applied sciences

MDPI

Article

Context-Aware Path Planning: A Unified Approach for Navigation in Heterogeneous Environments

Hamid Didari * and Gerald Steinbauer-Wagner

Institute of Software Technology, Graz University of Technology, 8010 Graz, Austria; gerald.steinbauer-wagner@tugraz.at
* Correspondence: hamid.didari@tugraz.at

Abstract

Autonomous navigation across heterogeneous environments remains challenging because different environments may require fundamentally different planning representations and action models. Structured roads are naturally represented as graphs with constrained connectivity, unstructured terrain requires continuous geometric reasoning, and service-specific areas such as charging stations can be described by discrete task states and symbolic actions. We propose a modular, context-aware planning framework that represents the environment as a set of navigation contexts, each with its own environment representation, planning state space, applicable actions, and dynamics. Contexts are connected through explicitly defined interfaces that determine where inter-context transitions can occur and how states are mapped between the corresponding representations. Planning is formulated as an incremental A*-style search over a joint state space comprising the current context, context-specific state, and internal resource variables such as battery level. The planner jointly considers intra-context actions and inter-context transitions while minimizing accumulated action and context-transition costs subject to modeled resource constraints. We evaluate the framework in simulated environments combining continuous geometric, graph-based, and discrete task-level representations. Compared with a hierarchical multi-context planning baseline using fixed representative interface states, the proposed method produced shorter routes, retained more battery at the goal, and required less planning time in both evaluated resource scenarios. In a restricted grid–topological evaluation, both the proposed method and a strengthened topological baseline found the shortest path in all 50 test cases, while the proposed method achieved lower average planning time. A nearest-node topological baseline frequently failed or produced suboptimal paths.

Keywords: autonomous navigation; context-aware path planning; multi-representation path planning; heterogeneous environments; context transitions; heterogeneous representations; resource-aware planning; A*-based path planning

Academic Editors: Cheng Xu, Shihong Duan and Augusto Ferrante

Received: 19 August 2026
Revised: 10 September 2026
Accepted: 17 September 2026
Published: 19 September 2026

1. Introduction

Autonomous mobile robots operate in a wide variety of environments, ranging, for instance, from structured urban roads and unstructured off-road terrain to subterranean spaces. Each domain imposes distinct requirements on perception, localization, and control. Although state-of-the-art autonomous navigation systems achieve strong performance in domain-specific settings [1–3], operating across heterogeneous environments remains challenging because different environments may require different representations, sensing

modalities, and planning abstractions [4]. Because real-world missions may span multiple such environments, autonomous robots require unified navigation frameworks that explicitly account for environmental context and employ context-appropriate representations, actions, and planning models while also considering internal factors that influence navigation decisions. Figure 1 illustrates this interaction: depending on its internal state and the context-specific navigation capabilities available along alternative routes, the robot may choose a route through a charging-station context rather than following the lower-cost direct route.

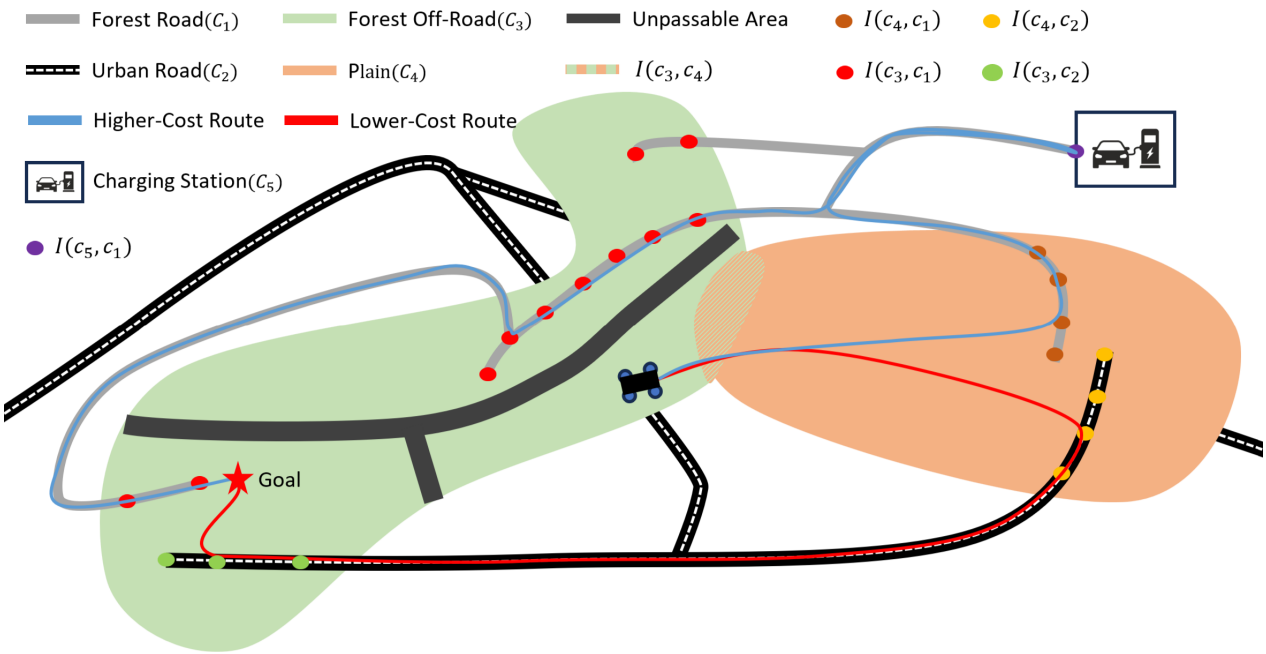


Figure 1. Example of a navigation problem in a heterogeneous environment composed of contexts with different navigation properties. Depending on its internal state and the context-specific navigation capabilities available along alternative routes, the robot may follow the higher-cost blue route through a charging-station context instead of the lower-cost red route. $I(c_i, c_j)$ denotes the interface between contexts c_i and c_j.

In contrast, human navigation naturally exploits contextual information when deciding how to move through different environments, a capability sometimes referred to as intelligent navigation [5]. In robotic navigation, one prominent way of incorporating such context is through semantic information. Semantic trajectories and maps enrich geometric or topological representations with high-level information [6,7] and have been used, for example, for graph-based planning [8] and navigation based on natural-language room descriptions [9]. More recent approaches also integrate semantic or mid-level visual representations directly into planning and control, for example through semantically informed MPC for exploration [10]. These works demonstrate the value of contextual information for navigation [4]. However, they primarily use semantics to enrich or guide navigation within an existing representation or planning formulation. The problem addressed in this work is more general: a single mission may traverse contexts whose appropriate state representations, action models, and dynamics differ fundamentally, requiring the planner to reason not only within each context but also about transitions between heterogeneous representations.

In parallel, several skill-based approaches address context-dependent navigation by selecting predefined behaviors, learned policies, or navigation modules according to the current environmental context, internal state, or high-level goal. Hierarchical and behavior-

based architectures, including those used in modern navigation stacks such as Nav2 [11], often employ finite state machines (FSMs) or behavior trees (BTs) to switch between skills based on context conditions (e.g., task phase, battery level, or terrain type) [11–13]. In such architectures, the executive determines which navigation skill to execute (e.g., path following vs. docking) and when to switch between them, while the individual modules typically plan within their respective representations or operating contexts and transitions are defined at the executive level. Hybrid frameworks that combine learning and modular control [14] further increase flexibility. However, these approaches primarily address the selection and coordination of navigation modes rather than jointly planning local actions and transitions across heterogeneous representations within a single search problem.

Grid–topological hybrid mapping [15] partially addresses scalability by combining divided 2D grid maps with a high-level topological graph for city-scale navigation. By switching between local grid maps via topological path planning, this approach reduces memory usage and planning time. However, transition points between regions are selected as nearest node pairs, which may be suboptimal and do not support region-level transitions where overlapping areas allow entry or exit anywhere along a shared boundary. Moreover, although the regions are connected through a high-level topological graph, navigation within the individual regions relies on a common grid-based representation and navigation strategy. This limits the ability to model contexts that require fundamentally different state representations, action models, or dynamics. Internal state variables (e.g., battery level, payload, or faults) are also typically not incorporated into the planning problem, although they may affect which routes are feasible or preferable.

To address the lack of a unified planning formulation for navigation across contexts with heterogeneous representations, action models, and dynamics, we introduce a modular, context-aware framework that models the environment as a set of distinct navigation contexts connected through explicitly defined interface zones. Each context specifies how the local environment and the robot state within it are represented, a set of actions applicable in that context, and a context-specific dynamics model, thereby allowing navigation behavior to be modeled according to the properties of the corresponding region. The planner selects context-specific navigation actions and context transitions according to the current environmental context, the overall navigation goal, and the robot's internal state. Context transitions can only occur within interface zones, where representation-aware mappings transform the robot state from the source context representation to the corresponding target context representation. Planning is performed incrementally using an A*-style search that considers both intra-context actions and context transitions within the same search graph. When a search state lies within an interface zone, the planner can generate a successor in the neighboring context by mapping the current state into the corresponding representation and accounting for the associated context-switching cost. Internal resource variables are included directly in the search state and updated during successor generation, allowing resource constraints to influence which actions and transitions remain feasible. In this way, the planner exploits the context-specific representations, actions, dynamics, and traversal costs to compute a minimum-cost feasible plan toward the overall navigation goal.

The main contributions of this paper are:

- We propose a modular, context-aware navigation framework that models environments as distinct navigation contexts, each represented by its own representation, applicable action set, and dynamics model, connected via explicit interface zones in which representation-aware mappings enable transitions between the states of neighboring contexts.
- We formulate context-aware path planning as an incremental A*-style search over a joint state space that explicitly represents contexts with different state representations,

applicable actions, dynamics, context transitions, and internal resources (e.g., battery). Under non-negative transition costs and an admissible heuristic, the search returns a minimum-cost plan, if one exists, that reaches the goal while satisfying the context-transition conditions and internal resource constraints represented in the search.

- We evaluate the framework in simulated environments composed of navigation contexts with different representations, state spaces, and traversal models. Compared with a hierarchical planning baseline using fixed representative interface states, the proposed method produces shorter paths, retains more of the modeled internal resource, and reduces planning time. In a restricted grid–topological evaluation, both the proposed method and a strengthened topological baseline recover the shortest path in all evaluated cases, while the proposed method achieves lower average planning time. A nearest-node topological baseline, in contrast, frequently fails or produces suboptimal paths because of its fixed interface-state selection.

In this work, we focus on context-aware planning under the assumption that an appropriate decomposition of the environment into contexts and corresponding interface zones is available. Determining or learning this decomposition automatically is outside the scope of the present work and constitutes a limitation that we plan to address in future work.

The remainder of this paper is organized as follows. Section 2 reviews related work on modular, context-aware, and heterogeneous navigation approaches. Section 3 introduces the proposed context-aware planning framework, including the context representation, state transitions, search formulation, and cost function. Section 4 describes the experimental setup and comparison baselines, while Section 5 presents and analyzes the evaluation results. Finally, Section 6 summarizes the main findings and discusses directions for future work.

