety

Near-field protection initially reads the planar scan from the rotating LiDAR, whose 1.0 m minimum range makes every closer obstacle invisible: the corner guard, which vetoes motion when a mapped obstacle approaches a body corner, fires zero times while the body grazes walls in simulation. The replacement reads the Euclidean distance field of the camera-only volumetric map over a 0.3 m to 3.0 m band [Ole+17; Mil+24]. Perceptive legged controllers use the same class of near-field terrain evidence [WH21; Mik+22b]. Wedge prevention addresses the related failure in which the body commands motion against an obstacle without achieving net displacement; the follower-side wedge state recovers from it (Section 6.3.3). Two further corrections follow: treating unknown cells as obstacles deadlocks cold-start rotation, so only known obstacles constrain motion; and the watchdog initially treats gate-blocked holding as a stuck state, so the final design separates blocked from stuck: blocked means the explorer holds because the held target fails a gate against the current map, which a later map can repair, and stuck means the body does not move under an accepted goal.

6.3.3. Holonomic Follower and Transform Hardening

The initial follower rotates before translating, although the simulated quadruped accepts lateral body-frame velocity. The replacement transforms path motion into (v_x, v_y), suppresses yaw near walls, limits alignment in open space, and uses a bounded strafe-away command from the known-obstacle distance-field gradient. One Webots episode still exposes an ordering defect in which a shorter stuck callback, 6 s against the 8 s wedge delay, clears the goal before the wedge state can activate.

A malformed quaternion that terminates the navigator motivates transform hardening. The listener now checks numeric type, finiteness, norm, and normalization before insertion, drops only the invalid transform, and catches insertion errors. Inspection

identifies the unsafe quaternion construction in the simulator publisher, and the source correction removes it. Tests cover malformed, missing, non-finite, zero-norm, and denormalized transforms.

6.3.4. Completion Latch and Termination

The first termination rule counts empty replans, which is sound only if an empty candidate set implies a finished map. A physical-robot basement trial disproves it: the vertical frontier band sits below the observed floor, the mapper emits no frontier, and the run latches on an almost empty map.

The final rule requires three conditions in one cycle. The support-cell history must span 480 s with every sample inside 0.75 % of the window mean. Air classification must cover at least 98.5 % of the remaining frontier voxels, with an empty residual passing vacuously. Every candidate tile, one cell of the 1.5 m tiling and the unit of this accounting, must reach a terminal state: visited, blacklisted, without progress after its final visit, or unreachable in the geodesic flood. A tile that has reached none of them is pending; each reachable tile receives one final lift for the whole run, which releases its blacklist or suppression so that the explorer attempts it once more, and a failure in that attempt settles it as without progress. The node logs the tiles it treats as unobservable.

In Webots simulation, an unbounded pending set leaves 33 tiles pending at the single retry, because such a set drains only through an arrival or the per-target stuck path, so completion stays false indefinitely; the bounded replacement passes 30 of 30 unit checks. A recorded near miss of 98.94 % air and 0.5257 % support deviation determines the looser gates above. The latch then fires in two Webots runs: one partial-coverage run logs 28 tiles as unobservable, and one high-coverage run finishes about 2 % below a teleoperation reference of 15 788 observed columns. A further run exceeds that reference and does not latch, processing roughly 4300 permanently unobservable air frontiers for more than 10 min. The conditions prove necessary but not sufficient, so the later frontier-seal rule removes air frontiers at their source.

6.4. Walkable-Goal Filtering

This layer contributes goal admissibility rather than goal selection: every goal must lie on mapped, walkable, geodesically reachable, head-clear ground, and a second node with an independent implementation audits every publication, that is, every accepted waypoint on `/way_point`, and reports the share of them that its audit accepts (Section 9.1.2). Ground exploration requires this constraint because traversability limits goal placement [Cao+22], and published legged systems project candidate graph edges onto mapped ground and reject edges without support beneath them [Kul+22]. The mapper remains a pure sensor-to-map component and derives a per-column support surface, headroom, and walkable mask from a GPU column pass over the distance-field query grid at 0.10 m resolution, while all filtering remains in the explorer. A second

elevation-mapping stack behind a cross-domain relay [FBH18; Mik+22a] remains an unused fallback.

6.4.1. Filter Design and Bag-Replay Validation

Per column, the support surface is the lowest occupied voxel with two unoccupied voxels above it and an observed-free first voxel above it, which prevents a tabletop from becoming the surface; headroom is the distance from it to the first occupied voxel above. From the height map, the mapper computes the three traversability layers of established elevation-map filter sets [Wer+16; FH16]: slope against 20°, step against 0.15 m, and roughness at 0.08 m.

The slope stencil provides the instructive parameter. Because the height map inherits voxel quantization, a half-width of one cell reads $\arctan(0.5) = 26.6°$ at every quantization step of any ramp shallower than 45° and rejects legitimate ramps, whereas a half-width of two averages that staircase to $\arctan(1/4) = 14°$. On a synthetic 12° ramp, the walkable fraction rises from 59 % to 100 %, which provides unit-test evidence. Map discretization noise rather than the robot datasheet therefore determines threshold calibration.

Four gates run per replan cycle. Gate zero snaps a frontier voxel to its column's support cell, requires a vertical offset between −0.2 m and 1.5 m, drops candidates whose column has no surface, and sets the goal height to the surface height. Gate one requires a 0.35 m metric disk in which at least 80 % of the cells are valid, meaning that their column yields a support surface, and at least 90 % of those valid cells are walkable, meaning that the same surface also passes the slope, step, and roughness tests (Section 9.1.2). Gate two floods eight-connected from the robot cell over walkable cells only, with a 0.15 m per-step height limit, and admits candidates that any of the last three floods reach, so one failing flood does not reject a still-reachable candidate. Gate three requires 0.75 m of headroom. The design specifies but does not implement a fifth gate against the manufacturer's legged obstacle grid. Flood cost also replaces Euclidean distance in the utility [ACM21; Zhe+26].

The filtering sequence in Figure 6.2 separates the reason for each rejection from the final waypoint publication. It also distinguishes the explorer's gates from the independent validator that audits the published goal and lists the terrain quantities that the gates read.

Every cycle re-evaluates every gate against the current snapshot for the held target and new candidates and re-snaps the held target in height. The explorer drops a failing held target instead of blacklisting it because dropping handles a target that an improving map invalidates, while blacklisting remains for demonstrated unreachability. That path closes a failure in which stale republication causes 33 walkability and 3 height-match rejections within one replay. A separate validator with its own gate implementation audits acceptance; it cannot import the explorer, re-checks every waypoint against its

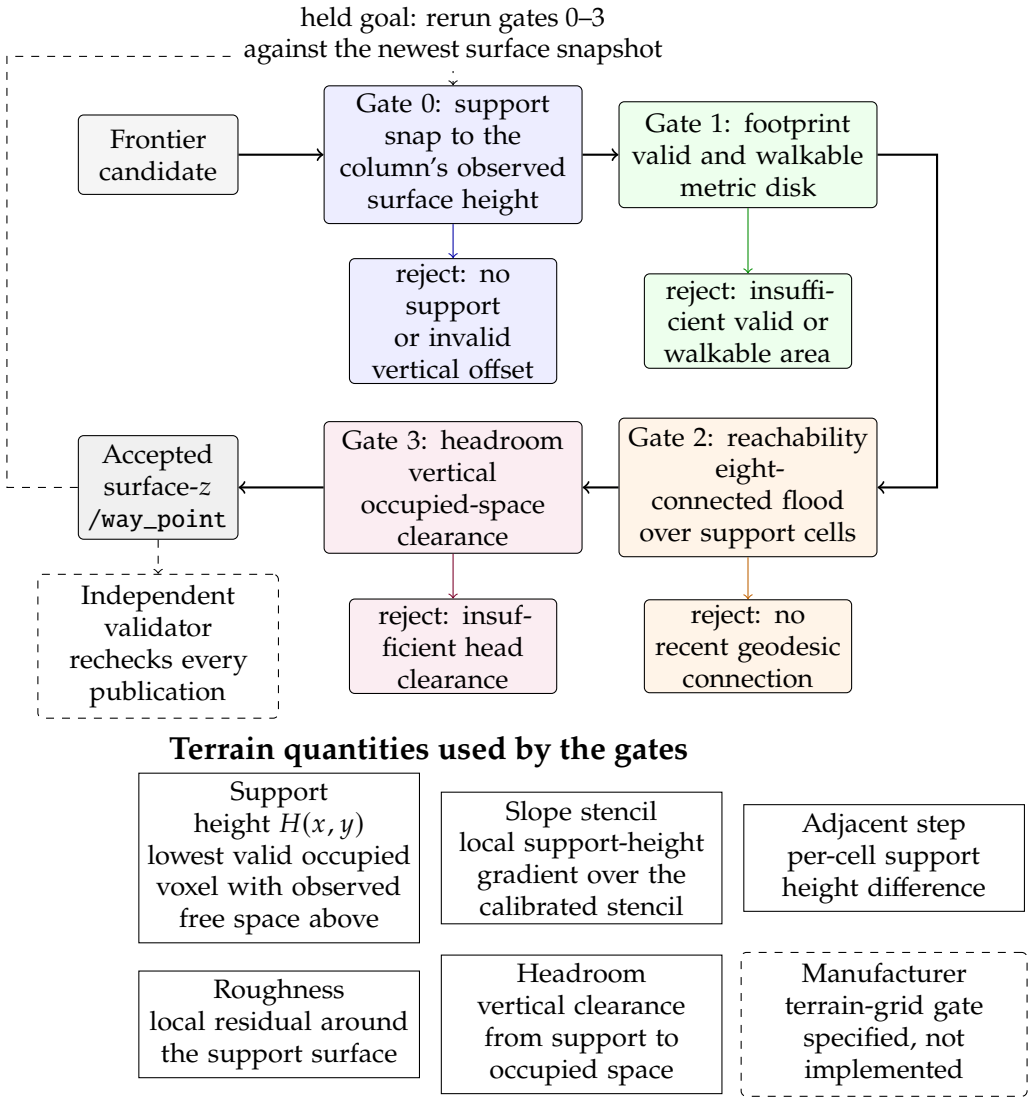


Figure 6.2: Walkable-goal gate sequence and terrain quantities used by the filter.

last three support-surface snapshots, and accepts it on the best of those three verdicts.

Four replay iterations on one indoor recording raise the validated-publication result from 30 of 40 to 112 of 118. Six brief reachability blips account for the residual failures, and the explorer drops those stale targets on the next cycle. Across the final three iterations, 464 publications contain no goal in air, on a tabletop, or under a table. Across Isaac Sim open-terrain episodes with the ESDF scorer, the footprint gate rejects 17.8 % to 33.7 % of candidates while reachability and headroom reject none; in that environment, the filter therefore reduces to a footprint test [PBF22].

6.4.2. Cold-Start Exploration from Depth Cameras

Gate zero also exposes a sensing limitation that no filter tuning can repair, because a goal cannot pass where sensors never observe the floor. In Webots, the simulated rotating LiDAR leaves the floor within roughly 2.6 m of the body unobservable, so no support surface exists beneath the robot at cold start and every frontier fails gate zero. The robot snap radius, within which the explorer replaces its own unsupported cell with an observed support column, is enlarged from 0.5 m to 4.0 m, which only mitigates this behavior, and the evaluation does not repeat the final setup with the original value. The physical robot has no such disk because five body-mounted depth cameras cover it [Bos24], as hardware evidence confirms. Enabling the same five cameras in Webots, together with a single-precision depth branch in the mapper's unit conversion, produces admissible goals without driving: the mapper reports 3447 support cells, 1976 walkable cells, and 998 admitted candidates from 3105; the first target lies 2 m ahead, and the validator passes all 47 publications in that slice.

6.5. Frontier-Seal Termination

Frontier sealing is the termination rule for residual frontier voxels that no longer correspond to ground the robot can reach. A frontier voxel is an observed-free cell beside unknown space. The seal allows such a voxel to stop counting as unexplored when it lies above an interior mapped floor or at a supported local crest. It must preserve the forward frontier ring at the sensor horizon, because that ring represents genuinely unobserved ground. Sealing therefore removes unreachable interior air from termination accounting without declaring the unknown volume free.

An Isaac Sim control episode in a walled 140 m^2 arena, meaning the same stack with sealing disabled, quantifies the termination problem. Support, the support-column count that measures coverage in Section 9.1.2, reaches 99.5 % of its final value at 517 s, but the episode continues to 5826.8 s and 1386.43 m. Most residual frontiers stand over mapped flat ground while the air above remains outside the low cameras' view.

The first seal ignores an unknown voxel when any occupied voxel lies within 1.0 m below it. Frontiers fall from 527 to 17, but no goal survives and support reaches only 22.5 % of the control. Ordinary floor beneath the forward sensor horizon satisfies the

Table 6.1: Goal-generation frameworks evaluated, by representation and integration fit.

System	Map boundary	Outcome of the evaluation
GBPlanner ground branch	Separate voxblox volume in ROS 1	Returned one vertex and no edge in the evaluated branch
GPU elevation mapping	Separate 2.5D elevation grid	Connectivity idea retained; separate process retired
TARE	Separate terrain cloud with one height per column	Kept as fallback after terrain-cloud geometry limits; the physical goal-only termination cause remains undetermined
Hydra	Separate free-space topology	Set aside when the topology layer did not form in replay

same test as floor beneath an interior air column, so the rule removes the frontier ring that continued exploration needs.

The revised seal applies that vertical test only to an interior column or a local-maximum crest. Interiority requires mapped ground throughout a 1.25 m neighborhood. The crest test requires 0.15 m of relief and no column more than 0.10 m higher within 2.0 m. The rule preserves all 180 horizon-frontier voxels in its decisive fixture and passes 86 of 86 checks.

In Isaac Sim, the revised seal retains 98.6 % of the control support after 57.08 m, leaves three frontier voxels, and produces 242 of 242 valid publications. It does not latch because the history guard demands that the sampled window span its full nominal width, which jittered sampling misses by about one sample interval, so the condition can never hold; accepting 99 % of that width instead passes 160 of 160 jittered histories. With that correction, a later run retains 96.9 % of the control support, leaves one frontier voxel, validates 301 of 301 publications, and latches near 740 s.

These results come from single Isaac Sim episodes in one arena and one seed. The seal defaults to disabled, the corrected guard exists only in a side variant, and the physical-robot record contains no latch.

6.6. Alternative Exploration Frameworks Evaluated

The evaluation considers four goal-generation systems before selecting the custom frontier explorer for the final system. They use different environments, sensors, and metrics, so Table 6.1 compares integration fit without ranking performance, listing the map boundary and the recorded outcome for each framework.

6.6.1. GBPlanner Ground Branch and Elevation Mapping

The evaluated GBPlanner ground branch runs in an isolated ROS 1 container with its own voxblox map [Dan+20; Ole+17]. The aerial control forms full graphs, but the

ground path returns one vertex and no edges on the same harness. This branch-specific result does not invalidate the useful separation of informative and commandable vertices [Kul+22].

GPU elevation mapping supplies grounding and reachability rather than exploration [Mik+22a]. Its connectivity plugin changes the selected direction in one offline replay, which does not demonstrate improved exploration. Moving support and reachability into the existing mapper retires the separate process but retains its connectivity flood.

6.6.2. TARE for Goal Generation

TARE serves as the goal-selection component for roughly six weeks and reaches the physical robot and Isaac Sim [Cao+21]. Its separate terrain cloud stores one height per column [Cao+22], so it cannot preserve an elevated deck over walkable floor. A physical goal-only run also terminates before motion, and the evaluation does not complete the planned diagnostic comparison, so the cause remains undetermined. The stack retains TARE as a fallback after that point.

6.6.3. Hydra Scene Graphs

The evaluation tests Hydra in physical-recording replay for free-space topology and loop-closure updates. Reconstruction continues, but the exploration topology remains empty across roughly 800 distance-field updates; this result applies only to the evaluated build.

7

Navigation

Autonomous exploration on legged platforms coordinates spatial representations across distinct scales: volumetric distance fields select frontier targets (Chapter 6), while locomotion maintains dynamic balance across irregular ground (Chapter 8). Navigation operates between these boundaries, evaluating geometric clearance over reconstructed terrain and generating velocity commands along traversable routes.

This chapter examines whether a two-dimensional route representation can support legged traversal of ramps, curbs, and dunes without discarding terrain constraints needed for safe execution. The answer depends on the interfaces among volumetric mapping, goal filtering, path planning, and learned locomotion.

The evaluation compares a terrain-aware ground-autonomy pipeline with a layered costmap pipeline that employs model predictive path integral (MPPI) control [Wil+16]. Evidence comes from Webots simulation, early physical trials on Spot, and Isaac Sim evaluation of the final Nav2 MPPI chain in dune and ramp environments. None of the evaluated configurations achieves autonomous ramp ascent. Offline bag replays validate upstream mapping and goal gating rather than route execution, and joint faults prevent a physical navigation rerun of the final floor-relative Nav2 configuration.

7.1. Navigation Requirements and Boundaries

The stack separates mobility into exploration, navigation, and locomotion. Exploration evaluates volumetric distance fields to select coordinates on supported terrain, while locomotion controls the joints and absorbs local surface variation. Navigation routes between those layers by evaluating geometric paths, enforcing body clearance, and emitting planar velocity commands.

These layers use strict interface contracts. Goal generation publishes `PointStamped` targets on `/way_point`, while navigation publishes `Twist` velocity commands. The safety supervisor alone owns `/spot/cmd_vel` on hardware and intercepts commands

on lidar proximity, stale transforms, or inadequate progress; simulation remaps the equivalent output topic. Here *watchdog* denotes a supervisory progress check that aborts or replaces a route when displacement stays below a threshold for a fixed window, and *recovery* the escape maneuver it triggers.

Legged exploration imposes five operational requirements on navigation: (1) operating into unmapped space toward frontier boundaries without aborting on unobserved cells; (2) ensuring body clearance in narrow corridors; (3) tolerating sensor latencies where volumetric fields refresh slower than control loops; (4) reconciling platform deadbands where low commands fail to induce walking, which requires a software command floor rather than a zero lower bound; and (5) decoupling contact mechanics from geometric routing.

Delegating contact mechanics to locomotion allows navigation to plan over continuous geometric representations rather than discrete contact graphs [Bou+20; WH21]. Published legged systems traverse rough and discontinuous footholds through their locomotion layer [Hut+16; Lee+20; Mik+22b]. In this stack, upstream goal gating verifies that target coordinates and the reachability flood lie on supported, walkable terrain; the locomotion layer then handles local roughness, small steps, and unmodeled inclines along intermediate route segments.

7.2. Early Navigation Methods

The first navigation configuration reused a planar stack, and hardware operation exposed four constraints before any terrain-aware component existed. The two subsections below report the baseline and the simulated comparison that followed, which separates the controller, the map representation, and the route execution.

7.2.1. Planar Navigation Baseline and Early Physical Findings

Initial exploration trials on the physical quadruped combine `explore_lite` [Yam97], Nav2 [Mac+20], and `slam_toolbox` [MJ21] in a containerized runtime [BDA24]. Hardware operation exposes four constraints: planners need unknown-space traversal for frontier goals; the dynamic-window-based (DWB) controller commands reverse rotations into unobserved space, which leads to Regulated Pure Pursuit [Mac+23] for forward tracking; LiDAR updates at 3 Hz cause transform timeouts in the 50 Hz controller loop and require a transform freshener; and an oriented rectangular footprint, the body polygon tested for collision (0.55 m by 0.25 m on hardware with 0.30 m inflation), represents doorway clearance more accurately than the default radius.

7.2.2. Comparison of Three Simulated Navigation Mechanisms

Webots simulation [Mic04] evaluates three mechanisms against a unified frontier interface with nvblox volumetric mapping [Mil+24], bounded by a simulated cap of $0.15\,\mathrm{m\,s^{-1}}$ linear speed and $0.40\,\mathrm{rad\,s^{-1}}$ angular rate. Nav2 costmap search over

a projected grid [Mac+20; Mac+23] reaches 4 clustered goals over 70.91 m without aborts at 38.44 % navigation CPU. Potential-field descent on the Euclidean distance field [Kha86; Ole+17] reaches 2 targets over 19.35 m at 12.86 % CPU but suffers from local-minimum oscillations. Geodesic Dijkstra search across support columns (0.10 m horizontal query-grid columns carrying a valid support surface, Chapter 9) reaches 3 coordinates but advances only 1.33 m at 10.18 % CPU because of mesh step discontinuities. This evaluation also exposes a topic collision that causes 72 costmap resizing events on the static layer, the persisted-map layer rather than a live obstacle layer; removing the collision and extending the goal-relay hold timers prevent aborts during transit.

7.3. The Terrain-Aware Ground-Autonomy Navigator

Planar route execution cannot express the terrain constraints that the preceding section identifies. This section describes the ground-autonomy architecture, the integration defects that appeared in simulation and on hardware, and a comparative evaluation against continuous mesh navigation.

7.3.1. Architecture of the Ground-Autonomy Stack

The limitations of planar navigation motivate evaluation of a three-dimensional ground-autonomy architecture. The stack integrates the Autonomous Exploration Development Environment [Cao+22] with the FAR route planner [Yan+22] and couples visibility routing with motion primitives. FAR maintains a visibility graph from point clouds, `localPlanner` [Cao+22] evaluates motion primitives, and `terrainAnalysis` fits planes across a 10 m by 10 m patch. The ground-autonomy follower, a pursuit controller, tracks collision-free paths directly on surface points without collapsing relief, though sparse returns can misjudge continuity.

7.3.2. Navigator Defects and Integration Fixes

Integration of this pipeline into Webots exposes five defects and corresponding corrections: (1) a 15.0 s point decay with a 1.5 m clear radius resolves near-field point-memory freezing; (2) net displacement (0.15 m over 30 s) replaces velocity in the escape trigger and exposes vehicle stalls during entrapment; (3) an explicit latching escape state with a 30 s cooldown prevents premature recovery cancellation; (4) a bridge watchdog eliminates stale route handoffs by reasserting waypoints directly to the local planner; and (5) narrowing the vehicle collision envelope from 0.90 m to 0.72 m restores doorway clearance and enables 2322 collision-free through-poses in corridor benchmarks.

7.3.3. Comparative Simulated Evaluation and Route Variance

Across 18 timed Webots simulation episodes, an alternative architecture that employs Move Base Flex [PSH18] with continuous mesh navigation [Püt+21] plateaus at 8751

support columns, roughly 55 % of the teleoperated baseline (15 788 columns in the Webots maze world), because corridor costs remain conservative. In contrast, the ground-autonomy pipeline traverses the interior doorway and achieves the highest coverage among the evaluated simulated routes, although no single episode satisfies all six acceptance criteria of the comparison simultaneously (an accepted completion latch plus coverage, walkability, validator, termination, and timing thresholds; Chapter 9). Its strongest completed episode records 15 478 support columns in 1043 s and validates all 1693 waypoints before completion 310 columns short of the human baseline; another run reaches 15 815 columns before an air-frontier cycle stalls it. Byte-identical deployments also produce terminal counts of 15 478, 15 296, and 14 239 columns, which shows that early routing decisions alter subsequent discovery sequences. These episodes evaluate configurations under specific route draws rather than rank algorithms deterministically.

