Towards Generalizable Embodied Navigation with Vision-Language Models

Zongtai Li

CMU-RI-TR-26-62

May 2026

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Ji Zhang, *chair*
Wenshan Wang
Zhixuan Liu

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To all those who have supported me.

Abstract

Embodied navigation asks an autonomous agent to move through unknown environments and accomplish tasks such as finding objects or following instructions. Reliable performance in real-world settings, from household assistance to warehouse logistics, requires the agent to tightly integrate perception, semantic reasoning, and long-horizon planning under cluttered layouts, ambiguous appearances, and robot-specific constraints. Vision-language models (VLMs) offer rich semantic priors for this task, but directly inserting them into the navigation loop often leads to inefficient exploration, unstable behavior, and limited transfer across platforms. This thesis argues that these failures stem from a multi-level *misalignment* between how VLMs reason and what navigation demands, and presents four complementary contributions to address it. **STRIVE** shows that object navigation improves substantially when the environment is summarized as a structured graph of objects, viewpoints, and rooms, letting the VLM reason at a semantic level while classical algorithms handle local exploration. **SysNav** extends this into a deployable system by decoupling semantic reasoning, room-level planning, and embodiment-specific control for robust cross-platform deployment. **IntentNav** shifts from prompting to learning, showing that navigation decisions become more stable when trained with intent-aligned supervision from human demonstrations. Recognizing that object-goal search underutilizes VLM reasoning, **Goal2Pixel** moves to instruction-guided navigation where longer, compositional instructions demand richer language grounding, and reformulates the task as pixel grounding so the model directly connects instruction understanding to executable motion. Together, these works trace a progression from structured representation through system integration and learned decision making to instruction-guided navigation, arguing that effective embodied navigation with VLMs requires aligning reasoning with the right representations, architectures, learning objectives, and task formulations.

Acknowledgments

I am deeply grateful to Professor Ji Zhang, Wenshan Wang, and Jean Oh for their guidance and support throughout my two years in the master's program. I have been fortunate to learn from them not only as researchers, but also as mentors. Professor Zhang's practicality, rigor, and attention to detail have shaped the way I approach research. I have also learned a great deal from Wenshan's patience and kindness, and from Jean's warmth, encouragement, and enthusiasm. Their mentorship has meant a great deal to me.

I would also like to thank Zhixuan, Haokun, Yuxin, Muyi, William, and many other friends who have shared this research journey with me. I will always remember the nervousness and excitement of our first real-robot experiments, the long nights spent recording demos, the many discussions about ideas, experiments, and writing, and the relief we felt whenever a project finally came together. Those days were often tiring, but they were also some of the most meaningful parts of my graduate experience. I learned a great deal from working with you, and I grew a great deal because of you.

I am also thankful to Alex, Kevin, Jingfan, Yusen, Zihan, Pranav, Ziyong, Jinxi, Yiyu, and all of my friends in the lab. I will always remember the time we spent playing sports, hanging out, eating together, trying new restaurants, going to work, and going home together. Our lab may not be the most productive, but it has always felt like the warmest one to me. I never expected a lab to be filled with so much laughter and joy. I feel very lucky to have met all of you. Thank you for making these years so memorable.

I would also like to thank all the friends who played soccer and badminton with me. In Pittsburgh, sports became one of the best parts of my life outside of research. Many of the friends I made here came from the soccer field or the badminton court. I know that after leaving school, it may be hard to find another time in life when so many energetic people can gather so easily to play, compete, and have fun together. That makes me cherish these final days as a student even more.

I am grateful to WeRide for giving me a meaningful and fulfilling internship experience, and to all the new friends I met during my months in the

Bay Area. You made that summer rich, busy, and unforgettable. I also want to thank my friends from the gaming world. Because of the time difference in Pittsburgh, I often had to join you late at night, but those games brought me a lot of joy. Once I move to the West Coast, I look forward to playing together more often.

Most importantly, I want to thank my parents. You raised me with endless love, care, and patience, and guided me as I grew into the person I am today. Through both your words and your actions, you taught me how to live with kindness, responsibility, and integrity. Your support and trust have given me the freedom and courage to pursue my own path. I will always be grateful for everything you have done for me, and I will always love you.

As I reach the end of this chapter, I am filled with gratitude. There is much more I wish I could say, but words are not enough.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Embodied navigation asks an autonomous agent to move through an unknown environment and accomplish a task goal—finding a target object, following a natural-language instruction, or reaching a specified location. Although the task descriptions are conceptually simple, reliable performance requires the agent to solve several tightly coupled problems at once: it must build a useful representation of the environment from incomplete observations, reason semantically about where the goal is likely to lie, and make efficient navigation decisions over long horizons. These requirements become even more demanding in realistic indoor environments, where cluttered layouts, ambiguous visual appearances, and robot-specific motion constraints all interact.

This combination of perception, reasoning, planning, and control makes embodied navigation a demanding testbed for artificial intelligence. It is not enough for the agent to recognize objects in the current view, nor is it enough to explore based only on geometry. The agent must continuously connect what it has already seen, what it infers about the still-unobserved parts of the scene, and which actions are worth taking next under limited exploration budgets. In other words, embodied navigation tests not only perceptual competence, but also whether semantic knowledge can be translated into effective embodied behavior.

The difficulty of embodied navigation is compounded by the fact that its sub-problems are not independent. Perception errors propagate into planning, planning errors propagate into control, and control errors can bring the agent to locations

where perception is even more challenging. This coupling means that improving any single component in isolation may not improve overall performance, and may even degrade it if the improvement introduces new failure modes that other components cannot handle. A central theme of this thesis is therefore that navigation should be understood as a system-level problem, where the interfaces between components matter as much as the components themselves.

1.1 The Role of Vision-Language Models

Recent progress in vision-language models (VLMs) [51, 61, 75] has made semantic reasoning a promising ingredient for navigation. VLMs provide broad prior knowledge about object-room co-occurrence, can interpret rich visual context, and can often explain why one region of an environment appears more promising than another. They can associate target categories with typical room types, interpret relationships among observed objects, and use broader world knowledge to guide exploration.

However, directly inserting a VLM into the navigation loop does not by itself produce an effective navigation agent. VLM calls are expensive and slow, and the amount of visual and textual context the model can process at once is limited, so semantic reasoning cannot be invoked indiscriminately throughout navigation. Current VLMs also remain weak at 3D spatial reasoning and directional grounding [11, 50, 104]; if low-level action selection relies on them too directly, the agent can make inconsistent local decisions or fall into looping behavior. Effective use of VLMs in embodied navigation therefore requires answering four foundational questions: how should the environment be represented so that semantic reasoning becomes genuinely useful, how should semantic reasoning be organized within a deployable system architecture, how can navigation decisions be learned from data rather than prompted, and what output interface should bridge the model's reasoning to executable motion. These four questions correspond to the four bottlenecks identified below, and each is addressed by one of the four technical chapters in this thesis.

1.2 Four Bottlenecks in VLM-Guided Navigation

This thesis identifies four progressively deeper bottlenecks that limit the effectiveness of VLMs in embodied navigation, and proposes four complementary solutions.

Representation bottleneck. VLM reasoning quality depends heavily on the context provided. If the input is limited to local observations without a structured summary of accumulated knowledge, the model may fail to reason consistently over a partially explored environment. The first question is therefore: how should the environment be represented so that semantic reasoning becomes genuinely useful?

System bottleneck. Even when representation is improved, real-world navigation is more than a benchmark task. A deployable system must handle noisy perception, long-horizon planning, embodiment-specific constraints, and real-time execution. The second question is: how should semantic reasoning be organized within a complete system architecture that works reliably across different robots and environments?

Learning bottleneck. Zero-shot VLM prompting and hand-designed heuristics have inherent limitations. They cannot guarantee stable long-horizon search behavior, and they do not improve from experience. The third question is: how can VLMs be trained to learn effective navigation policies from data rather than relying solely on prompting?

Output interface bottleneck. Most VLM-based navigation methods still predict low-level actions at every timestep, creating ambiguity, short-horizon bias, and high inference costs. The fourth question is: what output interface should serve as the bridge between VLM reasoning and executable motion?

1.3 Thesis Overview and Unifying Claim

This thesis addresses the four bottlenecks through four research contributions that form a coherent progression—from structured representation, through system-level integration, to learned decision making, and finally to output interface redesign.

STRIVE (Chapter 3) addresses the representation bottleneck by constructing a multi-layer environment representation of objects, viewpoints, and rooms, and performing VLM reasoning at the room level. It achieves state-of-the-art performance on four simulation benchmarks and demonstrates robustness in 120 real-world experiments.

SysNav (Chapter 4) addresses the system bottleneck by decoupling navigation into three cooperating levels—semantic reasoning, room-based planning, and base autonomy—and deploying the same high-level logic across wheeled, quadruped, and humanoid robots in 190 real-world experiments.

IntentNav (Chapter 5) addresses the learning bottleneck by shifting from zero-shot prompting to learning-based imitation. It trains a VLM on human demonstrations using a BEV-grounded candidate decision space with intent-aligned supervision, achieving state-of-the-art performance among learning-based methods and transferring across three robot embodiments.

Goal2Pixel (Chapter 6) broadens the scope from object navigation to vision-and-language navigation in continuous environments (VLN-CE), where longer, compositional instructions demand richer semantic reasoning than single-category object search. It addresses the output interface bottleneck by reformulating VLN-CE as navigable pixel grounding: rather than predicting actions, the VLM directly outputs a navigable pixel coordinate, achieving competitive state-of-the-art performance on R2R-CE and RxR-CE with approximately 6× fewer VLM calls.

These four contributions form a progression from representation through system integration and learned decision making to output interface design, arguing that effective embodied navigation depends on *alignment* at every level. Progress on any single dimension yields partial improvement; the full potential is realized only when representation, system architecture, decision learning, and output interface are addressed together. This perspective contrasts with the idea that navigation can be solved by scaling up end-to-end models; even with powerful pretrained backbones, it remains necessary to think carefully about what information is provided as input, what outputs the model produces, and how training is structured. Although the four works were developed with different VLM backbones, robot platforms, and evaluation benchmarks, the same design principles recur throughout, suggesting that these principles reflect genuine structural insights about embodied navigation rather

than artifacts of a particular experimental setup.

1.4 Chapter Organization

The rest of this thesis is organized as follows. Chapter 2 reviews the technical foundations: object navigation, VLM-guided navigation, structured scene representations, room-based planning, imitation learning for navigation, and vision-language navigation in continuous environments. Chapter 3 presents STRIVE and focuses on structured representation for efficient zero-shot VLM-guided navigation. Chapter 4 presents SysNav and focuses on system-level coordination for real-world, cross-embodiment object navigation. Chapter 5 presents IntentNav and focuses on learning navigation intent from human demonstrations. Chapter 6 presents Goal2Pixel and focuses on pixel-level grounding for vision-language navigation. Chapter 7 concludes the thesis and outlines future directions.

Chapter 2

Background and Related Work

2.1 Object Navigation

Object navigation requires an embodied agent to find an instance of a target object category in an unseen environment. In contrast to point-goal navigation, the target location is unknown and must be inferred during exploration. A successful agent therefore needs to balance at least three capabilities: semantic reasoning about likely target locations, geometric exploration of unknown free space, and efficient decision making over long horizons.

Approaches to object navigation can be broadly grouped into modular systems and end-to-end learned policies. Modular methods [4, 10, 45, 73, 113] decompose the task into components such as object detection, mapping, planning, and control, which improves interpretability and robustness in large or partially structured indoor spaces. A particularly influential line of modular work is frontier-based exploration [79], where the agent navigates toward boundaries between known and unknown space to systematically expand its map. Classical frontier methods are purely geometric, but they provide a simple and effective mechanism for systematic coverage that modern semantic navigation systems build upon [6, 13, 23, 24, 53]. End-to-end methods, typically trained via reinforcement learning or imitation learning, learn a direct mapping from observations to actions [54, 55, 59, 73, 82, 95]. While effective in controlled settings, they often struggle to generalize and produce policies that are difficult to interpret.

A more recent line of work enriches modular pipelines with structured scene representations that can support semantic reasoning. Semantic map methods [10, 99, 105] overlay category labels on occupancy grids, enabling category-aware planning while retaining geometric precision. Graph-based methods go further by constructing scene graphs or viewpoint graphs so that reasoning can operate over structured entities instead of raw sensory observations [28, 80]. Some works use scene graphs to summarize semantic information and prompt a language model to select among frontier locations [41, 84], while others explicitly construct viewpoint nodes that discretize the environment, enabling VLMs to reason over the graph and choose among candidate waypoints [1, 74]. The rise of foundation models has also pushed object navigation toward zero-shot and open-vocabulary settings [25, 34, 86, 88, 109], where LLMs or VLMs are used to reason over candidate goals, scene structure, or commonsense knowledge without any task-specific training. These methods demonstrate that pretrained semantic priors can meaningfully guide exploration, but they also reveal a recurring limitation: when VLMs are asked to make too many fine-grained decisions directly, they can produce redundant or unstable behavior because they do not reliably capture detailed 3D spatial structure [104].

The four technical chapters in this thesis build on the structured-representation tradition but go further in a specific sense: rather than treating representation, system architecture, training objective, and output interface as independent choices, they are designed jointly to align with the structure of the navigation problem. The claim is not that end-to-end learning is impossible, but that embodied navigation exposes strong structural regularities—room hierarchy, object-context relations, directional candidate geometry, and image-plane spatial structure—that benefit from being represented explicitly. The methods in this thesis therefore combine structured representations with varying degrees of learned decision making, from zero-shot VLM reasoning in STRIVE and SysNav to imitation-learned policies in IntentNav and Goal2Pixel.

2.2 Vision-and-Language Navigation in Continuous Environments

Vision-and-language navigation (VLN) requires an agent to follow natural-language instructions and reach a target location. While early VLN benchmarks such as Room-to-Room (R2R) [2] were built on discrete navigation graphs, recent benchmarks extend VLN to continuous environments (VLN-CE) [31], where the agent must perform fine-grained physical movements. This makes VLN-CE a language-conditioned spatial decision-making problem: the agent must ground language in egocentric observations, track its progress, and continuously decide where to move next.

Existing VLN-CE methods can be broadly grouped into three categories. *Zero-shot and training-free methods* [12, 14, 22, 35, 40] leverage pretrained foundation models for reasoning-based navigation without any task-specific fine-tuning. These methods demonstrate that commonsense knowledge and language priors can partially substitute for learned navigation experience, but they are limited by high inference cost and the difficulty of aligning open-ended reasoning with continuous control. *Non-VLM learning-based methods* [26, 27, 64, 81, 103] train task-specific navigation policies through reinforcement or imitation learning, often simplifying continuous navigation through waypoint graphs, topological maps, or low-level action prediction. Although effective in benchmark settings, these methods can accumulate errors during long-horizon execution and may not generalize well to unseen environments. *VLM-based methods* [17, 70, 91, 96, 101, 107] leverage pretrained vision-language backbones for embodied navigation, using the VLM's world knowledge to interpret instructions and ground them in visual observations. Most of these methods still follow an action-centric paradigm, querying the VLM at every timestep for low-level meta-actions such as *turn left/right 15°, move forward 25 cm, or stop*. A few recent works explore alternative output interfaces—NavFoM [98] predicts relative 3D coordinates, OmniNav [77] predicts continuous-space waypoints, and DualVLN [69] combines pixel-grounded waypoints with action execution—but these remain hybrid designs that retain substantial action-oriented components.