2. Related Work

Related work relevant to our problem can be grouped into four complementary directions: (1) modular and adaptive navigation architectures, (2) executive and behavior-based approaches for context-dependent navigation, (3) hybrid and multi-modal planning methods, and (4) approaches for representing or deriving environmental contexts and regions. We discuss these directions separately because they address different aspects of context-aware navigation: (1) how context-specific navigation capabilities are organized and selected, (2) how transitions between context-specific navigation capabilities are coordinated, (3) how planning across multiple modes is formulated, and (4) how the underlying contexts are represented or obtained. Our work focuses on integrating context-specific representations, state spaces, actions, dynamics, transitions, and internal resource states within a single planning problem.

2.1. Modular and Adaptive Navigation Architectures

Autonomous navigation is commonly implemented using modular architectures that separate global path planning, local path execution, and higher-level coordination. In the Robot Operating System (ROS) 1 ecosystem, Move Base [16] became a widely used navigation framework following this architecture: a global planner computes a path toward the navigation goal, while a local planner—more precisely, a controller—generates locally feasible motions for following this path using local environment information. Move Base supports configurable planner and controller plugins, but navigation is organized around a common cost-map-based representation and a fixed global–local planning structure. Nav2 [11] and Move Base Flex (MBF) [17] address limitations of earlier navigation stacks by increasing modularity and allowing different planners, controllers, and recovery behaviors to be configured or selected at run time. This enables a navigation system to choose among

multiple implementations according to the current task or operating conditions. However, this flexibility is primarily achieved at the level of component selection and coordination: the available modules remain part of a common navigation architecture, and transitions between navigation capabilities are handled by a higher-level executive rather than being jointly considered with local motion in a single planning problem. In contrast, our work treats context-specific state representations, applicable actions, dynamics, and transitions between them as part of the planning state and search itself. Autoware [18] provides an end-to-end modular stack for perception, planning, and control in autonomous driving. Its planning architecture is primarily designed around the structured representations and motion constraints of road environments. It therefore addresses modularity within a particular navigation domain, whereas our work considers missions that may traverse contexts requiring fundamentally different representations and navigation models.

Another line of work addresses context-aware navigation through run-time adaptation of the navigation architecture. These approaches respond to changes in the environment or system state by modifying navigation parameters, component configurations, or selected modules, thereby addressing one aspect of context-dependent navigation. MROS (Metacontrol for ROS Systems) [19], for example, provides a meta-control layer that adjusts navigation parameters and component configurations in response to environmental changes while retaining the underlying navigation architecture. Thus, context influences the configuration of an existing navigation system, but it is not represented as part of a planning problem in which different context-specific state representations, actions, and transitions are jointly considered. Bozhinoski et al. [20] build on MROS to realize context-based navigation by dynamically reconfiguring local planner parameters according to the perceived environment. This allows the navigation behavior to change with environmental context, but the context is primarily used to parameterize the local planner rather than to associate different contexts with different state representations and explicitly plan transitions between them. Brugali et al. [21] propose a model-based framework that encodes rules for run-time adaptation of generic robotic solutions and demonstrate it on a hospital logistics navigation system. Their approach addresses context-dependent reconfiguration at the architectural level, whereas our problem concerns how context-specific navigation properties and transitions between heterogeneous representations can be incorporated directly into path-planning. Gherardi and Hochgeschwender [22] present a model-based approach for run-time adaptation, enabling a quadcopter to recover from motor failures by reconfiguring its control architecture. This demonstrates how changes in the robot's internal condition can trigger architectural reconfiguration, but such internal state is not used as part of a joint search over navigation actions, context transitions, and alternative context-specific navigation models. Al Shukairi and Cardoso [14] introduce a hybrid framework that combines learned components with rule-based decision logic for navigation in complex scenarios. While this increases flexibility in selecting or coordinating navigation decisions, it does not explicitly formulate transitions between heterogeneous context-specific representations as decision variables of a unified planning search. Overall, these approaches use contextual information to adapt, configure, or select components of an existing navigation architecture. In contrast, our work incorporates the current context, context-specific navigation model, internal state, and transitions between heterogeneous representations directly into the planning state and search.

2.2. Behavior-Based and Executive Approaches for Context-Dependent Navigation

Behavior-based and hierarchical architectures have also been explored. Wasik and Saffiotti [23] present a hierarchical behavior-based system that composes several basic behaviors to perform multiple vision-based manipulation tasks. Scheutz and Andronache [24]

propose an architecture that supports dynamic changes in behavior-selection strategies. These works demonstrate how different behaviors can be selected or composed according to the current situation. However, context is primarily used to determine which behavior or strategy should be active. They do not address the planning problem considered here, in which context-specific state representations, applicable actions, dynamics, and the transitions and mappings between contexts are jointly considered within a single search while preserving the individual representations of each context.

Beyond component-level and parameter adaptation, several systems use finite state machines (FSMs) or behavior trees (BTs) as high-level executives for context-aware navigation. Höllmann et al. [12] address a problem closely related to ours in long-term agricultural autonomy. Their system combines geo-referenced semantic zones with a waypoint graph and a hierarchical SMACH state machine that selects zone-specific MBF planners, controllers, and recovery behaviors, including docking at a charging container. Their approach therefore supports context-dependent selection of navigation capabilities. In contrast, our work incorporates context transitions and transition states directly into the path-planning search and optimizes these decisions jointly with context-specific motion and internal resource state.

Internal resource states can also influence which navigation behavior should be executed. Existing executive-based approaches commonly handle such dependencies through predefined conditions and mode switches. Battery-aware BTs for aerial inspection, for example, switch between normal waypoint following and return-home-and-recharge behaviors when the battery reaches a predefined threshold [13]. Similarly, the Nav2 navigation stack provides BT condition nodes such as `isBatteryLow`, which can activate docking or recharging subtrees when the corresponding condition becomes true [11]. Comparative studies of BTs and FSMs likewise show how such architectures can encode resource-dependent mode switches, including low-battery recharge behavior [25]. These approaches are effective for selecting among predefined navigation modes based on the current resource state. In our formulation, by contrast, internal resources are included directly in the planning state and evolve together with the robot's motion. This allows the planner to evaluate how alternative actions, routes, and context transitions affect resource availability and to select a minimum-cost route that satisfies the modeled resource constraints, rather than reacting to resource conditions only through predefined mode switches.

2.3. Hybrid and Multi-Modal Planning

A closely related line of work addresses planning problems with multiple discrete modes and mode-specific feasible spaces. Hauser and Latombe [26] formulate multi-modal motion planning as the joint selection of a sequence of modes and continuous motions within each mode. Transitions between modes occur through transition configurations that satisfy the constraints of both adjacent modes, and their Multi-Modal-PRM explicitly samples both mode-specific configurations and transition configurations.

This formulation is conceptually related to our use of contexts and inter-context transitions. However, Hauser and Latombe consider geometric motion-planning modes represented as constrained submanifolds of a common robot configuration space. In our formulation, different contexts may instead use different planning state spaces and representations, such as continuous geometric states, graph states, or discrete task states. Inter-context transitions therefore require explicit representation-aware state mappings, which need not be bijective, and the planning state additionally includes internal resource variables such as battery level.

2.4. Representation and Generation of Environmental Contexts

Beyond navigation architectures, several works address how context representations and regions can be generated automatically from data. Polygonal regions can be extracted from occupancy grids and point clouds using clustering and surface analysis, then annotated semantically and used as planning entities [27,28]. Semantic zone-based map management approaches build on RGB-D SLAM systems and partition 3D maps into semantic zones, with recent work discussing how zone boundaries can be adapted based on robot trajectories and usage patterns [29]. Other methods segment maps into regions and use these as nodes in a topological graph, or employ semantic segmentation and vision–language models to classify the robot's context (e.g., indoor corridor, outdoor terrain, crosswalk) from onboard perception and select context-specific navigation behaviors [30]. These approaches suggest that the context and interface–zone structures assumed to exist by our framework could be at least partially derived or refined from data, rather than being entirely hand-coded.

Within the proposed framework, such perception- or map-based methods could provide the context decomposition and candidate interface regions required by the planner. For example, semantic segmentation or map-region extraction could identify context boundaries, while overlapping or adjacent regions could define candidate interface zones. The representation-aware mappings between the corresponding context-specific state spaces could then be constructed through geometric calibration, graph association, or application-specific semantic mappings. Automatically deriving and validating such mappings between fundamentally different representation types remains outside the scope of the present work and constitutes an important direction for future research.

3. Context-Aware Path Planning Framework

Autonomous navigation across heterogeneous contexts requires planning with context-specific representations, state spaces, actions, and dynamics. For instance, as illustrated in Figure 1, a robot may need to traverse multiple environments, each requiring its own specialized navigation strategy. Moreover, internal factors—such as battery level, payload status, or system faults—can significantly influence the optimal route, necessitating context-aware decision-making that accounts for both external environmental conditions and internal system states.

Each context presents distinct perceptual, planning, and control requirements and may therefore require different navigation models. For example, off-road areas may rely on traversability-based navigation, urban roads on structured graph-based path planning, and open terrain on continuous geometric motion planning. These contexts differ in their navigation properties, including their state representations, applicable actions, and dynamics, making a single fixed navigation model insufficient for representing all of them appropriately.

To address this challenge, we propose a context-aware planning framework that models the environment as a set of distinct contexts, each associated with its own state representation, applicable actions, and dynamics model. The planner reasons jointly about motion within the current context and transitions to other contexts in order to construct a route toward the overall navigation goal. In this way, the planning process can exploit the different navigation properties of the available contexts while also accounting for the robot's internal state.

3.1. Environment Modeling and Context Models

We formalize the environment as a structured composition of contexts, where each context specifies an environment representation, a context-specific robot state space, a set of context-specific actions, and local dynamics.

Let the environment $\mathcal{E}$ be modeled as a finite set of contexts:

$$\mathcal{C} = \{c_1, c_2, \ldots, c_N\}.$$

Each context c_i is defined as a tuple:

$$c_i = (\rho_i,\ \mathcal{X}_i,\ \mathcal{A}_i,\ f_i),$$

where ρ_i denotes the representation used to describe the local environment in context c_i (e.g., geometric, topological, or semantic), $\mathcal{X}_i$ denotes the corresponding state space used to represent the robot for planning in that context, $\mathcal{A}_i$ denotes the set of actions defined for context c_i, with the applicability of an action $a \in \mathcal{A}_i$ depending on the current state, and f_i defines the context-specific dynamics:

$$f_i : \mathcal{X}_i \times \mathcal{A}_i \to \mathcal{X}_i.$$

Membership in $\mathcal{A}_i$ specifies that an action belongs to the context-specific action model; it does not imply that the action is applicable at every state in $\mathcal{X}_i$. Action applicability is state dependent, and f_i is evaluated only for actions that are applicable at the current state.

The environment representation ρ_i and robot state space $\mathcal{X}_i$ are related but need not be identical. For example, a context represented by a 3D voxel map may use a continuous 3D robot pose as its planning state, whereas a road graph may represent the robot state by a graph node or edge, and a semantic context may use a discrete task state.

This formulation allows the same environment representation (e.g., geometric) to appear in multiple contexts—such as a corridor and a narrow tunnel—while the associated state spaces, action sets, dynamics, or constraints may differ. By separating the environment representation from the context identity and robot state space, the framework can model navigation properties specific to each context. Actions are interpreted within the context for which they are defined, while their applicability depends on the current state.

3.2. Contextual State Transitions

The agent is situated within a context $c_i \in \mathcal{C}$, which is associated with an environment representation ρ_i and a context-specific state space $\mathcal{X}_i$. The agent's local state in context c_i is denoted by

$$x^{c_i} \in \mathcal{X}_i.$$

The form of x^{c_i} depends on the state representation used for planning in the corresponding context. For example, it may represent a geometric pose in a continuous environment, a node or edge in a graph-based context, or a discrete task state in a semantic context.