7.4. Physical Deployment of Ground Autonomy

Following simulation verification, the ground-autonomy pipeline moves to the physical Boston Dynamics Spot quadruped with adjustments to eight environment bindings, including frames, clocks, and a point-cloud-to-scan converter. Staged bring-up validates static registration at LiDAR scan rate with terrain analysis at 10 Hz, followed by short-range goal dispatch tests under velocity caps.

Continuous exploration reveals four platform constraints: (1) actuator deadbands, the measured command range that produces no motion, ignore velocities below $0.15\,\mathrm{m\,s^{-1}}$, so a software command floor of $0.15\,\mathrm{m\,s^{-1}}$ and a $0.30\,\mathrm{m\,s^{-1}}$ hardware cap enforce motion; the Webots comparison uses a separate $0.15\,\mathrm{m\,s^{-1}}$ episode cap in Section 7.2.2; (2) tablet overrides and battery swaps revoke motor leases and reset odometry origins; (3) dynamic walking accumulates −0.11 m of vertical odometry error per 100 s, so the body-height belief moves from 0.68 m to 1.57 m while the true standing height stays near 0.68 m, which decouples world-anchored height thresholds; and (4) the ground-autonomy follower induces oscillatory turn-overshoot-turn cycling at low speeds.

A Webots simulation comparison evaluates a sequence of parameter settings against the oscillatory turn-overshoot-turn behavior from initial hardware walking. Increasing the look-ahead distance to 1.0 m, lowering yaw-rate gains, and adding a downstream adapter with slew-rate limits reduces command yaw sign reversals, counted as yaw command sign flips per meter traveled, by 62 %, from 19.9 to 7.6. The hardware configuration then adopts the selected parameters.

Continuous hardware exploration exposes three structural bottlenecks: the ground-autonomy follower idles every 15 s to 20 s at snippet boundaries, where a snippet is one short rolling segment of the reference path handed to the local controller; physical sensor noise accumulates spurious obstacles in FAR Planner's visibility graph;

and divergence between the local planner's sparse point cloud and nvblox's dense volumetric distance field repeatedly stalls routes.

7.5. The Nav2 Migration and Predictive Control

These hardware difficulties prompt a re-examination of the architecture: controller idling, fragile routing, and mismatches between the LiDAR cost cloud and the volumetric distance field. The revised architecture moves routing and control to Nav2 [Mac+20; Mac+23], retains nvblox TSDF/ESDF mapping [Mil+24], the walkable-goal filter, and the safety supervisor, and unifies the world representation in a layered costmap framework.

7.5.1. Model Predictive Path Integral Control and the Mini-Global Relay

An isolated simulation comparison evaluates the ground-autonomy follower against Regulated Pure Pursuit and MPPI control across a 7.56 m course. Relative to the ground-autonomy follower, MPPI reduces commanded yaw sign reversals from 7.41 to 0.44 flips per meter, raises drive duty, the share of the run during which a nonzero drive command is issued, from 0.59 to 0.98, and removes the recorded stop-and-go intervals by forward-simulating rollouts with path and cost critics [Wil+16; LHS14]. Short 1.0 m to 1.5 m rolling snippets nevertheless cause MPPI to loop and lose waypoint precision, so it completes only three of four legs.

To restore precision, a mini-global goal relay computes reference paths through eight-connected octile A* search [HNR68] with unknown space treated as traversable; the relay takes the global-planner role over the projected occupancy grid, supplying one grid-searched path per waypoint in place of the rolling snippet that MPPI otherwise tracks. The relay dispatches an action goal to MPPI and suppresses replanning within a 0.35 m arrival radius, which guides the platform through all legs in 8.54 m at 0.35 yaw flips per meter. Subsequent simulated runs resolve three costmap defects: an inflation radius of 0.40 m, the distance over which obstacle cost spreads into free space, with cost scaling 3.0, the rate at which that cost decays with distance, restores 1160 boundary cells that had collapsed under the earlier inflation settings; an oriented rectangular polygon (0.96 m by 0.36 m, larger in both dimensions than the 0.55 m by 0.25 m hardware footprint of Section 7.2.1) restores doorway clearance; and a 2.0 s replan interval with a 0.30 m drift trigger reduces replanning frequency from 368 to 578 calls per goal to 10 to 20. Exploration in the Webots maze world subsequently reaches 15 810 support columns in 10.5 min, matching the human baseline pace.

7.5.2. Sensor-Band Geometry and Hardware Phantom Obstacles

Porting Nav2 to hardware uncovers dense fields of non-clearing obstacle phantoms, obstacle cells that no later sensor ray removes, in the local costmap. Chassis-frame

height cropping tilts during walking pitch and captures ground reflections at range, while discarded out-of-range returns prevent ray clearing, the removal of an obstacle cell by a passing ray. Gravity-aligned cropping in the odometry vertical axis, `inf_is_valid=true` for clearing rays to 3.0 m, and a reduction in obstacle persistence, the time an unrefreshed obstacle cell survives, from 3.0 s to 1.0 s resolve the issue. These measures suppress near-field phantom accumulation, although some far-field camera noise remains; joint actuator faults prevent the physical quadruped from executing the subsequent autonomous navigation mission.

7.6. Terrain-Preserving Two-Dimensional Projection

Planar navigation over complex terrain depends on the projection connecting volumetric mapping to the costmap. Nav2 plans across a layered two-dimensional costmap [LHS14], which lacks an intrinsic model of terrain relief; the stack resolves this limitation by embedding terrain awareness into the projection itself and surrounding it with protective gates.

7.6.1. Floor-Relative Obstacle Slicing and Step-Gated Crest Recovery

Fixed-elevation slicing fails as vertical odometry drifts or the robot ascends slopes, either slicing into rising ground or floating above obstacles [Mil+24]. The stack replaces fixed slicing with a per-column floor-relative band anchored to a support surface $H(x, y)$, the local floor height from which the band measures clearance. The lowest occupied voxel with two unoccupied upper neighbors and an observed-free immediate upper neighbor defines this surface. The costmap evaluates obstacle clearance within a vertical window:

$$z \in \left[H(x, y) + 0.30\,\mathrm{m},\ H(x, y) + 0.80\,\mathrm{m}\right]. \tag{7.1}$$

A column is occupied if any observed voxel within this window has obstacle distance $D(x, y, z) \leq 0.10\,\mathrm{m}$. A chassis-anchored fallback band, $[z_{\mathrm{body}} - 0.38\,\mathrm{m}, z_{\mathrm{body}} + 0.12\,\mathrm{m}]$ with z_{body} the odometry body height, applies to unobserved columns lacking a floor. Because ground and obstacles enter the map during the same temporal epoch, their vertical separation remains invariant to odometry drift.

Downward depth cameras cannot observe airspace above convex ridges from below. Unobserved voxels above the crest cause support detection to fail, which activates the chassis-anchored fallback band and falsely labels the crest as an obstacle. The stack resolves crest occlusion through a two-stage correction: it marks columns with established support and no observed obstacle in their vertical band as free, while a 3×3 maximum-height fill over neighboring columns reconstructs isolated support

gaps on ridges under a step-continuity gate:

$$\Delta H \leq 0.15\,\mathrm{m}. \tag{7.2}$$

Here ΔH is the support-height difference between adjacent columns considered by the fill. In a controlled Isaac Sim evaluation, an ungated fill converts 38.9 % of lethal cells on a 0.40 m curb into traversable free space. Applying the selected 0.15 m step criterion retains every curb cell as lethal while restoring traversability in the tested crest cases.

7.6.2. Preserved Features, Discarded Properties, and Safety Limits

The projection preserves local ground clearance, metric inflation, and unknown space, but discards elevation, slope, roughness, step height, headroom, and negative obstacles. It describes where a body may stand rather than the full topography. Upstream, the walkable-goal filter (Chapter 6) audits target coordinates and its reachability flood against a 2.5D model [FH16; PBF22], using limits on slope (20°), step height (0.15 m), roughness (0.08 m, the elevation-map height-variation layer), and headroom (0.75 m, the distance from the support surface to the first occupied voxel above it). Those checks do not establish a continuous slope bound along every route. Downstream, the locomotion layer handles contact and balance with a policy that training exposes to terrain gradients up to 0.4, corresponding to 21.8° (Chapter 8) [Hwa+19; Rud+22].

The evaluated dune episodes stay inside this training envelope, with slopes peaking at 14.5° and freed crest cells at or below 14.3°. These results do not bound safety under three conditions: an intermediate route face steeper than the policy tolerates; an unobserved low overhang above known floor; or a descending edge that produces no positive obstacle return in the projection band. The first is absent from the grid, the known-floor rule may free the second, and the third depends on the upstream reachability flood.

The terrain-routing contract in Figure 7.1 keeps support and walkability tests outside the projected route grid while retaining a conventional planner and controller interface. The figure describes ownership and information loss; it is not a claim that the projected grid preserves complete terrain geometry or that the final chain was validated on the physical robot.

7.7. Final Configuration and Unresolved Limits

The final configuration combines the volumetric mapper, the floor-relative projection, the mini-global relay, and MPPI control that the preceding sections introduce. This section states that configuration with its verification results and then records the representational limits that the evaluation did not resolve.

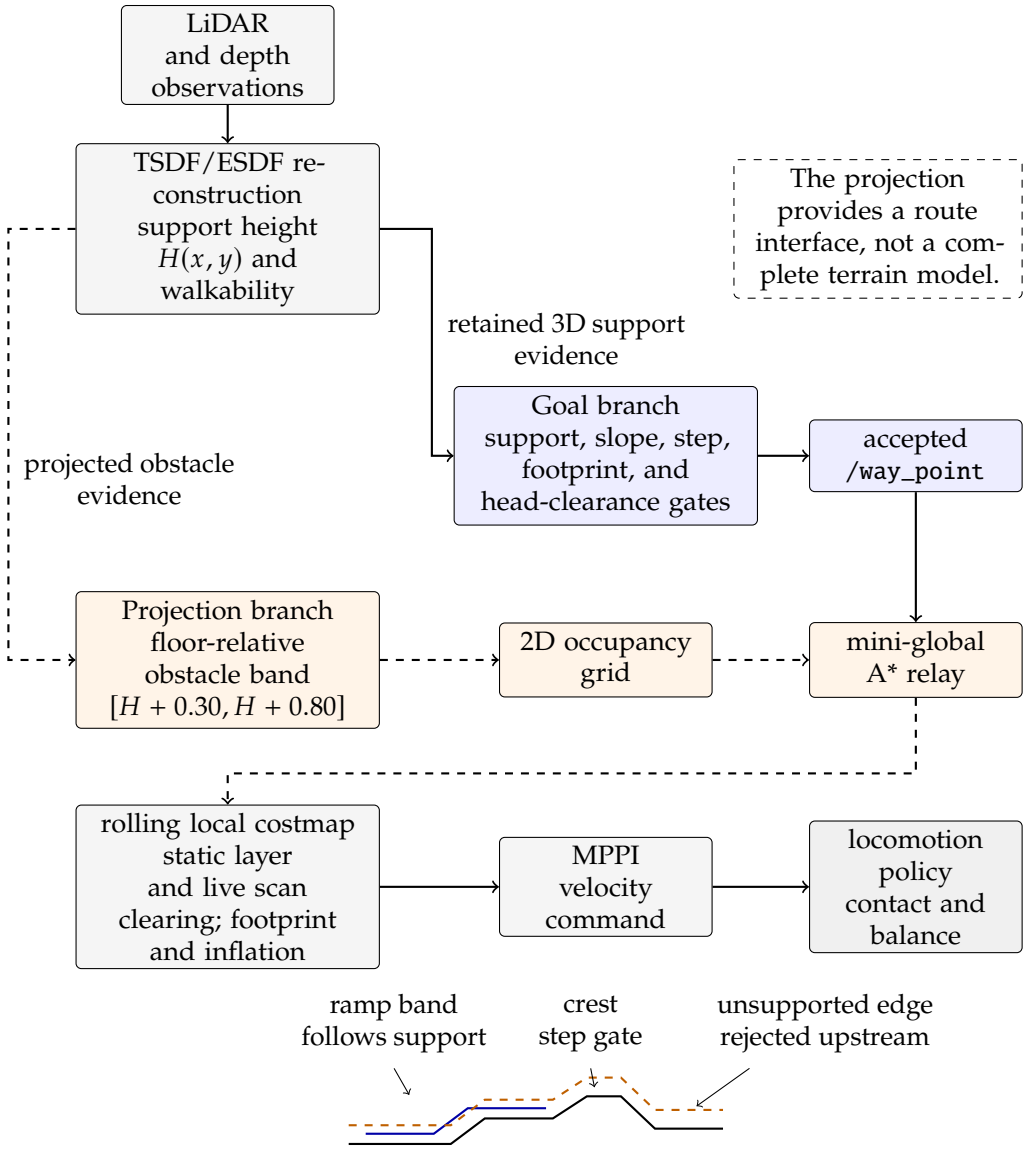


Figure 7.1: Terrain-preserving navigation contract. Solid paths retain three-dimensional support and goal evidence; dashed paths carry the floor-relative projection used by the two-dimensional route and costmap layers, and the lower panel sketches the ground plane (solid), the support surface $H(x, y)$ (blue), and the floor-relative band (dashed orange) for a ramp, a crest, and an unsupported edge.

7.7.1. Final Configuration and Verification

The final navigation architecture combines volumetric distance mapping, floor-relative projection, a mini-global A* relay, and MPPI local control. A five-check verification suite in Isaac Sim records settling errors, the terminal distance to the goal, of 0.24 m to 0.28 m on 3 m to 5 m transits, completion of one uphill goal in 10.4 s with 0.23 m error, stall-triggered reverse recovery, one publisher on each of nine navigation topics, and zero measured displacement during 300 zero-velocity commands over 15 s. Two configuration changes address observed stalls: the command floor rises from $0.25\,\mathrm{m\,s^{-1}}$ to $0.40\,\mathrm{m\,s^{-1}}$ under a $0.45\,\mathrm{m\,s^{-1}}$ cap, both higher than the hardware floor and cap of Section 7.4, and the watchdog timeout increases from 6 s to 15 s to accommodate gait initiation, the higher command threshold that starts a gait from rest than to sustain one.

7.7.2. Representational Limits and the Ramp-Preference Problem

The evaluated dune episodes do not resolve the ramp-preference problem. In an Isaac Sim ramp world, the floor-relative projection classifies a 12° incline as traversable floor, with support rising 0.15 m per meter. Given goals beyond the crest, however, the global planner routes across adjacent flat ground and terminates at ground level. During manual teleoperation, observation shows the policy traversing the incline, but the record contains no quantitative trajectory; chapter 8 therefore treats the observation as qualitative. The autonomous failure exposes a limitation of the unweighted occupancy projection in the implemented configuration: flat floor and ramp share zero traversal cost, while the crest landing rail contributes occupied cells, so graph search has no cost gradient that favors elevation gain. A terrain-gradient cost layer [FH16; Wer+16] or a surface planner would preserve information that the binary projection removes.

Physical Spot trials evaluate the earlier ground-autonomy and planar Nav2 configurations. Isaac Sim alone evaluates the final integrated architecture with floor-relative projection and MPPI control because joint faults remove robot access before the planned hardware navigation trials. The final Nav2 chain therefore carries no physical validation claim.

8

Reinforcement-Learning Terrain Locomotion

The navigation layer of Chapter 7 emits planar velocity commands, and the locomotion controller converts those commands into joint motion over whatever ground the route crosses. This chapter states the baseline controller used here, the terrain limits of that baseline, and the training stages that adapt it to non-planar ground. The retrained policy runs in simulation only, because joint actuator faults ended physical operation before its deployment.

8.1. Locomotion Limits on Non-Planar Terrain

Autonomous exploration requires motion beyond flat ground. Legged platforms negotiate broken terrain by placing footholds and adjusting posture [Bos24]; the locomotion layer therefore determines which geometrically admissible routes the robot can execute.

Isaac Lab develops the baseline quadruped controller for flat surfaces [OHB24; Mit+25]; that flat baseline policy was trained without a terrain curriculum and observes no terrain height. It natively accommodates micro-roughness between 0.02 and 0.05 m; macroscopic inclines, dune waves, and relief above 0.05 m require retraining. In Isaac Sim, the baseline stops before a twelve-degree ramp, remains stationary when it spawns on the slope, and tips prone on dune relief without recovering.

An earlier Webots trial exposes the same training boundary [Mic04]: an open-loop crawling gait climbs a ten-degree incline but rolls over at fifteen degrees because it lacks attitude feedback. Planar velocity interfaces can support slopes when coupled to a terrain-capable controller. The final design therefore retains the planar velocity interface and trains the locomotion policy on non-planar profiles.

8.2. Training Environment and Policy Interface

Preserving modularity requires a narrow interface between navigation and locomotion. The policy receives no terrain geometry and consumes forty-eight values: body linear velocity, angular velocity, projected gravity, and velocity commands (three each), followed by twelve joint positions, twelve joint velocities, and twelve previous actions [Lee+20]. The configuration disables observation noise. The network emits a dimensionless vector $a \in \mathbb{R}^{12}$, which a 0.2 rad scale converts before addition to the nominal stance q^{nominal}. A proportional-derivative model tracks joint targets with $K_p = 60.0$, $K_d = 1.5$, 45 N m hip limits, 0–8 ms randomized actuation delay, and an empirical knee-torque table. These actuator parameters come from the shipped Spot configuration and are not retuned in any stage. Training evaluates the policy at 50 Hz over a 500 Hz physics step through decimation 10.

The environment comprises 200 procedural 8 m × 8 m tiles across ten difficulty rows and twenty terrain-type columns: pyramid slopes up to gradient 0.4 (21.8°) account for 50 %, dune waves for 15 %, uniform noise for 15 %, and boxes and stairs for 10 % each, with stair risers capped at 0.15 m. A distance-based curriculum [Rud+22] promotes an environment after the robot traverses half a tile and demotes it when travel falls below half the commanded distance, so a curriculum level indexes terrain difficulty rather than policy performance.

Optimization uses PPO [Sch+17] with networks of dimensions 512, 256, and 128. The actor is warm-started from the flat baseline policy [OHB24], whose parameters load without modification because the observation layout is unchanged, while the critic starts from scratch. Across four stages, training accumulates 4,000 iterations (196,608,000 steps in 2 hours, 51 minutes, and 55 seconds) on a mobile GPU across 2,048 parallel environments.

The four-stage correction path is summarized in Figure 8.1. Each stage addresses a distinct failure observed in Isaac Sim; the values shown are stage-specific checks rather than a common reward score. The policy remained simulation-only because the physical Spot was unavailable for the retrained controller.

8.3. Initial Policy and Reward-Threshold Exploit

For the initial rough-terrain adaptation, fine-tuning runs the flat baseline policy for 1,500 PPO iterations (73,728,000 environment steps in 1 hour and 35 minutes). Foot-clearance calculation uses the lowest detected foot height to avoid climbing penalties, the world-level orientation penalty drops from -3.0 to -1.0 because gravity alignment conflicts with ascending inclines, and termination applies only to body collisions, which permits leg contacts on crests and stairs.

In simulation, this policy exhibits uncommanded forward drift: under zero commands, the robot accelerates to 0.64 m/s and strikes the boundary within twenty seconds.

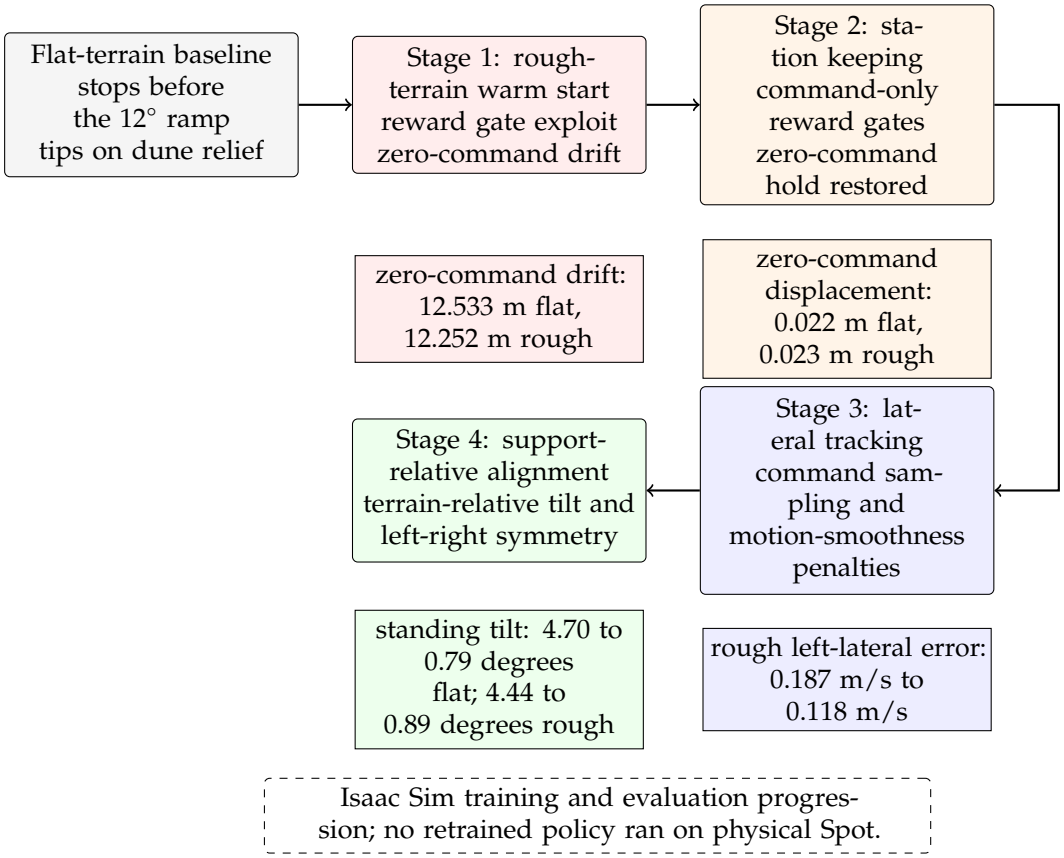


Figure 8.1: Simulation-only terrain-locomotion training progression. The stages address drift, station keeping, lateral tracking, and support-relative alignment in sequence; stage metrics are not a common cross-stage score.

Standard metrics mask this defect: average episode distance reaches 16.25 m with an upright fraction of 0.997 and fails to distinguish tracking from uncommanded motion.

Inspection reveals a threshold exploit in three inherited rewards, which share one activation gate, a condition on when a term applies: air-time (+5.0), gait (+10.0), and joint position penalties (-0.7). The disjunctive gate activates when velocity receives a command or base speed exceeds 0.5 m/s. Reward weights are per-step terms of one summed objective, so only relative magnitudes within a stage carry meaning. Under zero commands, speed above 0.5 m/s triggers gait and air-time rewards simultaneously (fifteen points per step), overpowering the 2.4-point tracking penalty at 0.64 m/s; higher speeds reduce tracking returns without increasing bonuses.