The persistence of the action-centric interface is notable because it introduces three limitations. First, *ambiguous action supervision*: many distinct action sequences can

reach the same goal, making the oracle trajectory an unnecessarily restrictive training target. Second, *myopic decisions*: optimizing for the next short-range movement discourages longer-horizon spatial reasoning. Third, *high inference cost*: because each prediction advances the agent only a few centimeters or degrees, the VLM must be invoked dozens of times per episode. These observations suggest that the bottleneck is not only the capacity of the VLM but the interface through which its reasoning becomes motion.

2.3 Vision-Language Model Foundations

The methods in this thesis build on recent advances in vision-language models, which combine visual perception with language understanding in a single pretrained architecture.

Pretrained VLMs. Models such as GPT-4V [75], InternVL [15, 16], Qwen2-VL [3], and LLaVA [38] have demonstrated strong capabilities in visual question answering, image captioning, and multimodal reasoning. These models are trained on massive web-scale datasets and acquire broad prior knowledge about object categories, room types, spatial relationships, and commonsense reasoning. In the context of navigation, this knowledge can be leveraged to infer where target objects are likely to appear, interpret the function of different rooms, and reason about the semantic relevance of observed objects.

InternVL3. InternVL3 [16] serves as the VLM backbone for both IntentNav and Goal2Pixel. It provides a scalable architecture with vision encoders, projection layers, and a large language model backbone. Its 2B variant is particularly attractive for navigation because it balances reasoning capability with computational efficiency, enabling deployment on laptop-class GPUs. The model supports fine-tuning with LoRA for task-specific adaptation and full fine-tuning of the language backbone for deeper domain adaptation.

From Zero-Shot to Fine-Tuned. A key theme across this thesis is the transition from zero-shot VLM usage to fine-tuned VLM policies. STRIVE and SysNav use

VLMs in a zero-shot manner, providing structured prompts and reading out navigation decisions without any parameter updates. This approach is flexible and generalizable but limited by the VLM's inherent spatial reasoning weaknesses and the quality of the prompt design. IntentNav and Goal2Pixel instead fine-tune VLMs on navigation-specific data, learning policies that are more stable and better suited to the navigation task. This transition reflects a broader principle explored throughout the thesis: hand-designed interactions provide a useful starting point, but learned interfaces ultimately produce more robust behavior.

2.4 Vision-Language Models for Navigation

Vision-language models have recently become attractive for navigation because they offer strong semantic priors and rich contextual reasoning. They can associate target categories with typical room types, interpret relationships among observed objects, and use broader world knowledge to guide exploration. This has led to a growing class of VLM-guided navigation methods.

These methods can be broadly grouped by how they use the VLM. *Zero-shot and training-free methods* [34, 84, 86, 88, 109] use pretrained VLMs directly as semantic reasoners, providing textual or visual prompts and reading out navigation decisions without any fine-tuning. As foundation models have become stronger, a number of zero-shot and open-vocabulary navigation methods have begun to use language or vision-language models for semantic decision making. These methods show that commonsense reasoning can improve exploration, but they also reveal a limitation: when VLMs are used too aggressively for low-level decision making, they can introduce redundant behavior because they do not reliably capture detailed 3D spatial structure. While flexible, they are limited by the quality of the prompt design and the VLM's inherent spatial reasoning weaknesses. *Learning-based methods* [19, 71, 102] fine-tune VLMs on navigation data, either from expert trajectories or human demonstrations. These methods can learn more stable behavior but require careful design of the training signal and decision interface.

However, VLMs are not a complete solution by themselves. Their reasoning quality depends heavily on the context provided to them. If the input is limited to local observations, they may fail to reason consistently about a partially explored

environment. If they are asked to choose among too many fine-grained candidates, they may produce redundant or unstable exploration decisions. These limitations motivate both the structured representation in STRIVE and the room-level reasoning strategy later systematized in SysNav.

2.5 Structured Scene Representation

A scene representation transforms raw sensory observations into a more useful internal description of the environment. For navigation, common representations include occupancy grids, semantic maps, frontier sets, scene graphs, and hierarchical combinations of these structures. A good representation should support both geometric planning and semantic reasoning.

Occupancy grids and metric maps provide precise geometric information for path planning but do not directly encode semantic categories or room structure [60, 99]. Semantic maps overlay category labels on metric maps, enabling category-aware planning but often at the cost of increased complexity and sensitivity to detection errors [10, 105]. Scene graphs represent environments as nodes (objects, rooms, regions) connected by edges (spatial relations, containment, visibility), providing a more compact and structured abstraction that is well suited to semantic reasoning [80, 84]. Graph-based methods go further by constructing scene graphs or viewpoint graphs so that reasoning can operate over structured entities instead of raw sensory observations [1, 41, 74]. Viewpoint nodes serve not just as generic topological anchors, but as a middle layer connecting room structure and object semantics. Hierarchical representations combine multiple levels of abstraction, such as room-level, viewpoint-level, and object-level, allowing different types of reasoning to operate at the appropriate scale [28, 72]. The thesis draws on all of these traditions: STRIVE and SysNav use hierarchical scene graphs, IntentNav uses BEV-grounded candidate spaces, and Goal2Pixel uses the image plane itself as the primary spatial representation.

Bird's-eye-view (BEV) representations deserve special mention because they appear in multiple chapters. A BEV map projects 3D observations onto a top-down 2D grid, preserving spatial layout while discarding some vertical information. BEV maps are particularly useful for frontier extraction, path planning, and candidate waypoint construction, because they provide a compact summary of explored and

unexplored regions. IntentNav uses a persistent BEV map to construct its candidate set, and the BEV representation also appears in SysNav's in-room exploration planning. The BEV perspective is complementary to egocentric first-person views: BEV provides global spatial structure while egocentric views provide detailed semantic evidence.

The works in this thesis are grounded in the view that representation should not merely record explored space, but should summarize the environment at multiple semantic and spatial granularities. In STRIVE, this appears as a three-layer representation of objects, viewpoints, and rooms. In SysNav, related ideas are extended to a richer scene representation that supports room-level reasoning and task constraints. In IntentNav, representation becomes more decision-centric through a BEV-grounded candidate space with egocentric visual memory. In Goal2Pixel, the image plane itself becomes the primary spatial interface.

The broader lesson is that representation determines what kinds of reasoning become possible. A purely geometric map supports frontier exploration, but it does not directly encode object-context relations. A purely semantic memory may store useful clues, but without navigational structure it is difficult to convert those clues into efficient motion. The thesis treats representation as an interface between semantics and action, and this is why structured representation appears repeatedly even though each chapter uses it in a different form.

2.6 Imitation Learning for Navigation

Imitation learning [58] offers an alternative to both hand-designed heuristics and reinforcement learning by training policies to mimic expert demonstrations. In navigation, expert data can come from oracle planners, human teleoperators, or heuristic policies, and the choice of data source has a strong influence on what the learned policy captures. Oracle trajectories, typically generated by shortest-path planners under full observability, provide optimal goal-reaching behavior but encode no search strategy; policies trained on them may fail to generalize when the target location is uncertain. Human demonstrations, in contrast, reflect bounded-rational search under partial observability—they contain rich latent intent such as search strategy, spatial memory, and target-directed reasoning—but this intent is not directly

observable in low-level action sequences.

Extracting useful training signals from raw demonstrations requires addressing two intertwined challenges. The first is the *labeling granularity*: methods that supervise low-level actions (e.g., move forward, turn left) inherit the ambiguity of action-level prediction, since many action sequences can reach the same goal, and optimizing for the next immediate action discourages long-range planning. Methods that instead supervise waypoint-level or goal-level decisions avoid these problems but require constructing appropriate waypoint-level labels from action-level demonstrations. IntentNav, presented in Chapter 5, addresses this through Frontier-based Human-Intent Labeling, which converts low-level human trajectories into candidate-level waypoint labels by analyzing frontier evolution in the demonstration.

The second challenge is the *policy backbone and input representation*. Recent work has explored using VLMs as the backbone for imitation-learned navigation policies [36, 47, 81, 89, 107], fine-tuning pretrained models on navigation data from first-person images or observation histories. While VLMs bring strong visual priors and enable language-conditioned policies, first-person inputs lack explicit spatial structure and low-level action prediction emphasizes short-horizon control, often leading to locally repetitive behaviors such as turning in place or backtracking. A broader concern is the distribution mismatch between training demonstrations and deployment conditions: oracle trajectories represent optimal behavior under full observability, while human demonstrations reflect bounded-rational search. IntentNav's approach of extracting high-level intent from human demonstrations provides a middle ground that captures the strategic value of human search while avoiding the noise in individual motor commands.

2.7 Real-World Embodied Navigation Systems

Deploying navigation methods on physical robots introduces challenges that are easy to overlook in simulation: sensor noise, sparse point clouds, imperfect detections, embodiment-specific motion constraints, and longer-horizon navigation all create failure modes absent from clean benchmark settings. The sim-to-real gap in navigation has been studied through both direct transfer [67] and simulation-to-simulation adaptation [30], and both lines of work confirm that policies trained in simulation

degrade substantially when confronted with real-world perception and control noise. Classical robot navigation systems address this by separating mapping, planning, and execution into modular pipelines built on reliable geometric primitives [7, 92, 93]. This decomposition improves interpretability and makes it easier to isolate and control failure modes, and it remains the dominant paradigm for real-world deployment. Recent efforts to bring object navigation to real homes [83] further confirm that system-level integration—robust perception, planning, and execution—is as important as any single algorithmic advance.

These observations motivate the system-level perspective of SysNav, presented in Chapter 4, which decomposes navigation into high-level semantic reasoning, mid-level planning, and low-level execution, treating semantic reasoning itself as an explicit architectural level rather than a subroutine. More generally, the real-world perspective changes the standard of success: in deployment settings, it matters not only whether the agent reaches the goal but also whether the system is interpretable, robust to sensing imperfections, portable across embodiments, and efficient enough to operate under practical time constraints. These concerns motivate the thesis emphasis on explicit interfaces and structured decomposition.

Chapter 3

STRIVE: Structured Representation Integrating VLM Reasoning for Efficient Object Navigation

3.1 Introduction

Object navigation requires an embodied agent to locate an instance of a target object category in an unknown environment under partial observability. The task is difficult because success depends on several capabilities at once: the agent must recognize meaningful semantic cues, understand enough of the environment layout to plan efficiently, and avoid wasting steps on repeated exploration or unnecessary backtracking. These demands are especially pronounced in large indoor environments, where the target may be many rooms away from the starting position.

Recent vision-language models are appealing for this problem because they provide strong semantic priors and commonsense reasoning. In principle, a VLM can infer that a refrigerator is likely to be in a kitchen, that a nightstand suggests a bedroom, or that a visible doorway may lead to more promising unexplored space. However, directly inserting a VLM into the navigation loop does not automatically yield an

efficient agent. Existing VLM-guided methods often suffer from two related problems. First, the VLM is typically given local or weakly structured observations, making it difficult to reason consistently over previously explored space. Second, the VLM is often used to choose among low-level frontier viewpoints at every step [84, 86, 88], even though these decisions require detailed 3D spatial understanding that current VLMs do not handle well [11, 50, 104].

STRIVE starts from the premise that semantic reasoning becomes more useful when it is grounded in a structured summary of the environment and applied at the right scale. The method therefore introduces two coupled ideas. The first is a multi-layer environment representation that incrementally organizes the scene into object nodes, viewpoint nodes, and room nodes. The second is a two-stage navigation policy in which the VLM is used for high-level room selection, while efficient in-room exploration is carried out at the viewpoint level. Together, these design choices aim to improve not only target-finding success but also path efficiency.

Within the broader thesis, STRIVE establishes that semantic reasoning needs structured context to be effective. It asks what kind of internal structure a navigation agent must build if semantic reasoning is to be genuinely useful. The answer proposed in this chapter is that semantic reasoning should not be attached to isolated observations after the fact; it should be supported by an explicit representation that organizes the environment into the units over which navigation decisions are naturally made.

3.2 Task Definition

In Object Navigation, the agent is required to find an instance of a given object category (e.g., "Find the *bed*") in an unknown environment. At each time step t, the agent receives a posed RGB-D observation $\mathbf{O}_t = \{I_t, D_t, P_t = \langle \mathbf{p}_t, \mathbf{R}_t \rangle\}$, where I_t is the RGB image, D_t is the depth map, and P_t is the camera pose. The navigation policy then predicts an action $a_t \in \{\texttt{move\_forward}, \texttt{turn\_left}, \texttt{turn\_right}, \texttt{stop}\}$. The task is considered successful if the agent stops within d_s meters of the target object in fewer than T steps.

3.3 Model

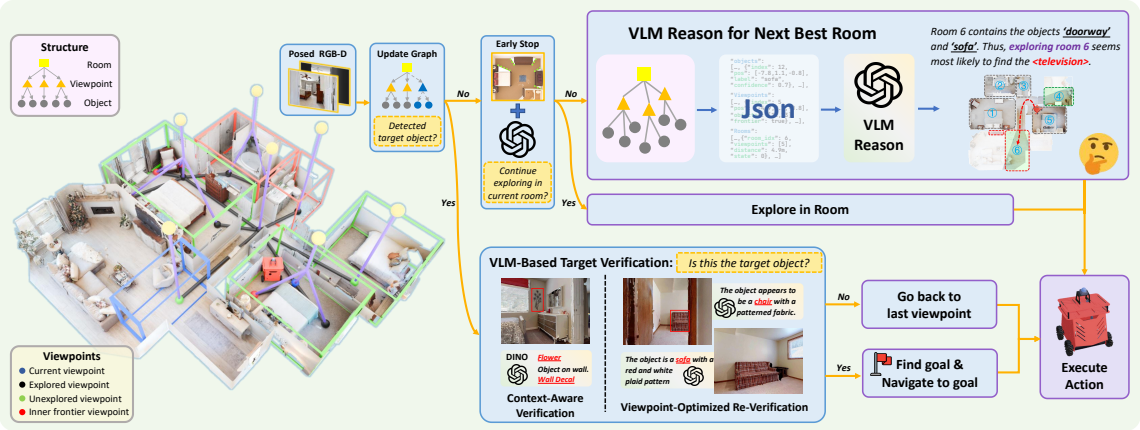

Figure 3.1: **Overview of the STRIVE pipeline.** The agent incrementally constructs a three-layer representation (object, viewpoint, and room nodes) and uses a two-stage policy: room-level VLM reasoning for cross-room selection and frontier-based exploration for in-room coverage.

The central idea of STRIVE is to incrementally construct a structured representation that exposes the environment at multiple granularities and then use this representation to separate high-level semantic reasoning from low-level exploration. The model is built around a three-layer graph representation $\mathcal{R}$ and a two-stage policy (Figure 3.1).

At a high level, STRIVE answers two questions. First, what information should be remembered from previous observations so that a VLM can reason about the environment coherently? Second, at what level should the VLM make decisions so that its semantic strengths are used without overloading it with geometric detail? The representation answers the first question, and the policy answers the second.