In addition to its context-dependent local state, the agent maintains a set of internal state variables θ, representing system-level conditions such as battery level, possession of objects (e.g., whether the agent has a key), or fault status. The full planning state is therefore expressed as

$$s = (c_i,\ x^{c_i},\ \theta),$$

where x^{c_i} denotes the robot state expressed in the state space $\mathcal{X}_i$ associated with the current context c_i, and θ contains internal variables that are not part of the context-specific state representation.

We distinguish between *intra-context transitions* and *inter-context transitions*. An intra-context transition remains within the current context c_i and is governed by the context-specific dynamics f_i and an applicable action $a \in \mathcal{A}_i$:

$$x^{c_i\prime} = f_i(x^{c_i}, a).$$

The internal state variables may additionally change according to the effects of the executed action.

An *inter-context transition* changes the active context from c_i to an overlapping context c_j. Such a transition can only be considered at an explicitly defined *interface zone*. For a directed transition from c_i to c_j, the interface zone is defined as

$$I_{ij} \subseteq \mathcal{X}_i.$$

A state $x^{c_i} \in I_{ij}$ represents a situation in which the robot state expressed in context c_i can also be represented in the state space associated with context c_j. Thus, the interface zone identifies the source-context states for which a transition between the two context-specific representations can be established. In geometric contexts, such an interface may correspond to an overlapping area or a shared boundary. Between heterogeneous representations, it may, for example, associate a geometric pose with a graph state or with a semantic task state.

For an interface I_{ij}, a representation-aware mapping is defined as

$$\mathrm{map}_{ij} : I_{ij} \to \mathcal{X}_j.$$

Given a source state $x^{c_i} \in I_{ij}$, the mapping provides the corresponding state in the target context:

$$x^{c_j} = \mathrm{map}_{ij}(x^{c_i}).$$

The mapping depends on the representations associated with c_i and c_j and may involve, for example, coordinate transformations, associating a geometric position with a graph node or edge, or converting a geometric entry state into a semantic task state.

The mapping is not required to be injective or bijective. In particular, multiple states in the source state space may map to the same state in the target state space:

$$x_1^{c_i} \neq x_2^{c_i} \quad \text{while} \quad \mathrm{map}_{ij}(x_1^{c_i}) = \mathrm{map}_{ij}(x_2^{c_i}).$$

This allows transitions between representations with different levels of granularity. For example, multiple geometric poses within an entrance area may correspond to the same semantic state *enter*, or several geometric states may be associated with the same node in a graph-based representation. Consequently, mappings in opposite directions are defined independently and are not generally inverses of one another.

The existence of an interface and a corresponding state mapping does not necessarily imply that an inter-context transition is permitted under the current planning state. We therefore define the context-transition predicate

$$\psi(s, c_i, c_j) = \begin{cases} 1, & \text{if the transition from } c_i \text{ to } c_j \text{ is permitted at } s, \\ 0, & \text{otherwise.} \end{cases}$$

The predicate may account for additional transition conditions associated with the current planning state, including internal state variables.

An inter-context transition from c_i to c_j is considered only if

$$x^{c_i} \in I_{ij} \quad \text{and} \quad \psi(s, c_i, c_j) = 1.$$

If both conditions are satisfied, the active context changes to c_j, the context-specific state is transformed using the representation-aware mapping, and the resulting planning state is

$$\hat{s} = \left(c_j, \text{map}_{ij}(x^{c_i}), \theta\right).$$

This separation distinguishes representational compatibility at an interface from state-dependent transition feasibility. Interface membership and the representation-aware mapping determine where a transition can be represented, whereas the transition predicate captures additional feasibility conditions determined by the robot's current state.

3.3. Search Space and Incremental Planning

To perform context-aware planning across heterogeneous context-specific representations, the planner constructs the search graph incrementally rather than relying on a fixed global graph. States and transitions are generated on the fly according to the state space, applicable actions, and dynamics of the currently considered context, while inter-context transitions are incorporated into the same construction process.

Let the incrementally constructed search graph be

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}),$$

where each node $v \in \mathcal{V}$ represents a planning state

$$\sigma(v) = s = (c_i, x^{c_i}, \theta).$$

Here, c_i denotes the current context, $x^{c_i} \in \mathcal{X}_i$ is the corresponding context-specific robot state, and θ contains the internal state variables.

The edge set consists of intra-context and inter-context transitions:

$$\mathcal{E} = \mathcal{E}_{\text{intra}} \cup \mathcal{E}_{\text{inter}}.$$

An intra-context edge represents the execution of a non-empty finite sequence of context-specific actions that is applicable from the current planning state:

$$e_{\text{intra}} = \left(s, s', \mathbf{a}\right),$$

where

$$\mathbf{a} = (a_1, \ldots, a_n) \in \mathcal{A}_i^+$$

is an ordered sequence of actions associated with context c_i, with $a_k \in \mathcal{A}_i$ for $k = 1, \ldots, n$.

Action applicability is state-dependent. An action sequence $\mathbf{a}$ is applicable from the current planning state $s = (c_i, x^{c_i}, \theta)$ only if every action a_k is applicable at the intermediate state resulting from the execution of the preceding actions $a_1, \ldots, a_{k-1}$. The actions are applied sequentially according to the context-specific dynamics f_i, while the internal state variables are updated according to the corresponding action effects.

Both endpoint states therefore belong to the same context:

$$s = (c_i, x^{c_i}, \theta), \qquad s' = (c_i, x'^{c_i}, \theta').$$

The context-specific state $x^{c_i\prime}$ and internal state θ' result from the sequential execution of the applicable action sequence $\mathbf{a}$ from s.

An inter-context edge represents a transition from context c_i to context c_j:

$$e_{\text{inter}} = \left(s, \hat{s}, \tau_{ij}\right),$$

where τ_{ij} denotes the inter-context transition from c_i to c_j. Such an edge can only be generated if

$$x^{c_i} \in I_{ij} \quad \text{and} \quad \psi(s, c_i, c_j) = 1.$$

The corresponding successor state is obtained through the representation-aware mapping:

$$\hat{s} = \left(c_j, \, \text{map}_{ij}(x^{c_i}), \, \theta\right).$$

Consequently, intra-context actions and inter-context transitions are successor alternatives within the same search. The planner can therefore reason jointly about progress within a context and the choice of when and where to transition to another context, rather than first computing a complete path within one context and subsequently selecting a context transition.

3.4. Cost Function

The planner follows the standard A^* evaluation rule. For each search state

$$s = (c_i, x^{c_i}, \theta),$$

the priority value is

$$f(s) = g(s) + h(s), \tag{1}$$

where $g(s)$ denotes the accumulated cost from the start state to s, and $h(s)$ is a heuristic estimate of the remaining cost to the goal. The heuristic used in this work is defined in the section titled Context-Connectivity Heuristic.

Let a plan be an ordered sequence of search-graph edges

$$P = (e_1, e_2, \ldots, e_T),$$

where edge e_m connects planning state s_{m-1} to s_m. The total cost of the plan is

$$C(P) = g(s_T) = \sum_{m=1}^{T} c(e_m), \tag{2}$$

where $c(e_m)$ denotes the cost associated with edge e_m.

Since the search graph distinguishes between intra-context and inter-context edges, their costs are defined accordingly:

$$c(e_m) = \begin{cases} c_{\text{effort}}(\mathbf{a}_m), & e_m \in \mathcal{E}_{\text{intra}}, \\ c_{\text{switch}}(c_i, c_j, s_{m-1}), & e_m \in \mathcal{E}_{\text{inter}}, \end{cases} \tag{3}$$

where $c_{\text{effort}}(\mathbf{a}_m)$ is the accumulated cost of executing the action sequence $\mathbf{a}_m$ associated with an intra-context edge. Depending on the application, this cost may represent travel distance, execution time, energy consumption, risk, or a weighted combination of these quantities.

For an inter-context edge from c_i to c_j, $c_{\text{switch}}(c_i, c_j, s_{m-1})$ represents the cost associated with performing the context transition at the current state. This cost can capture transition-specific effort or penalties that are not represented by the intra-context action cost.

All edge costs are assumed to be non-negative.

Context-Connectivity Heuristic

Because the context-specific state spaces may use fundamentally different representations, a geometric-distance heuristic cannot be applied in general across the complete search space. We therefore construct a directed context-connectivity graph

$$\mathcal{G}_{\mathcal{C}} = (\mathcal{C}, \mathcal{E}_{\mathcal{C}}), \tag{4}$$

where

$$(c_i, c_j) \in \mathcal{E}_{\mathcal{C}}$$

if an inter-context transition from c_i to c_j is structurally defined through an interface I_{ij} and the corresponding representation-aware mapping.

Each edge of the context-connectivity graph is assigned a non-negative lower bound $\underline{w}_{ij}$ on the cost of an inter-context transition from c_i to c_j. We define

$$\underline{w}_{ij} = \inf\{c_{\text{switch}}(c_i, c_j, s) \mid s = (c_i, x^{c_i}, \theta),\ x^{c_i} \in I_{ij},\ \psi(s, c_i, c_j) = 1\}. \tag{5}$$

Hence, $\underline{w}_{ij}$ is no greater than the cost of any admissible inter-context transition from c_i to c_j.

The navigation goal may be representable in more than one context. This can occur, for example, when the goal lies within an interface between two contexts. We therefore define the set of goal contexts as

$$\mathcal{C}_G = \{c_i \in \mathcal{C} \mid \exists\, s = (c_i, x^{c_i}, \theta) \text{ such that } G(s) = 1\}, \tag{6}$$

where $G(s)$ denotes the goal condition.

Importantly, membership in $\mathcal{C}_G$ does not imply that the current planning state satisfies the goal. A state may belong to a goal context while the actual goal cannot be reached directly from that state within the same context. The planner may therefore leave a goal context, traverse other contexts, and later re-enter it through another interface. Search termination occurs only when the current planning state satisfies $G(s) = 1$.

The context-level cost-to-go is initialized as

$$H_{\mathcal{C}}(c_i) = 0, \qquad c_i \in \mathcal{C}_G. \tag{7}$$

For all other contexts, $H_{\mathcal{C}}$ is the minimum accumulated lower-bound inter-context transition cost required to reach any context in $\mathcal{C}_G$:

$$H_{\mathcal{C}}(c_i) = \min_{(c_i, c_j) \in \mathcal{E}_{\mathcal{C}}} \Big(\underline{w}_{ij} + H_{\mathcal{C}}(c_j)\Big). \tag{8}$$

The values $H_{\mathcal{C}}$ are computed once for each planning query by running a multi-source Dijkstra search from all goal contexts $c_g \in \mathcal{C}_G$ on the reversed context-connectivity graph, which yields the minimum cost from each context to any goal context in the original graph. If no sequence of inter-context transitions connects a context c_i to any goal context, then

$$H_{\mathcal{C}}(c_i) = +\infty. \tag{9}$$

For a planning state

$$s = (c_i, x^{c_i}, \theta),$$

the context-connectivity heuristic is defined as

$$h(s) = H_{\mathcal{C}}(c_i). \tag{10}$$

The heuristic considers only lower bounds on inter-context transition costs and ignores intra-context action costs and internal-resource constraints. Since intra-context edge costs are non-negative and removing resource constraints can only enlarge the set of feasible

context sequences, the context-connectivity problem constitutes a relaxation of the original planning problem. Any goal-reaching plan induces a sequence of context transitions whose actual transition costs are at least as large as the corresponding lower bounds $\underline{w}_{ij}$. Consequently,

$$h(s) \leq C^*(s),$$

where $C^*(s)$ denotes the minimum cost of a goal-reaching plan from s. The heuristic is therefore admissible.