To isolate the defect, a per-axis Isaac Lab harness holds one command axis fixed across sixty-four simulated environments for 950 steps after a 100-step initialization. Its seven gates score the zero-command hold, the four linear holds, and the two yaw holds, against 0.10 m of zero-command hold displacement (net displacement over the hold), 0.15 m/s of linear error, and 0.15 rad/s of yaw error; later stages add gates for defects the earlier set missed, so the counts reach twelve and fifteen (Section 9.4). On flat ground, the zero-command hold drifts 12.533 m at a mean speed of 0.765 m/s and the policy passes three of seven gates; on rough terrain, it passes two gates.

8.4. Correction of Station Keeping

A 500-iteration fine-tuning stage corrects station keeping (24,576,000 environment steps in 14 minutes and 47 seconds). Command-only conditions replace speed-dependent gates across gait, air-time, and joint-position rewards, which prevents base velocity from triggering stepping bonuses under zero commands. The configuration also adds a -3.0 standing velocity penalty for commands below 0.05 m/s, increases standing environments from 0.10 to 0.25, makes longitudinal commands symmetric between -1.5 and +1.5 m/s, raises tracking weights from 5.0 to 8.0, and halves gait scaling to 5.0.

In per-axis evaluation, zero-command displacement falls from 12.533 m to 0.022 m on flat ground, a 570-fold reduction, and passes all seven gates; on rough terrain, displacement falls from 12.252 m to 0.023 m. The policy initially passes six of seven rough gates, failing left lateral hold at 0.181 m/s, and passes five of seven under the subsequent common task, one shared configuration that applies identical terrain, physics, and command settings to every later policy, where backward tracking reaches 0.159 m/s against the 0.15 m/s criterion. Forward hold distance on rough terrain remains at 13.332 m versus 14.190 m initially, while standing action rate, the mean norm of the change in the action vector between successive policy steps, drops from 1.558 to 0.004.

Interactive teleoperation in the Isaac Sim runtime corroborates these results: a forty-second zero-command hold on a flat plaza drifts 0.000 m, and a twenty-second hold on a 0.341 m dune crest drifts 0.001 m. In the same runtime, the quadruped climbs 0.278 m

onto a 0.341 m dune crest without falling. The evaluation retires distance traveled per episode because one quarter of environments receive a standing command, which naturally reduces distance.

8.5. Recovery of Lateral Tracking and Motion Smoothness

Although station keeping succeeds, manual teleoperation in the Isaac Sim runtime reveals lateral veer: reverse driving drifts laterally at 0.22 m/s, and an eighteen-second forward run finishes twenty-five degrees off heading. In per-axis evaluation, rough-terrain left lateral tracking fails at 0.181 m/s against a 0.15 m/s threshold. Training rarely produces straight-line holds: uniform sampling yields negligible lateral commands in five percent of environments, while a 1.0 linear kernel dilutes marginal drift penalties.

A tracking fine-tuning stage runs for 1,000 iterations (49,152,000 environment steps in 31 minutes and 37 seconds). Command sampling reserves twenty percent for pure longitudinal motion and fifteen percent for pure lateral strafing. Tracking rewards are exponential kernels whose widths, in m/s and rad/s, set how quickly the return decays with error; the linear kernel narrows from 1.0 to 0.5, raising marginal penalties 1.6 times, and the angular width becomes 1.0. Action-rate and body-motion penalties rise to -4.0, and joint-velocity penalties reach -3.0×10^{-2} across all joints, evaluated at curriculum level 5, the difficulty row from which this stage starts.

The policy passes eleven of twelve gates. On rough terrain, left lateral tracking error improves from 0.187 to 0.118 m/s, zero-command displacement falls to 0.007 m, and driving action rates drop 7 to 20 percent. Rough left-hold survival, the count of environments out of 64 that complete the full hold, reaches 50, up from 48 but below the threshold of 56, while right-hold survival reaches 62 and exposes lateral asymmetry. In the previous stage, the offline veer metric, the worst absolute mean body-frame lateral velocity over the forward and backward holds, passes its 0.05 m/s gate at 0.046 m/s yet under-reports Isaac Sim runtime drift of 0.22 m/s by roughly a factor of five. Its reduction to 0.017 m/s on flat terrain and 0.016 m/s on rough terrain therefore provides directional evidence rather than a verified runtime bound. Isaac Sim runtime testing rejects an exponential action filter because it reduces neither lateral drift nor body roll.

8.6. Support-Relative Alignment and Symmetry

Despite improved tracking, simulation teleoperation reveals persistent roll lean: 7 to 9 degrees while driving and 4.7 degrees while standing. Measurement against world level causes this defect: on a 0.4 gradient slope, a terrain-aligned body experiences pitch deviation $\sin(\arctan 0.4) \approx 0.37$. A world-gravity penalty therefore punishes

climbing posture; reducing that penalty to -1.0 enables climbing but leaves the optimizer indifferent to standing tilt.

Resolving this conflict requires evaluation of orientation relative to local terrain. Diagonal vector cross products over four foot coordinates estimate a support plane, and tilt against that plane replaces the world-level term, which scores the horizontal component of projected gravity. Because a diagonal trot lifts paired feet in unison, diagonal orientation remains stable across the stride cycle and avoids discontinuous contact gating. Penalties of -8.0 dynamically and -10.0 while standing, for commands below 0.05 m/s, target terrain-relative tilt; world-level penalties fall to -0.5. The support plane enters only the reward, since the policy observes no foot positions. Left-right mirror data augmentation flips lateral velocities, roll and yaw rates, and commands and swaps opposite leg joints with hip-abduction negation; asymmetric leg geometry rules out fore-aft mirroring.

This stage runs for 1,000 iterations (49,152,000 environment steps in 30 minutes and 24 seconds) and passes thirteen of fifteen gates. Standing tilt falls from 4.70 to 0.79 degrees on flat terrain and from 4.44 to 0.89 degrees on rough terrain against a 1.5-degree threshold. Rough survival becomes nearly symmetric: left survival rises from 50 to 56 and right survival settles from 62 to 57 out of 64. Two gates fail. Walking roll bias reaches 2.36 degrees on flat terrain and 1.54 degrees on rough terrain, missing the 1.0-degree requirement despite eliminating unidirectional lean. Walking roll peak also misses the configured thresholds, with 6.63 degrees flat and 18.57 degrees rough against 5.0 and 8.0 degrees, because of a metric mismatch: world-level maxima determine the thresholds, but the script scores terrain-relative roll, where four-foot swing introduces support-normal discontinuities. These failures remain unresolved.

8.7. Stack Integration and Sim-to-Real Boundaries

Training runs in Isaac Lab and deployment in the Isaac Sim runtime, a separate process that shares the articulated Spot model and its actuator configuration. The autonomous stack in Isaac Sim therefore integrates the levelness-tuned policy by substituting the compiled network file into the robot controller, which preserves actuator models and gains. Runtime evaluation reveals a gait-initiation deadband: linear commands below 0.35 m/s fail to initiate stepping from rest. Forward motion requires a concurrent angular command of at least 0.04 rad/s and sustains trotting speeds between 0.18 and 0.22 m/s. The navigation stack therefore uses a linear velocity floor of 0.40 m/s, a ceiling of 0.45 m/s, and periodic zero-integral yaw dither, a small alternating yaw command with zero net rotation.

Interactive Isaac Sim trials use a 24 m×24 m dune arena that this evaluation authors with 133 mounds, slopes up to 15.9°, and heights up to 0.314 m. The measured slope result records a 0.278 m climb onto a 0.341 m dune crest. In the ramp world, autonomous navigation routes around a twelve-degree incline because the two-dimensional costmap

encodes the ramp and floor identically (Chapter 7). During teleoperation, observation shows stable ramp traversal, but the record contains no quantitative trajectory, so this remains a qualitative observation. These observations distinguish the route-selection failure from the measured dune-climb result without establishing general incline capability.

The retrained policy runs exclusively within Isaac Sim; joint faults prevent deployment on the physical Spot. Training and deployment also use different control frequencies. Training runs the policy at 50 Hz over 500 Hz physics with decimation 10, whereas the deployed runtime runs the world at 200 Hz physics with the same decimation, so the deployed policy runs at 20 Hz, since decimation fixes the control period in physics steps. Parsing the policy configuration produces the logged 0.002 s step value but does not override the runtime world timestep.

Miller et al. deploy reinforcement learning (RL) on physical Spot hardware using the Spot RL Researcher Development Kit for joint-level motor access, latency compensation, and parameter optimization on low-poly noisy terrain around flat ground [Mil+25]. An onboard proportional-derivative law converts their joint-position commands to torques. The policy in this thesis also outputs joint-position targets for an articulated proportional-derivative model, but simulation alone evaluates it. Physical rough-terrain deployment would require the unavailable joint-level hardware interface together with matched actuator and contact calibration.

9

Experiments and Evaluation

This chapter reports experimental outcomes from the designs in Chapters 4 to 8. Four distinct evidence domains provide the measurements: the physical Spot, the Webots maze, the Isaac Sim terrain worlds, and the Isaac Lab training environment. The first section defines the evaluation unit, metrics, stopping conditions, and rules for comparison. The simulation results come first because the final configuration of every layer runs only in simulation: Isaac Sim mapping, Isaac Sim exploration and navigation, and the retrained locomotion policy. The Webots maze and the physical robot follow, reporting the earlier configurations and the evidence that only real hardware provides. The final section states what the assembled runs support for each research question (RQ) in Chapter 1. Negative outcomes, rejected configurations, and run-to-run variance constrain every claim to the same extent as successful runs.

9.1. Evaluation Methodology and Metrics

This section defines every quantity before reporting results: the evidence domain, evaluation unit, metric computation, stopping conditions, and comparison rules. All later sections apply these definitions without restatement.

9.1.1. Evidence Domains and Evaluation Units

Table 4.3 defines each evidence domain, and every result retains the domain of its source run.

An episode is one continuous run of a declared stack configuration in one world. It begins when the mapper starts and ends at one declared stopping condition. The record includes the sensor configuration, map resolution, goal-scoring variant, navigator, locomotion source, and any controlled random seed. A replay iteration passes one recorded sensor stream through a fixed software revision, thereby removing trajectory variation and isolating a software change. A fixture check applies a deterministic unit test to a stored map or synthetic geometry and reports a pass count over a fixed

number of cases.

9.1.2. Metrics

The support-column count measures coverage: at 0.10 m resolution, it counts horizontal query-grid columns for which the mapper derives a valid support surface under the rule in Section 6.4. The count includes each column once, so it measures mapped extent rather than dwell time. The walkable-cell count includes only support columns that also pass the slope, step, and roughness gates, and the walkable ratio divides this count by the support-column count. Each result reports both count and ratio because a ratio alone does not reveal mapped extent. Later sections use support column and support cell for this same grid unit, reporting a map total in Webots and cells gained over an interval in Isaac Sim, and they refer to the walkable ratio as the walkable share or the support walkable ratio. The physical rows instead report disk coverage, the share of the scored disk that carries mapped floor.

The validated-publication ratio measures goal quality. Its numerator counts published goals that the independent validator accepts after applying its own gate implementation to the final three map snapshots; its denominator counts all published goals. This ratio describes geometric admissibility. Its unit of accounting is one publication event, called a goal or a waypoint depending on the section, rather than one distinct target: a held target is republished and rechecked each cycle (Section 6.4.1), so checked counts can exceed issued-goal counts within one episode. The validator is a separate node with its own implementation of the gate tests, and its record separates an accepted publication, a rejected publication, and a publication it cannot re-evaluate. Per-gate rejection shares instead divide rejections by the candidates that a gate evaluated. Goal arrival remains separate, and run completion depends on the termination logic. For termination, the evaluation reports the residual frontier-voxel count at episode end and the latch state, which indicates whether the completion condition in Section 6.5 fires.

Episode duration, in simulated or wall-clock seconds, and route length, integrated from the body-frame path, quantify cost. Defect counts within a declared evaluation mask quantify map integrity: sub-floor crossings, sub-floor mesh vertices, hole count, and maximum hole area in cells. Every result names its mask. Defect counts also lie inside a declared vertical band, the evaluation window that the run applies to the map; the primary band is the window applied by the reported run, and runs collected before the floor-relative reduction use the earlier band variant (Section 5.2.3). Counts of discrete events, including escape-state activations, route stalls, and operator interventions, together with stage update rates, describe execution. The escape rule is the navigator's stall-recovery trigger, which reacts to sustained low net displacement and latches a recovery state with a cooldown (Section 7.3). Command-tracking error, traversed distance, and achieved climb height describe locomotion. Survival counts how many of 64 environments complete the full hold. The evaluation log provides

mean action-change rate over a driving hold. Terrain-relative roll and pitch during zero command determine standing tilt, while the worst absolute mean terrain-relative roll over the driving holds determines walking roll bias.

9.1.3. Stopping Conditions and Comparison Rules

The protocol recognizes five stopping conditions: the completion latch, a fixed time or distance budget, a coverage plateau when the support count reaches a stated fraction of its final value, an operator-issued stop, and an external abort such as a hardware fault. Every reported episode names its stopping condition because a truncated budget and a latch do not represent equivalent outcomes. A latch is a completion request from the state machine in Section 6.5: the chapter reports whether the supervisory watchdog accepted it, and a rejected latch does not count as a completed episode. Watchdog outcomes, such as the route-loop verdict and the enforced cap, are named as such in the termination column of the episode tables.

Two runs support direct numerical comparison only when they share the environment, world and seed, sensor configuration, map resolution, evaluation mask, and stopping condition, while differing in the single factor under study. The chapter reports results from different environments side by side without subtracting or ranking them because their sensor, contact, and timing models differ. When several episodes use one configuration, the evaluation reports their spread rather than a best case. Development runs form a separate class because their configurations changed during collection; they demonstrate failure modes but do not support performance ordering.

The teleoperated Webots maze episode serves as a reference rather than a competing method. It bounds the coverage available from the same sensing and mapping path when an operator supplies goal selection and route execution. A percentage of that reference therefore expresses how much of a manually attainable map an autonomous configuration produces. Relative figures name their denominator: a reference episode, a control run, or the run's own final value. A control run repeats the protocol with the factor under study removed, so percentages of control in Table 9.3 divide by the support reached by the sealing-disabled episode in the same arena. An algorithmic baseline instead runs an alternative navigation configuration under the same protocol.

Table 9.1 assigns each research question (RQ) to the evidence domains and metrics that answer it. The final section collects the boundaries that limit those answers.

9.2. Isaac Sim Mapping Results

The evaluation begins in simulation because Isaac Sim holds the final configuration of every layer. This section addresses the first and second research questions for that configuration: it asks whether the camera-only map preserves walkable dune crests and real obstacles well enough for the frontier layer to use. Section 5.2.3 describes the mapper rule under test.

Table 9.1: Research questions with their evidence domains and metrics; the RQ column follows the question numbering in Chapter 1, and the metrics named here are those defined in Section 9.1.2.

RQ	Evidence domain	Metrics
1	Physical Spot; Webots maze; Isaac Sim terrain	Loop-closure counts, support-column count, map-integrity defect counts, stage update rates
2	Physical Spot walking loops; Webots maze drives; Isaac Sim range comparison	Walkable ratio with its support count, sub-floor crossings and vertices, hole count and maximum hole area, support-height error
3	Replay of physical recordings; Isaac Sim ramps, dunes, and walled arena; fixture checks	Validated-publication ratio, per-gate rejection shares, residual frontier voxels, latch state
4	Webots maze; physical Spot bring-up and exploration; Isaac Sim terrain	Support-column count, episode duration, route length, event counts for stalls, escapes, and interventions
5	All four domains, compared by declared scope rather than by score	Presence or absence of an executed experiment per claim, with the metrics of the owning question

The simulation figures in this chapter are top-down maps, so Figure 9.1 first shows each simulated world in the same top-down form, as it was authored. Panel (a) is the large dune field, panel (b) the small walled arena, and panel (c) the floor plan of the Webots maze; the perspective views of the same worlds are Figure 4.3 and Figure 4.2. The terrain colour scale matches the support-height scale of the later maps, so a well-built map should reproduce the dunes of panel (a) wherever the robot has looked. The lettered boxes mark the regions that later figures enlarge, and each later caption names its world and region. The later maps also draw the authored dune footprint as a dashed outline, which separates terrain the map has missed from terrain that does not exist.

Figure 9.2 shows the same three worlds as complete three-dimensional maps in RViz. The meshes come from the same world definitions and are coloured by height in the manner of the nvblox mesh display of Figure 9.8, so they show what a complete reconstruction of each world would contain.

9.2.1. Crest Correction in Isaac Sim

One Isaac Sim session runs the crest experiment in the large dune field of Figure 4.3(b). A closed-loop waypoint follower drives the robot along the same planned route over two dune massifs for every configuration, and only the mapper restarts between configurations. The evaluation scores the grid and support surface against the authored height field. A trajectory-independent common mask, the cells within 2.5 m of every run's walked path, covers 21,534 cells, including 1,001 crest cells whose authored height is at least 0.20 m. The camera-only baseline marks 0.1 % of crest cells free, 26.8 % occupied, and 73.1 % unknown. The adopted known-floor rule with the step-gated crest guard marks 100 % of crest cells free, reduces flat-ground false occupied cells

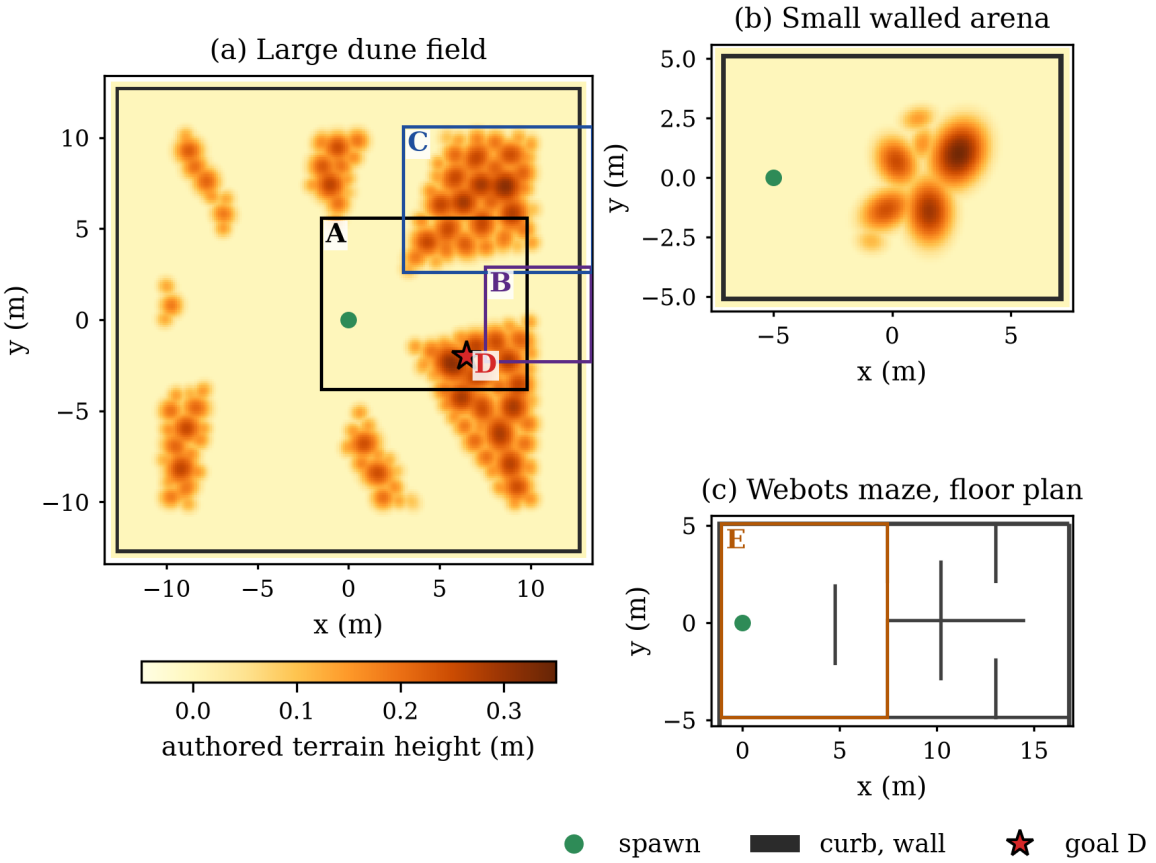
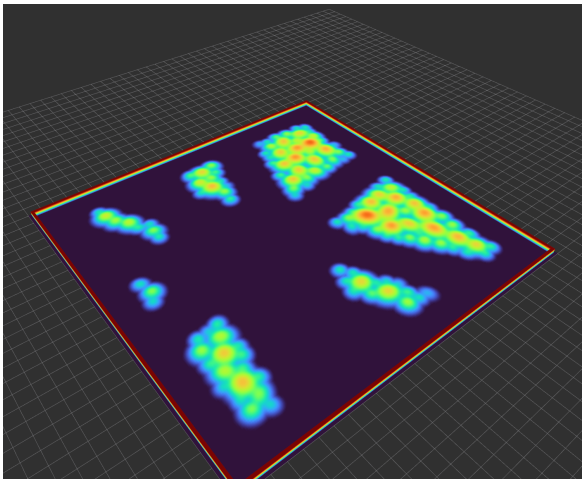
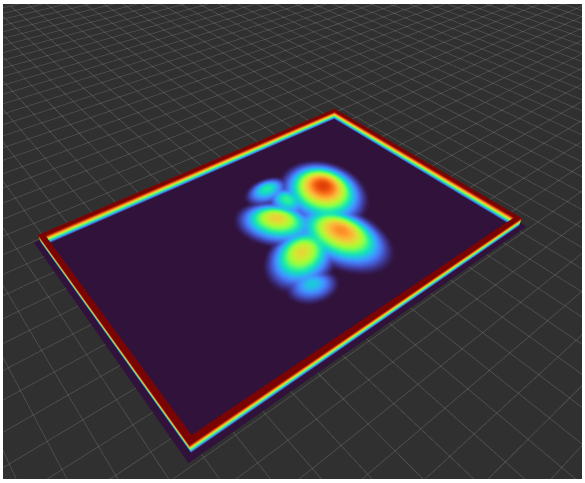


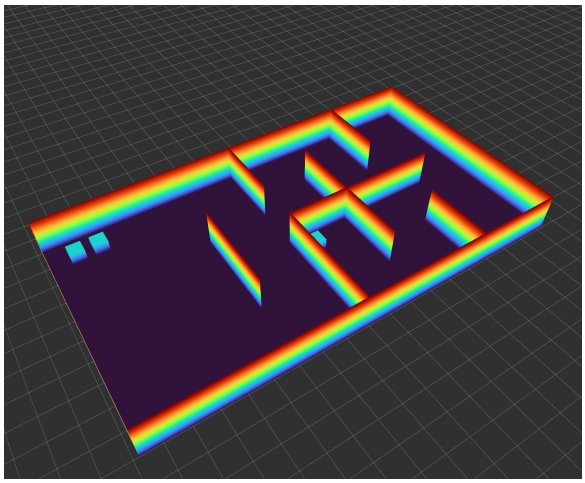
Figure 9.1: Top-down reference views of the simulated worlds, drawn from the authored world definitions in the map frame of the episodes. (a) Large dune field with its 0.40 m boundary curb; box A is the crest route of Figure 9.3(a,b), box B the curb route of Figure 9.3(c,d), box C the goal-candidate view of Figure 9.4, and D the goal of the locomotion run in Figure 9.7; Figure 9.5 covers the whole field. (b) Small walled arena of Figure 9.6, with the flat spawn pad on the left. (c) Webots maze walls from the world file; box E is the start room of Figure 9.8, and Figures 9.10 and 9.12 cover the whole maze. Green dots mark the spawn.