3.3.1 Representation Design

The STRIVE representation is organized around three node types.

Viewpoint Nodes Viewpoint nodes discretize the environment into representative locations from which local exploration can be organized. Inspired by skeleton

graphs [80], STRIVE defines a coverage range ζ_{cover} as the maximum distance within which objects are assumed to be reliably detected. Each viewpoint node thus controls a circular region of radius ζ_{cover}. The construction process works as follows. First, the region controlled by the current viewpoint is excluded from the observed point cloud, dividing the remaining cloud into separate regions. For regions containing frontiers [79], the detected frontiers are clustered into frontier edge segments and organized into a visibility graph. The Maximum Clique is iteratively removed from this graph, and for each removed clique, its center is added as a new viewpoint node serving as the exploration anchor for all frontier segments within the clique. For regions without frontiers, the center is directly added as a viewpoint node. Edges between viewpoint nodes are established based on straight-line traversability. This construction reduces the action space and provides a structured way to cover a room without repeatedly revisiting already understood regions. Figure 3.2 visualizes the viewpoint selection process, and Algorithm 1 summarizes the algorithm.

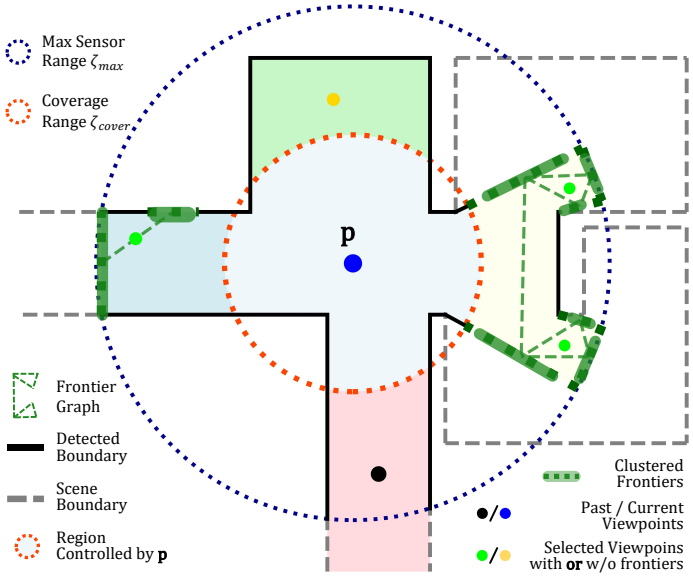

Figure 3.2: **Visualization of the viewpoint selection algorithm.** *Green* and *yellow* nodes are the selected viewpoints. Green dashed lines form the frontier visibility graph $G_{frontier}$.

Room Nodes Room nodes provide the highest-level abstraction in the representation. Each room node summarizes a semantically meaningful region of the environment

Algorithm 1 Viewpoint Construction Process

Require: Position p, Point Cloud $\mathcal{P}$
Ensure: Updated Viewpoint Nodes $\mathcal{V}^{vp}$
Calculate controlled region P_c
$\hat{\mathcal{P}} \leftarrow \mathcal{P} - P_c$
Regions $\mathcal{R} \leftarrow \texttt{Cluster}(\hat{\mathcal{P}})$
for region r_i in $\mathcal{R}$ **do**
    if r_i has no frontiers **then**
        $v^{vp} \leftarrow \texttt{Center}(r_i)$, Add new v^{vp} to $\mathcal{V}^{vp}$
    else
        $\mathcal{C}_i^F \leftarrow \texttt{FindFrontierCliques}(r_i)$
        for $c_{i,j}^F$ in $\mathcal{C}_i^F$ **do**
            $v^{vp} \leftarrow \texttt{Center}(c_{i,j}^F)$, Add new v^{vp} to $\mathcal{V}^{vp}$
        end for
    end if
end for

and supports room-level reasoning. Room nodes are constructed by identifying all walls in the environment through planar surface fitting and analyzing the distribution of point clouds along the vertical axis [28, 72]. By iteratively dilating the detected wall regions, the initially connected space is partitioned into multiple disconnected components, each treated as an individual room. This is a crucial design choice in STRIVE. Indoor object navigation is not naturally a sequence of isolated frontier choices; it is often better understood as a sequence of room-to-room hypotheses. For many targets, semantic priors are strongly tied to room categories and co-occurring objects. By introducing room nodes, STRIVE gives the VLM a more appropriate reasoning unit than individual frontiers. The room segmentation process works by first identifying all walls in the environment through planar surface fitting on the 3D point cloud and analyzing the distribution of point clouds along the vertical (z) axis. The algorithm detects horizontal planar surfaces that correspond to walls, then iteratively dilates these wall regions in the 2D top-down projection. As the wall regions expand, the initially connected walkable space is progressively partitioned into multiple disconnected components. Each component is treated as an individual room and added as a room node to the representation. Edges are then added between each room node and the viewpoint nodes located within the corresponding room,

establishing the hierarchical connection between the viewpoint and room layers. This wall-based segmentation is robust in typical indoor environments where rooms are separated by physical walls and doorways, though it may produce less meaningful partitions in open-plan layouts.

Object Nodes Object nodes are constructed using open-vocabulary detection and segmentation methods [29, 106]. Given an RGB image, Grounding DINO [106] performs open-vocabulary object detection, and the detected bounding boxes are fed into SAM [29] to obtain segmentation masks. By combining the depth map and camera pose, the 3D point cloud of each detected object is reconstructed. Each object node stores attributes including 3D position, point cloud, predicted label, confidence score, and 3D bounding box. Newly instantiated nodes are merged with previously observed nodes if they correspond to the same physical object. An edge is added between a viewpoint node and an object node if the object is within the viewpoint's coverage range and visible from it. An object can be associated with multiple viewpoints, which provides redundant semantic evidence that improves robustness. If an object is not connected to any viewpoint—which can occur when the object is detected at the boundary of the sensor range—it is connected to the closest visible viewpoint to ensure it remains accessible to the navigation policy. Newly instantiated object nodes are merged with previously observed nodes if they correspond to the same physical object, using a combination of spatial proximity and semantic label matching. This merging process is important for maintaining a compact and consistent representation as the agent revisits areas and re-observes the same objects from different viewpoints.

The three node layers are connected by three edge types.

Viewpoint–Viewpoint Edges e^{v-v} Viewpoint–viewpoint edges connect viewpoint nodes that are mutually reachable via straight-line paths within a room, supporting efficient intra-room navigation.

Room–Viewpoint Edges e^{r-v} Room–viewpoint edges connect each room node to the viewpoint nodes located within it, providing the hierarchical link between room-level reasoning and viewpoint-level exploration.

Viewpoint–Object Edges e^{v-o} Viewpoint–object edges connect a viewpoint to each object within its coverage range that is visible from it, enabling the policy to retrieve semantic associations for target verification. Together, these nodes and edges form a hierarchical graph over which the two-stage navigation policy operates.

3.3.2 Navigation Policy

STRIVE uses a two-stage navigation policy that mirrors the structure of the representation. High-level semantic reasoning operates over rooms, while local exploration operates over viewpoint nodes within the selected room.

In-Room Exploration

Inside a room, the agent performs viewpoint-level exploration rather than asking the VLM to score every possible low-level action. STRIVE distinguishes between true frontiers, which indicate genuinely unexplored room boundaries, and inner frontiers, which are typically caused by occlusion from furniture or clutter. The agent first resolves true frontiers to cover the room systematically. If only inner frontiers remain, a VLM-assisted early-stop mechanism uses the current visual and contextual evidence to decide whether further exploration inside the room is necessary. This prevents the system from spending too many steps resolving small occluded regions with low semantic value.

The distinction between true and inner frontiers is made through geometric analysis of the frontier's location relative to the room boundary. True frontiers lie along the boundary between the current room and unexplored or adjacent regions, typically corresponding to doorways, hallways, or open passages. Inner frontiers arise within the room itself, caused by furniture such as sofas, bookshelves, or kitchen islands that occlude small regions of space behind them. The early-stop mechanism queries the VLM with the current observation context and asks whether further exploration of the room is necessary. The VLM's response determines whether the agent continues exploring or exits the room to pursue cross-room navigation. This mechanism is important because it prevents a common failure mode of frontier-based exploration: spending excessive steps resolving small occluded regions that have no semantic relevance to the task. In practice, the VLM correctly identifies that inner

frontiers behind a couch are unlikely to warrant continued search for a refrigerator, or that frontiers behind a bookshelf are unlikely to conceal a bed, allowing the agent to terminate exploration early and move to a more promising room.

Cross-Room Selection

When a room has been sufficiently explored without finding the target, STRIVE switches to high-level room selection. The task-relevant context is consolidated into a structured prompt containing four components: (1) the target object instruction (e.g., "Find the *bed*"), (2) the agent's current viewpoint and position, (3) the navigation history as a sequence of visited viewpoints with positions, and (4) the environment representation $\mathcal{R}$ formatted as a JSON file describing rooms, viewpoints, and objects with their attributes. The VLM is explicitly instructed to evaluate two factors: the semantic similarity between objects in each room and the target object, and the distance from the agent's current position to each room. Using a Chain-of-Thought reasoning strategy, the VLM selects the most suitable unexplored room. Finally, the viewpoint closest to the current position in the selected room is chosen as the next action viewpoint.

To further reduce wasteful motion, STRIVE introduces a penalized distance mechanism. In the later navigation stage, continuing forward is more effective than backtracking, as remaining steps may not allow long detours. The penalized distance is defined as

$$d_{\text{pen}}(v_j^{room}) = \left(1 + \frac{t}{2T_{\max}}\right)^{n_j} \cdot d_{\text{geo}}(v_j^{room}), \tag{3.1}$$

where t is the current step number, $T_{\max}$ is the maximum allowed steps, n_j is the number of previously explored viewpoints along the shortest path to candidate room v_j^{room}, and $d_{\text{geo}}(v_j^{room})$ is the geodesic distance from the agent's current position to the nearest viewpoint in room v_j^{room}. The time ratio $t/(2T_{\max})$ grows as the episode progresses, and the exponent n_j penalizes paths that traverse previously explored regions. In early navigation stages, when t is small, the penalty factor is close to unity and the VLM can freely select rooms based on semantic promise. In later stages, rooms reachable via shorter and less-explored paths are preferred, discouraging backtracking to distant rooms that were bypassed earlier in the trajectory. This design encourages forward exploration momentum while preserving the VLM's ability

to make semantically informed exceptions.

Target Verification

Because open-vocabulary detection can generate false positives, STRIVE uses VLM-based target verification as an additional reliability layer. The first stage uses contextual visual evidence around a detected object to determine whether it is a plausible instance of the target category. The second stage performs a single re-verification from an optimized viewpoint, unlike prior methods such as SG-Nav [84] which observe from multiple viewpoints. Specifically, STRIVE samples candidate points along the path from the agent's current position to the detected object and selects the first point (traversed in reverse from the object) that provides sufficient visibility of the target's point cloud and a bounding box no smaller than at the original viewpoint. This single-viewpoint design reduces verification cost while providing the VLM with a clearer and more informative observation than the initial detection.

3.4 Implementation Details

STRIVE uses Gemini [61] (gemini-2.0-flash) as the VLM backbone for both room-level reasoning and target verification. For open-vocabulary object detection and segmentation, it employs MM-Grounding-DINO-L [106] and SAM-2.1-L [29]. Each episode allows a maximum of 500 steps with a success distance threshold of 1.0 m. The agent observes the environment using 640×480 RGB-D images with depth values ranging from 0.5 m to 5.0 m and a horizontal field of view of 79°. The camera is mounted 0.88 m above the ground. The agent moves forward by 0.25 m per step and rotates by 30°. All experiments are conducted on RTX 4090 GPUs.

For real-world deployment, STRIVE is deployed on a Mecanum wheel platform [92] equipped with a Ricoh Theta Z1 360-degree camera for RGB image capturing and a Livox Mid-360 LiDAR for 3D point cloud acquisition. To maintain compatibility with the simulation input format, the collected LiDAR point clouds are converted into depth maps when necessary. The sparsity of LiDAR-captured point clouds compared to simulated depth maps introduces additional noise in both object detection and frontier identification, making the real-world evaluation a meaningful test of robustness.

3.5 Experiments

STRIVE is evaluated on four widely used simulation benchmarks. HM3D-v1 [78] includes 2,000 episodes across 20 high-fidelity scenes with 6 target object categories. HM3D-v2 [48] includes 1,000 episodes across 36 scenes with 6 categories. RoboTHOR [21] includes 1,800 episodes across 15 scenes with 12 categories. MP3D [9] includes 2,195 episodes across 11 scenes with 21 categories. The experiments use standard ObjectNav metrics including Success Rate (SR), Success weighted by Path Length (SPL), SoftSPL, and Distance to Goal (DTG). The VLM backbone is used without any task-specific fine-tuning, making STRIVE a purely zero-shot method. In addition to simulation, STRIVE is validated through 120 real-world experiments across 15 target categories and ten indoor environments—including offices, meeting rooms, classrooms, lounges, dining areas, corridors, and kitchens.

The experiments are designed to answer three questions. First, does the proposed structured representation and two-stage policy improve both target-finding success and path efficiency relative to prior methods? Second, does room-level VLM reasoning provide a better operating point than viewpoint-level VLM planning? Third, do the auxiliary components, namely early stopping, penalized distance, and VLM-based verification, each make measurable contributions?

3.5.1 Main Benchmark Comparison

STRIVE achieves state-of-the-art performance across all four simulated benchmarks.

On HM3D-v1 it reaches 62.9% SR and 34.2% SPL, improving +6.4% SR and +3.6% SPL over the second-best baseline. On HM3D-v2 it reaches 79.6% SR and 38.7% SPL, a gain of +13.1% SR and +6.2% SPL. On RoboTHOR it reaches 68.1% SR and 36.3% SPL, improving +20.6% SR and +12.3% SPL. On MP3D it reaches 52.3% SR and 23.1% SPL, gaining +11.2% SR and +5.5% SPL (Table 3.1). The gains are especially notable in SPL across all benchmarks, which indicates that STRIVE improves not only whether the agent succeeds, but how efficiently it reaches the target.

Table 3.1: **Comparison with SOTA methods with different settings on four ObjectNav benchmarks.** Best results are in **bold**, second best are underlined. VLFM♯ replaces the pre-trained PointNav module with a shortest-path planner for fair comparison. STRIVE* denotes evaluation restricted to episodes where the agent's starting position and the target object are located on the same floor.