To establish consistency of the heuristic with respect to the complete search graph, we consider both intra-context and inter-context edges. Recall that consistency requires

$$h(s) \leq c(e) + h(s')$$

for every search-graph edge e from state s to successor state s'.

For an intra-context edge $e = (s, s', \mathbf{a})$, the active context does not change. Since the context-connectivity heuristic depends only on the active context,

$$h(s) = H_{\mathcal{C}}(c_i) = h(s').$$

Because all edge costs are non-negative,

$$h(s) = h(s') \leq c(e) + h(s').$$

Thus, the consistency condition is satisfied for intra-context edges.

For an inter-context edge from c_i to c_j, the shortest-path property of $H_{\mathcal{C}}$ gives

$$H_{\mathcal{C}}(c_i) \leq \underline{w}_{ij} + H_{\mathcal{C}}(c_j).$$

Furthermore, by construction,

$$\underline{w}_{ij} \leq c_{\text{switch}}(c_i, c_j, s).$$

Since $h(s) = H_{\mathcal{C}}(c_i)$, $h(s') = H_{\mathcal{C}}(c_j)$, and $c(e) = c_{\text{switch}}(c_i, c_j, s)$, it follows that

$$\begin{aligned} h(s) &= H_{\mathcal{C}}(c_i) \\ &\leq \underline{w}_{ij} + H_{\mathcal{C}}(c_j) \\ &\leq c_{\text{switch}}(c_i, c_j, s) + H_{\mathcal{C}}(c_j) \\ &= c(e) + h(s'). \end{aligned}$$

Thus, the consistency condition is also satisfied for inter-context edges.

When additional lower bounds on intra-context costs are available, the heuristic can be strengthened. Let

$$\underline{d}_{c_i}(x^{c_i}, I_{ij})$$

denote a lower bound on the intra-context cost of reaching interface I_{ij} from the current state x^{c_i}. Furthermore, if $c_i \in \mathcal{C}_G$, let

$$\underline{d}_{G,c_i}(x^{c_i})$$

denote a lower bound on the cost of reaching a state satisfying the goal condition within the current context. If no such lower bound is available, $\underline{d}_{G,c_i}$ is set to zero for goal contexts; for contexts not in $\mathcal{C}_G$, it is set to $+\infty$.

The strengthened heuristic can then be written as

$$h_{\mathrm{str}}(s) = \min\left\{ \underline{d}_{G,c_i}(x^{c_i}),\ \min_{(c_i,c_j)\in\mathcal{E}_{\mathcal{C}}} \left[\underline{d}_{c_i}(x^{c_i}, I_{ij}) + \underline{w}_{ij} + H_{\mathcal{C}}(c_j) \right] \right\}. \tag{11}$$

The first term represents the possibility of reaching the goal directly within the current context, whereas the second term represents the possibility of first leaving through an interface and continuing through other contexts. Consequently, the formulation also covers cases in which the current state already belongs to a goal context but the minimum-cost or only feasible route requires leaving that context and re-entering it later.

If $\underline{d}_{c_i}(x^{c_i}, I_{ij})$ is unavailable, it may be set to zero. The strengthened heuristic remains admissible provided that all local quantities used in it are valid lower bounds. If consistency of the strengthened heuristic is required, the local lower-bound functions must additionally satisfy the corresponding consistency conditions with respect to intra-context transitions.

For $d_{c_i}(x^{c_i}, I_{ij})$ to be a valid lower bound, it must not exceed the minimum accumulated intra-context cost required to reach any state in interface I_{ij} from the current state. Similarly, $d_{G,c_i}(x^{c_i})$ must not exceed the minimum intra-context cost required to reach a state satisfying the goal condition within c_i. These quantities may be obtained from a relaxation of the corresponding local planning problem, provided that the relaxation cannot overestimate the true feasible cost. If no informative lower bound is available, zero may be used.

Under these conditions, any goal-reaching plan must either reach the goal within the current context or first leave through some outgoing interface I_{ij}. In the first case, $d_{G,c_i}(x^{c_i})$ lower-bounds the cost of that plan. In the second case, $d_{c_i}(x^{c_i}, I_{ij}) + w_{ij} + H_C(c_j)$ lower-bounds the cost of the plan when I_{ij} is its first inter-context transition. Since h_{str} takes the minimum over these alternatives, $h_{\mathrm{str}}(s) \leq C^*(s)$, and the strengthened heuristic is therefore admissible.

If the strengthened heuristic is admissible but its consistency cannot be guaranteed, A* can retain optimality provided that previously expanded states are reopened whenever a lower g-value is discovered. Without such reopening, graph-search A* with an inconsistent heuristic does not generally retain its optimality guarantee.

When the local lower bounds are non-negative, the strengthened heuristic incorporates additional information about the cost of reaching an interface or the goal within the current context and can therefore be more informative than the context-only heuristic. If the local bounds are set to zero, the formulation reduces to the simpler context-level estimate. The experiments in this work use the context-connectivity heuristic h, for which consistency has been established above; empirical evaluation of alternative strengthened local bounds is left for future work.

The complete planning procedure, including the construction of the context-connectivity heuristic and the joint expansion of intra-context and inter-context transitions, is summarized in Algorithm 1. The algorithm first identifies the set of goal contexts and constructs the directed context-connectivity graph, from which the heuristic values $H_{\mathcal{C}}$ are computed by a reverse multi-source Dijkstra search. The start state is then inserted into the A* open list. At each iteration, the state with minimum $f(s) = g(s) + h(s)$ is selected. If the goal condition is satisfied, the corresponding plan is returned. Otherwise, the planner generates both inter-context and intra-context successors. Inter-context successors are generated when the current state lies in a valid interface and the corresponding transition predicate is satisfied, while intra-context successors are obtained from action sequences applicable from the current planning state. For every feasible successor, the accumulated cost is updated and the state is inserted into or updated in the open list. The search continues until a goal state is reached or the open list becomes empty.

Algorithm 1 Context-Aware Incremental Planning with Reverse Context Heuristic

Require: Contexts $\mathcal{C} = \{c_1, \ldots, c_N\}$, directed interfaces I_{ij} and mappings map_{ij}, start state s_{start}, goal condition G

Ensure: A minimum-cost goal-reaching plan over the generated search graph that satisfies the modeled context-transition and resource constraints, if such a plan exists.

Determine the set of goal contexts

$$\mathcal{C}_G = \{c_i \in \mathcal{C} \mid \exists\, s = (c_i, x^{c_i}, \theta) \text{ such that } G(s) = 1\}$$

if $\mathcal{C}_G = \emptyset$ **then**
  return failure
end if
Construct the directed context-connectivity graph $\mathcal{G}_{\mathcal{C}} = (\mathcal{C}, \mathcal{E}_{\mathcal{C}})$ from the defined interfaces and representation-aware mappings
Assign each context edge $(c_i, c_j) \in \mathcal{E}_{\mathcal{C}}$ its non-negative lower-bound transition cost $\underline{w}_{ij}$
Compute $H_{\mathcal{C}}$ by running multi-source Dijkstra from all $c_g \in \mathcal{C}_G$ on the reversed context graph
Let $s_{\text{start}} = (c_{\text{start}}, x^{c_{\text{start}}}, \theta_{\text{start}})$
if $H_{\mathcal{C}}(c_{\text{start}}) = +\infty$ **then**
  return failure
end if
Initialize the open list with s_{start}
$g(s_{\text{start}}) \leftarrow 0$
while open list is not empty **do**
  Select and remove a state $s = (c_i, x^{c_i}, \theta)$ with minimum $f(s) = g(s) + h(s)$
  if $G(s) = 1$ **then**
    return reconstructed plan P
  end if

  // Inter-context expansion
  for each $(c_i, c_j) \in \mathcal{E}_{\mathcal{C}}$ **do**
    if $x^{c_i} \in I_{ij}$ **and** $\psi(s, c_i, c_j) = 1$ **and** $H_{\mathcal{C}}(c_j) < +\infty$ **then**
      $x^{c_j} \leftarrow \text{map}_{ij}(x^{c_i})$
      $\hat{s} \leftarrow (c_j, x^{c_j}, \theta)$
      $c_{\text{inter}} \leftarrow c_{\text{switch}}(c_i, c_j, s)$
      $g_{\text{new}} \leftarrow g(s) + c_{\text{inter}}$
      if $\hat{s}$ has not been reached **or** $g_{\text{new}} < g(\hat{s})$ **then**
        $g(\hat{s}) \leftarrow g_{\text{new}}$
        Store s as predecessor of $\hat{s}$ and τ_{ij} as the corresponding edge
        Insert or update $\hat{s}$ in the open list
      end if
    end if
  end for

  // Intra-context expansion
  Generate candidate action sequences $\mathbf{a} \in \mathcal{A}_i^+$ that are applicable from s
  for each candidate action sequence $\mathbf{a}$ **do**
    Simulate $\mathbf{a}$ by repeated application of f_i to obtain x'^{c_i}
    $\theta' \leftarrow \text{update}_\theta(\theta, \mathbf{a})$
    if (x'^{c_i}, θ') satisfies the modeled state and resource constraints **then**
      $s' \leftarrow (c_i, x'^{c_i}, \theta')$
      $c_{\text{intra}} \leftarrow c_{\text{effort}}(\mathbf{a})$
      $g_{\text{new}} \leftarrow g(s) + c_{\text{intra}}$
      if s' has not been reached **or** $g_{\text{new}} < g(s')$ **then**
        $g(s') \leftarrow g_{\text{new}}$
        Store s as predecessor of s' and $\mathbf{a}$ as the corresponding edge
        Insert or update s' in the open list
      end if
    end if
  end for
end while
return failure

3.5. Computational Complexity and Scalability

The computational cost of the proposed planner depends primarily on the number of planning states generated during the incremental search. Let $N = |\mathcal{C}|$ denote the number of contexts and $|E_C|$ the number of directed context connections. Computing the context-connectivity heuristic using multi-source Dijkstra requires $O((N + |E_C|) \log N)$ time with a binary heap.

For the main A* search, let $|V_R|$ and $|E_R|$ denote the number of reached planning states and generated transitions, respectively. Priority-queue operations require approximately $O(|E_R| \log |V_R|)$ time, in addition to the application-specific cost of successor generation. The effective branching factor depends on both the number of applicable intra-context actions and the number of feasible inter-context transitions. Finer discretization of interface regions and internal resource variables further increases the joint state space. As with conventional A*, the worst-case search effort may therefore grow exponentially with solution depth.

For larger problems, stronger admissible heuristics, weighted or anytime A* variants, incremental replanning, and pruning or adaptive sampling of large interface regions may be used to trade optimality, computation time, and memory consumption. A systematic evaluation of these strategies is left for future work.

4. Experimental Setup

The proposed framework combines several aspects of context-aware planning, including context-specific state and environment representations, applicable actions and dynamics, representation-aware inter-context transitions, and internal resource variables. Since no single baseline considered in this study supports all of these aspects simultaneously, we evaluate the proposed approach in two complementary settings, each designed to isolate a different part of the framework.