(a) Large dune field



(b) Small walled arena



(c) Webots maze

Figure 9.2: Complete three-dimensional maps of the simulated worlds in RViz, built from the authored world definitions rather than from a reconstruction: (a) the large dune field with its 0.40 m boundary curb, (b) the small walled arena, and (c) the Webots maze with its walls and its 0.5 m image stands. As in the nvblox mesh display, colour encodes height from dark purple at floor level to red at the highest surface of each world, which is 0.40 m in (a) and (b) and 1.70 m in (c). Grid cells are 1 m wide.

from 55 to 0, and leaves support-height RMS unchanged at 0.058 m. The obstacle control explains the gate: a second route approaches the 0.40 m boundary curb of the same world, and on that curb the baseline and step-gated guard keep all 360 scored curb cells occupied, while the ungated guard frees 140.

The corridor mask reports 75.6 % crest cells free for the adopted configuration because simulated locomotion stalls after 15.4 m, while other configurations walk up to 20.2 m; this mask mixes mapper output with route coverage. The rejected long-range scanner configuration also marks 100 % of crest cells free, at the range-dependent support-height and false-occupancy cost reported in Section 5.2.3.

Figure 9.3 shows these runs as top-down planning grids, rendered from the recorded maps. Before the correction, the two dune massifs south-east of the route appear as unknown and occupied patches: the downward cameras leave the air above each convex crest unobserved, so the reducer cannot confirm the column as free and the planner would detour around walkable ground. With the known-floor rule and the step gate, the same crest cells read free while the flat ground stays clean. The lower panels show why the gate is needed. Without it, the support height of the narrow curb top spreads into the neighbouring columns, lifts their obstacle band above the curb, and opens a gap in a real obstacle; with the gate, the whole scored curb face remains occupied.

All crest evidence comes from one authored dune world in simulation. It includes the scored crest band and one 0.40 m curb control, with a maximum authored slope of 14.5°, and lacks a physical rerun. The runs show that the corrected camera-only map keeps walkable crests available to the frontier layer without erasing the one real obstacle in the world.

9.3. Isaac Sim Exploration and Navigation Results

Isaac Sim runs closed-loop episodes with autonomous goal selection and motion, including one large-dune episode that the operator stops early. These runs address the third and fourth research questions on open uneven terrain and constrain part of the answer to the fifth. The stack uses the articulated Spot model with its actuator and gain models, five body-mounted depth cameras, camera-only volumetric mapping with floor-relative projection, the walkable-goal filter, the mini-global relay with predictive local control, and the selected levelness-tuned locomotion policy (Chapters 5 to 8). Two authored worlds provide the terrain, shown top-down in Figure 9.1(a) and (b): a 24 m × 24 m dune arena with 133 mounds, slopes to 15.9°, and crest heights to 0.314 m; and a walled 140 m^2 dune arena for shorter termination episodes. Each row in Table 9.2 requires interpretation within its own world and duration because the arenas differ in scale and episode length differs by more than an order of magnitude.

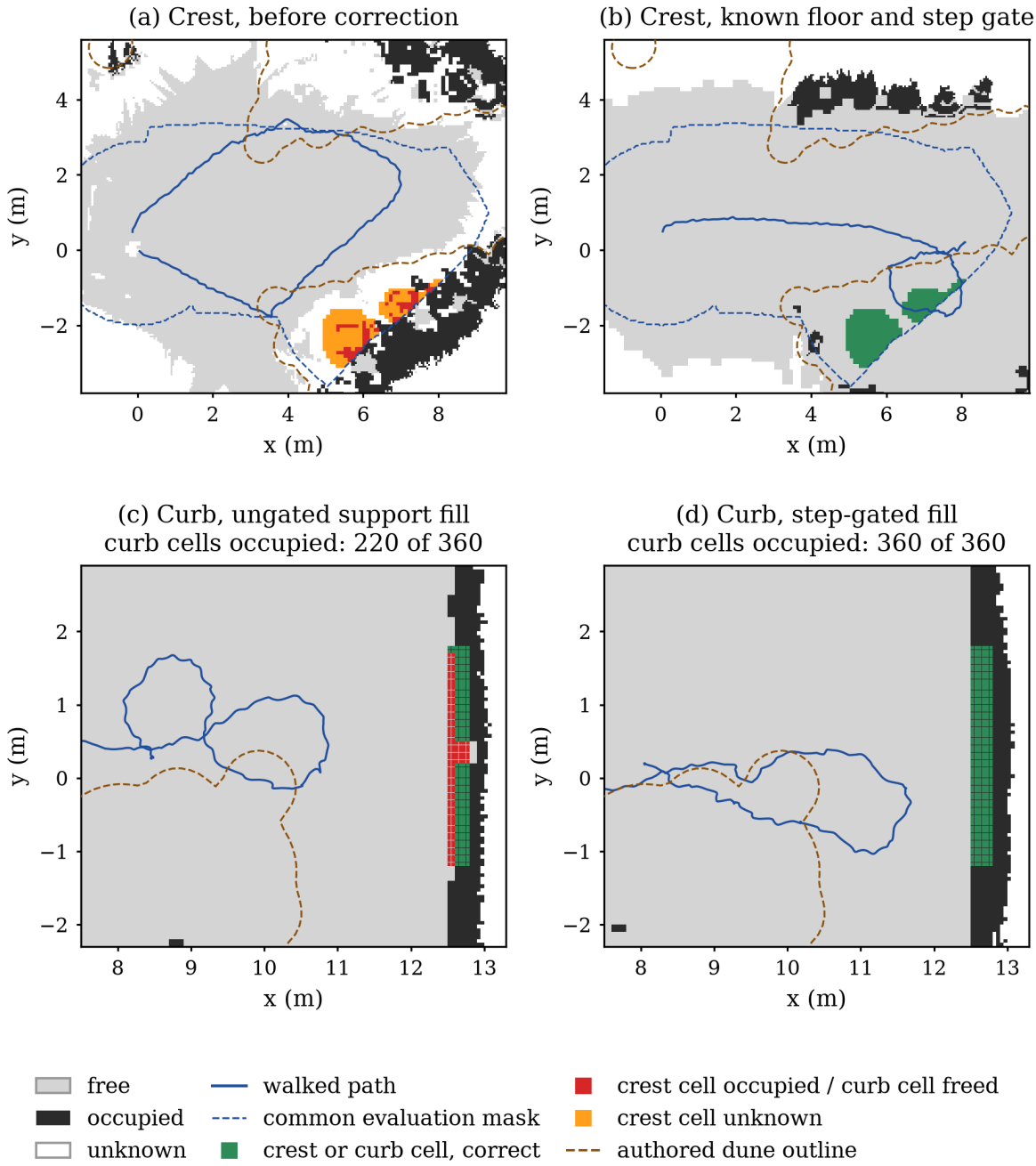


Figure 9.3: Top-down planning grids from the Isaac Sim crest experiment in the large dune field, rendered from the recorded runs: panels (a) and (b) show region A and panels (c) and (d) region B of Figure 9.1(a). (a) and (b) show the dune route before and after the known-floor rule with the step-gated crest guard; crest cells inside the common evaluation mask are coloured by their classification. (c) and (d) show the curb route; the scored cells of the 0.40 m boundary curb are green when occupied and red when the ungated fill frees them. Grey is free space, black is occupied, white is unknown, and the dashed brown line outlines the authored dunes.

Table 9.2: Isaac Sim autonomous episodes with the integrated stack. The goal columns count distinct goals issued and arrived, while the validator column counts publication events rechecked against map snapshots; its entries sum to the 12 396 checks reported in Section 9.3.2.

Configuration	Arena	Simulated duration	Path	Goals issued / arrived	Validator-accepted publications
Retained explorer, baseline settings	Large dune	1160 s	221.0 m	129 / 105	1,170 of 1,170
Retained explorer, tuned hold and decay	Large dune	3605 s	649.3 m	330 / 245	3,620 of 3,620
Ground-frontier variant, stopped early	Large dune	1220 s of 3600 s planned	405.4 m	97 / 40	1,212 of 1,212
Seal disabled, termination control	Small dune	5826.8 s	1386.4 m	485 / 388	5,849 of 5,849
First seal rule	Small dune	287.0 s	0.58 m	1 / 1	2 of 2
Revised seal rule	Small dune	1267.9 s	57.1 m	26 / 20	242 of 242
Revised seal with corrected span guard	Small dune	1062.1 s	61.6 m	27 / 16	301 of 301

9.3.1. Crest Observation and Goal Placement

The truncated dune episode measures the goal-layer effect of the crest defect before the mapper correction in Section 9.2.1. By 840 s, the support surface reconstructs crest ground to 0.45 m, but 103 of 104 unique goals lie at support heights no greater than 0.10 m; none lie in the 0.20 m to 0.45 m crest band. At selection, one-meter discs around issued goals contain 35.1 % unknown area and retain 19.7 % after approach. Across 62 footprint samples, unknown cells occupy a mean 13.6 % beneath the standing robot. The crest band therefore remains absent from every tested explorer's reachable set; candidate ranking alone cannot place goals there.

After adoption of the correction, the termination-control episode in the small dune arena places 9.0 % of its goals on authored ground in the 0.20 m to 0.34 m crest band and 24.2 % on ground at or above 0.10 m. The corrected mapper therefore demonstrates goal placement above crest height in simulation; the earlier configuration does not.

9.3.2. Walkable Goals and Route Execution

Across every Isaac Sim episode, the independent validator re-checks each published goal with its own gate implementation and records no validation failure or unverifiable publication among 12,396 checks. This record concerns only published-target admissibility. Arrival counts in Table 9.2 and the latch states below remain separate outcomes, and several episodes with complete validator records arrive at well under nine tenths of their goals. In these open-terrain episodes, only the footprint gate rejects candidates,

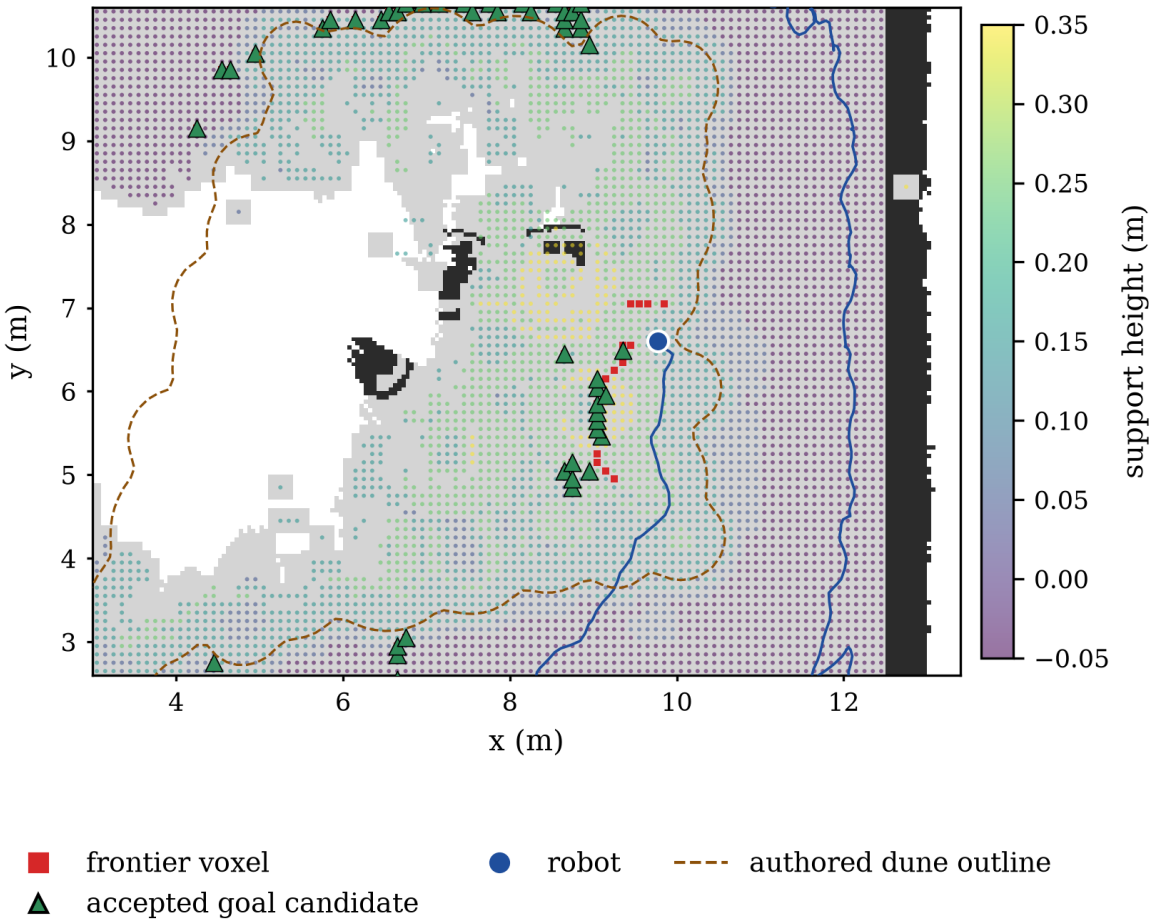


Figure 9.4: Goal candidates around the robot in region C of the large dune field (Figure 9.1(a)) in a representative episode. Points show the mapped support surface coloured by height, red squares are frontier voxels, green triangles are candidates that passed the walkable-goal gates, and the dashed brown line outlines the authored dunes. The background grid uses the colours of Figure 9.3.

with shares between 17.8 % and 33.7 %; the reachability flood and headroom test reject none.

Figure 9.4 shows how the filter places goals in a representative large-dune episode of the same configuration; the view is region C of Figure 9.1(a), on the eastern side of the north-eastern dune massif. The robot stands on the eastern lane beside that massif; the frontier voxels trace the edge of the observed crest, and the accepted candidates lie a short distance behind that edge, inside the authored dune outline, on dune ground whose support surface is already mapped. None of the candidates sits on a frontier voxel itself, because a frontier voxel marks the boundary to unobserved space rather than ground the robot can stand on.

Five checks in the large dune arena verify route execution before the episodes. Transits over 3 m to 5 m settle with 0.24 m to 0.28 m error. A previously unfinished uphill flank goal completes in 10.4 s with 0.23 m error. An induced stall triggers a watchdog escape and explicit failure inside the retry cap rather than a silent deadlock. The

publisher audit finds exactly one publisher on each of nine navigation topics, and 300 zero commands produce 0 m measured drift. All episodes use the command floor, ceiling, and yaw dither required by the selected policy's gait deadband, as derived in Section 9.4.

9.3.3. Dune Episodes

The baseline dune episode gains 30,380 support cells over 221.0 m, or 137.5 support cells per meter walked (cells gained divided by route length over the same interval), arrives at 105 of 129 goals, and recovers from its single hard goal failure without intervention through the watchdog, relay resend, and explorer blacklist chain. The tuned episode gains 57,958 support cells over 649.3 m and extends goal coverage from a maximum of 4.8 m to 11.8 m along the previously unvisited axis. Its full-run efficiency of 89.3 cells per meter reflects arena saturation by roughly 35 simulated minutes and a 570 s immobilization against the authored boundary curb, from which it recovers unattended. A matched cut compares two episodes over the same elapsed interval, taken at the shorter of the two durations, so that duration does not account for the difference. At a 1165 s cut matching the baseline duration, the tuned episode has gained 24,326 cells at 107.0 per meter and arrived at 131 of 154 goals.

The ground-frontier variant changes one variable from the tuned episode, and the operator stops it at 1220 s of the planned 3600 s. At the matched cut, it gains 54,983 support cells (142.7 per meter) against 33,841 (138.1 per meter); arrival reaches 41 % against 87 %.

Accepted candidate markers lie within 0.5 m of a frontier voxel in 96.2 % of 6,893 samples. This comparison cannot support a final ordering of the two candidate generators: no run provides the retained explorer's marker control, the first curb incident in the tuned episode occurs later than the truncated comparison, and the curb-standoff question remains unanswered. The evidence does show that the replacement generates geometrically frontier-adjacent candidates and accumulates support faster over the matched interval, but with a lower arrival share.

Figure 9.5 shows what such episodes produce over roughly their first twenty minutes; it covers the whole large dune field and can be compared directly with the authored terrain of Figure 9.1(a), which uses the same colour scale. The background is the mapped support surface coloured by height, so the dune massifs appear as the orange regions and the flat lanes as the pale ground between them, and the dashed outlines mark the authored dunes. The retained explorer first sweeps the flat perimeter lane and then turns inward across the eastern massifs; a large central region is still unobserved at the end of the shown interval, and occupied cells remain where a massif borders that unobserved region. The ground-frontier variant instead sends the robot repeatedly across the northern massifs, so its path stays on dune ground for a larger share of the interval, and the outlines in its panel show the southern dunes it has not yet reached.

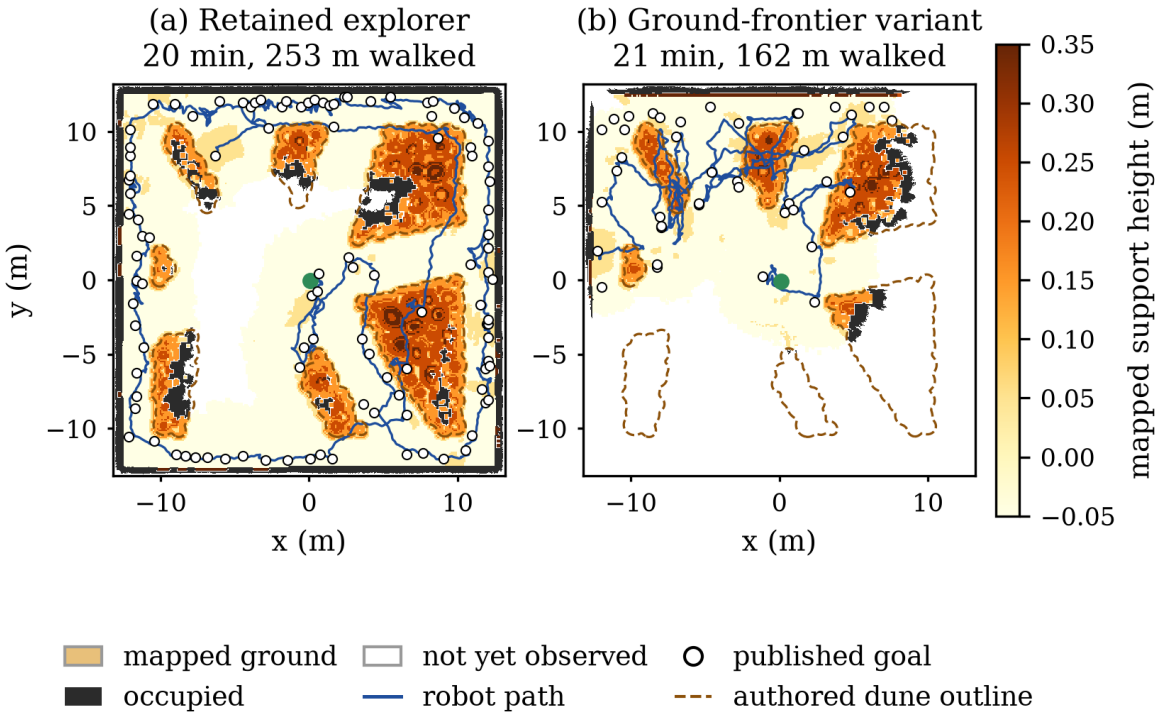


Figure 9.5: Maps built by representative episodes in the large dune field (Figure 9.1(a)) after about twenty simulated minutes: (a) the retained explorer and (b) the ground-frontier variant. Colour encodes the mapped support height on the scale of the reference view, black cells are occupied, white is not yet observed, the line is the robot path, circles are published goals, and the dashed brown line outlines the authored dunes.

In these particular episodes, the variant also walks a shorter route and maps a smaller area than the retained explorer, the reverse of the measured matched-cut comparison above. The figure shows representative episodes rather than the measured episodes of Table 9.2; with one episode per configuration on each side, the two comparisons bound route variance and show where each generator sends the robot, but they do not rank the generators.

9.3.4. Frontier Sealing and Termination

Frontier sealing is a termination rule, not a claim that unknown space is free. It removes frontier voxels above an interior mapped floor or a supported local crest from the unexplored set, while preserving the forward frontier ring that represents genuinely unobserved ground.

With sealing disabled, the termination-control episode reaches 99.5 % of its final support within 517 s and continues to 5826.8 s and 1386.4 m. After the plateau, 88.3 % of accepted goals add only 33 support cells over 1228.4 m. Interior voxels account for 1,042 of 1,044 residual frontiers, and 83.6 % stand over flat mapped ground rather than crests. A stopping rule must remove this population.

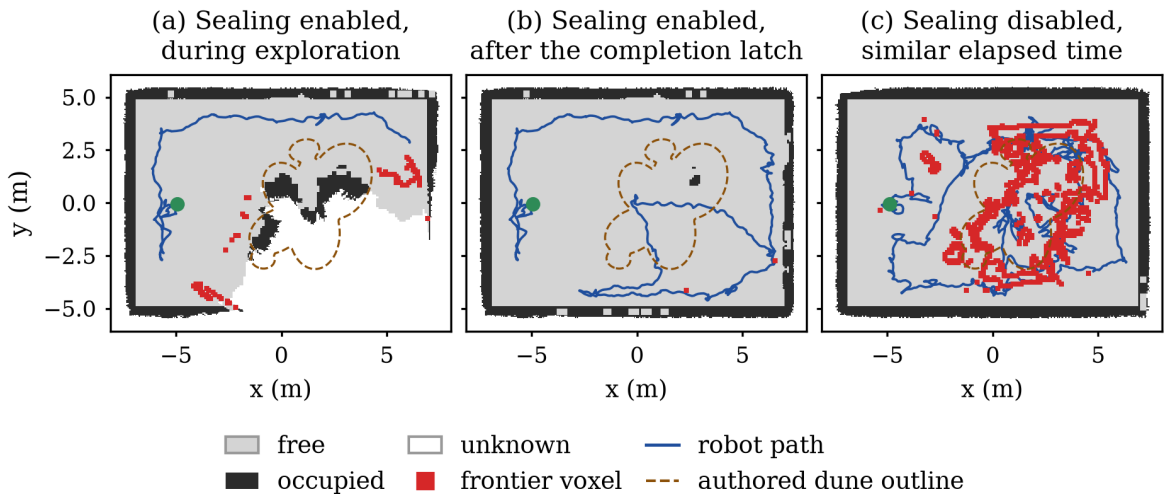


Figure 9.6: Planning grids and frontier voxels in representative episodes in the small walled arena (Figure 9.1(b)): (a) sealing enabled during exploration, (b) sealing enabled after the completion latch, and (c) sealing disabled at a similar elapsed time. Red squares are frontier voxels and the dashed brown line outlines the authored dunes. The quantitative results are those of Table 9.3.