Method	Open-Set	Zero-Shot	HM3D-v1		HM3D-v2		RoboTHOR		MP3D	
			SR (%) ↑	SPL (%) ↑	SR (%) ↑	SPL (%) ↑	SR (%) ↑	SPL (%) ↑	SR (%) ↑	SPL (%) ↑
SemEXP [10]	✗	✗	-	-	-	-	-	-	36.0	14.4
PONI [53]	✗	✗	-	-	-	-	-	-	31.8	12.1
ZSON [44]	✓	✗	25.5	12.6	-	-	-	-	15.3	4.8
L3MVN [88]	✗	✓	50.4	23.1	36.3	15.7	41.2	22.5	34.9	14.5
ESC [109]	✓	✓	39.2	22.3	-	-	38.1	22.2	28.7	11.2
VoroNav [74]	✓	✓	42.0	26.0	-	-	-	-	-	-
VLFM [86]	✓	✓	52.5	30.4	63.6	32.5	-	-	36.4	17.5
VLFM♯ [86]	✓	✓	50.9	23.6	56.9	27.5	-	-	32.5	15.9
SG-Nav [84]	✓	✓	54.0	24.9	49.6	25.5	47.5	24.0	40.2	16.0
OpenFMNav [34]	✓	✓	54.9	24.4	-	-	44.1	23.3	37.2	15.7
TriHelper [100]	✓	✓	56.5	25.3	-	-	-	-	-	-
InstructNav [39]	✓	✓	-	-	58.0	20.9	-	-	-	-
DORAEMON [25]	✓	✓	55.6	21.4	66.5	20.6	-	-	41.1	15.8
STRIVE	✓	✓	**62.9**	**34.2**	**79.6**	**38.7**	**68.1**	**36.3**	**52.3**	**23.1**
STRIVE*	✓	✓	**72.7**	**38.2**	-	-	-	-	-	-

3.5.2 Representation Ablation

The representation ablation examines how each node type contributes to navigation performance.

V^{vp}	V^{obj}	V^{room}	SR ↑	SPL ↑	S-SPL ↑	DTG(m) ↓
✓	✗	✗	71.3	33.2	35.2	1.86
✓	✓	✗	72.4	34.0	36.1	1.95
✓	✗	✓	72.9	33.8	35.4	1.86
✓	✓	✓	**75.0**	**34.9**	**36.5**	**1.80**

Table 3.2: **Representation ablation on HM3D.** Using a viewpoint-level policy for consistency. Object and room nodes contribute complementary information.

In the representation ablation (Table 3.2), using only viewpoint nodes yields 71.3 SR and 33.2 SPL. Adding object nodes or room nodes individually each improves performance, and using both together yields the strongest result at 75.0 SR and 34.9 SPL. This supports the claim that object semantics and room structure contribute complementary information.

3.5.3 Policy Ablation

The policy ablation examines the contribution of each policy component (Table 3.3). Replacing the room-level planning policy with viewpoint-level VLM planning reduces performance while also sharply increasing token usage from 8,068 to 22,935 tokens per episode. This result is particularly important for the thesis narrative: the VLM is most effective when used for high-level room reasoning rather than dense low-level planning. Additional ablations show that removing early stopping, penalized distance, or VLM verification each degrades performance, confirming that these components improve efficiency and robustness in complementary ways.

	SR ↑	SPL ↑	S-SPL ↑	DTG(m) ↓	Tokens ↓
w/o Early Stop	74.8	34.8	36.4	1.62	-
w/o Penalized Dist	73.7	36.1	36.9	1.47	-
w/o VLM-Verify	72.1	32.7	34.1	1.83	-
Viewpoint Policy	75.0	34.9	36.5	1.80	22935
STRIVE	**79.6**	**38.7**	**38.9**	**1.29**	**8068**

Table 3.3: **Policy ablation on HM3D.** Room-level planning reduces token usage by 65% while improving performance.

3.5.4 Real-World Evaluation

Real-world experiments further show that STRIVE is not only a benchmark method. Across 120 episodes in 10 indoor environments, the system remains robust in diverse layouts. Runtime evaluation shows high success rates with short average completion times, demonstrating that the structured representation and room-level reasoning remain effective under realistic sensing conditions. Figure 3.3 visualizes a representative navigation episode.

The key thesis-level conclusion from these results is not merely that STRIVE scores well. Rather, the evidence shows why it works: the structured representation grounds semantic reasoning, and the two-stage policy assigns the VLM to the level of decision making where it provides the most value.

This interpretation is important because it separates STRIVE from a purely benchmark-driven contribution. The chapter does not only report that one system

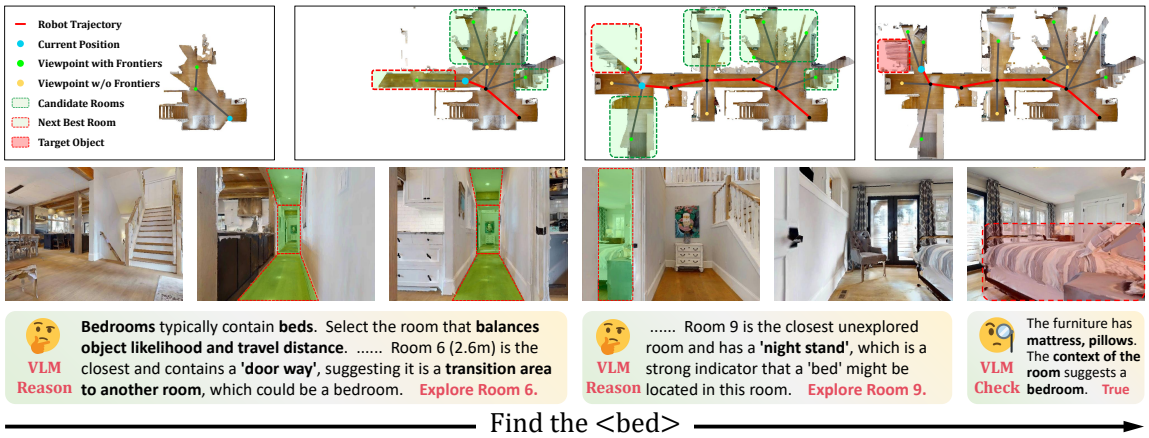

Figure 3.3: **Qualitative visualization of STRIVE's navigation process.** The VLM jointly considers room layout, semantic cues, and travel cost when selecting rooms, and the early-stop mechanism prevents excessive exploration of low-value inner frontiers.

outperforms another. It shows that the gain comes from a more appropriate interface between semantics and navigation structure. That observation becomes the foundation for the rest of the dissertation. Detailed qualitative analysis and token usage studies are provided in Appendix A.

3.6 Conclusion

STRIVE establishes the first major claim of this thesis: VLM-guided object navigation becomes substantially more effective when semantic reasoning is grounded in a structured, multi-level representation of the environment. By organizing observations into object, viewpoint, and room nodes, STRIVE provides the VLM with a coherent summary of the scene rather than fragmented local evidence. By combining this representation with room-level planning and efficient viewpoint-level exploration, it improves both success and path efficiency.

Within the larger thesis narrative, STRIVE therefore serves as the representation chapter. It demonstrates that better structure leads to better semantic reasoning and better navigation. At the same time, it also reveals the next limitation: even a well-structured navigation method still needs a full system architecture if it is to scale to real deployment settings. The next chapter builds on this insight and asks

the next question in the progression: if room-level semantic reasoning is effective, how should it be embedded in a deployable real-world navigation system that works across multiple robot embodiments?

Chapter 4

SysNav: Multi-Level Systematic Cooperation for Real-World, Cross-Embodiment Object Navigation

4.1 Introduction

Real-world object navigation is more than a benchmark task. A deployable system must handle noisy perception, long-horizon planning, embodiment-specific constraints, semantic ambiguity, and real-time execution. These requirements expose the limitations of viewing object navigation as a single-policy problem. Even a method that performs well in simulation can fail in practice if its perception is brittle, if it overuses semantic reasoning at the wrong scale, or if it cannot transfer across robot embodiments. Moreover, the real world introduces terrain diversity: a wheeled robot, a quadruped, and a humanoid face fundamentally different mobility constraints, and a single-platform system may fail to reach the target purely for locomotion reasons even when its navigation reasoning is sound.

SysNav addresses these gaps by treating object navigation as a system-level problem rather than a single learned policy. The central premise is that real-world

navigation combines several qualitatively different subproblems: semantic reasoning about where a target is likely to appear, planning over partially explored environments, and low-level embodiment-specific motion execution. Asking one model to solve all of these simultaneously is both inefficient and fragile. SysNav instead decouples these functions into three cooperating levels while preserving a consistent room-based planning logic across embodiments.

A second key insight concerns the appropriate use of VLM reasoning. Although VLMs exhibit strong commonsense reasoning abilities, many existing methods over-rely on them for fine-grained exploration decisions—scoring frontiers, selecting viewpoints, or generating low-level waypoints at every step. While such strategies may work in clean simulation, they perform poorly under real-world noise and do not leverage the VLM's actual strengths. VLMs lack precise 3D spatial understanding, and involving them in dense geometric decisions wastes computation while introducing brittleness. SysNav argues that VLM reasoning should be reserved for the spatial level where it provides the most value: semantically grounded decisions over rooms. Inside a room, efficient classical exploration algorithms handle geometric coverage without VLM involvement.

Within the thesis narrative, this chapter marks the transition from representation design to deployable system design. STRIVE showed that room-level semantic reasoning works better when grounded in a structured representation. SysNav asks the next question: how can this idea be organized into a robust system that works on real robots, in long-range settings, and across multiple embodiments? That shift in emphasis is important. The problem is no longer only whether semantic reasoning can improve navigation decisions, but whether it can do so under the operational constraints of an actual robot system—handling imperfect sensing, coordinating multiple planning horizons, and preserving the same high-level logic across very different robot bodies.

4.2 Model

The core idea of SysNav is multi-level systematic cooperation (Figure 4.1). Rather than forcing one model to solve semantic reasoning, path planning, and motion execution simultaneously, SysNav decomposes the task into three levels: a high-level

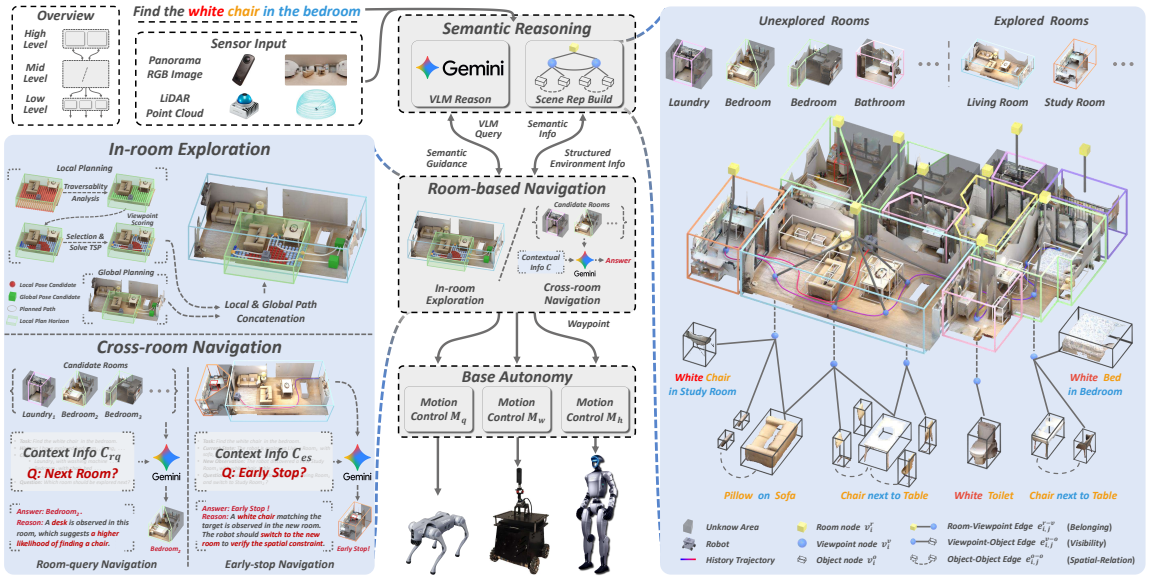

Figure 4.1: **System overview of SysNav.** The system decomposes the task into three levels: high-level semantic reasoning with structured scene representation, mid-level room-based navigation, and low-level base autonomy across multiple embodiments.

semantic reasoning module, a mid-level room-based navigation module, and a low-level base autonomy module. Each level is assigned a specific role, and the levels communicate through structured intermediate representations.

4.2.1 High-Level Semantic Reasoning

At the high level, SysNav organizes environmental information into a structured scene representation and uses a VLM to provide semantic-grounded guidance. The representation is built incrementally as the agent explores and consists of three node types with associated edges.

Room nodes v^r: To decompose the environment into semantically meaningful high-level structural units and exploit this structure for efficient exploration, SysNav introduces room nodes at the top layer of the scene representation, where each room node represents an individual room. Room nodes are constructed by fitting planar surfaces to detect wall structures and analyzing point cloud distributions along the vertical (z) axis [28, 72]. By dilating the detected wall regions, the initially

connected space is partitioned into multiple disconnected components, each treated as an individual room and added to the representation $\mathcal{R}$ as a room node v_i^r. For each room node, SysNav maintains a set of attributes $\boldsymbol{A}(v_i^r) = \{m_i^r,\ c_i^r,\ I_i^r\}$, where m_i^r is a 2D top-down room mask, c_i^r is the room category (e.g., kitchen or bedroom), and I_i^r is a representative RGB image that provides the best view of the room.

Viewpoint nodes v^v: To efficiently store semantic information and discretely represent explored areas, viewpoint nodes are introduced at the middle layer. Each viewpoint node corresponds to a previously visited location, capturing surrounding semantic and geometric information within a defined range. These nodes together form a compact discrete representation of the environment. A coverage distance d_{cover} defines the range within which voxels are considered fully observed, both semantically and geometrically, forming the coverage region $\mathcal{C}(v_i^v)$. During exploration, the system evaluates the coverage region at the robot's current pose against the accumulated coverage of all existing viewpoints, $\mathcal{C}_{prev} = \bigcup_{v_i^v \in \mathcal{R}} \mathcal{C}(v_i^v)$. If the newly observed region provides sufficient novel coverage, i.e., $|\mathcal{C}_t \setminus \mathcal{C}_{prev}| > \epsilon$, the current pose is added as a new viewpoint node. Each viewpoint node maintains attributes $\boldsymbol{A}(v_i^v) = \{p_i,\ \mathcal{C}_i,\ I_i\}$, where p_i is the position, $\mathcal{C}_i$ is the coverage region, and I_i is a panorama image captured at the viewpoint. This design stores RGB images only at representative viewpoints rather than along the entire trajectory, enabling efficient storage of semantic information. The formal coverage criterion is defined in Appendix B.

Object nodes v^o: To model object instances, object nodes are introduced at the lowest layer. Each object node corresponds to an object instance in the environment, constructed using open-vocabulary detection and segmentation methods [18, 57, 63]. At each time step, the 3D point cloud of each detected object is reconstructed from predicted masks, depth data, and robot pose. Newly detected objects are added as object nodes and merged with existing nodes if they represent the same physical instance. Each object node maintains attributes $\boldsymbol{A}(v_i^o) = \{c_i,\ conf_i,\ P_i,\ bbox_i,\ I_i,\ (\varphi_1, \varphi_2, \ldots)\}$, where c_i is the object category, $conf_i$ is the detection confidence score, P_i is the reconstructed 3D point cloud, $bbox_i$ is the 3D bounding box, I_i is a representative RGB image providing the best view, and $(\varphi_1, \varphi_2, \ldots)$ are on-demand self-attributes such as color and material. To keep the representation compact, self-attributes are inferred only when the task instruction specifies attribute constraints ϕ for category c_i: the corresponding object nodes are

retrieved and their RGB images are used to prompt the VLM for attribute inference, appending the results φ to each node.