Direct quantitative comparison with semantic, learning-based, or general multi-modal navigation systems is not straightforward because these approaches typically address partially different components of the navigation problem. Many semantic and learning-based methods focus on perception, semantic representation, behavior selection, or policy generation [10,14,30], whereas the present work assumes the context decomposition as input and focuses on planning jointly across context-specific state representations, actions, and representation-aware transitions. Likewise, classical multi-modal motion-planning formulations generally consider multiple modes within a common configuration-space representation [26]. The selected baselines are therefore intended as controlled comparisons that isolate the planning decisions central to this work rather than as a comprehensive benchmark against all contemporary navigation architectures.

First, we evaluate the complete framework in an environment composed of five navigation contexts using three different representation types and battery as an internal resource variable. In this setting, we compare against a hierarchical multi-context planner to investigate the effect of jointly searching intra-context actions and inter-context transition states rather than separating context-level and local planning.

Second, we consider a restricted grid–topological setting without internal resource modeling to enable a focused comparison with the navigation approach presented in [15]. This setting evaluates the treatment of transitions between regions, in particular the difference between selecting fixed entry/exit states and reasoning over the available interface states during planning.

4.1. Evaluation of Context-Aware Planning

To evaluate the proposed context-aware planning framework, we construct a simulation environment (Figure 2) containing five navigation contexts with different environment representations, planning state spaces, applicable actions, and dynamics. Forest-Offroad and Plain use continuous polygon-based representations, Urban-Road and Forest-Road use graph-based representations, and the Charging-Station context is modeled by discrete semantic task states.

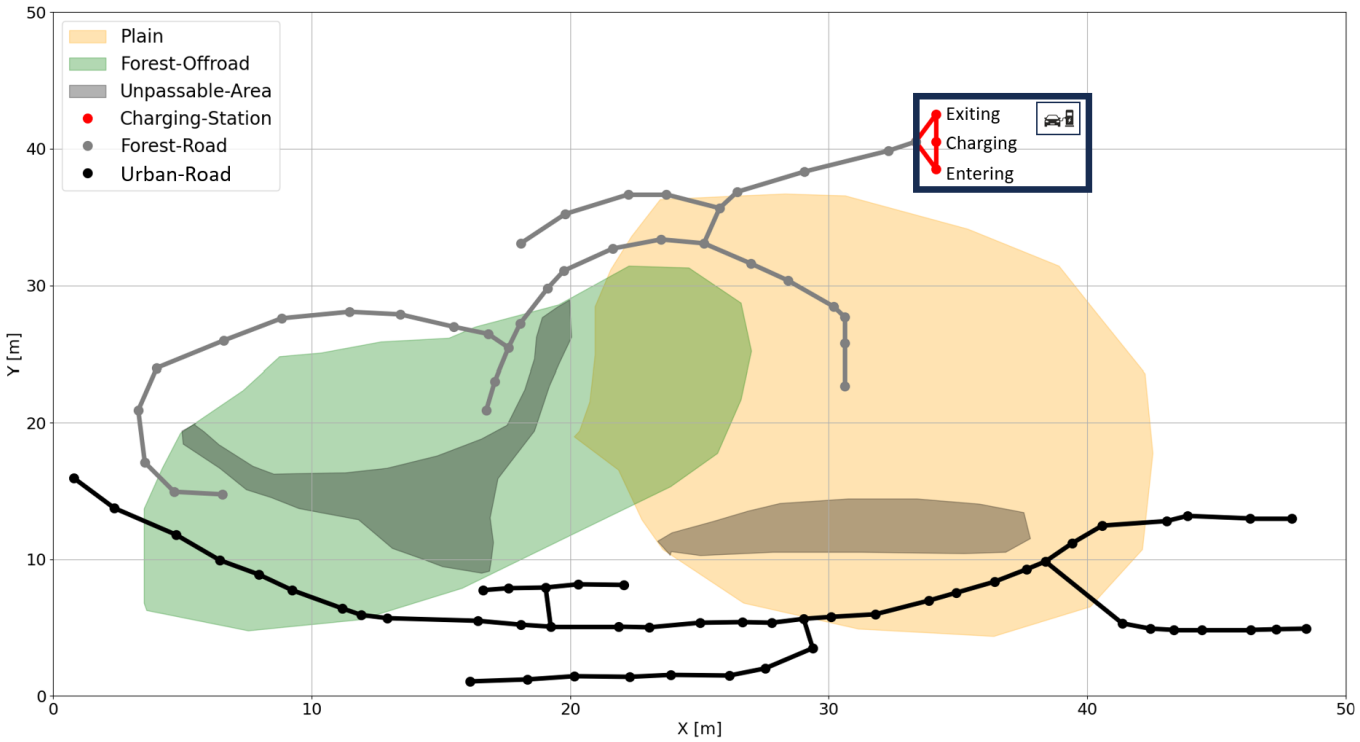


Figure 2. Simulation environment illustrating the different navigation contexts and their interfaces across heterogeneous representations.

The Forest-Offroad and Plain contexts represent continuous traversable areas. The robot state is expressed geometrically, and successor states are generated by applying kinematic motion actions in continuous space. Although both contexts use geometric state spaces, their applicable motions and traversal properties may differ according to the corresponding terrain.

Urban-Road and Forest-Road are represented as discrete graphs. In these contexts, the robot state is represented by a graph node or edge, and intra-context actions correspond to traversing graph connections. Thus, moving from a continuous terrain context to a road context requires not only changing the traversal model but also mapping the robot state from a continuous geometric state space to a discrete graph-based state space. Urban-Road and Forest-Road additionally use different traversal dynamics to represent the different properties of structured urban roads and rough forest roads.

The Charging-Station context does not perform geometric path planning. Instead, it represents the service procedure as a discrete task-level state space,

$$x^{c_{\mathrm{ch}}} \in \{\mathrm{enter}, \mathrm{charge}, \mathrm{exit}\}.$$

The charging operation is modeled as a discrete symbolic action rather than as a continuous-time charging process. Each execution of the charging action updates the robot's internal battery state according to a predefined charging effect and incurs the corresponding

fixed action cost. Repeated charging actions are permitted when additional battery capacity is required. The remaining symbolic actions change the task-level state between *enter*, *charge*, and *exit*. Entering or leaving this context therefore requires a representation-aware mapping between a spatial navigation state and the corresponding task-level state.

The resulting planning problem is therefore not defined only by different traversal costs in different parts of the environment. A complete route may contain continuous geometric motion, discrete graph traversal, symbolic task actions, and inter-context transitions between their corresponding state representations. The proposed planner considers all of these successor types within the same search while preserving the context-specific representations and dynamics. Traversal and energy costs are then used to distinguish alternative plans within this heterogeneous planning problem.

4.1.1. Internal State Modeling

Beyond the context-specific planning state, the robot maintains battery level as an internal resource variable. The battery level is constrained to the modeled range $0 \leq b \leq 100$, where b denotes the remaining battery percentage. The battery depletes incrementally with each action, with the rate of consumption determined by both the context's dynamics and the action's traversal effort. High-effort contexts, such as off-road regions and forest roads, drain the battery more rapidly, while structured contexts such as urban roads enable more energy-efficient movement. Including battery in the planning state allows resource availability to influence which intra-context actions, inter-context transitions, and goal-reaching plans remain admissible.

4.1.2. Scenario Configurations

To evaluate how internal resource state influences the resulting plan, we define two scenario configurations that differ only in the robot's initial battery level. Both scenarios use the same environmental layout, context definitions, start state, goal condition, and planning-cost parameters. The robot starts in the *Forest-Offroad* context on the right side of the environment, while the goal is located in the *Forest-Offroad* context on the left side, as shown in Figure 2.

In the first scenario, the robot starts with sufficient battery to reach the goal without recharging. The planner therefore selects the minimum-cost route according to the defined planning objective while satisfying the modeled resource constraints. Depending on the chosen cost metric, this route need not correspond to either the geometrically shortest or the most energy-efficient path.

In the second scenario, the robot starts with a battery level that is insufficient for any goal-reaching plan that avoids recharging. The resource constraint therefore requires the planner to include the *Charging-Station* context in the plan. The planner must jointly determine how to reach the charging station, where to transition between the involved contexts, and how to continue toward the goal after charging, while minimizing the same planning objective.

4.1.3. Planning Objective and Evaluation Metrics

The cost function introduced in Section 3.4 is instantiated using fixed context-specific action costs and context-switching costs that remain unchanged across all experiments. These costs define the planning objective and determine the preference between alternative routes.

Battery is modeled separately as an internal resource variable and constrains which plans are admissible. To characterize the resulting plans and the computational performance of the planner, we report traveled distance, number of context transitions, remaining battery

at the goal, and planning runtime. Search expansions and local planner calls are additionally reported to describe computational effort.

4.1.4. Hierarchical Multi-Context Planning Baseline

To evaluate the proposed joint planning formulation, we introduce a hierarchical multi-context planning baseline for comparison. The *Hierarchical Multi-Context Planner* (HMCP) is a deliberately simplified hierarchical baseline inspired by the separation between high-level context selection and context-specific navigation found in architectures such as the system presented by Höllmann et al. [12]. HMCP is not a reimplementation of that system and is not intended to represent the full class of hierarchical context-aware navigation approaches. Instead, it is constructed to compare the proposed joint search with a planning structure that separates context-level transition selection from context-specific planning.

HMCP uses the same context decomposition, context-specific state spaces, applicable actions and dynamics, interface zones, representation mappings, costs, and battery model as the proposed method. The difference lies in how these elements are combined during planning. Whereas the proposed method searches intra-context actions and inter-context transitions jointly, HMCP performs a high-level search over interfaces and invokes a context-specific planner to connect successive interface states.

To obtain a finite high-level search graph, HMCP assigns one fixed representative state to each directed interface I_{ij}. For polygonal interfaces, the geometric centroid is used when it lies inside the interface; otherwise, a point-on-surface operation provides a state inside the interface. For a graph-based context, this geometric location is associated with the nearest graph node or edge whose geometric embedding lies within the corresponding interface. Reachability of the representative state from the current state is subsequently evaluated by the context-specific planner.

The HMCP high-level state is

$$q = (c_i, x_{\text{in}}^{c_i}, b),$$

where c_i denotes the current context, $x_{\text{in}}^{c_i} \in \mathcal{X}_i$ is the robot state at which the current context was entered, and b is the remaining battery level. For the start context, $x_{\text{in}}^{c_i}$ is the initial robot state; after an inter-context transition, it is obtained through the corresponding representation-aware mapping.

For every outgoing interface I_{ij}, let

$$\bar{x}_{ij}^{c_i} \in I_{ij}$$

denote its fixed representative state in the state space of context c_i. The context-specific planner then searches for an intra-context action sequence

$$\mathbf{a}_{ij} \in \mathcal{A}_i^+$$

that transforms the entry state into the representative interface state according to the dynamics f_i:

$$x_{\text{in}}^{c_i} \xrightarrow{\mathbf{a}_{ij}} \bar{x}_{ij}^{c_i}.$$

The context-specific local planner therefore computes an intra-context action sequence $\mathbf{a}_{ij}$ as defined in Section 3.3.