The first seal rule removes the forward frontier ring together with interior air: the robot walks 0.58 m, support reaches 22.5 % of the control, 45.1 % of the arena remains unknown, and the mixed residual frontier population blocks the completion latch permanently. The revised column-gated rule retains 98.6 % of control support after 57.1 m with three residual frontier voxels, but it does not latch because the stability guard demands exact coverage of a jittered history window. After correction of that guard, a further episode retains 96.9 % of control support after 61.6 m, leaves one frontier voxel, and latches near 740 s.

Table 9.3 collects these results. Figure 9.6 shows the same effect in the planning grid of representative episodes in the small walled arena of Figure 9.1(b), whose dune massif lies inside the dashed outline. During exploration with sealing enabled, the frontier voxels lie at the edge of the observed area, where unobserved ground genuinely remains. After the completion latch, the arena is mapped and only a few isolated voxels are left. With sealing disabled, at a similar elapsed time and with the arena equally mapped, a dense population of frontier voxels remains above the interior floor that the robot has already crossed; these voxels border the unobserved air above the floor, which the low cameras never see, so a stopping rule that waits for an empty frontier set never fires.

Each seal result comes from one episode, one world, and one seed, so the difference between the two revised-rule episodes bounds route variance rather than isolating the guard correction. Sealing remains disabled by default, and the corrected guard exists only in a side variant of the explorer. No physical-robot episode records a latch.

Table 9.3: Isaac Sim frontier-sealing results. Support percentages divide by the control support, the value reached by the sealing-disabled episode in the same arena, except in that row, which is relative to its own final support. A dash marks a row without a validator entry in this comparison, so it denotes a missing record rather than zero failures.

Configuration	Support retained	Path (m)	Residual frontiers	Validator	Latch
Seal disabled	99.5% of final	1386.4	1044	–	No
First rule	22.5% of control	0.58	mixed population	–	No
Revised column-gated rule	98.6% of control	57.1	3	242/242	No
Revised rule with guard correction	96.9% of control	61.6	1	301/301	Yes, near 740 s

9.3.5. Ramp Preference and Evidence Boundary

In the ramp world, the floor-relative projection classifies the 12° incline as traversable floor: all 13 sampled centreline cells remain free, and support height rises to 0.63 m near the landing. Given goals beyond the crest, the planner routes across adjacent flat ground on both attempts and terminates at ground level. No evaluated configuration achieves autonomous ramp ascent. The only measured climb comes from the interactive dune trials in Section 9.4 and does not extend to ramp route selection.

All results in this section come from simulation with the declared actuator, contact, and sensor models. No trial exercises the closed-loop stack on the physical robot or reruns the final floor-relative navigation configuration on hardware. Isaac Sim alone validates frontier sealing, so these measurements support no transfer claim.

9.4. Locomotion Evaluation on Uneven Terrain

Three non-interchangeable instruments produce the locomotion evidence that supports the velocity-command portions of the fourth and fifth research questions.

9.4.1. Instruments

The per-axis harness runs in the training environment: 64 environments, one command axis held for 950 steps after 100 discarded steps, resampling disabled, gates at 0.10 m zero-command displacement, 0.15 m s^{-1} linear error, and 0.15 rad s^{-1} yaw error. Survival counts the environments out of 64 that complete a hold, and the later configurations share one common evaluation task. Interactive runtime checks substitute a trained network into the simulator runtime's own loader and measure scripted pulses or teleoperation from odometry, generally once per condition, while full-stack observations add the navigation graph so that the controller issues the velocity. Gate counts are single-seed results against internal thresholds, and each harness defines its own gate inventory, so a passed fraction applies only within the stage that reports it.

Table 9.4: Representative per-axis metrics before and after each locomotion correction. The first group reports seven per-axis gates for each terrain condition under that stage's harness, and the later groups report the stage totals cited in the text under the common evaluation task, so passed fractions from different groups are not comparable.

Quantity	Criterion	Before	After
First rough-terrain to station-keeping correction, that stage's own harness			
Zero-command hold displacement, flat	at most 0.10 m	12.533 m	0.022 m
Per-axis gates passed, flat	7 of 7	3 of 7	7 of 7
Per-axis gates passed, rough	7 of 7	2 of 7	6 of 7
Station-keeping to tracking correction, common evaluation task			
Left lateral tracking error, rough	at most $0.15\,\mathrm{m\,s^{-1}}$	0.187	0.118
Left hold survival, rough	at least 56 of 64	48 of 64	50 of 64
Action rate, rough driving holds	no increase	0.947 to 1.252	0.812 to 1.071
Tracking to levelness correction, common evaluation task			
Standing tilt magnitude, flat and rough	at most 1.5°	4.70° and 4.44°	0.79° and 0.89°
Walking roll bias, worst flat and rough	at most 1.0°	11.68° and 12.37°	2.36° and 1.54°
Left and right hold survival, rough	reported	50 and 62 of 64	56 and 57 of 64

9.4.2. Correction Sequence

The simulator supplies a blind policy trained on flat ground with 0.02 m to 0.05 m noise and no terrain curriculum. Under four diagnostic conditions, including one spawn on the 12° wedge, it neither enters nor remains on the slope and does not self-right (Section 4.5). The first retrained configuration crosses relief but drifts under zero command: 12.63 m over 40 s in the runtime without a command publisher and 12.533 m over a 19 s harness hold. Distance traveled per episode, then the active metric, scores this defect as success. A reward discontinuity causes the behavior: the air-time, gait, and joint-position terms share a gate that activates under a nonzero command or body speed above $0.5\,\mathrm{m\,s^{-1}}$, and drift settles just above that threshold. The replacement harness first reproduces this known defect, thereby validating its detection.

Table 9.4 groups rows by configuration pair because the earliest attitude and symmetry values belong to the tracking configuration. The station-keeping correction replaces disjunctive gates with command-only conditions and raises the standing-environment proportion from 0.10 to 0.25. Under the later common task, that network passes 5 of 7 gates on rough terrain and records backward tracking error of 0.159. Rough-terrain forward-hold distance falls 6 % to 13.332 m.

Lateral behavior separates the harness from the runtime. The harness veer statistic passes its $0.05\,\mathrm{m\,s^{-1}}$ gate for a configuration that later drifts about $0.22\,\mathrm{m\,s^{-1}}$ sideways in the runtime and finishes an 18 s open-loop run about 25° off the commanded heading. This roughly fivefold under-report makes later reductions directional evidence rather than a verified bound. The tracking correction passes 11 of 12 gates, while target survival moves only from 48 to 50 of 64 against a threshold of 56. Right-hold survival rises from 55 to 62 of 64, which identifies left-right asymmetry as one plausible shared

cause.

The levelness correction treats attitude and asymmetry together and passes 13 of 15 gates. The evaluation scores tilt against a support plane through the four foot positions rather than world level, so it does not penalize a slope-aligned body for the pitch that climbing requires. A separate world-level attitude penalty still charges that pitch, and the support plane enters only the reward, leaving the observation blind. Standing tilt magnitude and worst walking roll bias both fall, as Table 9.4 records, but the correction reduces rather than removes the lean: roll biases of 2.36° and 1.54° still fail the 1.0° threshold. Roll peak provides the second failed gate under an unsound statistic: world-level maxima define its thresholds, while the evaluation scores a terrain-relative extremum. The record retains that failure. Symmetry indicators disagree: the rough survival gap closes from twelve environments to one, and the mirror-consistency measure falls 57 %, while a probe of the exported networks remains flat.

9.4.3. Integration, Slope Results, and Boundaries

Integration exposes a gait-initiation deadband that the per-axis gates do not capture. A purely linear command never starts a gait from rest: $0.30\,\mathrm{m\,s^{-1}}$ produces 0.00 m in 12 s, whereas the same magnitude with $0.04\,\mathrm{rad\,s^{-1}}$ of yaw produces $0.18\,\mathrm{m\,s^{-1}}$. A speed ladder places the threshold between $0.35\,\mathrm{m\,s^{-1}}$ and $0.40\,\mathrm{m\,s^{-1}}$. The stack therefore applies a linear floor of $0.40\,\mathrm{m\,s^{-1}}$, a cap of $0.45\,\mathrm{m\,s^{-1}}$, and yaw dither of at least $0.06\,\mathrm{rad\,s^{-1}}$ with zero net integral. At the raised floor, the previously incomplete uphill goal in Section 9.3.2 completes; two attempts with a $0.25\,\mathrm{m\,s^{-1}}$ floor fail to finish within 100 s. Repeated drive paths range from 15.4 m to 20.2 m, and uphill starts still stall intermittently.

The station-keeping configuration in the dune arena provides the only numeric slope result for a retrained policy; the authored arena reaches a maximum local slope of 15.9°. One closed-loop drive reaches a 0.341 m crest with 0.38 m final error in 20 s, raises body height by 0.278 m at a maximum roll of 8.7° without falling, and then holds the crest for 20 s. This result covers one crest and one run. No configuration achieves autonomous ramp ascent: for a goal beyond the crest, the robot routes around the ramp twice because one two-dimensional costmap encodes the ramp and adjacent floor identically as free (Chapter 7). The teleoperated ramp traversal has no quantitative trajectory and therefore remains a qualitative observation.

Figure 9.7 shows the selected policy on the same kind of terrain in a representative run: given goal D of Figure 9.1(a), on the south-eastern dune massif of the large dune field, the robot walks from the flat spawn plaza, steps onto the dune flank, and stands on the crest, all under the velocity commands of the navigation chain. The camera looks back from beyond that massif towards the spawn. The frames are qualitative; they show the gait on relief but carry no measurement.

Two boundaries close this contribution. Deployment uses a 20 Hz control cadence:

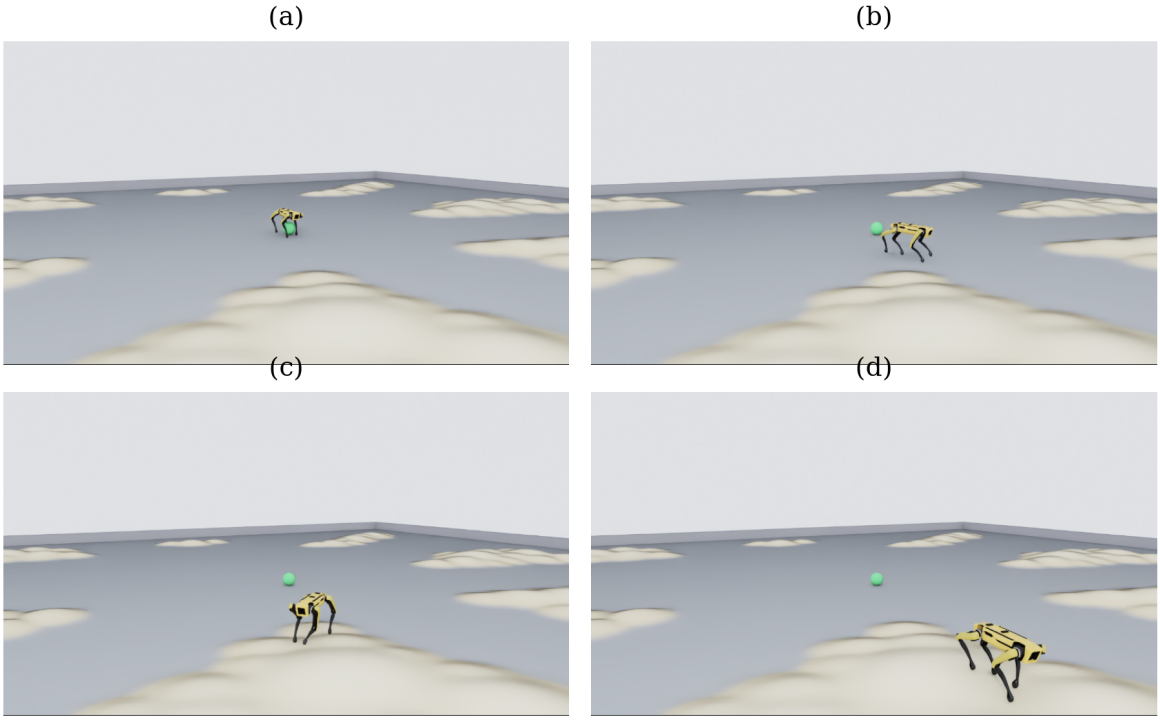


Figure 9.7: Simulator frames of the selected locomotion policy walking towards goal D of Figure 9.1(a) in the large dune field (perspective view in Figure 4.3(b)): (a) at the spawn, (b) walking out across the plaza, (c) stepping onto the dune flank, and (d) standing on the crest. The green sphere marks the spawn point.

the runtime initializes world physics at 200 Hz with decimation 10, while training evaluates the network at 50 Hz over a 500 Hz physics step with the same decimation. The 0.002 s step parsed from the training log does not govern the active runtime world. Documentation for the supplied policy asset identifies sensitivity to this setting. No retrained configuration runs on the physical robot after joint faults end access, so every figure here remains a simulator measurement. The retuned rewards also depart from the actuator-validated configuration supplied with the asset (Table 4.3).

9.5. Webots Maze Evaluation

The Webots maze [Mic04] of Figure 4.2 provides the structured indoor world for measuring exploration coverage, goal admissibility, and route execution, thereby addressing the third and fourth research questions in an enclosed environment. The section opens with two mapping results that rely on the exact robot pose that the Webots supervisor provides: the timing isolation and the sensing-configuration comparison, which the physical walking-loop trial complements. Every exploration episode uses the cameras-only volumetric mapping configuration [Mil+24] with five body depth streams, graph odometry, the exploration layer in Chapter 6, and the walkable-goal filter in Section 6.4. Each episode starts from a full cold restart of the graph. The comparison in Section 9.5.4 varies the navigation link. Explorer settings, validator activation, and the linear speed cap also differ between the experiment series where noted, so the analysis groups episodes by series rather than pooling them.

9.5.1. Timing Isolation in Webots

The Webots driver derives published odometry from the supervisor pose, so this environment provides an odometry oracle and separates map damage from pose error. A 90 s probe against the independent simulated positioning and inertial topics records millimetre-scale position error and 0.155° attitude RMS, excluding simulated odometry drift as the source of the observed floor defects.

Publish-time offsets range from −32 to +160 ms for the rotating scanner and reach +256 ms for a depth stream, so no constant offset represents the error. Stamping each message when its buffer changes reduces raw scanner-floor motion from 9.95 cm to 0.097 cm peak to peak.

Corrected timing does not restore map integrity. In the comparable 180 s trial, the support walkable ratio falls from 89.9 % to 42.9 %, sub-floor crossings in the primary band rise from 0 to 1,167, and mesh vertices in that band rise from 14 to 24,543. Per-cloud best-fit offsets have a median of 0 ms. No residual timing term remains, which implicates projective integration of geometry accumulated during motion as the source of the remaining damage.

Table 9.5: Mapping outcome by sensing configuration.

Configuration	Environment	Floor quality	Defect metric
Fused scanner and cameras	Webots, 180 s	Walkable 79.2 % to 30.7 %	367 to 4,584 mesh vertices
Scanner only	Webots, 180 s	Walkable 27.5 % to 51.1 %	368 to 2,146 mesh vertices
Cameras only	Webots, 180 s	Walkable 97.3 % to 94.5 %	0 to 0 mesh vertices
Fused scanner and cameras	Physical loop	88.9 % disk coverage	46 holes, largest 52 cells
Cameras only	Physical loop	99.6 % disk coverage	3 holes, largest 4 cells

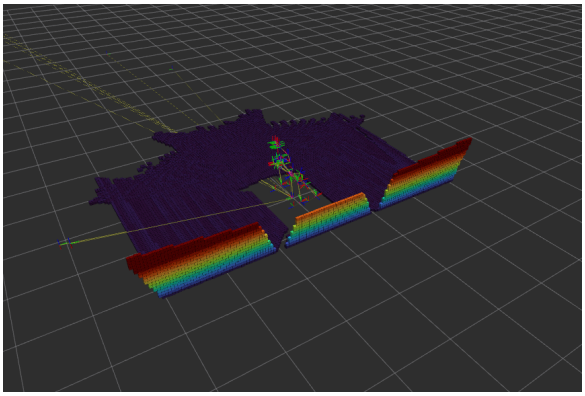
9.5.2. Sensing Configuration Comparison

Each Webots episode starts with a fresh simulation and map and a 60 s stationary warm-up, then follows the same 180 s scripted drive with yaw bouts of at most 18 s separated by pauses. For each physical attempt, the operator drives the walking loops from a fresh map, and the evaluation scores the mesh inside a 2.5 m-radius disk around the trajectory. Table 9.5 reports stationary and moving figures for each Webots configuration and the disk score for each physical configuration.

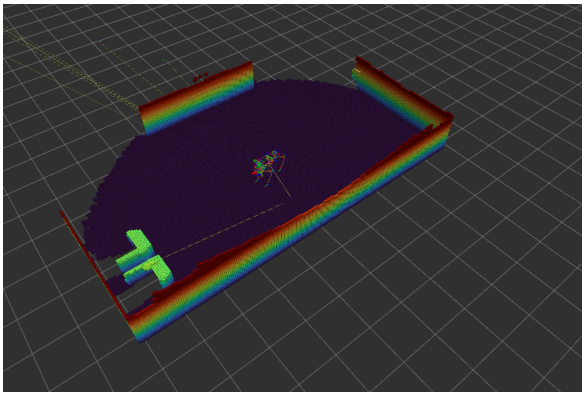
Figure 9.8 shows the maps behind the Webots rows. All six views come from the start room of the maze in Figure 4.2, which is box E of the floor plan in Figure 9.1(c), captured in RViz after the warm-up and after the drive; mesh colour encodes height, so dark purple is floor level and the warm colours are walls. The cameras-only map keeps one continuous floor after the drive. The scanner-only map starts as a sparse ring of wall returns and fills the floor unevenly. The fused map develops the strongest floor artefacts during motion, visible as the bright blue patches in its after-drive view, which sit above the true floor height. The images show the defects that the walkable ratio and sub-floor counts in Table 9.5 measure.

Each row admits interpretation only within its own environment; no comparison combines the Webots and physical rows. In Webots, the scanner-only configuration neither reaches the camera-assisted stationary coverage nor removes the motion-bound phantom bands. The protocol omits the conditional leg-sensing episode because it requires a stable scanner-only map. In the physical trial, the fused map also contains a visibly duplicated floor surface. The deployed configuration prioritizes floor completeness inside the scored region, absence of sub-floor geometry, and retention of long-range returns for closure. Camera-only dense mapping meets the first two criteria in both tested environments, while the rotating scanner supplies the third. The deployed stack therefore uses depth cameras for the dense near-field volume and retains the scanner for scan refinement, geometric closure, and long-range sensing.

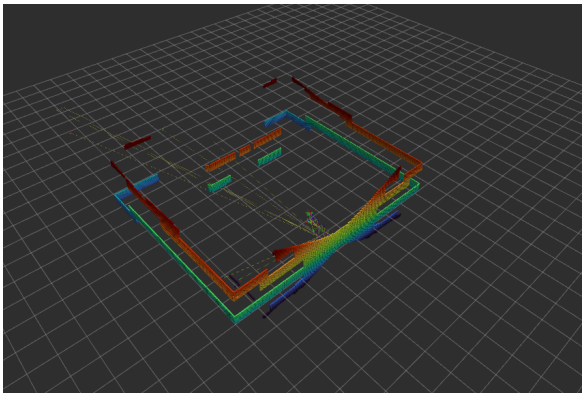
Platform rotation trials start from a fresh map and record 68 floating voxels at 0.4 rad s^{-1}



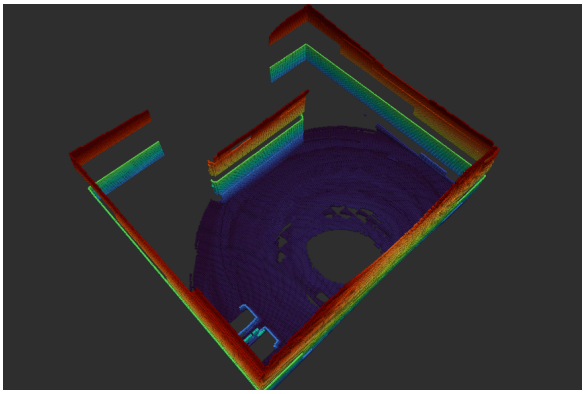
(a) Cameras only, after warm-up



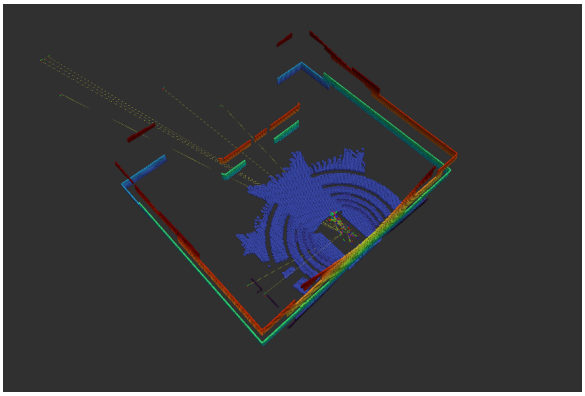
(b) Cameras only, after drive



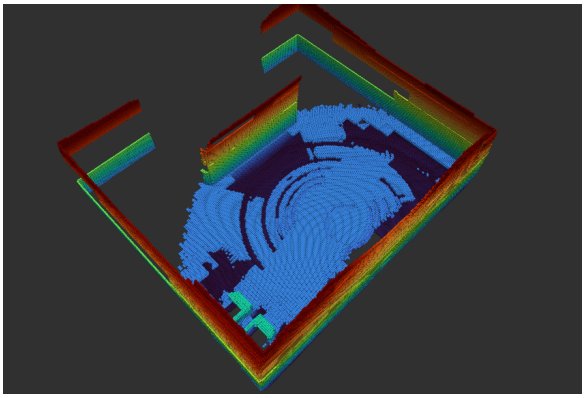
(c) Scanner only, after warm-up



(d) Scanner only, after drive

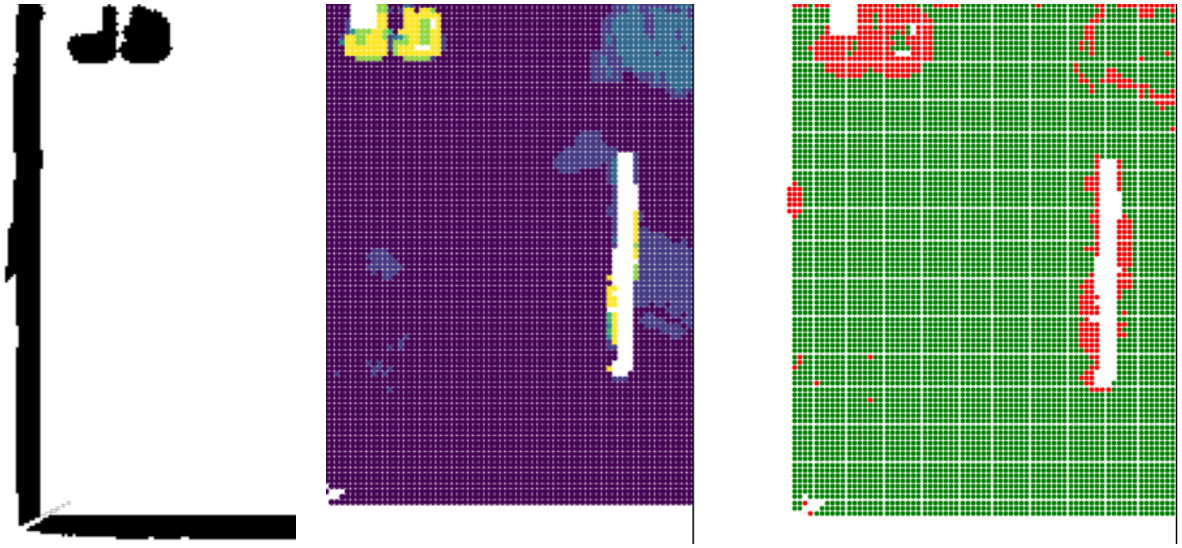


(e) Fused, after warm-up



(f) Fused, after drive

Figure 9.8: Webots sensing comparison in the start room of the maze (box E in Figure 9.1(c); perspective view in Figure 4.2): dense map meshes after the 60 s warm-up and after the 180 s scripted drive for each sensor configuration. Mesh colour encodes height.