The three node layers are connected by five edge types:

Room–room edges e^{r-r}: To model room connectivity, an edge $e^{r-r}_{i,j}$ is added in $\mathcal{R}$ when two room nodes correspond to rooms connected in the environment via a doorway.

Room–viewpoint edges e^{r-v}: To model viewpoint-room affiliation, an edge $e^{r-v}_{j,i}$ is added to $\mathcal{R}$ when viewpoint v^v_i lies in room v^r_j.

Room–object edges e^{r-o}: To model object-room containment, an edge $e^{r-o}_{i,j}$ is added in $\mathcal{R}$ when object node v^o_j lies within room v^r_i.

Viewpoint–object edges e^{v-o}: To model object-viewpoint visibility, an edge $e^{v-o}_{i,j}$ is added in $\mathcal{R}$ when the robot at viewpoint v^v_i can observe object node v^o_j.

Object–object edges e^{o-o}: To model spatial relationships between object instances, object-object edges are added conditionally to avoid excessive graph density. If the task instruction specifies a spatial constraint ϕ (e.g., "the cup is on the table") between v^o_i and v^o_j, the system retrieves viewpoint nodes $\mathcal{V}_{i,j} = \{v^v_k \mid e^{v-o}_{k,i} \in \mathcal{R},\ e^{v-o}_{k,j} \in \mathcal{R}\}$ that observe both objects and uses their RGB images to prompt the VLM. If ϕ holds, an edge $e^{o-o}_{i,j}$ is added in $\mathcal{R}$ with attribute $\boldsymbol{A}(e^{o-o}_{i,j}) = \{\varphi\}$ indicating the satisfied spatial relation.

An important extension in SysNav is that the task setting is not restricted to simple category-level goals. The system supports semantic constraints such as attributes ("find a *red* chair") or spatial relations ("find the cup *on the table*"), making the representation more than a navigation map; it becomes a semantic interface that the VLM can reason over.

4.2.2 Mid-Level Room-Based Navigation

At the mid level, SysNav plans exploration strategies and waypoints. Rooms are treated as the minimal semantic decision units. The module includes two components: an in-room exploration policy and a cross-room navigation policy.

In-room exploration adopts a two-level planning structure inspired by TARE [6], consisting of local and global planning. A local planning horizon is defined as a set of sampled pose candidates $\mathcal{H} = \{c_1, \ldots, c_n\}$ within the current room. The

traversable candidates within the room are filtered by collision checking and the room mask: $\mathcal{H}_{trav}^{v_j^r} = \{c_i \in \mathcal{H} \mid c_i \text{ is traversable and within room } v_j^r\}$. For each candidate, a coverage score measures how many currently uncovered surface points—the boundary between free and non-free space—it can observe. The surface point set $\mathcal{S}$ represents the generalized boundary between explored free space and occupied or unknown regions. For each candidate c_i, the coverage score is $w_{cov}(c_i) = |\mathcal{S}_{cov}(c_i) \cap \hat{\mathcal{S}}|$, where $\hat{\mathcal{S}}$ is the set of currently uncovered surface points. Stochastic sampling guided by coverage scores selects pose candidates, iteratively updating the uncovered surface set until all remaining scores fall below a threshold δ. A Traveling Salesman Problem is then solved over the selected candidates to generate a local exploration path. This procedure is repeated K times, and the minimum-cost path is chosen. This stochastic sampling with multiple restarts provides robustness against local minima in the coverage landscape, which is important in cluttered rooms where the optimal exploration order is not obvious. Local and global planning are coordinated through a rolling window mechanism: chosen pose candidates leaving the local horizon are transferred to the global horizon, where another TSP generates the global exploration path.

This decomposition is central to the design. Rooms are semantically meaningful units, and many target objects have strong room-level priors. At the same time, efficient coverage inside a room is better handled by geometric exploration rather than repeated semantic prompting.

Cross-room navigation uses two VLM-guided decision modes that coordinate exploration order across rooms. The VLM acts as a high-level reasoner, evaluating each room's relevance to the current task online.

Early-stop mode. When exploring room v_i^r and discovering a new room v_j^r, the VLM is provided with contextual information

$$\mathcal{C}_{es} = \{\boldsymbol{A}(v_i^r),\ \{v_u^o \mid e_{i,u}^{r-o} \in \mathcal{R}\},\ \boldsymbol{A}(v_j^r),\ \{v_v^o \mid e_{j,v}^{r-o} \in \mathcal{R}\},\ \mathcal{G}\},$$

where $\boldsymbol{A}(\cdot)$ and $\{v_\cdot^o \mid e_{\cdot,\cdot}^{r-o} \in \mathcal{R}\}$ denote room attributes and the object sets belonging to the corresponding room respectively, and $\mathcal{G}$ is the task goal. Based on $\mathcal{C}_{es}$, the VLM decides whether to terminate exploration in the current room and switch to the new one. This semantic-aware interruption mechanism enables dynamic focus

adjustment and avoids redundant exploration of semantically irrelevant rooms.

Room-query mode. If in-room exploration in v_i^r completes without locating the target, the system switches to a global reassessment mode. The VLM is provided with contextual information

$$\mathcal{C}_{rq} = \{\{\boldsymbol{A}(v_k^r) \mid v_k^r \in \mathcal{R}_{uncov}\},\ Traj,\ \mathcal{G}\},$$

including attributes of all uncovered rooms, the robot's trajectory, and the task goal. Based on this context, the VLM performs semantic reasoning over the candidate rooms and selects the room most likely to contain the target object. By restricting semantic reasoning to room-level decision points, the system leverages VLM strengths while preserving overall navigation efficiency.

4.2.3 Low-Level Base Autonomy

At the low level, planned waypoints are executed by embodiment-specific motion control modules through a base autonomy system [7]. This system translates planned waypoints into embodiment-specific motion commands and incorporates waypoint-following, collision avoidance, and terrain traversability analysis to ensure safe and efficient execution. The separation between mid-level planning and low-level control is what allows the same system logic to transfer across wheeled, quadruped, and humanoid robots.

This low-level abstraction is one of SysNav's most important practical contributions. For the wheeled robot, it handles differential drive control and 2D LiDAR-based obstacle avoidance. For the Unitree Go2 quadruped, it manages legged locomotion over uneven terrain. For the Unitree G1 humanoid, it coordinates bipedal walking with balance maintenance. In all cases, the high-level semantic reasoning and mid-level navigation planning remain identical; only the execution layer changes. This means that semantic reasoning and room-based planning do not have to be redesigned for each embodiment.

4.2.4 Scene Representation and Constraints

SysNav expands the representation perspective of STRIVE into a richer system component. It uses room, viewpoint, and object abstractions while also supporting semantic constraints such as attributes and inter-object relations. In the source formulation, object self-attributes can be inferred on demand, and object-object relations can be verified only when required by the task. This keeps the system compact while still allowing richer semantic goals than plain object category navigation.

From the perspective of this thesis, this is an important shift. In STRIVE, structure mainly improved efficiency and semantic grounding for category-level navigation. In SysNav, structure also becomes a mechanism for scaling semantic reasoning to richer tasks and more realistic deployments.

4.2.5 Room-Based Navigation as a System Principle

One of the most important conceptual contributions of SysNav is that room-level reasoning is no longer treated as a local method choice. Instead, it becomes a system principle. The VLM is reserved for decisions over semantically meaningful structural units, while geometric coverage and embodiment-specific execution are handled elsewhere in the stack. This yields a cleaner allocation of responsibility and helps explain the system's real-world robustness.

This design directly extends the argument of the previous chapter. STRIVE showed that room-level semantic reasoning is more efficient than dense viewpoint-level VLM planning. SysNav elevates that observation into an organizing principle for the whole navigation system.

4.3 Implementation Details

SysNav uses open-vocabulary detection and segmentation methods including YOLO-World [18], YOLOE [63], and SAM-2 [57] for object instance extraction. Point clouds are obtained from LiDAR sensors on real robots and from depth cameras in simulation. The base autonomy system provides waypoint-following, collision avoidance, and terrain traversability analysis for each embodiment. For the wheeled robot, it handles

differential drive control and 2D LiDAR-based obstacle avoidance. For the Unitree Go2 quadruped, it manages legged locomotion over uneven terrain with body pose stabilization. For the Unitree G1 humanoid, it coordinates bipedal walking with balance maintenance using the robot's built-in locomotion controller. In all cases, the high-level semantic reasoning and mid-level navigation planning remain identical; only the execution layer changes.

The VLM used for high-level semantic reasoning and cross-room navigation is Gemini [61] (gemini-2.5-flash), a general-purpose pretrained model used without task-specific fine-tuning. The coverage distance d_{cover} is set to 1.5 m for real-world deployment. The TARE-inspired in-room planner uses $K = 5$ stochastic sampling iterations and solves TSP using standard solvers. The system runs on an onboard laptop with an NVIDIA GPU for real-time inference.

4.4 Experiments

SysNav is evaluated in both simulation and real-world settings. The simulation evaluation spans four benchmarks: HM3D-v1 [78], HM3D-v2 [48], MP3D [9], and HM3D-OVON [87], totaling 8,195 episodes. HM3D-OVON is an open-vocabulary benchmark that extends the standard ObjectNav setting to a larger set of object categories (49 target categories across 36 scenes), testing the system's ability to generalize beyond a fixed category list. The real-world evaluation is particularly important: the system is deployed on three embodiments—a custom wheeled robot [92], the Unitree Go2 quadruped, and the Unitree G1 humanoid. Across these platforms, the source paper reports 190 real-world experiments: 78 episodes for qualitative evaluation and 112 episodes for quantitative evaluation. The qualitative episodes include 4 in floor-scale environments, 41 on the wheeled robot, 19 on the quadruped, and 14 on the humanoid, across 15 scenes with an average area of 432 m^2.

The experimental design tests not only benchmark performance but also deployment robustness. The main questions are as follows. First, can a multi-level system outperform prior methods in success and efficiency? Second, does room-level semantic guidance remain useful under real-world sensing noise and building-scale exploration? Third, can the same navigation logic transfer across multiple embodiments without redesigning the entire system?

The paper evaluates performance using SR and SPL in simulation. In real-world settings where shortest paths are not directly available, it additionally reports Success Penalized by Time ($SPT = SR \cdot (1 - t/T_{timeout})$), which rewards fast completion while penalizing failures, along with Average Time (AT) over successful episodes. This aligns well with the thesis emphasis on efficiency in addition to raw success.

Table 4.1: **Quantitative comparison of SysNav with state-of-the-art methods on four ObjectNav benchmarks.** Best results in **bold**, second-best underlined.

Method	Open-Set	Zero-Shot	HM3D-v1		HM3D-v2		MP3D		HM3D-OVON	
			SR (%)	SPL (%)	SR (%)	SPL (%)	SR (%)	SPL (%)	SR (%)	SPL (%)
L3MVN [88]	✗	✓	50.4	23.1	36.3	15.7	34.9	14.5	-	-
ESC [109]	✓	✓	39.2	22.3	-	-	28.7	11.2	-	-
VoroNav [74]	✓	✓	42.0	26.0	-	-	-	-	-	-
VLFM [86]	✓	✓	52.5	30.4	63.6	32.5	36.4	17.5	35.2	19.6
SG-Nav [84]	✓	✓	54.0	24.9	49.6	25.5	40.2	16.0	-	-
OpenFMNav [34]	✓	✓	54.9	24.4	-	-	37.2	15.7	-	-
TriHelper [100]	✓	✓	56.5	25.3	-	-	-	-	-	-
InstructNav [39]	✓	✓	-	-	58.0	20.9	-	-	-	-
MTU3D [114]	✓	✓	-	-	-	-	-	-	40.8	12.1
ApexNav [102]	✓	✓	59.6	**33.0**	76.2	**38.0**	39.2	17.8	-	-
SysNav	✓	✓	**63.7**	30.5	**80.8**	37.2	**50.7**	**18.1**	**54.9**	**26.1**

4.4.1 Simulation Results

Table 4.1 presents the quantitative comparison with state-of-the-art methods on four simulation benchmarks. SysNav consistently outperforms all baselines across all benchmarks. On HM3D-OVON, the most challenging open-vocabulary benchmark with 49 target categories, SysNav achieves 54.9% SR and 26.1% SPL, improving over the strongest baseline (ApexNav) by 14.1% in SR and 6.5% in SPL. On the closed-set benchmarks, SysNav achieves 63.7% SR on HM3D-v1, 80.8% on HM3D-v2, and 50.7% on MP3D, all ranking first. The smaller improvement in SPL compared to SR arises from the system's real-world-oriented design, which adopts stricter coverage strategies to handle environmental complexity and sensor noise; this can cause slight over-coverage in simulation and reduce path efficiency. Nevertheless, the results demonstrate that the multi-level architecture benefits are not limited to one deployment platform or environment type.

4.4.2 Real-World Results

For quantitative evaluation, 112 episodes are conducted on the wheeled robot platform across 10 target object categories and 3 real-world environments, categorized into easy (24 episodes, single room, target mostly in initial view), medium (40 episodes, single room requiring moderate exploration), and hard (48 episodes, three rooms requiring cross-room exploration) difficulty levels. Table 4.2 presents the results. SysNav significantly outperforms all baselines across all difficulty levels. In the hard setting, SR improves from 37.2% (best baseline) to 98.3%, SPT from 20.7% to 71.8%, and AT decreases from 97.4 seconds to 67.6 seconds. Notably, the slightly better performance in the hard setting than medium for SysNav arises because multi-room layouts pose limited additional difficulty for the room-based system, while the medium scene contains denser obstacles that slow the robot.

Table 4.2: **Real-world comparison of SysNav with baselines across Easy, Medium, and Hard difficulty levels.** SR (%), SPT (%, higher is better), and AT (seconds, lower is better) are reported.

Method	Easy			Medium			Hard		
	SR	SPT	AT	SR	SPT	AT	SR	SPT	AT
VLFM [86]	58.3	40.3	52.1	47.5	34.4	75.3	25.6	12.9	97.4
InstructNav [39]	45.8	29.5	67.4	55.0	27.8	84.6	37.2	20.7	103.4
SysNav	**100.0**	**83.8**	**29.1**	**97.5**	**67.9**	**72.8**	**98.3**	**71.8**	**67.6**

In real-world deployment, SysNav delivers roughly four-fold gains in navigation efficiency over existing baselines. Just as importantly, it is reported as the first system capable of reliably and efficiently completing floor-scale long-range object navigation in complex real-world environments. The qualitative real-world results are particularly illustrative. In floor-scale scenarios (average area 1007 m^2, average geodesic distance 84 m), the system successfully coordinates multi-room exploration with semantic constraints such as attribute matching ("find a *red* chair") and spatial relations ("find the cup *on the table*"). The VLM reasoning trace shows that the system correctly identifies room types based on observed objects, re-evaluates exploration priorities when new rooms are discovered, and uses attribute constraints to filter candidate objects.