If such an action sequence exists and satisfies the modeled resource constraints, it generates a successor of the high-level search state:

$$q' = \left(c_j, \text{map}_{ij}(\bar{x}_{ij}^{c_i}), b - E(\mathbf{a}_{ij})\right),$$

with high-level transition cost

$$c_H(q, q') = c_{\text{effort}}(\mathbf{a}_{ij}) + c_{\text{switch}}(c_i, c_j, \bar{x}_{ij}^{c_i}).$$

Uniform-cost search is performed over the resulting interface-level search graph. Repeated visits to the same context are permitted because a context may be entered through different interfaces, at different context-specific states, and with different remaining resource levels. Consequently, two high-level states associated with the same context need not represent the same planning situation. Reaching a context in which the goal can be represented is not sufficient for termination; the context-specific planner must additionally connect the current entry state to a state satisfying the goal condition.

The use of a single representative state per interface is a deliberate configuration of the HMCP baseline rather than an inherent property of hierarchical planning. A hierarchical variant could instead consider multiple candidate transition states per interface, potentially improving solution quality at the cost of a larger high-level search and additional context-specific planning queries. In this evaluation, HMCP uses one representative state per interface to provide a simple hierarchical baseline in which interface-state selection is fixed before the corresponding context-specific planning query is performed.

4.2. Comparison with Topological Navigation Baselines

The second evaluation isolates the treatment of transitions between local navigation areas. For this comparison, the environment is restricted to a grid–topological representation similar to [15]. The areas $M_1, \ldots, M_5$ are represented as local grid maps, while a topological graph specifies which areas can be connected. A topological edge therefore indicates that a transition between two local maps is possible; the actual motion between an entry and an exit state is computed within the corresponding local map.

For this experiment, the cost of a high-level transition is defined by the length of the corresponding local path. The same distance-based cost metric is used for all compared planners. Accordingly, cost-optimal refers to the shortest overall path within the considered grid–topological planning problem.

Figure 3 shows the experimental environment. Across the 50 test cases, the start state is sampled from M_1, while the goal lies in M_4. The environment contains alternative sequences of local maps connecting start and goal. In particular, one alternative requires traversal through the obstacle-rich area M_5, while another avoids it. The experiment therefore evaluates both whether a planner can identify connected transition states and whether it can choose transition states whose resulting local paths lead to a lower-cost overall plan.

Topo-Map-n: For every transition between neighboring local maps, a single entry/exit pair is selected according to geometric proximity. Planning then proceeds using these fixed transition states. This may lead to failure if the selected states cannot be connected by the local planner, and even when they are connected, the selected pair may lead to a higher-cost local path than another transition state within the same interface.

Topo-Map-s: For each transition between neighboring local maps, Topo-Map-s considers all available discrete candidate entry/exit state pairs associated with the corresponding interface. For each candidate pair, the local planner determines whether the states can be connected and, if so, computes the corresponding path length. All connected candidate pairs are retained as alternatives in the high-level search, which selects the combination that minimizes the total path length.

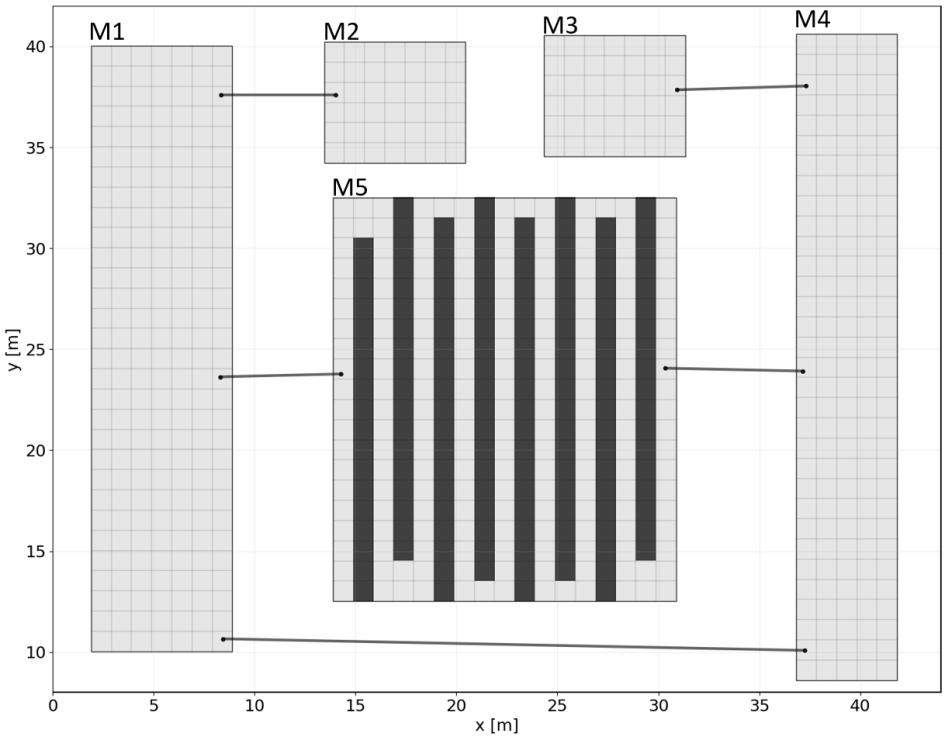


Figure 3. Simulation environment used to evaluate planning performance against the topological navigation baselines. The local grid-map areas M1 and M4 contain the possible start and goal locations, M5 is obstacle-rich, and M2 and M3 form dead-end branches.

5. Results

We evaluate the proposed framework in a five-context heterogeneous environment under two initial battery conditions and compare it with HMCP. Additional experiments compare the method with purely topological baselines in a restricted grid–topological setting.

5.1. Evaluation of Context-Aware Navigation

This section examines the behavior of the proposed planner in a heterogeneous environment with varying traversal costs, representation types, and explicitly defined interface zones. Two scenarios with different initial battery levels are analyzed to examine how the internal resource state affects the joint selection of intra-context actions and inter-context transitions under the same planning objective.

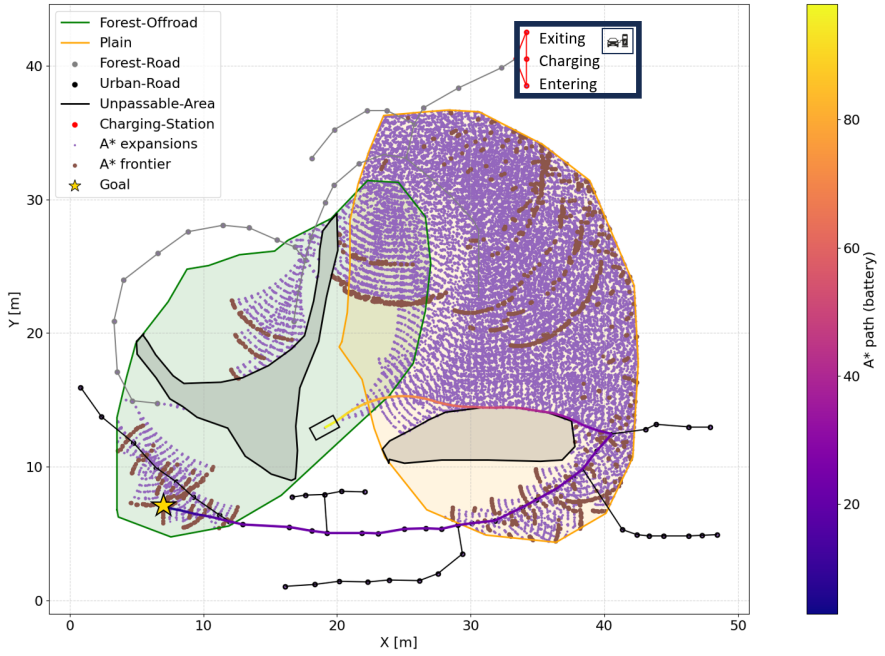
5.1.1. Behavior Analysis

Both scenarios use the same environment, context definitions, start and goal conditions, and planning-cost parameters; only the initial battery level is changed.

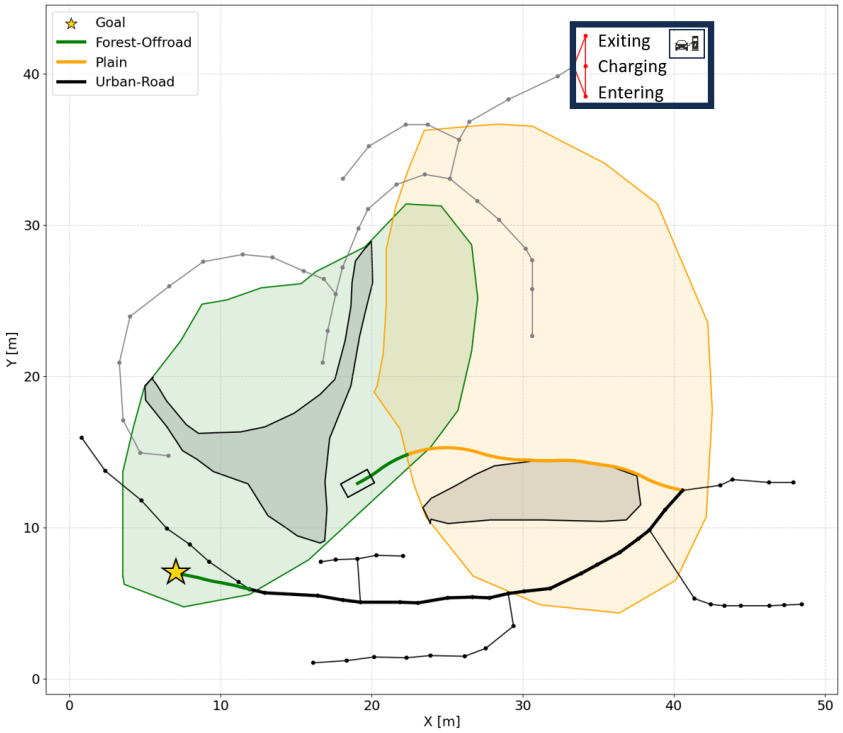
In the *direct-goal* scenario, the resulting plan is

$$\texttt{Forest-Offroad} \rightarrow \texttt{Plain} \rightarrow \texttt{Urban-Road} \rightarrow \texttt{Forest-Offroad}.$$

The route travels 58.592 m, contains three inter-context transitions, and reaches the goal with 2.80% battery remaining. Under the defined planning costs, the planner uses the *Urban-Road* context for a large part of the route because traversal there is less costly than through the continuous terrain contexts (Figure 4).



(**a**)



(**b**)

Figure 4. Direct-goal scenario: search behavior and resulting plan of the proposed planner across heterogeneous navigation contexts. The planner jointly selects intra-context actions and inter-context transitions to reach the goal without entering the Charging-Station context. (**a**) A* search expansions and frontier, with the resulting path colored by remaining battery level. (**b**) Resulting path with segments colored by the active navigation context.

In the *resource-constrained* scenario, reaching the goal without recharging is not admissible. The resulting plan is

$$\texttt{Forest-Offroad} \rightarrow \texttt{Plain} \rightarrow \texttt{Forest-Road} \rightarrow \texttt{Charging-Station} \rightarrow$$

$$\texttt{Forest-Road} \rightarrow \texttt{Forest-Offroad}.$$

The battery decreases to 12.61% before charging and reaches 43.46% at the goal. The route travels 105.546 m and contains five inter-context transitions. Repeated charging is permitted but incurs additional cost; in this scenario, one visit to the Charging-Station context is sufficient (Figure 5).