(a) Occupancy grid (b) Support height (c) Walkability

Figure 9.9: Teleoperated coverage reference in the Webots maze (Figure 4.2; floor plan in Figure 9.1(c)). The panels show the common scored map region; in (b), color denotes support height from 0 m to 0.5 m, and in (c), green marks the 13 970 walkable columns while red marks the 1818 rejected columns.

and 3 at 0.25 rad s^{-1}, implicating camera integration during fast yaw. Suppressing 63 camera frames in the gated trial reduces the count to zero. This finding concerns near-field floor integration for this sensor geometry in the two tested environments.

9.5.3. Protocol and Reference Episodes

The second gate implementation audits goal admissibility in this world. A supervisory watchdog terminates an episode when a route-loop condition occurs, defined as less than 5 cm of net displacement over 180 s while a goal remains active, and enforces a 90 min cap.

Two reference episodes define the coverage scale. An operator drives one traverse under the same mapping configuration; it records 15 788 support columns and 13 970 walkable columns, and the gate audit admits 418 of 438 waypoints. The three panels in Figure 9.9 show the occupancy grid, support-height field, and walkability classification over the same mapped region; the classification rejects 1818 columns. This episode provides a coverage reference rather than a competing navigation method because an operator supplies route selection. It also has the weakest goal-validity record reported here: roughly 4.6 % of audited waypoints fail the gates, whereas the active validator admits every waypoint from each autonomous configuration below.

The autonomous ground-explorer configuration that precedes the navigator comparison provides the second reference. It records 15 809 support columns and 14 464 walkable columns, admits 1251 of 1251 waypoints, and runs for approximately 1394 s. The navigator comparison applies a predefined six-part criterion set: an accepted completion latch, support of at least 15 788 columns, a walkable share of at least

85 %, validator failures below 1 %, no route-loop termination or wall contact, and a time below 1394 s to reach 15 788 columns. The teleoperated reference defines the coverage threshold, and this autonomous reference defines the timing threshold. The walkable-share and validator criteria use the ratios of Section 9.1.2, where the validator criterion is the complement of the accepted share of checked publications.

9.5.4. Route Execution Comparison

Eighteen timed episodes compare the terrain-aware ground-autonomy stack [Cao+22; Yan+22] with continuous mesh navigation under Move Base Flex [PSH18; Püt+21]. The mesh route requires six chain revisions before the platform moves; those attempts terminate between 2218 and 3435 support columns without motion. The third revision produces the first driven mesh episode, reaches 8751 support columns, approximately 55 % of the teleoperated reference, and admits 1156 of 1156 waypoints. At 1177.7 s, the explorer emits a partial-coverage latch that the watchdog rejects. The explorer also records 28 tiles as unobservable because the mesh cost layers and controller margins refuse the narrow corridors.

Figure 9.10 shows why the mesh route plateaus. It draws the navigation mesh that the episode exported at three stages over the maze floor plan of Figure 9.1(c), which is taken from the Webots world file. The mesh covers the start room early in the run and grows into the first corridor, but it reaches the eastern half of the maze only as a narrow strip beyond the northern doorway, and the robot path stays inside the area the mesh already covers. The planner can only route over mesh faces, so unmeshed doorways bound the reachable area.

The ground-autonomy route reaches higher coverage in the same world. Its strongest episode uses the corrected doorway envelope, records a peak of 15 478 support columns at 1042.8 s with 13 815 walkable columns, admits 1693 of 1693 waypoints, and traverses the interior pocket doorway with a minimum body-center clearance of 0.329 m. Within this maze world and criterion set, the ground-autonomy configuration therefore produces the highest coverage among the compared routes. This result neither orders the navigation frameworks nor extends to the terrain worlds.

Two defects qualify that episode. It remains 310 columns short of the coverage criterion. Its completion latch fires at 1726.6 s, and the watchdog rejects it at 1727.0 s with terminal support of 15 442. The latch coincides with a simulator physics event at the arena outer wall, where body height rises from about 0.52 m to 2.225 m within 0.55 s. The event confirms that the completion state machine fires, but wall contact contaminates the terminal samples. One earlier episode exceeds the teleoperated coverage figure before wedging without a latch; inconsistent archived peaks exclude it from Table 9.6.

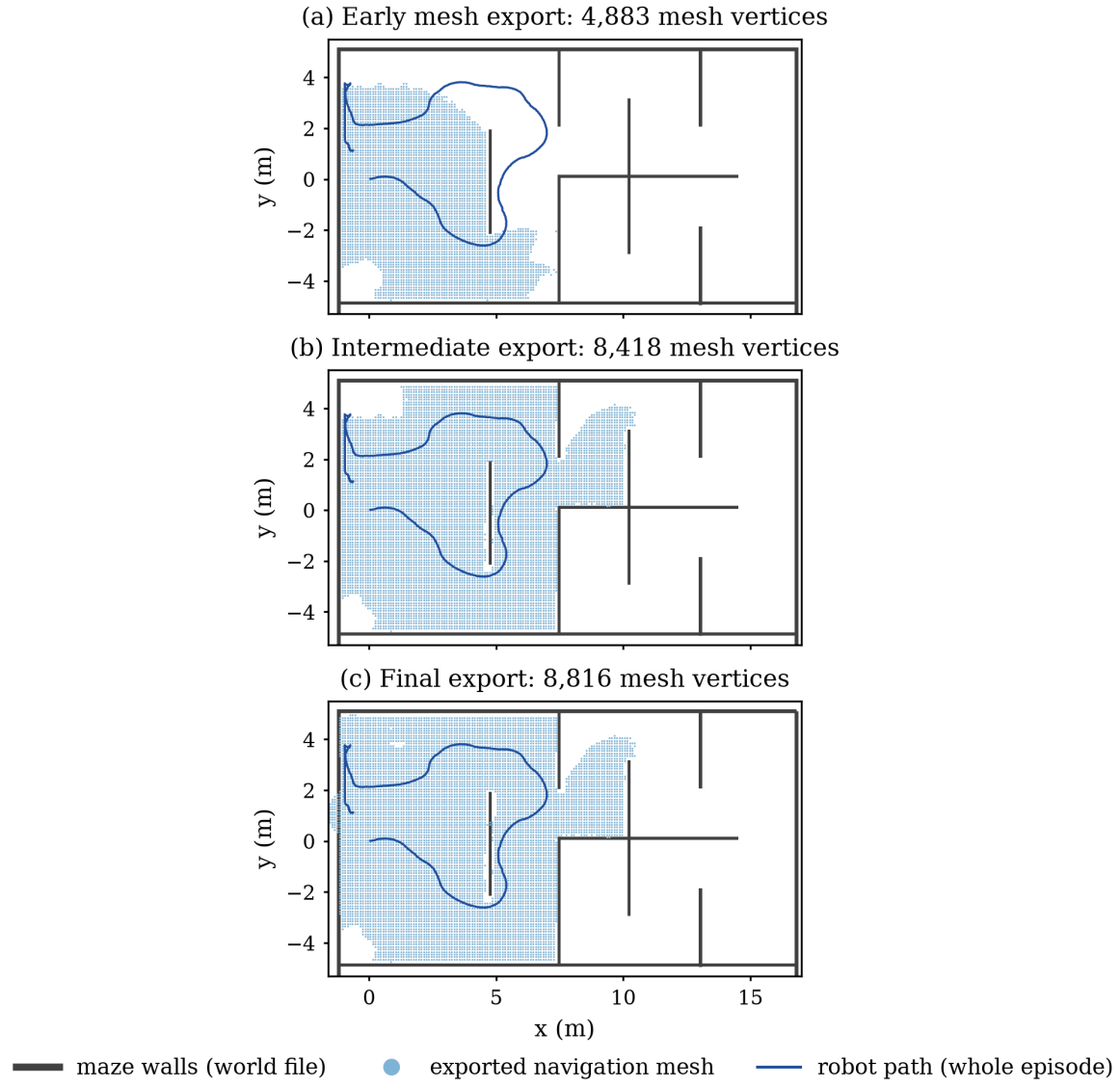


Figure 9.10: Mesh-navigation episode in the Webots maze (floor plan in Figure 9.1(c), perspective view in Figure 4.2). Blue points are the vertices of the navigation mesh exported at an early, an intermediate, and the final stage of the episode; the dark line is the whole robot path, and the walls come from the maze world file.

9.5.5. Route Variance

A repetition of the strongest configuration uses sources that checksum confirms as byte-identical and terminates at 14 239 support columns after a route-loop verdict at 1205.5 s. A long-loop condition rather than a stationary one causes that verdict: within the terminal window, the platform accumulates 12.31 m of path while returning to within 0.040 m of its window start pose. Divergence appears early, at 8737 support columns after ten minutes against 13 907 for the preceding episode on the same sources.

Across the strongest episode, its predecessor configuration, and this repetition, peak coverage spans 15 478, 15 296, and 14 239 columns. The 1239-column spread exceeds the differences attributed to individual parameter changes. Single episodes in this world therefore cannot provide a deterministic ranking of those changes. Target ordering, revisit timing, live terrain decay, and the detection gap between the escape rule and watchdog window produce material variance at fixed configuration. Live terrain decay here denotes the time-based ageing of accumulated near-field obstacle memory, and the detection gap arises because the escape rule reacts over a window shorter than the watchdog's route-loop window (Section 7.3, Section 9.5.3).

Figure 9.11 compares the peak support columns of the representative episodes with the teleoperated reference, and Figure 9.12 shows where the three ground-autonomy episodes actually walked. The strongest episode sweeps both halves of the maze and passes the interior pocket doorway; the predecessor configuration reaches the eastern rooms along a different order of targets and ends early in a route loop; the byte-identical repetition returns repeatedly to the start room before crossing to the east. The repetition and the strongest episode run the same software, yet their routes differ visibly, and their coverage difference follows from those routes rather than from a configuration change.

9.5.6. Nav2 Migration Evidence

The evaluation first exercises the Nav2 migration [Mac+20] on an isolated 7.56 m four-leg course and then in full exploration episodes. On that course, replacing the pursuit follower with model-predictive path integral control [Wil+16] cuts commanded yaw sign reversals from 7.41 to 0.44 per meter and removes all three stop-and-go intervals. It reaches only three of four legs over 29.6 m while streaming the rolling local path as the complete plan. The mini-global relay sends one fixed-goal reference path per waypoint from a grid search [HNR68] and stops re-sending inside the arrival radius; it restores four of four legs in 8.54 m at 0.35 reversals per meter.

The exploration episodes in this migration use a relaxed coverage criterion of 14 000 columns, an eighteen-minute cap, and a linear speed cap of $0.30\,\mathrm{m\,s^{-1}}$, double the reference cap. Their figures therefore show that the migrated control link sustains full exploration rather than ranking it against the preceding episodes. The first integrated predictive-control episode reaches 15 750 peak support columns and records

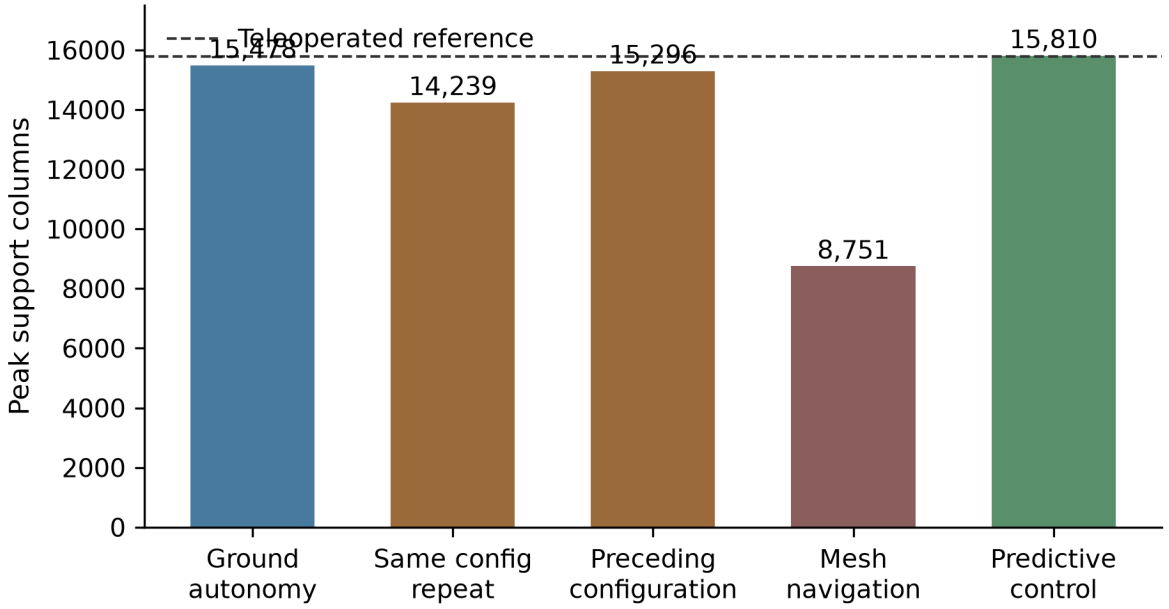


Figure 9.11: Peak support columns of representative episodes in the Webots maze (Figure 9.1(c)). The dashed line is the teleoperated coverage reference. The episodes use different stopping rules and route protocols, so the chart does not establish a universal ranking.

eleven path-following aborts because a straight-line reference seed crosses maze walls. Replacing that seed with grid search initially produces a weaker episode at 12 435 columns. After the footprint correction in Section 7.5.1, the strongest episode in this series reaches 15 810 support columns near 10.5 min. It belongs to this series and relaxed criterion and remains separate from the 15 478-column navigator-comparison episode. Neither it nor the pursuit-follower episode, which reaches 15 736 support columns in 657 s under the lower cap, runs with an active validator; neither therefore carries a goal-admissibility record.

9.5.7. Evidence Boundary

No episode in this world satisfies all six acceptance criteria at once, although different episodes satisfy each criterion individually. The highest-coverage navigator-comparison episode falls 310 columns short and ends in a wall-contact event. The episode that exceeds the coverage figure never latches, and the watchdog rejects both recorded latches. This evidence therefore supports no single winner; the coverage ordering applies only to this maze world and criterion set. Unit tests alone verify the ground-frontier explorer revision prepared for the next comparison; no episode runs it in this world. The floor-relative configuration derived from this work also receives no physical rerun.

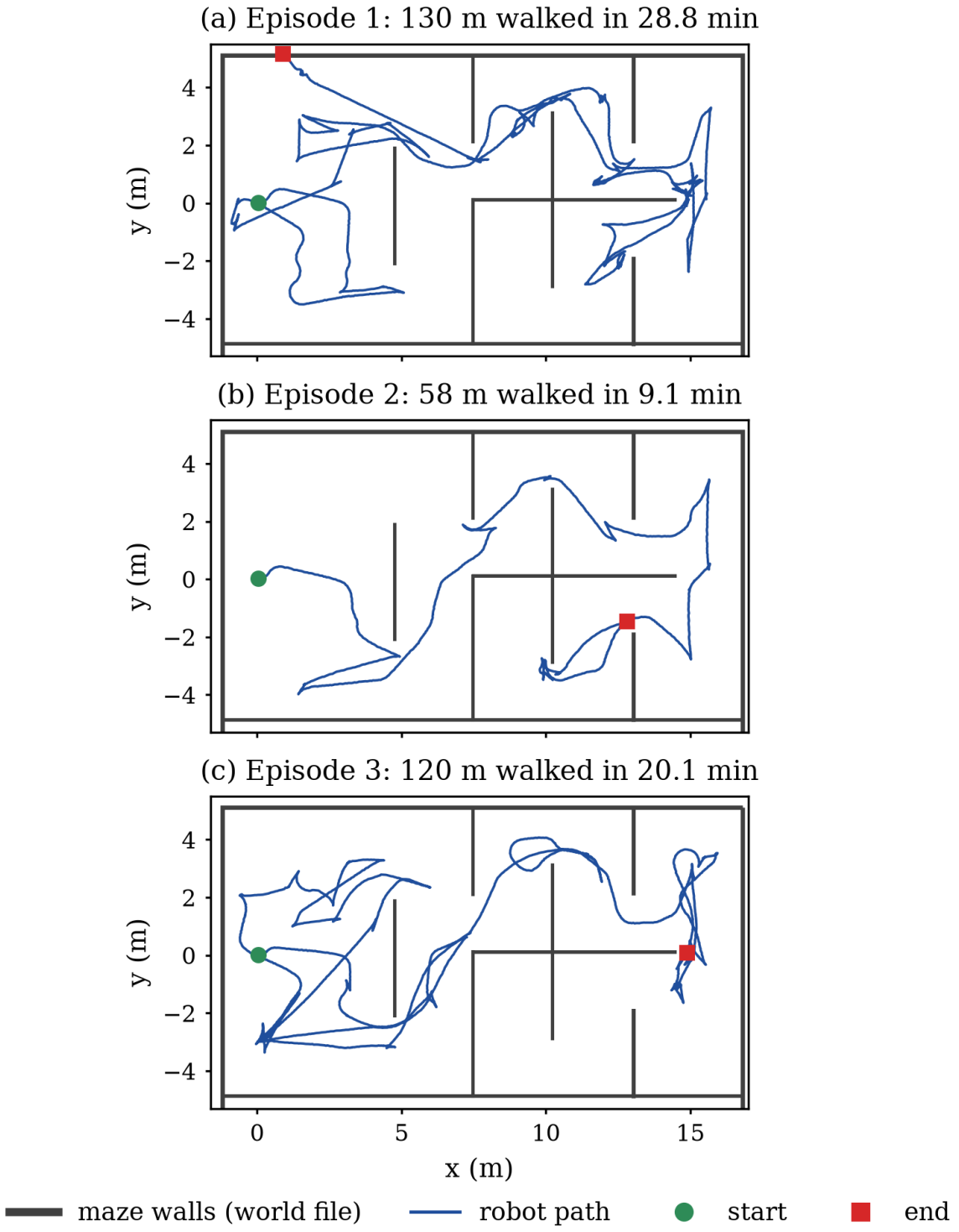


Figure 9.12: Robot paths of three ground-autonomy episodes in the Webots maze (floor plan in Figure 9.1(c), perspective view in Figure 4.2). Episode 1 is the strongest episode, Episode 2 the predecessor decay configuration, and Episode 3 the byte-identical repetition of Episode 1; they reach 15 478, 15 296, and 14 239 peak support columns. Walls come from the maze world file.

Table 9.6: Representative Webots maze episodes under the cameras-only mapping configuration. Support and walkable report the counts reached by each episode, percentages of the teleoperated reference use its 15 788 support columns as the denominator, and the validator column reports accepted and checked publication events (Section 9.1.2), where *no validator* marks a series in which the validator was not active and which therefore carries no admissibility record rather than zero failures. Termination names the stopping condition that ended the run, and unobservable tiles are explorer candidate units counted by the completion rule in Section 6.3.4.

Configuration	Support	Walkable	Validator accepted / checked	Termination
Teleoperated coverage reference	15 788	13 970	418/438	Traverse ended by operator
Ground-explorer autonomous reference	15 809	14 464	1251/1251	Reference coverage crossed at 1086.6 s
Ground autonomy, doorway envelope	15 478	13 815	1693/1693	Latch rejected at 1727.0 s; terminal wall-contact physics event
Same configuration repeated	14 239	12 984	1178/1178	Route-loop verdict at 1205.5 s
Preceding decay configuration	15 296	13 913	534/534	Route-loop verdict at 544.9 s
Mesh navigation, third chain revision	8751	8113	1156/1156	Latch at 1177.7 s rejected; 28 tiles unobservable
Pursuit follower, as-deployed parameter set	15 736	14 273	no validator	Episode stopped at 657 s
Predictive control, footprint corrected	15 810	14 340	no validator	Relaxed criterion met near 10.5 min at the doubled speed cap

9.6. Physical-Robot Evidence

Physical trials on the Boston Dynamics Spot quadruped cover the parts of the fifth research question that simulation cannot answer and bound the physical support for the first and fourth. Four evidence classes remain after the mapping comparison of Section 9.5.2: graph closure and odometry on physical recordings, staged perception and goal-only validation with actuation suppressed, continuous supervised exploration under velocity caps, and integration findings from the layered-costmap port. No physical measurement below characterizes the final integrated pipeline.

9.6.1. Graph Closure and Correction Magnitude

The live physical deployment runs graph simultaneous localization and mapping on the rotating sensor alone, with the leg-kinematic prior [LM19], and records 35 geometric local-space closures and no visual appearance closures. An offline replay of a 46.5 m physical multi-sensor recording produces a 102-node graph with 4 appearance closures and 31 geometric closures, so only offline evidence demonstrates appearance closure.

An observation-only replay of the same recording attaches the volumetric mapper and frontier layer and records 122 nodes and 50 geometric closures. Sampling corrections to the parent map transform every 0.5 s yields a median of 0.082 m and a maximum of 0.368 m across the 107 above-threshold samples. The frontier count grows from 11 to 25 voxels without phantom growth. Because the deployed mode records no appearance closure, the evaluation never exercises a meter-scale correction from a long unclosed loop against the non-deforming voxel volume.