The cross-embodiment evaluation is one of the most distinctive aspects of SysNav. The same high-level semantic reasoning and mid-level navigation modules are deployed on all three platforms—wheeled, quadruped, and humanoid—with only the low-level base autonomy module being embodiment-specific. This demonstrates that the navigation intelligence is decoupled from the physical platform, a property that is essential for practical deployment. Platform-specific differences in traversal speed, sensor height, and terrain handling are discussed in Appendix B.

4.4.3 Qualitative Real-World Analysis

Figure 4.2 shows qualitative results from 78 real-world episodes across 15 scenes. The first three rows illustrate floor-scale object navigation on the wheeled robot platform. In floor-scale scenarios involving multiple rooms and long corridors, the system successfully coordinates multi-room exploration with semantic constraints. The VLM reasoning trace reveals several important behaviors. First, when the agent enters a new room, it rapidly identifies the room type based on observed objects (e.g., recognizing a kitchen from the presence of a refrigerator, stove, and countertop). Second, when the early-stop mode triggers—for example, when a bedroom is discovered while exploring a living room—the VLM correctly weighs the semantic relevance of the new room against the remaining unexplored portions of the current room. Third, the room-query mode enables strategic reassessment: after exhausting local exploration, the VLM considers all uncovered rooms and their object contents to select the most promising next destination.

The system also demonstrates robustness to real-world challenges. LiDAR point clouds are substantially sparser than simulated depth maps, leading to noisier frontier detection and less precise wall identification. Despite this, the room segmentation algorithm produces reasonable room boundaries in most environments. Object detection is also more challenging in the real world due to varying lighting conditions, partial occlusions, and diverse object appearances. The system handles these challenges through the same VLM-based verification mechanisms introduced in STRIVE, adapted for the richer scene representation of SysNav.

From the perspective of this thesis, the most important conclusion is not only that SysNav improves performance, but that it explains how to operationalize semantic

reasoning in the real world. The room-based reasoning idea remains central, but in SysNav it succeeds because it is embedded inside a complete architecture with explicit interfaces among semantic reasoning, planning, and control.

This is the main reason the chapter matters beyond its headline results. SysNav turns a methodological insight from STRIVE into a system design principle. In doing so, it narrows the gap between benchmark success and practical embodied deployment, which is one of the central concerns of this dissertation.

4.5 Conclusion

Within the thesis narrative, SysNav establishes the system layer of the argument. STRIVE showed that structured representation improves VLM-guided navigation. SysNav shows that this representation-and-reasoning perspective can be elevated into a robust system architecture for real-world deployment, long-range navigation, and cross-embodiment transfer.

The core lesson of this chapter is that object navigation should not be treated as a single undifferentiated policy problem. Instead, it becomes more robust when semantic reasoning, navigation planning, and control are explicitly separated and coordinated. This sets up the next technical chapter of the thesis. Once representation and system decomposition are in place, the next remaining question is whether navigation decisions should continue to rely on zero-shot VLM prompting, or whether they can be learned from data to produce more stable long-horizon behavior. That question is addressed by IntentNav, which turns from system organization to learning navigation intent from human demonstrations.

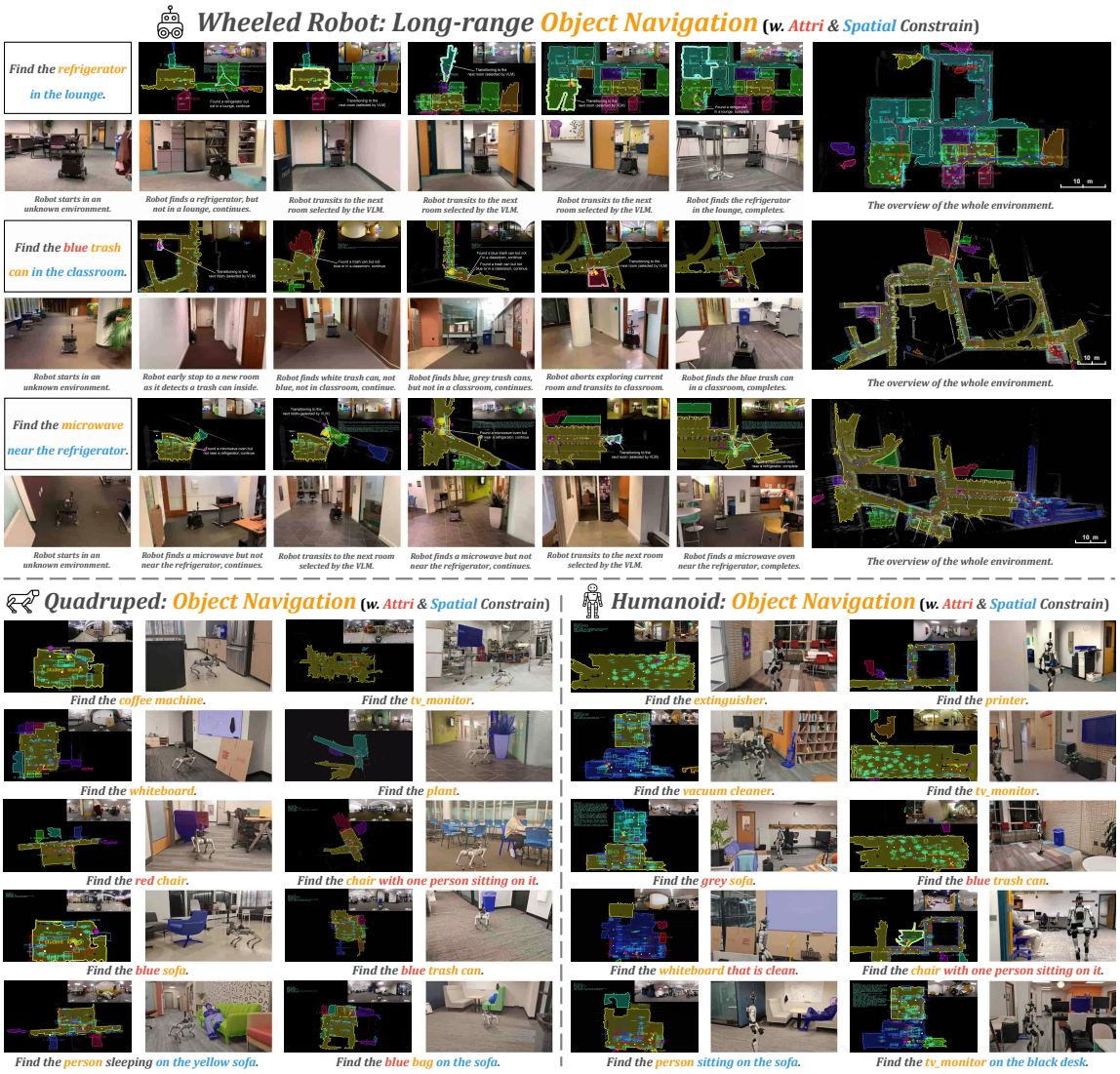

Figure 4.2: **Qualitative results from real-world deployment of SysNav.** The first three rows show floor-scale object navigation with multiple constraints on a wheeled robot, including step-by-step VLM reasoning analysis. The last five rows demonstrate cross-embodiment performance on quadruped and humanoid robots. For each episode, the final scene representation, first-person view and global overview at task completion are visualized. **Red** denotes self-attribute constraints, **blue** denotes spatial relationship constraints, and **orange** denotes target object categories.

Chapter 5

IntentNav: Learning Spatial-Visual Object Navigation from Human Demonstrations

5.1 Introduction

Object navigation in unseen environments requires an agent to search for a target object category while exploring under partial observability. The agent must decide where to explore next, balancing the need to cover new ground against the evidence accumulated so far. The central challenge is to perform long-horizon, target-conditioned search before the target is observed, inferring promising search directions from partial observations and continually adapting as new evidence appears.

Humans are naturally adept at this form of search. Imagine entering an unfamiliar apartment and being asked to find a mug. Rather than exhaustively traversing every reachable area, people use spatial and visual cues to guide the search: moving toward regions resembling kitchens or dining areas, pausing at informative viewpoints to scan the surroundings, and briefly probing uncertain rooms or hallways before deciding whether to continue. Their strong spatial awareness enables them to keep track of where they have been, which regions remain unexplored, and how the environment is laid out. Semantic common sense further helps them infer where the target object is

likely to appear, allowing promising areas to be inspected carefully while unlikely ones are passed over quickly. Effective ObjectNav should resemble this kind of human-like exploration: selectively probing visually promising frontiers while relying on spatial memory to avoid redundant revisits.

STRIVE and SysNav demonstrated that structured representation and room-level VLM reasoning can substantially improve navigation efficiency. However, both approaches rely on zero-shot VLM prompting—the VLM is given a hand-crafted textual or structured prompt and asked to reason about the next room or waypoint. While effective, this paradigm has inherent limitations. VLM reasoning alone does not guarantee stable long-horizon, target-conditioned search behavior, because VLMs still have limited spatiotemporal understanding [11, 104]. Existing VLM-based policies often rely on first-person RGB observations [36, 62, 68, 76, 95, 97] or text-based map summaries that abstract away geometric details, making it difficult to maintain structured spatial-visual memory over long horizons. As a result, these policies may produce locally plausible decisions at individual steps yet fail to maintain coherent strategy over time, leading to repetitive behaviors such as turning in place, backtracking, or repeatedly inspecting the same region.

IntentNav addresses these limitations by shifting from zero-shot prompting to learning-based imitation. Instead of asking a pretrained VLM to reason from scratch at each step, IntentNav fine-tunes a VLM to learn human-like navigation policies from human demonstrations. The key challenge is that human demonstrations contain rich search intent but provide only low-level action sequences rather than waypoint-level decisions. IntentNav bridges this gap through Frontier-based Human-Intent Labeling, which looks ahead into the demonstration and infers the high-level search direction that best explains the demonstrator's future behavior.

Within the thesis narrative, IntentNav marks the transition from designed reasoning to learned reasoning. STRIVE showed that VLM-guided navigation requires structured context and room-level decision granularity. SysNav showed how this principle can be embedded in a deployable multi-level system. IntentNav asks the next question: can the navigation decision itself be learned from data rather than prompted? The answer is that learning from human demonstrations at the candidate waypoint level, in a spatially grounded decision space, produces more stable and more human-like search behavior than zero-shot prompting.

5.2 Model

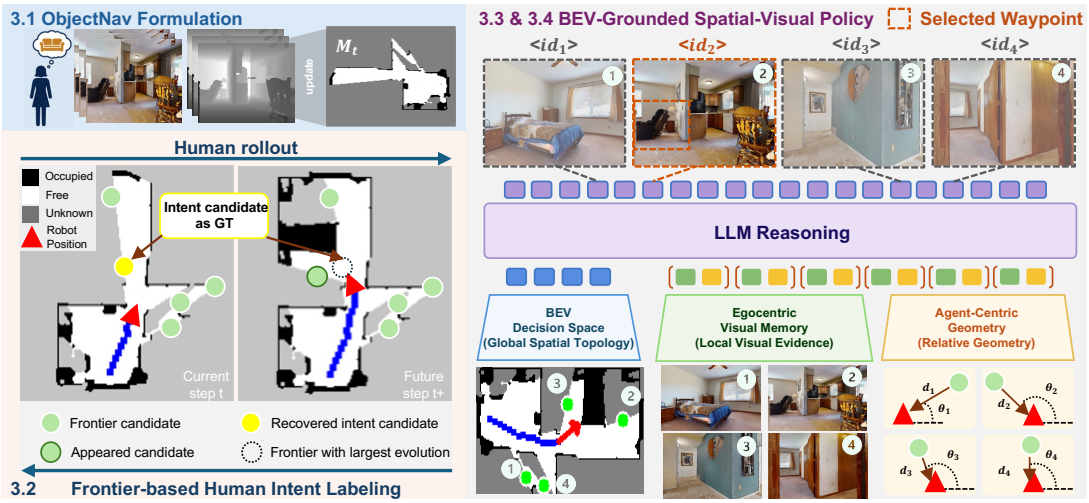

Figure 5.1: **Architecture overview of IntentNav.** The agent maintains a persistent BEV map, constructs a unified candidate set of frontiers and target hypotheses, and uses a VLM with candidate visual memory and agent-centric geometry to predict a candidate waypoint index.

The core idea of IntentNav is to infer the underlying high-level search intent from low-level human actions and express it as a candidate-level waypoint decision in a unified BEV space (Figure 5.1). The method has four main components: a candidate-level ObjectNav formulation, Frontier-based Human-Intent Labeling, a BEV-grounded spatial-visual candidate policy, and an Intent-Aligned Objective.

5.2.1 Candidate-Level ObjectNav Formulation

At each decision timestep t, the agent receives RGB-D observations and updates a persistent BEV map M_t that records explored free space, obstacles, trajectory history, and unknown regions. The BEV map is constructed by projecting depth observations onto a top-down grid and classifying points as free space, obstacles, or unknown based on their height relative to the ground plane. From M_t, a unified BEV candidate set $\mathcal{C}_t = \{x_i\}_{i=1}^{n_t}$ is constructed containing frontier candidates (on navigable boundaries between explored and unknown regions) and target candidates (verified

target hypotheses). Each candidate $x_i \in \mathbb{R}^2$ denotes a location in the BEV space and is paired with a candidate-specific egocentric RGB image I_i^{ego} and an agent-centric relative-geometry feature $\mathbf{f}_i$. Detailed BEV construction procedures are provided in Appendix C.

Given the target category c, the VLM predicts a candidate index over the current candidate set:

$$\hat{k} = \arg\max_{k \in [n_t]} p_\theta(k \mid M_t, \{x_i, I_i^{\text{ego}}, \mathbf{f}_i\}_{i=1}^{n_t}, c), \qquad \hat{x}_t = x_{\hat{k}}. \tag{5.1}$$

The selected waypoint $\hat{x}_t$ is then handed to a platform-specific local planner and controller for execution, decoupling high-level decision making from embodiment-specific control and enabling transfer across different physical embodiments. This decoupling follows the same principle established in SysNav, where high-level navigation logic remains identical across embodiments and only the execution layer changes.

5.2.2 Frontier-Based Human-Intent Labeling

Human demonstrations contain rich object-search intent but provide only low-level action sequences. To bridge this gap, IntentNav replays Habitat-Web ObjectNav human demonstrations [54] on MP3D scenes within the Habitat simulator [49] to construct candidate-level supervision. If a target candidate is present in the current candidate set, its index is used directly as supervision. Otherwise, the method infers the human's exploration intent from future frontier evolution.

Let $\mathcal{F}_t = \{f_i^t\}$ denote the frontier set at step t, and define the point-to-set distance $d(x, \mathcal{F}) = \min_{f \in \mathcal{F}} \text{geo}(x, f)$, where $\text{geo}(\cdot, \cdot)$ denotes geodesic distance over navigable free space. Starting from step t, the method advances past short local adjustments by requiring the demonstrator to have executed at least six `move-forward` actions, reaching step t_{adv}. It then identifies the earliest subsequent step t^+ where at least one current frontier no longer aligns with the future frontier set:

$$t^+ = \min \{t' \geq t_{\text{adv}} \mid \exists f \in \mathcal{F}_t,\ d(f, \mathcal{F}_{t'}) > \sigma_{\text{evo}}\}, \tag{5.2}$$

where σ_{evo} is a frontier-evolution threshold. The intent label y_t is then assigned over

the current candidate set:

$$y_t = \arg\max_i \; d(x_i, \mathcal{F}_{t^+}). \tag{5.3}$$

A large distance $d(x_i, \mathcal{F}_{t^+})$ indicates that candidate x_i is most inconsistent with the future frontier set, suggesting that the demonstrator continued exploration through that region, causing the frontier to evolve significantly. This converts low-level human trajectories into candidate-level proxy waypoint labels for training the VLM policy.