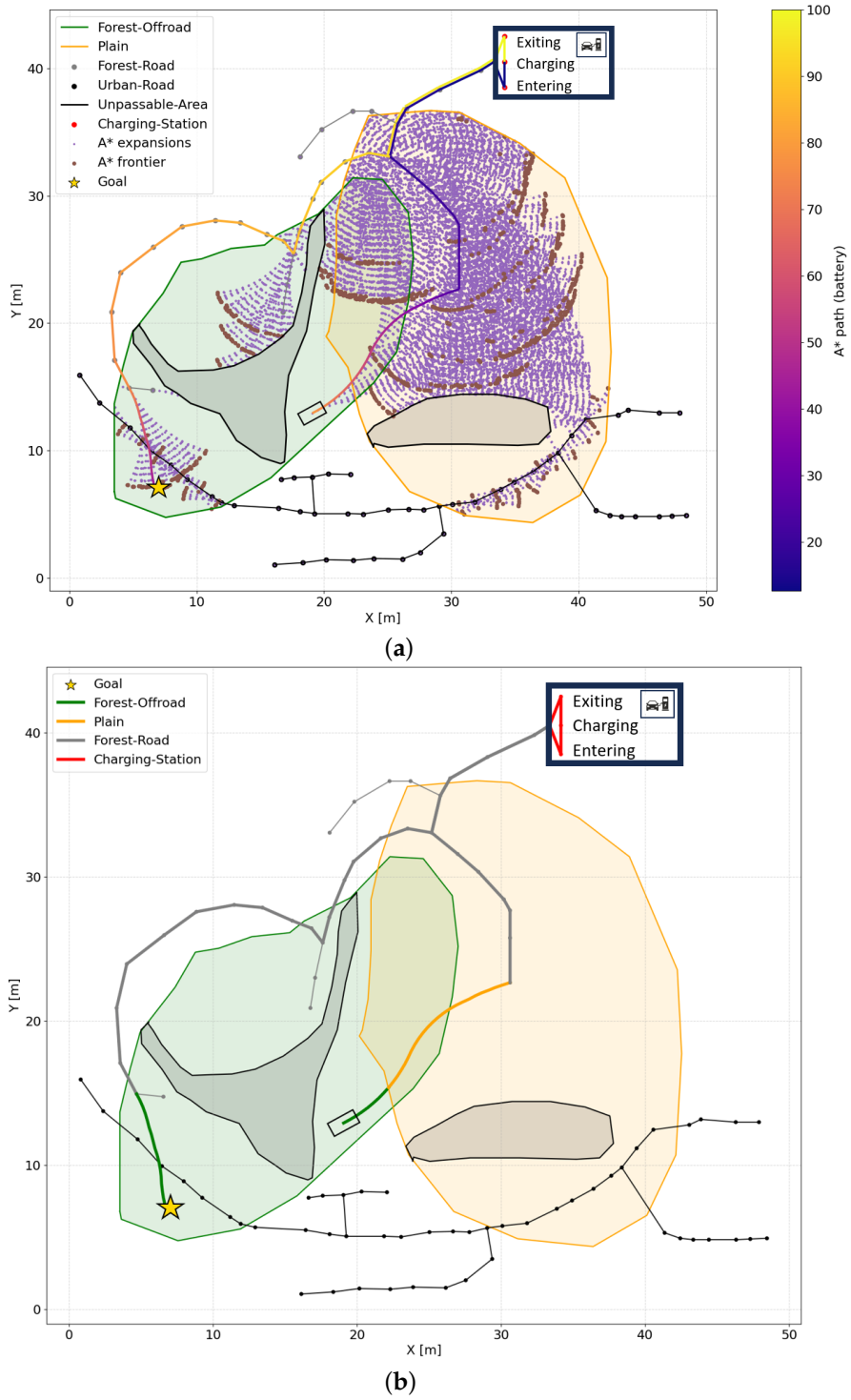


Figure 5. Resource-constrained scenario: search behavior and resulting multi-context plan of the proposed planner. The internal resource constraint changes the admissible search space, causing the planner to include the Charging-Station context before continuing toward the goal. (**a**) A* search expansions and frontier, with the resulting path colored by remaining battery level. (**b**) Resulting path with segments colored by the active navigation context.

The repeated occurrence of *Forest-Road* does not represent the same planning state being revisited, since the robot re-enters the context with a different internal battery state.

Table 1 summarizes the key metrics. Planning time is lower in the battery-constrained scenario because the resource constraint prunes approximately one third of the candidate actions, reducing the explored search space.

Table 1. Comparison of the proposed planner and HMCP in the five-context scenarios.

	Direct-Goal		Battery-Constrained	
Metric	**Proposed**	**HMCP**	**Proposed**	**HMCP**
Distance (m)	58.592	75.210	105.546	116.324
Context transitions	3	3	5	5
Final battery (%)	2.80	1.70	43.46	41.25
Local planner calls	–	5	–	12
Runtime (s)	0.81	2.93	0.49	0.93

5.1.2. Cost Evaluation

Table 1 compares the proposed planner with HMCP in the direct-goal and battery-constrained scenarios. Both planners reach the goal while satisfying the modeled battery constraints and use the same number of inter-context transitions. However, the proposed planner produces shorter routes and retains more battery at the goal.

In the direct-goal scenario, the proposed planner travels 58.592 m, compared with 75.210 m for HMCP, and reaches the goal with 2.80% battery instead of 1.70%. In the battery-constrained scenario, the corresponding distances are 105.546 m and 116.324 m, with 43.46% and 41.25% battery remaining, respectively.

The difference in route length mainly results from the treatment of interface states. HMCP commits to one predefined representative state per interface, whereas the proposed planner considers interface states directly within the joint search. The use of one representative state is a deliberate baseline configuration and not an inherent restriction of hierarchical planning.

The proposed planner requires 0.81 s and 0.49 s for the direct-goal and battery-constrained scenarios, respectively, compared with 2.93 s and 0.93 s for HMCP. Although HMCP operates on a smaller high-level search space, each high-level expansion can require several context-specific planning queries. In the evaluated scenarios, HMCP invokes the local A*-style planner five times in the direct-goal scenario and twelve times in the battery-constrained scenario. The observed runtime difference is therefore consistent with the repeated local planning required by the evaluated hierarchical implementation rather than with the use of a different local planning algorithm.

Because the two planners operate at different search-space granularities, their raw state-expansion counts are not directly comparable. We therefore use runtime and the number of context-specific local planner calls to characterize their computational behavior. These results describe the evaluated implementations and should not be interpreted as a general computational disadvantage of hierarchical planning architectures.

5.1.3. Context-Wise Cost Analysis

This analysis evaluates the intended asymmetry between structured roads and continuous terrain. In the direct-goal case, *Urban-Road* accounts for 31.06 m at a cost of 52.074, corresponding to approximately 0.18% of the total cost, while *Plain* and *Forest-Offroad* account for 59.2% and 40.6% of the total cost, respectively. The effective cost per unit distance is approximately 1.68 on *Urban-Road*, compared with approximately 977 on *Plain* and 1263 on *Forest-Offroad*.

In the battery-constrained case, *Forest-Road* contributes approximately 68.9% of the traveled distance and 78.3% of the total cost, whereas the *Charging-Station* contributes only approximately 0.62% of the total cost. More importantly, the resulting plan demonstrates how the proposed search combines context-specific motion, inter-context transitions, and internal resource state within the same planning problem. The low initial battery makes routes without recharging inadmissible, causing the search to select a transition into the *Charging-Station* context and subsequently continue planning from the updated battery state. Thus, the charging decision and the surrounding context transitions are determined as part of the joint search rather than being imposed by a separate high-level rule or executive.

5.1.4. Runtime Considerations

The proposed planner completed the direct-goal and battery-constrained queries in 0.81 s and 0.49 s, respectively, whereas HMCP required 2.93 s and 0.93 s. Both methods use the same A*-style intra-context search with the same context-specific state spaces, actions, dynamics, and cost models. The runtime difference therefore does not result from using a less efficient local planner for HMCP, but from how the search is organized.

HMCP invokes a separate intra-context search whenever the high-level planner evaluates a transition to a representative interface state. In the evaluated scenarios, this results in five local planner calls in the direct-goal case and twelve in the battery-constrained case. In contrast, the proposed method expands intra-context actions and inter-context transitions within one incremental search, so alternative interface transitions are evaluated as part of the same search process. In the present implementation, this avoids the repeated execution of complete local planning queries and accounts for the observed runtime difference.

The continuous *Plain* and *Forest-Offroad* contexts are computationally more expensive to expand than graph-based contexts because successor generation requires simulation of kinematic actions rather than a single graph-edge traversal. The reported runtimes should therefore be interpreted as results for the evaluated implementations rather than as a general computational advantage over hierarchical planning.

5.2. Comparison with Topological Navigation Baselines

We compare the proposed planner with two topological navigation variants. In the *Topo-Map-n* configuration, each transition between local maps is represented by a single spatially nearest entry/exit node pair. This commitment may lead to an unreachable or suboptimal transition when other interface states provide better connectivity.

The strengthened *Topo-Map-s* configuration considers all available discrete entry/exit candidates associated with an interface. For each candidate, the local planner evaluates connectivity and path length, and all connected candidates are retained in the high-level search. Topo-Map-s can therefore optimize over the available discrete interface alternatives, but requires additional local-planning evaluations.

Table 2 summarizes the results over 50 test cases. *Topo-Map-n* fails in 41 cases, returns eight suboptimal paths, and finds the shortest path in only one case. In contrast, both *Topo-Map-s* and the proposed planner reach the goal and obtain the shortest path in all 50 cases.

Table 2. Planner performance over 50 connectivity-challenging cases.

Planner	Fail	SubOpt	Opt	T_{avg} (s)
Proposed	0	0	50	0.54
Topo-Map-s	0	0	50	1.36
Topo-Map-n	41	8	1	0.81

The main difference between the latter two methods is therefore computational rather than solution quality in this experiment. *Topo-Map-s* explicitly evaluates the discrete candidate interface states using repeated local planning, resulting in an average runtime of 1.36 s. The proposed method incorporates interface-state selection directly into the joint search and requires 0.54 s on average. These results show that both approaches can recover the shortest paths in the evaluated setting, while the proposed formulation avoids a separate enumeration and evaluation of interface candidates.

The present evaluation is intended primarily to validate the proposed planning formulation and to isolate the effects of joint context and interface-state reasoning. The complete heterogeneous framework is evaluated in one five-context environment under two resource conditions, while the restricted grid–topological evaluation considers 50 test cases to examine interface-state selection separately. These experiments therefore demonstrate the intended planning behavior, but they do not establish scalability to large heterogeneous environments. A broader statistical evaluation over procedurally generated environments with randomized context layouts, interfaces, start and goal states, and resource levels is left for future work.

6. Conclusions and Future Work

We presented a unified planning formulation that performs a single incremental search across heterogeneous representations while explicitly modeling interfaces between contexts, context-switching costs, and internal resource constraints such as battery level. In the evaluated heterogeneous environment, the planner jointly selected context-specific actions and inter-context transitions according to the planning objective and internal resource state. Under a low initial battery level, routes without recharging became inadmissible, and the planner selected a longer route through the Charging-Station context to satisfy the battery constraint. The resulting action pruning also reduced the explored search space and planning time in this scenario.