9.6.2. Odometry on a Multi-Level Recording

The offline odometry comparison of Table 5.3 includes a 236 m physical recording that walks through several connected rooms and changes floor level. Figure 9.13 accumulates the rotating-sensor points of that recording in the map frame twice: once with the platform's leg odometry alone and once with leg-assisted GenZ-ICP, which initializes each scan registration from the leg odometry. With leg odometry alone, heading and position drift accumulate, so every wall appears several times at different angles and the rooms smear into each other. With scan registration, the walls collapse into single lines and the room layout becomes readable; a few straight streaks remain where registration briefly slips. This recording is the only multi-level case in the comparison, and the figure shows why the drifting leg estimate alone cannot anchor a map: the deployed stack corrects it with RTAB-Map rather than with this offline method.

9.6.3. Staged Perception and Goal-Only Validation

Staged validation follows the sequence in Section 3.6: read-only sensor and transform inspection, perception without command components, and goal publication isolated

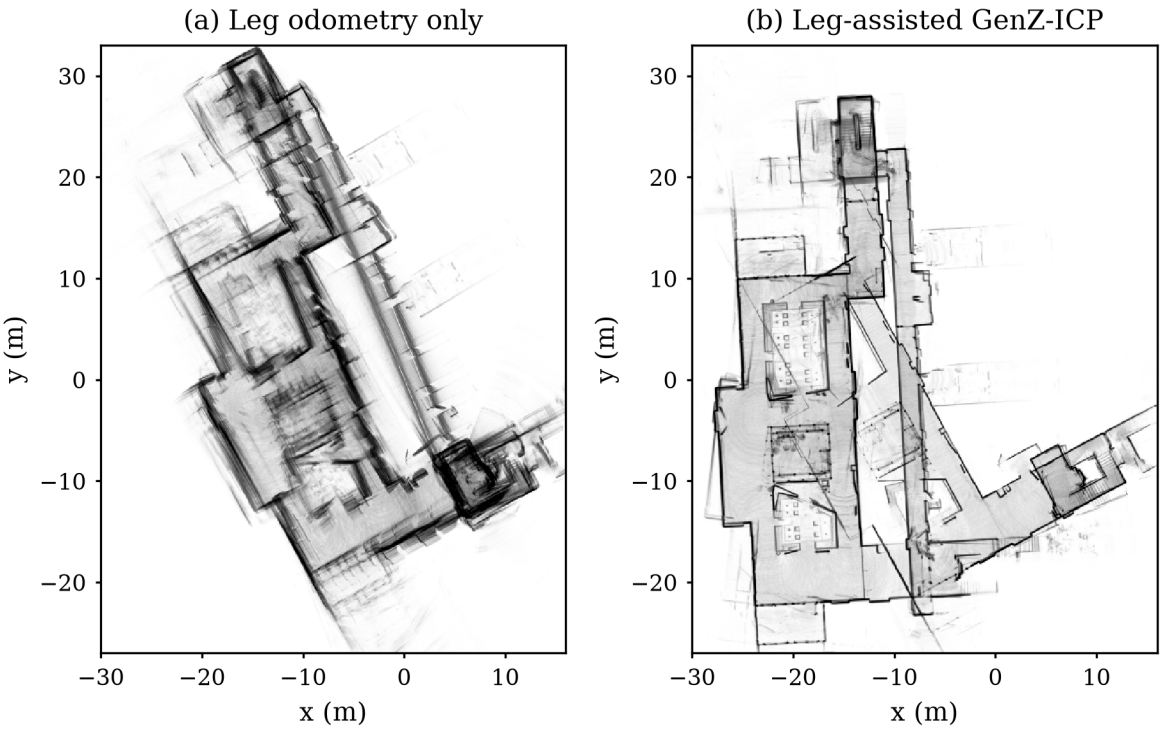


Figure 9.13: Top-down density of the rotating-sensor points accumulated over a 236 m multi-level physical recording, (a) with leg odometry only and (b) with leg-assisted GenZ-ICP. Darker pixels hold more points. Offline processing of a physical recording in a real indoor building; unlike the simulated worlds of Figure 9.1, no authored layout exists for it, so the straight walls of panel (b) are the only layout reference.

on a dedicated topic. The emergency stop and motor lease retain their operator-safety roles.

Preflight recordings establish the sensor baseline. The rotating sensor publishes at approximately 9.7 Hz to 9.9 Hz with roughly 29 000 points per scan. A 19.85 s stationary recording of 195 scans shows a maximum horizontal drift of 1.6 mm and no multi-parent conflict on the required transform chain. A 29.76 s manual-driving recording covers 8.216 m.

Perception-only operation uses a measured terrain seed of −0.700 m with obstacle classification enabled. Registered ground settles at approximately −0.685 m, terrain output runs at approximately 9.7 Hz, and static-obstacle voxel persistence reaches mean Jaccard indices of 0.930 and 0.973 in the two checked regions. The same recordings expose registered-scan loss during motion: 276 raw scans produce 263 registered scans. This loss does not affect goal-only observation, but it remains an unresolved precondition for any command-producing stage.

Goal-only operation confirms the geometric contract of each published target rather than its execution. One checked goal lies 2.34 m from the body, 0.696 m above supporting terrain, with 0.822 m of horizontal clearance to the obstacle cloud. A later goal in a fresh area lies 0.065 m horizontally from the nearest terrain support and 1.020 m from the nearest obstacle return in the configured vertical band. The command topic carries zero publishers throughout every staged run.

The record preserves two coverage outcomes without correction. After manual relocation, the separate-terrain-cloud goal selector (the TARE terrain cloud, Section 6.6.2) treats the bounded visible area as explored and returns its terminal state. In the later configuration, all five body depth cameras feed the terrain input, yet the same selector reports exploration and return-home completion before the robot moves and places the goal at the map origin. Depth transport remains active, with 12 000 to 14 000 merged filtered points per sample and 1,132 registered-cloud publications against 36 stamped-transform drops. This run therefore records immediate terminal output despite active depth transport. No sensing-disabled comparison isolates goal selection from the mapped-input state. Both physical goal-only runs use the earlier TARE (Technologies for Autonomous Robot Exploration) selector [Cao+21] rather than the custom frontier and walkable-goal layer evaluated later.

9.6.4. Continuous Supervised Navigation

Continuous exploration runs the terrain-aware ground-autonomy chain under operator supervision and the velocity limits in Section 7.4. A graduated sweep of 653 commands locates the actuation deadband between 0.13 m s^{-1} and 0.15 m s^{-1}, which fixes the symmetric linear floor. The command shaper cuts 248 recorded follower yaw sign reversals to zero. Direct goal handoff to the local planner cuts waypoint reassertions from 33 to zero and blacklist entries from 12 to zero. Because physical transits take 20 s

to 50 s, the goal hold timer uses 60 s. Frontier filtering cuts the air-classified fraction from 97 % to 6 % to 8 %. The strongest recorded five-minute window grows support from approximately 5300 to 7300 columns, visits 8 goals, and records no safety stop or stuck event.

Three behaviors persist after every applied fix: follower idling every 15 s to 20 s at path-snippet boundaries, goal reselection of previously rejected regions, and fragmented long-range route visualization. These observations motivate the architectural change in Section 7.5. They do not rank the two navigation stacks because no matched physical protocol exercises both.

9.6.5. Hardware-Specific Integration Findings

The layered-costmap port exposes findings that appear only on hardware. During walking, vertical odometry drifts by approximately −0.11 m per 100 s; while standing, it remains near 3 mm min^{-1}. The drift persists across driver restarts and reaches a body-height belief of 1.57 m against a true height near 0.68 m. Appearance-based closure fails at revisit with missing visual features until the mapping launch enables the camera path, and no platform run records a global appearance closure. Duplicate grid, scan, and command publishers include two safety nodes that own command output simultaneously, establishing the need for the launch-time and teardown sweeps in Section 3.6. Chassis-frame height cropping tilts with gait and produces ring-shaped floor returns; gravity-aligned cropping removes them. Discarding out-of-range returns prevents clearing rays until the mapper accepts infinite ranges as valid. A 1.8 m range cap reduces but does not eliminate camera-derived far-field noise. The record contains no corresponding rerun.

9.6.6. Evidence Boundary

Table 9.7 states what each physical observation supports. Two boundaries apply throughout. Hardware validates the earlier floor-relative band variants only for box detection and speckle suppression. Fixture checks and Isaac Sim verify the per-column support-relative variant and crest recovery gate, but neither receives a live trial. Joint actuator faults end platform availability before the final navigation rerun, so the integrated closed loop has no physical result.

9.7. Cross-Environment Synthesis

This section states what the reported runs support for each research question in Chapter 1 and where that support ends, applying the domain boundaries in Section 4.6 and Table 4.3. It reports results from different environments side by side without treating them as comparable scores because their sensor models, contact models, worlds, and stopping conditions differ. Chapter 10 interprets these positions and priorities.

Table 9.7: Physical evidence and permitted inference.

Physical observation	Claim it supports
Sensor rates, transform ownership, stationary and manual recordings	Interface and timing preconditions of the perception chain on the platform
Perception-only terrain output and obstacle persistence	Terrain and obstacle classification operate on real returns at the stated rates
Earlier TARE goal-only geometric checks	Targets have support and clearance, while the command topic has no publisher; no execution claim and no claim for the later custom frontier layer
Camera-only against fused floor coverage	Sensor split for dense near-field mapping on the tested indoor floor
Angular-rate gate spin trials	Fast-yaw camera integration correlates with the observed floating voxels, and the gate prevents them in the checked run
Command deadband sweep and shaped output	Platform velocity floor and removal of follower dither at the actuator
Vertical odometry drift measurement	Absolute height thresholds are unusable; height decisions need a body or floor anchor
Costmap band and clearing findings	Failure modes of chassis-anchored slicing and range-limited clearing on hardware
Supervised ground-autonomy exploration windows	Behavior of that navigation configuration under supervision; no cross-stack ranking
Final floor-relative projection with predictive control	No physical evidence; platform access ended before the rerun

The integrated odometry and volumetric mapping path maintains a representation that the frontier layer can consume in every tested configuration. On the physical robot, graph simultaneous localization and mapping supplies the corrected global pose while the driver retains the local body edge, and live operation records geometric closures (Section 5.1.2). Webots obtains the published pose from the supervisor, while Isaac Sim uses simulated odometry with an identity relation between map and odometry frames. Neither simulator therefore exercises the graph correction that runs on hardware, although both maps generate and publish frontier candidates. The physical mapping runs use the earlier band reduction, while Isaac Sim alone evaluates the floor-relative reduction and crest correction (Section 5.2.3). The evidence therefore establishes map usability per configuration, not for one configuration across both domains.

The sensing comparison has the widest agreement across domains. The Webots repeated-drive comparison, physical walking-loop trial, and later Isaac Sim range study all favor camera depth integration over scanner integration or fusion for volumetric floor quality (Section 9.5.2, Section 9.2.1). Agreement across three environments strengthens confidence in the selection for this platform and voxel resolution. It neither establishes a general ordering of sensing modalities nor shows that each environment exhibits the same mechanism.

Under the gates in Section 6.4, the frontier layer produces supported, reachable, and head-clear goals. A physical indoor recording provides replay evidence, while Isaac Sim episodes provide open-terrain evidence. The validator record covers published-goal admissibility rather than arrival or completion. On dunes, only the footprint test rejects candidates; the reachability and headroom gates reject none. One Isaac Sim dune-world mapping session measures crest-visibility retention. Single episodes in one small arena at one seed evaluate sealing. Sealing remains disabled by default, and no physical-robot episode records a latch (Section 6.5).

Navigation behavior has the most uneven evidence. Physical trials of ground autonomy and planar Nav2 expose actuator deadbands, lease and odometry-origin resets, body-height drift, and gravity-dependent sensor-band phantoms (Section 7.4, Section 7.5.2). In simulation, predictive control with a mini-global relay improves command continuity and waypoint precision (Section 7.5.1). Coverage varies between episodes of the same configuration, and no maze episode satisfies every acceptance criterion at once, so route variance forms part of the result rather than noise around one number (Section 9.5.5). The final architecture that combines floor-relative projection with predictive control receives no physical rerun, and every evaluated configuration retains the ramp-preference limitation (Section 7.7.2).

Sensor-modality selection, loop-closed odometry, mapping under real timing and compute load, and navigation bring-up carry physical evidence. Crest handling, frontier sealing, the retrained locomotion policy, and the full closed exploration loop carry simulation evidence only because hardware access ends before those trials

Table 9.8: Supported conclusions and evidence boundaries per research question; the RQ column follows the question numbering in Chapter 1.

RQ	Supported conclusion	Evidence boundary
1	The odometry and mapping path produces maps usable by the frontier layer in each tested configuration	Physical runs use the earlier band reduction; Isaac Sim alone evaluates the floor-relative variant
2	The stack selects camera depth integration over scanner integration and fusion for floor quality, with agreement in Webots, on hardware, and in Isaac Sim	Applies to this sensor set and voxel resolution; establishes no general modality ordering or shared mechanism
3	The frontier layer publishes validator-accepted goals on dune terrain; the corrected mapper retains crest visibility	Crest retention and sealing have Isaac Sim evidence only; sealing uses one run per configuration in one small arena and one seed, remains disabled by default, and has no physical latch; no trial demonstrates autonomous ramp ascent or ramp-goal placement
4	Earlier configurations execute goals on hardware under stated constraints; predictive control with a goal relay improves command continuity in simulation	Coverage varies between episodes; no maze episode meets all criteria at once; the final chain has no physical rerun; ramp preference remains unresolved
5	Sensing, odometry, mapping, and navigation bring-up carry physical evidence; crest handling, sealing, learned locomotion, and the closed loop carry simulation evidence	The declared models bound simulation results, which cannot certify contact forces, actuator response, sensor timing, or transfer

(Chapter 8).

Interface compatibility divides into demonstrated and unverified items. Hardware demonstrates single-owner frame ownership between the graph back end, driver, and mapping volume; the mapper-to-explorer column interface under replay; and the planar velocity interface to the manufacturer controller. Simulation demonstrates the explorer-to-navigator goal interface, the projection-to-costmap interface, and substitution of the retrained policy behind the same planar velocity interface as the scripted Webots gait. The corrected termination guard latches in Isaac Sim, but it remains disabled by default and absent from the main explorer configuration, which therefore carries no verified frontier-completion stopping rule. Physical deployment of the learned policy and the final projection and control chain remain unverified. Table 9.8 records these positions and their evidence boundaries.

10

Discussion and Conclusion

10.1. Thesis Summary

An exploration robot in an unknown environment must build a map of what it has observed, decide where to observe next, plan a route to that place, and walk there. On uneven ground these tasks depend on each other: a place is a useful goal only if the robot can stand there, a route is useful only if the robot can walk it, and a map that marks an obstacle where the terrain is passable hides the ground that exploration needs.

The thesis develops and evaluates such a stack on one Boston Dynamics Spot quadruped, which carries a rotating LiDAR and five body depth cameras that look downward at the nearby ground. A localization layer estimates the body pose, and its deployed component, RTAB-Map, corrects accumulated drift when the robot revisits a place. A mapping layer integrates the measurements into a three-dimensional grid of small cells, called voxels, using the GPU mapping library nvblox, and derives a floor height for each horizontal cell, called the support surface. An exploration layer detects frontiers, the boundaries between mapped free space and unobserved space, selects one as the next goal, and checks that the ground beneath it can carry the robot. A navigation layer plans a route on a two-dimensional grid and issues velocity commands, and a locomotion layer turns them into joint motions that keep the robot upright.

Four sources supply evidence: the physical Spot robot under real sensor timing and compute load; the Webots maze for repeated experiments on integration and coverage; the Isaac Sim terrain worlds for authored dunes and a ramp; and the Isaac Lab training environment for the walking controller. A fault in the joint actuators ended access to the physical robot before the final experiments, so the final configuration, which combines a floor-relative map reduction, a predictive sampling controller, and the retrained walking controller, was evaluated in simulation only.

The thesis makes five contributions, namely a sensing comparison that selects depth-

camera integration as the sole dense input to the mapping volume under the tested conditions; a mapper-side correction that keeps the ground above a terrain crest visible to the goal layer; a walkable-goal filter that moves frontier candidates onto ground the robot can stand on; a frontier-seal rule that removes frontier cells above already mapped interior ground while preserving the frontier ring at the sensor horizon; and a blind rough-terrain policy for Spot that receives no terrain geometry in its observation and keeps the existing velocity-command interface.

10.2. Contributions and Findings

Sensing configuration for the mapping volume. The first contribution concerns the dense mapping volume, which turns measurements into the floor estimate used by goal placement and by the route grid. Three configurations were compared: the five depth cameras alone, the rotating LiDAR alone, or both together. The two sensor types observe the ground differently: the depth cameras see the nearby floor at steep angles and return dense measurements close to the robot, whereas the LiDAR sees the floor at grazing angles over long distances, where a small change of body attitude moves the reconstructed surface by several centimetres and can shift or carve it. In the physical trial, the camera-only map covered 99.6 % of the scored area around the walked path, against 88.9 % for the map that also used the LiDAR, which contained a visibly doubled floor. The rotating LiDAR was not removed: it refines the localization links, supplies the planar scan used for near-field safety, and keeps long-range sensing. The finding applies to this sensor set and resolution, not to sensor types in general (Chapter 5 and section 9.5.2).

Crest visibility. The second contribution addresses ground the downward cameras cannot observe: the surface and the air directly above a convex crest, such as the top of a dune, lie to the side of their view, so the volume never records an observation above the crest. The baseline mapper treated that unobserved space as a possible obstruction and marked the crest as an obstacle, so the goal layer rejected the crest as a place to stand. The correction changes the default for a column whose supporting floor is known: the column counts as free unless an obstacle was observed in it, with obstacle evidence taking precedence, and a support height may spread to a neighbouring column only when the height difference stays at or below a gate of 0.15 m. The gate is required because propagation without it wrongly freed cells of a 0.40 m curb that the platform cannot step onto, while the gated rule kept every curb cell occupied. On the evaluated dune mask the corrected mapper marked all crest cells free, where the baseline had marked almost none, and removed the false obstacle cells on flat ground; the evidence comes from one authored dune world without a physical trial (Sections 5.2.3 and 9.2.1).

Goals on ground that can carry the robot. Frontier candidates indicate where new information may lie, but they are not places to stand: a frontier voxel records a boundary between free space and unobserved space, so it commonly lies in the air

or above ground that cannot support the robot. The third contribution converts each candidate into a goal on supported ground through four checks: snap it to the support surface of its map column; require a footprint around it to be covered mostly by valid, walkable cells, where walkable means that the support surface also passes the slope, step, and roughness limits; flood reachability from the robot over walkable cells only, so a goal behind a wall is rejected; and require free vertical space, called headroom, above the goal. A cluster that fails the checks is projected inward onto nearby valid ground instead of being discarded, and an independent validator with its own implementation of the checks audits every published goal. In replay of a physical indoor recording the validator-accepted share rose from 30 of 40 to 112 of 118, and in the simulated dune episodes the validator rechecked 12 396 publications without rejecting one; there the footprint check was the only one that rejected candidates. A fifth check against the manufacturer's obstacle grid was specified and not implemented, and the audit covers admissibility, not arrival (Section 6.4).

A stopping rule for residual frontiers. Residual frontiers are the reason the obvious stopping rule fails. Above interior floor the robot has already mapped, the air still borders unobserved space, so those frontier cells never disappear and a rule that waits for an empty frontier set never fires. A control episode with the rule disabled illustrates the problem: coverage stopped growing within the first ten minutes, yet the episode continued for about another one and a half hours, and 1042 of its 1044 residual frontiers stood above interior ground the robot had already mapped, most of them above flat floor. The fourth contribution removes frontier cells that lie above an interior mapped floor or at a supported local crest from the set of unexplored ground while preserving the frontier ring at the sensor horizon, which is genuinely unobserved ground. A first form also removed the horizon ring and starved the robot after 0.58 m; the revised rule retained almost all of the coverage of the disabled control, left a single residual frontier cell, and allowed the completion request to fire. The rule ships disabled by default, and each sealing result comes from one episode in one small arena and seed (Section 6.5).

From map to route. Between the map and the walking controller sits the route layer, which operates on a two-dimensional grid because its planners and controllers expect one. The projection therefore measures obstacle clearance in a band above the local floor height of each column, from 0.30 m to 0.80 m above the support surface, and falls back to a body-relative band in columns without a floor estimate. This keeps a usable route interface on slopes, because the band follows the local floor, but it discards elevation, slope, step height, and headroom (Chapter 7 and section 7.6.2).

Navigation and learned locomotion, the fifth contribution. Two navigation configurations were compared in the Webots maze. A terrain-aware stack that plans over the three-dimensional surface reached the highest coverage measured in that world under its criterion set, while an alternative that routes on a continuous mesh

reached about half of the teleoperated coverage reference. Neither wins universally: the ordering holds only for that maze world and criterion set, and repetitions showed that coverage varies between runs, with the strongest coverage episode still failing an acceptance criterion and ending after contact with the arena wall. Two runs of byte-identical software with one configuration ended at 15 478 and 14 239 mapped cells, a spread larger than the differences attributed to individual parameter changes, so single episodes cannot rank parameter settings there. The walking side contributes a learned policy trained in the Isaac Lab environment on rough terrain and retrained through four correction stages, which reduced uncommanded drift, restored station keeping, and improved tracking. It receives no terrain geometry, only body-sensed values such as joint positions and body velocity, and it returns joint targets behind the existing velocity interface. Integration then exposed a deadband the per-axis checks had not covered: a purely linear command never starts a gait from rest, and the threshold lies between $0.35\,\mathrm{m\,s^{-1}}$ and $0.40\,\mathrm{m\,s^{-1}}$, so the stack commands a floor above it together with a small alternating yaw command. Every policy figure is a simulator measurement, and the retrained policy never ran on the physical robot (Sections 7.3.3 and 9.5.5 and chapter 8).

10.3. Answers to the Research Questions

The five questions of Chapter 1 are restated and answered below with their evidence domains; Table 9.8 records the positions with their boundaries in full.

The first research question asks whether the integrated odometry and mapping path can maintain a representation that frontier-based exploration can use under the tested physical and simulated configurations. It can: on the physical robot the localization layer corrected the drift accumulated by leg-based odometry, and in both simulators the mapper published frontier candidates. The boundary is that physical runs used the earlier vertical band reduction, while only the Isaac Sim experiments evaluated the floor-relative reduction.

The second research question asks how the available depth-camera and LiDAR sensing configurations affect the suitability of the resulting map for exploration. Camera depth integration produced the most complete and cleanest floor in the Webots comparison, the physical walking-loop trial, and the later Isaac Sim range study, while scanner-only mapping and fusion degraded it. The result is a selection for this sensor set and map resolution, not a general ordering of the two sensor types.

The third research question asks whether a true-three-dimensional frontier layer can produce supported, reachable, and collision-free goals on ramps and dunes, including regions observed only above a crest. On dune terrain it can, because the independent validator accepted every publication it checked, and after the crest correction goals appeared on ground above crest height, whereas the configuration before the correction placed none there. The ramp part has a negative answer: no evaluated configuration

placed a goal that required ascending the ramp, and the planner routed across adjacent flat ground instead. Crest retention and sealing rest on simulation evidence only.