5.2.3 BEV-Grounded Spatial-Visual Candidate Policy

When searching for objects, humans combine spatial memory with local visual evidence. IntentNav therefore represents each candidate with two complementary forms of grounding.

Candidate visual memory. To ground each BEV candidate x_i with localized semantic evidence, it is paired with the egocentric RGB image I_i^{ego} captured when x_i was first instantiated. This memory augments the BEV waypoint with local visual context, such as room type, nearby objects, and scene layout, allowing the policy to assess the candidate's relevance to the target category. Candidate IDs are defined within the current step candidate set rather than tracked persistently across timesteps, since frontier proposals can move, merge, or split as the map evolves.

Agent-centric geometry. Although the BEV map contains each candidate's spatial relation to the agent, this geometry may not be reliably extracted by the VLM from the map feature alone. IntentNav therefore explicitly augments each candidate with an agent-centric relative-geometry feature. For a candidate at map position $\mathbf{p}_i$, this feature is computed from the agent pose $(\mathbf{p}_a, \psi_a)$ via

$$\mathbf{f}_i = \left[\frac{\|\mathbf{p}_i - \mathbf{p}_a\|}{W},\ \sin\theta_i^{\text{rel}},\ \cos\theta_i^{\text{rel}}\right], \quad \theta_i^{\text{rel}} = \text{wrap}(\text{atan2}(-\Delta y, \Delta x) - \psi_a), \tag{5.4}$$

where $(\Delta x, \Delta y) = \mathbf{p}_i - \mathbf{p}_a$ and W is the local BEV crop width. This encodes the candidate's normalized distance and relative bearing from the agent.

Candidate-aligned VLM input. The VLM receives a shared BEV crop followed by serialized candidate blocks, each containing the candidate's visual memory and agent-centric geometry feature. This organization preserves a one-to-one correspondence between each candidate, its local semantic evidence, and its explicit spatial grounding. The BEV crop and candidate memory images are processed by separate vision encoders to handle the distinct visual statistics of top-down maps and egocentric views.

Reserved candidate-ID selection. IntentNav reserves 32 candidate-ID tokens in the model vocabulary, covering the maximum number of candidates. The language model predicts a distribution over these reserved tokens, converting waypoint prediction into controlled candidate classification and avoiding free-form generation of indices. This design choice stands in contrast to STRIVE and SysNav, where the VLM outputs structured text (JSON or chain-of-thought reasoning); here the output is a discrete candidate index in a fixed vocabulary.

Unlike STRIVE and SysNav, which organize the decision space around rooms and viewpoints constructed from geometric map features, IntentNav's decision space is the BEV candidate set itself, where each candidate is grounded by both its first-observed visual context and its agent-centric geometry. This enables the VLM to jointly reason over semantic evidence, spatial relations, and search history within a single unified representation.

5.2.4 Intent-Aligned Objective

Human search behavior often reflects directional intent rather than a commitment to one exact waypoint. Nearby candidates along the demonstrated direction should be treated as plausible alternatives, but standard cross-entropy penalizes all incorrect candidates equally. IntentNav therefore constructs an angular soft target over the candidate set using a von Mises distribution centered on the labeled direction. Let θ_i and θ_{y_t} be the bearings from the agent to candidate x_i and the labeled candidate x_{y_t}, and define $\Delta\theta_i = \mathrm{wrap}(\theta_i - \theta_{y_t})$. The soft target is

$$q_i = \frac{\exp\left(\kappa(\cos\Delta\theta_i - 1)\right)}{\sum_j \exp\left(\kappa(\cos\Delta\theta_j - 1)\right)}, \qquad \kappa = \sigma_{\mathrm{rad}}^{-2}, \tag{5.5}$$

with $\sigma_{\text{rad}} = 25^\circ \approx 0.436\,\text{rad}$. This target assigns partial credit to candidates near the demonstrated direction while preserving strong penalties for large directional errors.

If the label is a frontier candidate, the final loss combines hard candidate supervision with the angular soft target:

$$\mathcal{L} = (1-\lambda)\big[-\log p_\theta(y_t)\big] + \lambda \left[-\sum_{i\in[n_t]} q_i \log p_\theta(i)\right]. \tag{5.6}$$

For target candidate labels, only hard cross-entropy is used to encourage decisive commitment to the target, since smoothing probability mass toward neighboring frontiers would weaken commitment to visible goals. This design reflects the observation that exploration decisions have directional structure (nearby frontiers are similar), while target commitment should be unambiguous.

IntentNav is fine-tuned from InternVL3-2B [16] using LoRA for 3 epochs on 4 A100 GPUs. The training corpus is derived from Habitat-Web ObjectNav human demonstrations [54] replayed on 28 MP3D scenes with 21 target categories, yielding 2,363,108 candidate-level training samples after filtering. Detailed LoRA configuration, learning rate schedule, and training hyperparameters are provided in Appendix C.

The training data construction converts low-level human actions into candidate-level waypoint labels through Frontier-based Human-Intent Labeling, and the Intent-Aligned Objective further smooths supervision by distributing probability mass among directionally similar candidates. This two-level structuring of the training signal is what enables the VLM to learn human-like search behavior from raw action sequences. A detailed discussion of training signal structuring is provided in Appendix C.

5.3 Experiments

IntentNav is evaluated on three widely used Habitat ObjectNav validation benchmarks: MP3D [8], HM3D-v1 [78], and HM3D-v2 [48]. Training demonstrations are replayed only on MP3D-train scenes, while MP3D-Val is used for in-domain evaluation. Evaluation on HM3D measures zero-shot transfer to unseen scene domains. The evaluation reports standard metrics including Success Rate (SR) and Success weighted by Path Length (SPL), along with four trajectory-level search metrics computed

offline from the agent state log, BEV exploration history, and the scene navmesh.

- **Exploration Coverage Ratio** (ECR): fraction of reachable free-space area (from the navmesh) observed before episode termination.
- **Local Scanning Ratio** (LSR): fraction of length-W=6 windows where translation <0.3 m and accumulated heading change >60°.
- **Probing Motion Ratio** (PMR): fraction of length-W=15 windows where the agent moves >1.0 m from the window start but returns within 0.5 m by the window end.
- **Redundant Revisit Ratio** (RRR): fraction of steps that both (i) do not expand the explored area ($\Delta\text{explored} \leq 0$) and (ii) bring the agent within 0.3 m of a previously visited position.

Higher ECR, LSR, and PMR indicate more active exploration; lower RRR indicates fewer unproductive revisits. The opening 12-step panoramic sweep is excluded from LSR. These metrics are used only for diagnostic analysis and are not optimized during training.

The experiments are designed to answer three questions. First, does learning from human demonstrations at the candidate waypoint level produce better navigation behavior than zero-shot prompting or low-level action prediction? Second, do the individual components—BEV grounding, candidate visual memory, agent-centric geometry, reserved candidate-ID selection, and the Intent-Aligned Objective—each contribute to the final performance? Third, does the candidate-level BEV waypoint interface transfer across different robot embodiments without additional VLM fine-tuning?

5.3.1 Results

IntentNav achieves state-of-the-art performance among learning-based methods across all three benchmarks, as shown in Table 5.1.

Table 5.1: **Main ObjectNav benchmark results.** Methods are grouped by training regime: training-free VLM systems, learning-based policies, generic-navigation models, and IntentNav. Best in **bold**, second-best underlined.

Method	Learning	MP3D		HM3D-v1		HM3D-v2	
		SR (%)	SPL (%)	SR (%)	SPL (%)	SR (%)	SPL (%)
VLFM [86]	✗	36.4	17.5	52.5	30.4	63.5	32.5
SG-Nav [84]	✗	40.2	16.0	54.0	24.9	49.6	25.5
OpenFMNav [34]	✗	37.2	15.7	54.9	24.4	-	-
TriHelper [100]	✗	-	-	56.5	25.3	-	-
BeliefMapNav [110]	✗	37.3	17.6	61.4	30.4	-	-
ApexNav [102]	✗	39.2	17.8	59.6	33.0	76.2	38.0
WMNav [46]	✗	45.4	17.2	58.1	31.2	72.2	33.3
PIGEON [47]	✗	41.6	14.4	57.9	32.3	79.2	36.8
STRIVE [111]	✗	52.3	23.1	62.9	34.2	79.6	**38.7**
SysNav [112]	✗	50.7	18.1	63.7	30.5	80.8	37.2
DD-PPO [73]	✓	21.6	8.7	-	-	27.9	14.2
Habitat-Web [54]	✓	14.6	4.9	-	-	-	-
ZSON [44]	✓	15.3	4.8	25.5	12.6	-	-
Navid [97]	✓	-	-	32.5	21.6	-	-
PixNav [5]	✓	-	-	37.9	20.5	-	-
ESC [109]	✓	28.7	14.2	39.2	22.3	-	-
PONI [53]	✓	31.8	12.1	-	-	-	-
SemExp [10]	✓	36.0	14.4	-	-	-	-
CompassNav [36]	✓	42.0	17.5	56.6	27.6	-	-
Hydra-Nav-SFT [68]	✓	49.0	16.5	-	-	72.9	27.7
UniGoal [85]	✓	41.0	16.4	-	-	54.5	25.1
Uni-Navid [95]	✓	-	-	-	-	73.7	37.1
OmniNav [76]	✓	-	-	-	-	56.1	30.0
IntentNav	✓	**53.8**	**23.1**	**70.5**	**34.6**	**82.2**	38.5

On MP3D it reaches 53.8% SR and 23.1% SPL. On HM3D-v1 it reaches 70.5% SR and 34.6% SPL. On HM3D-v2 it reaches 82.2% SR and 38.5% SPL. Although trained only on MP3D demonstrations, IntentNav transfers strongly to both HM3D-v1 and HM3D-v2, demonstrating robust cross-domain generalization. Compared with existing learning-based policies, the substantial SPL gains suggest that learning from human demonstrations helps the agent adopt more efficient search behaviors. It is also worth noting that IntentNav outperforms the SysNav baseline on both MP3D (53.8 vs. 50.7 SR, 23.1 vs. 18.1 SPL) and HM3D-v2 (82.2 vs. 80.8 SR, 38.5 vs. 37.2

SPL), demonstrating that learned decision-making provides additional performance beyond what structured representation and system design alone can achieve.

The ablation studies clarify where these improvements come from. Table 5.2 examines how the decision interface affects performance on HM3D-v2. The baseline predicts low-level action tokens from a uniformly sampled 5-frame RGB history. BEV-based variants instead select from the candidate set, either randomly, as free-form text coordinates, or as reserved candidate-ID tokens. Random BEV candidate selection already improves over the egocentric baseline by a large margin, demonstrating the *interface premium* of frontier and target candidate selection. Text-coordinate prediction further improves performance, but the reserved candidate-ID interface performs best, indicating that ranking discrete candidate options is more reliable than generating free-form waypoint coordinates.

Table 5.2: **Decision-interface ablation on HM3D-v2.** The results isolate the effect of BEV spatial-visual grounding and reserved candidate-ID selection.

Spatial interface	**Output format**	**SR (%)**	**SPL (%)**
Egocentric RGB history	Action token	35.4	14.2
BEV candidate space	Random candidate	58.7	22.3
BEV candidate space	Text coordinate	68.4	26.9
BEV candidate space	**Reserved candidate-ID token**	**82.2**	**38.5**

Table 5.3 reports component and objective ablations on HM3D-v2, including trajectory-level search behavior metrics. Among deployable component variants, removing candidate visual memory causes the largest performance drop, suggesting that egocentric evidence around each candidate is essential for assessing target relevance. Replacing visual memory with oracle frontier semantics yields the highest SR and SPL, indicating that more accurate semantic grounding could further improve results. Removing agent-centric geometry or replacing reserved candidate-ID tokens with text indices also degrades performance, showing that both visual evidence and a reliable decision interface are necessary for effective search. For the training objective, removing the angular soft target reduces both SR and SPL. The full IntentNav model, which combines the Target-Safe Angular objective with all components, achieves the best performance across all metrics.

Table 5.3: **Component and objective ablation on HM3D-v2.** SR/SPL and trajectory-level search behavior metrics for architecture and objective variants. *Visual*: per-candidate visual signal (none, privileged ground-truth semantics, or RGB birth-view memory).

	Architecture				*Performance*		*Search Behavior*			
Method	BEV	Visual	Geom.	Token	SR (%)	SPL (%)	ECR	LSR	PMR	RRR
Interface baseline										
Ego-History Action Policy	✗	–	–	–	35.4	14.2	0.51	0.08	0.01	0.44
Component ablation										
w/o Frontier Info	✓	None	✓	✓	70.2	29.4	0.62	0.14	0.02	0.29
Oracle Frontier Semantics	✓	GT Sem.	✓	✓	84.6	40.2	0.70	0.22	0.04	0.16
w/o Agent-Centric Geometry	✓	RGB	✗	✓	76.4	33.7	0.66	0.17	0.03	0.24
w/o Candidate-ID Tokens	✓	RGB	✓	✗	78.6	35.1	0.68	0.20	0.04	0.22
Objective ablation										
Hard CE Only	✓	RGB	✓	✓	79.6	36.2	0.70	0.19	0.03	0.20
Full IntentNav	✓	RGB	✓	✓	**82.2**	**38.5**	**0.71**	**0.21**	**0.05**	**0.18**
Human reference										
Human Demo (300)	–	–	–	–	*92.7*	*42.5*	*0.60*	*0.13*	*0.07*	*0.10*

5.3.2 Search Behavior Analysis

The four trajectory-level search metrics defined above (Table 5.3) provide a more detailed picture of how different methods explore.

As shown in Table 5.3, the egocentric action baseline achieves low ECR and high RRR, reflecting a tendency to revisit the same areas repeatedly without expanding coverage. This is a well-known failure mode of action-level policies, where the agent gets trapped in local loops of turning and moving forward without making progress toward the goal. Random BEV candidate selection already improves ECR substantially and reduces RRR, demonstrating that the candidate-level interface itself—not just the learned policy—contributes to better exploration. The full IntentNav model further improves ECR and reduces RRR compared to random selection, showing that the learned policy makes more directed choices than chance.

An important finding from Table 5.3 is that removing candidate visual memory reduces ECR and increases RRR, demonstrating that visual evidence around each candidate helps the policy distinguish promising directions from unpromising ones. Without visual memory, the policy cannot assess whether a frontier leads toward a kitchen or a bathroom, so it makes less directed exploration choices. Replacing visual

memory with oracle semantic labels further improves performance, yielding the lowest RRR among model variants. This suggests that more accurate semantic grounding—for example, through better object detection or scene understanding—could further improve navigation behavior.