Compared with the hierarchical planning baseline HMCP, which separates context-level transition selection from context-specific local planning, the proposed joint search produced shorter routes, retained more battery at the goal, and required less planning time in both evaluated scenarios. HMCP commits to fixed representative interface states, whereas the proposed method jointly considers local actions, interface states, context transitions, and resource levels within the same search.

In the restricted grid–topological evaluation, both the proposed method and the strengthened topological baseline found the shortest path in all evaluated cases. However, the strengthened baseline explicitly evaluates the available discrete interface candidates through repeated local-planning queries, whereas the proposed method incorporates interface-state selection directly into the joint search and achieved lower average planning time. The nearest-node topological baseline, in contrast, frequently failed or produced suboptimal paths because of its fixed interface-state selection.

Future work will evaluate the framework on larger and procedurally generated environments with randomized start states, goal states, battery levels, and interface configurations. We also plan to investigate multiple candidate interface states for hierarchical baselines, additional internal resources and task constraints, and real-robot integration with navigation frameworks such as Nav2. A further direction is the automatic extraction and refinement of contexts and interface zones from semantic maps and onboard perception, together with the automatic construction and validation of representation-aware mappings between the resulting context-specific state spaces.

https://doi.org/10.3390/app16189295

Author Contributions: Conceptualization, H.D. and G.S.-W.; methodology, H.D. and G.S.-W.; software, H.D.; validation, H.D.; formal analysis, H.D.; investigation, H.D.; visualization, H.D.; writing—original draft preparation, H.D.; writing—review and editing, H.D. and G.S.-W.; supervision, G.S.-W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Austrian Research Promotion Agency (FFG) grant number Sustainable Autonomous Forestry Logistics/FO999898697 under the THINK.WOOD. program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code, simulation configurations, and data supporting the findings of this study are available from the corresponding author upon request.

Acknowledgments: The authors acknowledge support from the TU Graz Open Access Publishing Fund for the publication of this article. ChatGPT (GPT-4o, OpenAI) was used solely for language editing and text correction to improve grammar, clarity, and readability. The authors are fully responsible for the manuscript's content, analysis, and conclusions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Xiao, Y.; Codevilla, F.; Gurram, A.; Urfalioglu, O.; López, A.M. Multimodal End-to-End Autonomous Driving. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 537–547. https://doi.org/10.1109/TITS.2020.3013234.
2. Meng, X.; Hatch, N.; Lambert, A.; Li, A.; Wagener, N.; Schmittle, M.; Lee, J.; Yuan, W.; Chen, Z.; Deng, S.; et al. TerrainNet: Visual Modeling of Complex Terrain for High-speed, Off-road Navigation. *arXiv* **2023**, arXiv:2303.15771.
3. Dixit, A.; Fan, D.D.; Otsu, K.; Dey, S.; Agha-Mohammadi, A.A.; Burdick, J.W. STEP: Stochastic Traversability Evaluation and Planning for Risk-Aware Navigation; Results From the DARPA Subterranean Challenge. *IEEE Trans. Field Robot.* **2025**, *2*, 81–99. https://doi.org/10.1109/TFR.2024.3512433.
4. Crespo, J.; Castillo, J.C.; Mozos, O.M.; Barber, R. Semantic Information for Robot Navigation: A Survey. *Appl. Sci.* **2020**, *10*, 497. https://doi.org/10.3390/app10020497.
5. Huang, B.; Liu, N. Mobile Navigation Guide for the Visually Disabled. *Transp. Res. Rec. J. Transp. Res. Board* **2004**, *1885*, 28.
6. Nüchter, A.; Hertzberg, J. Towards semantic maps for mobile robots. *Robot. Auton. Syst.* **2008**, *56*, 915–926. https://doi.org/https://doi.org/10.1016/j.robot.2008.08.001.
7. Bogorny, V.; Renso, C.; de Aquino, A.R.; de Lucca Siqueira, F.; Alvares, L.O. CONSTAnT—A Conceptual Data Model for Semantic Trajectories of Moving Objects. *Trans. GIS* **2014**, *18*, 66–88. https://doi.org/https://doi.org/10.1111/tgis.12011.
8. Rogers, J.G.; Christensen, H.I. Robot planning with a semantic map. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013*; IEEE: New York, NY, USA, 2013; pp. 2239–2244. https://doi.org/10.1109/ICRA.2013.6630879.
9. Talbot, B.; Lam, O.; Schulz, R.; Dayoub, F.; Upcroft, B.; Wyeth, G. Find my office: Navigating real space from semantic descriptions. In *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016*; IEEE: New York, NY, USA, 2016; pp. 5782–5787. https://doi.org/10.1109/ICRA.2016.7487802.
10. Goel, Y.; Vaskevicius, N.; Palmieri, L.; Chebrolu, N.; Arras, K.O.; Stachniss, C. Semantically Informed MPC for Context-Aware Robot Exploration. In *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, MI, USA, 1–5 October 2023*; IEEE: New York, NY, USA, 2023; pp. 11218–11225. https://doi.org/10.1109/IROS55552.2023.10341564.
11. Macenski, S.; Martín, F.; White, R.; Clavero, J.G. The Marathon 2: A Navigation System. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021*; IEEE: New York, NY, USA, 2020; pp. 2718–2725. https://doi.org/10.1109/IROS45743.2020.9341207.
12. Höllmann, M.; Kisliuk, B.; Krause, J.C.; Tieben, C.; Mock, A.; Pütz, S.; Igelbrink, F.; Wiemann, T.; Martinez, S.F.; Stiene, S.; et al. Towards context-aware navigation for long-term autonomy in agricultural environments. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 69–74.
13. Rocamora, B.M.; Simplicio, P.V.G.; Pereira, G.A.S. A Behavior Tree Approach for Battery-Aware Inspection of Large Structures Using Drones. In *Proceedings of the 2024 International Conference on Unmanned Aircraft Systems (ICUAS), Chania, Greece, 4–7 June 2024*; IEEE: New York, NY, USA, 2024; pp. 234–240. https://doi.org/10.1109/ICUAS60882.2024.10557083.
14. Al Shukairi, H.; Cardoso, R.C. ML-MAS: A Hybrid AI Framework for Self-Driving Vehicles. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23, London, UK, 29 May–2 June 2023*; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2023; pp. 1191–1199.

15. Niijima, S.; Umeyama, R.; Sasaki, Y.; Mizoguchi, H. City-Scale Grid-Topological Hybrid Maps for Autonomous Mobile Robot Navigation in Urban Area. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021*; IEEE: New York, NY, USA, 2020; pp. 2065–2071. https://doi.org/10.1109/IROS45743.2020.9340990.
16. Marder-Eppstein, E.; Lu, D.; Ferguson, M.; Hoy, A. Move_Base—Ros Wiki. 2016. Available online: http://wiki.roseorg/move_base (accessed on 10 September 2026).
17. Pütz, S.; Santos Simón, J.; Hertzberg, J. Move Base Flex A Highly Flexible Navigation Framework for Mobile Robots. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018*; IEEE: New York, NY, USA, 2018; pp. 3416–3421. https://doi.org/10.1109/IROS.2018.8593829.
18. Kato, S.; Tokunaga, S.; Maruyama, Y.; Maeda, S.; Hirabayashi, M.; Kitsukawa, Y.; Monrroy, A.; Ando, T.; Fujii, Y.; Azumi, T. Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems. In *Proceedings of the 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), Porto, Portugal, 11–13 April 2018*; IEEE: New York, NY, USA, 2018; pp. 287–296. https://doi.org/10.1109/ICCPS.2018.00035.
19. Silva, G.R.; Garcia, N.H.; Bozhinoski, D.; Deshpande, H.; Oviedo, M.G.; Wasowski, A.; Montero, M.R.; Corbato, C.H. MROS: A framework for robot self-adaptation. In *Proceedings of the 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Melbourne, Australia, 14–20 May 2023*; IEEE: New York, NY, USA, 2023; pp. 151–155. https://doi.org/10.1109/ICSE-Companion58688.2023.00044.
20. Bozhinoski, D.; Wijkhuizen, J. Context-based navigation for ground mobile robot in semi-structured indoor environment. In *Proceedings of the 2021 Fifth IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan, 15–17 November 2021*; IEEE: New York, NY, USA, 2021; pp. 82–86. https://doi.org/10.1109/IRC52146.2021.00019.
21. Brugali, D.; Capilla, R.; Mirandola, R.; Trubiani, C. Model-Based Development of QoS-Aware Reconfigurable Autonomous Robotic Systems. In *Proceedings of the 2018 Second IEEE International Conference on Robotic Computing (IRC), Laguna Hills, CA, USA, 31 January–2 February 2018*; IEEE: New York, NY, USA, 2018; pp. 129–136. https://doi.org/10.1109/IRC.2018.00027.
22. Gherardi, L.; Hochgeschwender, N. RRA: Models and tools for robotics run-time adaptation. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015*; IEEE: New York, NY, USA, 2015; pp. 1777–1784. https://doi.org/10.1109/IROS.2015.7353608.
23. Wasik, Z.; Saffiotti, A. A hierarchical behavior-based approach to manipulation tasks. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Taipei, Taiwan, 14–19 September 2003*; IEEE: New York, NY, USA, 2003; Volume 2, pp. 2780–2785. https://doi.org/10.1109/ROBOT.2003.1242013.
24. Scheutz, M.; Andronache, V. Architectural mechanisms for dynamic changes of behavior selection strategies in behavior-based systems. *IEEE Trans. Syst. Man, Cybern. Part B (Cybernetics)* **2004**, *34*, 2377–2395. https://doi.org/10.1109/TSMCB.2004.837309.
25. Iovino, M.; Förster, J.; Falco, P.; Chung, J.J.; Siegwart, R.; Smith, C. Comparison Between Behavior Trees and Finite State Machines. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 21098–21117. https://doi.org/10.1109/TASE.2025.3610090.
26. Hauser, K.; Latombe, J.C., Multi-modal Motion Planning in Non-expansive Spaces. In *Algorithmic Foundation of Robotics VIII: Selected Contributions of the Eight International Workshop on the Algorithmic Foundations of Robotics*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 615–630. https://doi.org/10.1007/978-3-642-00312-7_38.
27. Deeken, H.; Puetz, S.; Wiemann, T.; Lingemann, K.; Hertzberg, J. Integrating Semantic Information in Navigational Planning. In *Proceedings of the ISR/Robotik 2014; 41st International Symposium on Robotics, Munich, Germany, 2–3 June 2014*; IEEE: New York, NY, USA, 2014, pp. 1–8.
28. Wiemann, T.; Sprickerhof, J.; Hertzberg, J. A Toolkit for Automatic Generation of Polygonal Maps from Laser Data. In *Proceedings of the Proceedings of the 7th German Conference on Robotics (ROBOTIK), Munich, Germany, 21–22 May 2012*; IEEE: New York, NY, USA, 2012, pp. 1–6.
29. Yun, H.; Yoo, S. Semantic Zone based 3D Map Management for Mobile Robot. *arXiv* **2025**, arXiv:2512.12228.
30. Sathyamoorthy, A.J.; Weerakoon, K.; Elnoor, M.; Zore, A.; Ichter, B.; Xia, F.; Tan, J.; Yu, W.; Manocha, D. CoNVOI: Context-aware Navigation using Vision Language Models in Outdoor and Indoor Environments. In *Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, United Arab Emirates, 14–18 October 2024*; IEEE: New York, NY, USA, 2024; pp. 13837–13844. https://doi.org/10.1109/IROS58592.2024.10802716.