The fourth research question asks how the tested navigation configurations execute exploration goals on uneven terrain and which map, controller, and interface constraints determine their suitability. Earlier configurations executed goals on the physical robot under the stated velocity and supervision limits, and in simulation predictive control with a grid-searched goal relay improved command continuity and waypoint precision. Coverage varied between episodes of one configuration and no maze episode satisfied every acceptance criterion at once, so the comparison orders nothing outside its world and criterion set. The two-dimensional route grid, which gives a ramp and the floor beside it the same traversal cost, remains the constraint that limits route execution.

The fifth research question asks which parts of the stack simulation can evaluate and which claims require evidence from the physical robot. Physical trials support sensing behavior, loop-closed odometry, mapping under real timing and compute load, and navigation bring-up, while crest handling, frontier sealing, the retrained controller, and the closed loop carry simulation evidence only. The declared actuator, contact, and sensor models cannot certify real contact forces, actuator response, sensor timing, or transfer to hardware.

10.4. Limitations

Results remain inside the domain that produced them: physical trials constrain sensing, localization, and supervised navigation; the Webots maze constrains integration and coverage; the Isaac Sim worlds constrain crest handling, sealing, and the final chain; the training environment constrains the walking controller. The following items remain open.

- *No physical rerun of the final chain.* The final configuration combines the floor-relative map reduction, the sampling controller, and the retrained walking controller, and no physical measurement covers it. No physical run exercised a large correction against the mapping volume either, such as one following a long loop through a revisited place, because the volume stores measurements without the pose at which each was taken and therefore cannot re-pose them after a localization correction (Sections 9.6.1 and 9.6.6).
- *Ramp behavior.* No evaluated configuration climbed the ramp autonomously or placed a goal that required the climb, because the route planner chose the flat ground beside the ramp on both attempts. The reason for that preference is not resolved, since the route grid gives a ramp and the adjacent floor the same traversal cost. A teleoperated climb was observed without a measured trajectory (Sections 7.7.2 and 9.3.5).
- *Frontier sealing.* Each sealing result comes from one episode in one small simulated

arena under one seed, the corrected stability guard exists only in a side variant, and the rule ships disabled. No physical episode records a completion request that the supervisory watchdog accepts (Section 9.3.4).

- *Early termination of a physical goal-only run.* Both physical runs that published goals without commanding motion used the earlier goal-selection component TARE, which stores a single ground height per map column and cannot represent a raised deck over walkable floor. In the later run the component reported that exploration and the return home were complete before the robot moved and placed its goal at the map origin, although depth measurements were arriving, and the planned diagnostic comparison was not completed, so the cause remains undetermined (Section 6.6.2).
- *Coverage in the maze.* No episode satisfied all six acceptance criteria at once. The highest-coverage episode finished 310 mapped cells short of the coverage criterion and ended when the body contacted the arena wall, so the ordering applies only to that world and criterion set (Section 9.5.7).
- *Walking controller.* Every figure is a simulator measurement, and the controller never ran on the physical robot. Two attitude gates remain failed: walking roll bias, a steady sideways lean while walking, and roll peak, the largest momentary sideways tilt. The harness statistic for sideways drift under-reported the runtime drift by roughly a factor of five, so its reductions are directional evidence; training evaluated the controller at fifty hertz against a deployed twenty hertz, and the retuned rewards depart from the actuator configuration supplied with the robot model (Section 9.4.3).

10.5. Future Work

A physical rerun of the final chain would test the floor-relative reduction with the sampling controller on hardware, allow the seal to be enabled with the corrected guard, and record the completion request that no physical episode currently shows. Deploying the retrained walking controller would require a joint-level hardware interface, which was unavailable, together with actuator and contact calibration (Chapter 8).

Terrain-aware routing needs a cost term over elevation, either as a cost layer derived from the terrain gradient or through a surface planner, so that route search can prefer ground the platform can climb. The route check should extend to intermediate faces steeper than the training envelope and to descending edges that produce no positive obstacle return in the projection band (Sections 7.6.2 and 7.7.2).

The map side needs a representation that survives a large correction. A pose graph of rigid submaps would allow integrated geometry to be re-posed after a correction, appearance-based place recognition would have to run in the live deployment, and the offline split of pose degrees of freedom between the sensor sources remains available

for a live graph to evaluate (Sections 5.1.2 and 5.3.1).

Route variance at one fixed configuration makes single episodes unusable for ranking parameters, so the next measurement step is a protocol of repeated runs across seeds with a control marker for each candidate generator. Such a protocol should also settle how far the robot should stand from curbs and the undetermined cause of the early goal-only termination.

For the walking controller, the two failed attitude gates need statistics that score the same quantity as their thresholds, because world-level tilt peaks and terrain-relative scoring disagree, and repeated seeds plus alignment of the training and deployment control cadence would make the policy comparable across runs. Frontier sealing should become a default only after a completion request has been validated in more than one arena and seed, and the fifth goal-filter check against the manufacturer's obstacle grid remains specified and unimplemented.

References

[HNR68] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

[Kha86] Oussama Khatib. "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots". In: *The International Journal of Robotics Research* 5.1 (1986), pp. 90–98. DOI: 10.1177/027836498600500106.

[LC87] William E. Lorensen and Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: *ACM Computer Graphics (SIGGRAPH '87)* 21.4 (1987), pp. 163–169. DOI: 10.1145/37401.37422.

[BM92] Paul J. Besl and Neil D. McKay. "A Method for Registration of 3-D Shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[CL96] Brian Curless and Marc Levoy. "A Volumetric Method for Building Complex Models from Range Images". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. ACM, 1996, pp. 303–312. DOI: 10.1145/237170.237269.

[Yam97] Brian Yamauchi. "A frontier-based approach for autonomous exploration". In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. IEEE. 1997, pp. 146–151. DOI: 10.1109/CIRA.1997.613851.

[LaV98] Steven M. LaValle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. Iowa State University, 1998. URL: https://lavalle.pl/rrt/.

[Mic04] Olivier Michel. "WebotsTM: Professional Mobile Robot Simulation". In: *International Journal of Advanced Robotic Systems* 1.1 (2004), p. 5. DOI: 10.5772/5618. URL: https://arxiv.org/abs/cs/0412052.

[GSB07] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improved techniques for grid mapping with Rao-Blackwellized particle filters". In: *IEEE Transactions on Robotics* 23.1 (2007), pp. 34–46.

[Hol+10] Dirk Holz et al. "Evaluating the efficiency of frontier-based exploration strategies". In: *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. VDE. 2010, pp. 1–8.

[KF11] Sertac Karaman and Emilio Frazzoli. "Sampling-based Algorithms for Optimal Motion Planning". In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894. DOI: 10.1177/0278364911406761. URL: https://arxiv.org/abs/1105.1186.

[Koh+11] Stefan Kohlbrecher et al. "A flexible and scalable SLAM system with full 3D motion estimation". In: *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. IEEE. 2011, pp. 155–160.

[New+11] Richard A. Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.

[GT12] Dorian Gálvez-López and Juan D. Tardós. "Bags of binary words for fast place recognition in image sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197. DOI: 10.1109/TRO.2012.2197158.

[KSK12] Matan Keidar, Eran Sadeh-Or, and Gal A. Kaminka. "Fast Frontier Detection for Robot Exploration". In: *Advanced Agent Technology*. Vol. 7068. Lecture Notes in Computer Science. Springer, 2012, pp. 281–294. DOI: 10.1007/978-3-642-27216-5_20.

[Hor+13] Armin Hornung et al. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous robots* 34.3 (2013), pp. 189–206. DOI: 10.1007/s10514-012-9321-0.

[LM13] Mathieu Labbé and François Michaud. "Appearance-based loop closure detection for online large-scale and long-term operation". In: *IEEE Transactions on Robotics* 29.3 (2013), pp. 734–745. DOI: 10.1109/TRO.2013.2242375.

[Nie+13] Matthias Nießner et al. "Real-time 3D Reconstruction at Scale Using Voxel Hashing". In: *ACM Transactions on Graphics (TOG)* 32.6 (2013), Article 169. DOI: 10.1145/2508363.2508374.

[LHS14] David V. Lu, Dave Hershberger, and William D. Smart. "Layered Costmaps for Context-Sensitive Navigation". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2014, pp. 709–715. DOI: 10.1109/IROS.2014.6942636.

[ZS14] Ji Zhang and Sanjiv Singh. "LOAM: Lidar odometry and mapping in real-time". In: *Robotics: Science and Systems*. 2014. DOI: 10.15607/RSS.2014.X.007.

[Hen+15] Lionel Heng et al. "Efficient Visual Exploration and Coverage with a Micro Aerial Vehicle in Unknown Environments". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015. DOI: 10.1109/ICRA.2015.7139309.

[MMT15] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. "ORB-SLAM: A versatile and accurate monocular SLAM system". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[Whe+15] Thomas Whelan et al. "ElasticFusion: Dense SLAM without a pose graph". In: *Robotics: Science and Systems XI*. Vol. 11. 2015. DOI: 10.15607/RSS.2015.XI.001.

[Bir+16] Andreas Bircher et al. "Receding Horizon "Next-Best-View" Planner for 3D Exploration". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1462–1468. DOI: 10.1109/ICRA.2016.7487281.

[Cad+16] Cesar Cadena et al. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[FH16] Péter Fankhauser and Marco Hutter. "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation". In: *Robot Operating System (ROS): The Complete Reference*. Ed. by Anis Koubaa. Springer, 2016. Chap. 5, pp. 99–120. DOI: 10.1007/978-3-319-26054-9_5.

[Hes+16] Wolfgang Hess et al. "Real-time loop closure in 2D LIDAR SLAM". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1271–1278.

[Hut+16] Marco Hutter et al. "ANYmal - a highly mobile and dynamic quadrupedal robot". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 38–44. DOI: 10.1109/IROS.2016.7758092.

[Tab+16] Wennie Tabib et al. "Computationally Efficient Information-Theoretic Exploration of Pits and Caves". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 3722–3727. DOI: 10.1109/IROS.2016.7759548.

[Wer+16] Martin Wermelinger et al. "Navigation Planning for Legged Robots in Challenging Terrain". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1184–1189. DOI: 10.1109/IROS.2016.7759199.

[Wil+16] Grady Williams et al. "Aggressive Driving with Model Predictive Path Integral Control". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1433–1440. DOI: 10.1109/ICRA.2016.7487277.

[MT17] Raul Mur-Artal and Juan D Tardós. "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras". In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.

[Ole+17] Helen Oleynikova et al. “Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1366–1373. DOI: 10.1109/IROS.2017.8202315. URL: https://arxiv.org/abs/1611.03631.

[Sch+17] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017). arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[FBH18] Péter Fankhauser, Michael Bloesch, and Marco Hutter. “Probabilistic Terrain Mapping for Mobile Robots with Uncertain Localization”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3019–3026. DOI: 10.1109/LRA.2018.2849506.

[Pen+18] Xue Bin Peng et al. “Sim-to-Real Transfer of Robotic Control with Dynamics Randomization”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3803–3810. DOI: 10.1109/ICRA.2018.8460528. URL: https://arxiv.org/abs/1710.06537.

[PSH18] Sebastian Pütz, Jorge Santos Simón, and Joachim Hertzberg. “Move Base Flex: A Highly Flexible Navigation Framework for Mobile Robots”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3416–3421. DOI: 10.1109/IROS.2018.8593829.

[TIK18] Anirudh Topiwala, Pranav Inani, and Abhishek Kathpal. “Frontier Based Exploration for Autonomous Robot”. In: *arXiv preprint arXiv:1806.03581* (2018). ArXiv implementation paper; peer-reviewed venue not independently verified. URL: https://arxiv.org/abs/1806.03581.

[Del+19] Jeffrey Delmerico et al. “Current state and future directions of autonomous robotic systems for search and rescue”. In: *Advanced Robotics* 33.23 (2019), pp. 1193–1207.

[Hwa+19] Jemin Hwangbo et al. “Learning Agile and Dynamic Motor Skills for Legged Robots”. In: *Science Robotics* 4.26 (2019), eaau5872. DOI: 10.1126/scirobotics.aau5872. URL: https://arxiv.org/abs/1901.08652.

[LM19] Mathieu Labbé and François Michaud. “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation”. In: *Journal of Field Robotics* 36.2 (2019), pp. 416–446. DOI: 10.1002/rob.21831.

[Vel19] Velodyne Lidar, Inc. *Puck VLP-16 Datasheet*. Document 63-9229 Rev. K. 2019. URL: https://ouster.com/downloads/velodyne-downloads.

[Bou+20] Amanda Bouman et al. “Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 2518–2525. DOI: 10.1109/IROS45743.2020.9341361.

[Dan+20] Tung Dang et al. "Graph-Based Subterranean Exploration Path Planning Using Aerial and Legged Robots". In: *Journal of Field Robotics* 37.8 (2020), pp. 1363–1388. DOI: 10.1002/rob.21993.

[Kim+20] Donghyun Kim et al. "Vision Aided Dynamic Exploration of Unstructured Terrain with a Small-Scale Quadruped Robot". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 2464–2470. DOI: 10.1109/ICRA40945.2020.9196777.

[Lee+20] Joonho Lee et al. "Learning Quadrupedal Locomotion over Challenging Terrain". In: *Science Robotics* 5.47 (2020), eabc5986. DOI: 10.1126/scirobotics.abc5986. URL: https://arxiv.org/abs/2010.11251.

[Mac+20] Steve Macenski et al. "The Marathon 2: A Navigation System". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2718–2725.

[Mil+20] Ian D. Miller et al. "Mine Tunnel Exploration Using Multiple Quadrupedal Robots". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2840–2847. DOI: 10.1109/LRA.2020.2972872. URL: https://arxiv.org/abs/1909.09662.

[Ram+20] Milad Ramezani et al. "Online LiDAR-SLAM for Legged Robots with Robust Registration and Deep-Learned Loop Closure". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 4158–4164. DOI: 10.1109/ICRA40945.2020.9196769. URL: https://arxiv.org/abs/2001.10249.

[Rei+20] Victor Reijgwart et al. "Voxgraph: Globally Consistent, Volumetric Mapping Using Signed Distance Function Submaps". In: *IEEE Robotics and Automation Letters* 5.1 (2020), pp. 227–234. DOI: 10.1109/LRA.2019.2953859.

[Sha+20] Tixiao Shan et al. "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5135–5142. DOI: 10.1109/IROS45743.2020.9341176.

[ACM21] Héctor Azpúrua, Mario F. M. Campos, and Douglas G. Macharet. "Three-dimensional Terrain Aware Autonomous Exploration for Subterranean and Confined Spaces". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 2443–2449. DOI: 10.1109/ICRA48506.2021.9561099.

[Cao+21] Chao Cao et al. "TARE: A Hierarchical Framework for Efficiently Exploring Complex 3D Environments". In: *Robotics: Science and Systems XVII*. 2021. DOI: 10.15607/RSS.2021.XVII.018. URL: https://www.roboticsproceedings.org/rss17/p018.pdf.

[Kör+21] Marian Körber et al. “Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning”. In: *arXiv preprint arXiv:2103.04616* (2021). URL: https://arxiv.org/abs/2103.04616.

[LLA21] Iván Lluvia, Elena Lazkano, and Ander Ansuategi. “Active mapping and robot exploration: A survey”. In: *Sensors* 21.7 (2021), p. 2445.

[MJ21] Steve Macenski and Ivona Jambrecic. “SLAM Toolbox: SLAM for the dynamic world”. In: *Journal of Open Source Software*. Vol. 6. 61. 2021, p. 2783.

[Mak+21] Viktor Makoviychuk et al. “Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning”. In: *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*. 2021. URL: https://arxiv.org/abs/2108.10470.

[Püt+21] Sebastian Pütz et al. “Continuous Shortest Path Vector Field Navigation on 3D Triangular Meshes for Mobile Robots”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2256–2263. DOI: 10.1109/ICRA48506.2021.9560981.

[WH21] Lorenz Wellhausen and Marco Hutter. “Rough Terrain Navigation for Legged Robots using Reachability Planning and Template Learning”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 6914–6921. DOI: 10.1109/IROS51168.2021.9636358.

[XZ21] Wei Xu and Fu Zhang. “FAST-LIO: A Fast, Robust LiDAR-inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter”. In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 3317–3324. DOI: 10.1109/LRA.2021.3064227. URL: https://arxiv.org/abs/2010.08196.

[Zho+21] Boyu Zhou et al. “FUEL: Fast UAV Exploration Using Incremental Frontier Structure and Hierarchical Planning”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 779–786. DOI: 10.1109/LRA.2021.3051563. URL: https://arxiv.org/abs/2010.11561.

[Cao+22] Chao Cao et al. “Autonomous Exploration Development Environment and the Planning Algorithms”. In: *2022 IEEE International Conference on Robotics and Automation (ICRA)*. 2022, pp. 8921–8928. DOI: 10.1109/ICRA46639.2022.9812330. URL: https://arxiv.org/abs/2110.14573.

[HCC22] Nathan Hughes, Yun Chang, and Luca Carlone. “Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization”. In: *Robotics: Science and Systems XVIII*. 2022. DOI: 10.15607/RSS.2022.XVIII.050. URL: https://arxiv.org/abs/2201.13360.

[Kul+22] Mihir Kulkarni et al. "Autonomous Teamed Exploration of Subterranean Environments Using Legged and Aerial Robots". In: *2022 IEEE International Conference on Robotics and Automation (ICRA)*. 2022, pp. 3306–3313. DOI: 10.1109/ICRA46639.2022.9812401. URL: https://marco-tranzatto.github.io/publications/fr_2021_gbplanner_2.pdf.

[Mik+22a] Takahiro Miki et al. "Elevation Mapping for Locomotion and Navigation using GPU". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2273–2280. DOI: 10.1109/IROS47612.2022.9981507.

[Mik+22b] Takahiro Miki et al. "Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild". In: *Science Robotics* 7.62 (2022), eabk2822. DOI: 10.1126/scirobotics.abk2822. URL: https://arxiv.org/abs/2201.08117.

[Mob22] Mobile Autonomous Systems and Cognitive Robotics Institute. *webots_ros2_spot: Webots Simulation of Spot*. https://github.com/MASKOR/webots_ros2_spot. Software repository, accessed 19 September 2026. 2022.

[PBF22] Miloš Prágr, Jan Bayer, and Jan Faigl. "Autonomous Robotic Exploration with Simultaneous Environment and Traversability Models Learning". In: *Frontiers in Robotics and AI* 9 (2022), p. 910113. DOI: 10.3389/frobt.2022.910113.

[Rud+22] Nikita Rudin et al. "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning". In: *Proceedings of the 5th Conference on Robot Learning*. Vol. 164. PMLR, 2022, pp. 91–100. URL: https://arxiv.org/abs/2109.11978.

[Xu+22] Wei Xu et al. "FAST-LIO2: Fast Direct LiDAR-Inertial Odometry". In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2053–2073. DOI: 10.1109/TRO.2022.3141876. URL: https://arxiv.org/abs/2107.06829.

[Yan+22] Fan Yang et al. "FAR Planner: Fast, Attemptable Route Planner using Dynamic Visibility Update". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 9–16. DOI: 10.1109/IROS47612.2022.9981574. URL: https://arxiv.org/abs/2110.09460.

[Yu+22] Wenhao Yu et al. "Visual-Locomotion: Learning to Walk on Complex Terrains with Vision". In: *Proceedings of the 5th Conference on Robot Learning*. Vol. 164. PMLR, 2022, pp. 1291–1302. URL: https://proceedings.mlr.press/v164/yu22a.html.

[Mac+23] Steve Macenski et al. "From the Desks of ROS Maintainers: A Survey of Modern and Capable Mobile Robotics Algorithms in the Robot Operating System 2". In: *Robotics and Autonomous Systems* 168 (2023), p. 104493. DOI: 10.1016/j.robot.2023.104493. URL: https://arxiv.org/abs/2307.15236.

[Viz+23] Ignacio Vizzo et al. “KISS-ICP: In Defense of Point-to-Point ICP—Simple, Accurate, and Robust Registration If Done the Right Way”. In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 1029–1036. DOI: 10.1109/LRA.2023.3236571.

[WCF23] David Wisth, Marco Camurri, and Maurice Fallon. “VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots”. In: *IEEE Transactions on Robotics* 39.1 (2023), pp. 309–326. DOI: 10.1109/TRO.2022.3193788.

[BDA24] BDAI Institute. *spot_ros2: ROS 2 Driver for Boston Dynamics Spot*. https://github.com/bdaiinstitute/spot_ros2. Accessed: 2024. 2024.

[Bos24] Boston Dynamics. *Spot – The agile mobile robot*. https://bostondynamics.com/products/spot/. Accessed: 2024. 2024.

[Cyb24] Cyberbotics Ltd. *Webots: Open-source robot simulator*. https://cyberbotics.com. Accessed: 2024. 2024.

[Mil+24] Alexander Millane et al. “nvblox: GPU-Accelerated Incremental Signed Distance Field Mapping”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 2698–2705. DOI: 10.1109/ICRA57147.2024.10611532. URL: https://arxiv.org/abs/2311.00626.

[OHB24] Oyindamola Omotuyi, David Hoeller, and Ted Burnham. *Closing the Sim-to-Real Gap: Training Spot Quadruped Locomotion with NVIDIA Isaac Lab*. NVIDIA Technical Blog. Published 17 June 2024. June 2024. URL: https://developer.nvidia.com/blog/closing-the-sim-to-real-gap-training-spot-quadruped-locomotion-with-nvidia-isaac-lab.

[LLH25] Daehan Lee, Hyungtae Lim, and Soohee Han. “GenZ-ICP: Generalizable and Degeneracy-Robust LiDAR Odometry Using an Adaptive Weighting”. In: *IEEE Robotics and Automation Letters* 10.1 (2025), pp. 152–159. DOI: 10.1109/LRA.2024.3503254.

[Mil+25] A. J. Miller et al. “High-Performance Reinforcement Learning on Spot: Optimizing Simulation Parameters with Distributional Measures”. In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 9981–9988. DOI: 10.1109/ICRA55743.2025.11128575. URL: https://arxiv.org/abs/2504.17857.

[Mit+25] Mayank Mittal et al. “Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning”. In: *arXiv preprint arXiv:2511.04831* (2025). arXiv: 2511.04831 [cs.RO]. URL: https://arxiv.org/abs/2511.04831.

[Zac+26] Kevin Zacharia et al. “OmniPlanner: Universal Exploration and Inspection Path Planning across Robot Morphologies”. In: *arXiv preprint arXiv:2603.04284* (2026). arXiv: 2603.04284 [cs.RO]. URL: https://arxiv.org/abs/2603.04284.

[Zhe+26] Han Zheng et al. “TravExplorer: Cross-Floor Embodied Exploration via Traversability-Aware 3-D Planning”. In: *arXiv preprint arXiv:2605.19958* (2026). ArXiv preprint; peer-reviewed venue and DOI not independently verified. URL: https://arxiv.org/html/2605.19958v1.

A

Launch Files and System Configuration

B

Rosbag Datasets and Topic Tables

C

Benchmark Scoreboards and Episode Logs

D

RL Training Configurations

E

Docker Images and Software Versions