Human demonstrations achieve the lowest RRR and the highest PMR, reflecting decisive search with purposeful probing and few redundant revisits. However, their ECR is lower than several learned policies because the demonstrations are success-only and typically terminate once the target is found. This is an important observation: human search is not characterized by exhaustive coverage but by directed, efficient probing. IntentNav narrows the gap between learned policies and human behavior on these metrics, suggesting that the Frontier-based Human-Intent Labeling successfully captures key aspects of human search intent.

5.3.3 Case Study

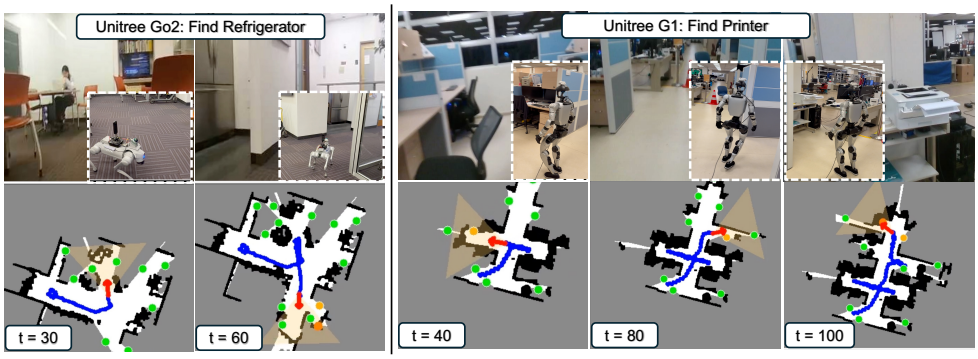

Figure 5.2: **Case study comparing navigation trajectories between IntentNav and zero-shot methods.** IntentNav selects frontier candidates based on learned visual memory and navigates more directly to targets, while zero-shot methods rely on room-level semantic priors.

Comparison of navigation trajectories (Figure 5.2) reveals complementary strengths between IntentNav and zero-shot methods. IntentNav selects frontier candidates based on learned visual memory and navigates more directly to targets in typical scenarios, while zero-shot VLM methods can reason about unusual placements using commonsense knowledge. This trade-off motivates the hybrid future system discussed in Chapter 7. Detailed case study analysis is provided in Appendix C.

5.3.4 Real-World Deployment

IntentNav is deployed on three physical embodiments: a wheeled robot [92], a Unitree Go2 quadruped, and a Unitree G1 humanoid, all running the same MP3D-trained checkpoint without additional VLM fine-tuning. All deployments use the same candidate-level BEV waypoint interface, with platform-specific sensing calibration and local control. The VLM policy runs on an RTX 4090 Laptop GPU (16 GB VRAM), and a ROS 2 bridge converts registered LiDAR scans, SLAM odometry, and egocentric camera images into the same BEV and candidate representation used during training, then publishes the selected waypoint to the robot's local planner. Unlike training-free pipelines that often rely on multi-stage prompting or repeated VLM queries, IntentNav performs low-frequency candidate-ID prediction over a compact waypoint set. The robots are tested on 8 target object categories across 10 different real-world scenes.

Real-time decision pipeline. The system runs as asynchronous ROS 2 nodes: LiDAR/SLAM updates the BEV map, the camera node projects RGB images, the detector runs in parallel, and the VLM node periodically queries the policy. The VLM is triggered when the robot comes within 0.5 m of the current waypoint, or when a 5 s periodic timer expires, whichever comes first. The detector runs asynchronously; the local planner continues executing the current waypoint during VLM inference, so decision latency overlaps with the final approach motion.

Table 5.4 reports the per-module latency breakdown averaged over 20 real-world episodes on the wheeled robot. The total decision cycle (BEV update + frontier extraction + rendering + VLM inference) takes 1404.4 ± 182.6 ms. VLM inference dominates the decision cycle at 1279.1 ± 148.3 ms, and scales approximately linearly with the number of candidates:

$$\text{Latency}_{\text{VLM}} \approx 59.7\,\text{ms} \times |\mathcal{C}_t| + 712\,\text{ms}. \tag{5.7}$$

This yields below 1 s for 3–4 candidates, $\sim$1.2 s for 8, and $\sim$2.6 s for 32. The target detector and camera pipeline run asynchronously and do not block waypoint execution. At the average robot speed of 0.27 m/s, the remaining approach time from the 0.5 m trigger distance is $\sim$1.85 s, which exceeds the decision cycle, so the VLM

Table 5.4: **Per-module latency breakdown on the RTX 4090 Laptop GPU, averaged over 20 real-world episodes.** Values are mean ± std. The VLM latency excludes the first cold-start inference; the detector and camera pipelines run asynchronously and do not block waypoint execution.

Module / statistic	Value
LiDAR BEV map update	53.7 ± 10.6 ms
Frontier extraction + filtering	68.6 ± 20.0 ms
BEV RGB rendering	3.0 ± 0.7 ms
VLM inference	1279.1 ± 148.3 ms
Total decision cycle	1404.4 ± 182.6 ms
Target detector (YOLOE, parallel)	11.5 ± 4.6 ms
Camera decode + projection (parallel)	2.5 ± 1.3 ms
Camera pipeline total (parallel)	3.7 ± 1.8 ms
Peak GPU memory	5.58 GB
LiDAR scan frequency	3.35 ± 0.27 Hz
Robot average speed	0.27 ± 0.19 m/s
Waypoint reached threshold	0.5 m

typically finishes before the robot arrives at the current waypoint. The quadruped and humanoid platforms have lower locomotion speed, making the overlap even more favorable. Detailed physical deployment procedures, including LiDAR-based BEV construction, frontier stabilization for real scans, and target localization without metric depth, are provided in Appendix C.

IntentNav has two acknowledged limitations. First, it assumes a mostly static environment when maintaining the BEV map and candidate-specific visual memories; dynamic changes may make stored map regions and visual memories stale, affecting frontier extraction and candidate ranking. Second, the current system decouples high-level waypoint selection from platform-specific local planning and control. While this abstraction enables cross-embodiment transfer, execution failures, detours, or local planner constraints are not explicitly fed back into the VLM policy. Future work should incorporate temporal memory updates for dynamic scenes and tighter feedback between high-level candidate selection and low-level execution.

5.4 Conclusion

Within the overall thesis, IntentNav addresses the decision-learning layer. STRIVE showed that semantic reasoning needs structured context. SysNav showed that structured reasoning needs a proper system architecture. IntentNav adds that the navigation decision itself should be learned from human demonstrations rather than prompted from scratch, showing that learned policies produce more stable and human-like behavior than zero-shot prompting.

The central lesson of this chapter is that zero-shot VLM prompting, while effective for room-level reasoning, has inherent limitations for fine-grained navigation decisions. By learning from human demonstrations at the candidate waypoint level, in a spatially grounded BEV decision space with intent-aligned supervision, IntentNav produces more coherent long-horizon search behavior. The Frontier-based Human-Intent Labeling bridges the gap between low-level human actions and high-level navigation intent, and the Intent-Aligned Objective respects the directional structure of exploration decisions.

This chapter also reveals the next question in the thesis progression. The decision-interface ablation in Table 5.2 shows that even random frontier selection already achieves 58.7% SR on HM3D-v2—a result that exceeds most training-free VLM systems in Table 5.1. This suggests that object navigation, where the goal is a single target category such as "chair" or "mug," places limited demands on the VLM's semantic reasoning. The task can be largely solved by structured exploration and simple semantic priors without requiring deep language understanding or compositional reasoning. A natural question then arises: can the VLM's reasoning be better tested by a more demanding task paradigm? The next chapter addresses this question through Goal2Pixel, which shifts from object navigation to vision-and-language navigation in continuous environments (VLN-CE). Unlike ObjectNav's single-word goal specification, VLN-CE requires the agent to follow multi-clause natural-language instructions, track progress across multiple landmarks, and handle compositional spatial constraints—a task paradigm that truly activates the VLM's semantic reasoning capabilities.

Chapter 6

Goal2Pixel: Grounding Goals to Pixels for Vision-Language Navigation

6.1 Introduction

Vision-and-language navigation in continuous environments requires an embodied agent to follow natural-language instructions and reach a target through fine-grained physical movements. Unlike static vision-language understanding, VLN-CE is a language-conditioned spatial decision-making problem: the agent must ground language in egocentric observations, track its progress, and continuously decide where to move next. Vision-language models offer a natural foundation for this problem, and recent VLN-CE methods increasingly build on pretrained VLMs to improve instruction-following and generalization.

Yet the interface between high-level VLM reasoning and executable motion has barely evolved. Most VLM-based methods still query the VLM at every timestep for low-level meta-actions such as *turn left/right 15°, move forward 25 cm, or stop.* This action-centric interface has three fundamental limitations. First, *ambiguous action supervision*: many distinct action sequences can reach the same goal, making the oracle trajectory an unnecessarily restrictive training target. Second, *myopic*

decisions: optimizing for the next short-range movement discourages longer-horizon spatial reasoning. Third, *high inference cost*: because each prediction advances the agent only a few centimeters or degrees, the VLM must be invoked dozens of times per episode. The bottleneck is therefore not only the capacity of the VLM but the interface through which its reasoning becomes motion.

Goal2Pixel asks: **what should serve as the interface between VLM reasoning and executable motion in VLN-CE?** The answer is the image plane itself—the VLM's native input—where a single pixel can ground a forward navigation target in the visual observation and be back-projected into a 3D waypoint for execution. Goal2Pixel reformulates VLN-CE as *navigable pixel grounding*: rather than predicting actions, the VLM predicts a visible navigable pixel, which is back-projected into a 3D waypoint via camera geometry and tracked by a lightweight local planner. For non-forward decisions such as turning and stopping, auxiliary directive regions are appended to the image plane, unifying all action spaces in a single pixel-prediction interface.

Within the thesis narrative, Goal2Pixel addresses the output interface bottleneck. STRIVE showed that structured representation and room-level reasoning improve VLM-guided navigation. SysNav showed how these principles can be embedded in a deployable multi-level system. IntentNav showed that navigation decisions should be learned from data rather than prompted. Goal2Pixel takes the final step by rethinking *what the VLM should output*: not a room label, not a candidate index, not a discrete action, but a pixel coordinate in the image plane. This progression reflects a deepening insight about the relationship between VLM reasoning and robot motion—that the most natural output interface is one that is native to the VLM itself.

6.2 Vision-Language Navigation Task Definition

Vision-and-language navigation in continuous environments requires an embodied agent to follow a natural language instruction $\mathcal{I}$ using an egocentric RGB observation history $\mathcal{O}_t = \{o_0, \ldots, o_t\}$ at each time step t. In the standard VLN-CE setting, the agent acts in a low-level action space $\mathcal{A} = \{$`Move_Forward` (25 cm), `Turn_Left` (15°), `Turn_Right` (15°), `Stop`$\}$. After executing an action, the agent receives a new

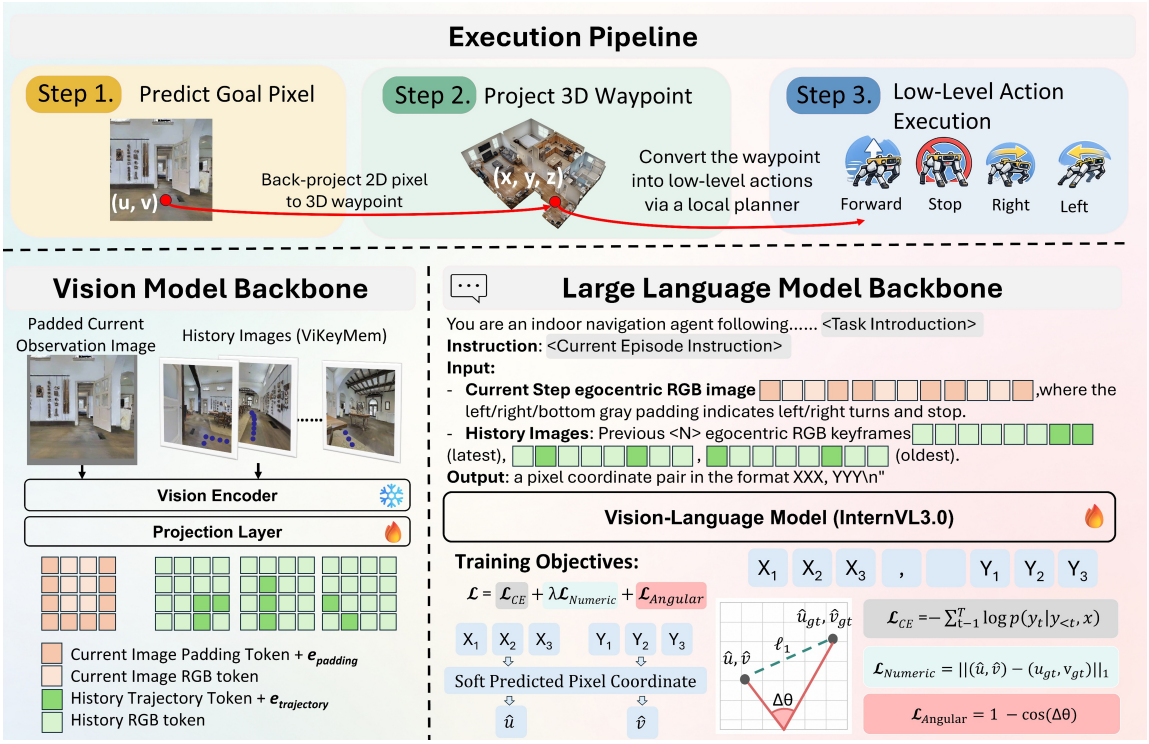

Figure 6.1: **Pipeline overview of Goal2Pixel.** The VLM autoregressively outputs a pixel coordinate as text ("XXX, YYY"), which is back-projected into a 3D waypoint via camera geometry. Auxiliary directive regions appended to the image unify turning and stopping decisions in the same pixel-prediction space.

observation, and the process repeats until the agent outputs `Stop`. An episode succeeds if the agent stops within the instruction-specified target region.

6.3 Data Collection

Oracle trajectories are derived from the training splits of R2R-CE [32] and RxR-CE [33] (English subset) in MP3D [9], constructing 2.64M training samples in total. Each sample is represented as a tuple $(\mathcal{I}, O_{\text{cur}}, O_{\text{hist}}, p)$, where $\mathcal{I}$ is the instruction, O_{cur} is the padded current egocentric RGB image, O_{hist} is the sequence of historical images constructed by ViKeyMem, and p is the ground-truth pixel.

6.3.1 Ground-Truth Pixel Construction

The target pixel p is defined by projecting the farthest visible future waypoint w onto the current image. Visibility is verified by a project–reproject consistency check: the waypoint w is projected to a pixel p, and p is then back-projected using its depth to obtain $\hat{w}$; w is considered visible if $\|w - \hat{w}\|_2 < 0.6$ m. This encourages the model to predict a farther navigable goal, directly addressing the myopia of action-level prediction.

To represent non-forward decisions within the same pixel-prediction space, auxiliary directive regions are appended to the left, right, and bottom sides of the image. These regions are used in two geometric fallback cases: (1) *Task Completion*: if the agent is within 1.0 m of the final destination, p is assigned to the bottom region to indicate `Stop`; (2) *Out-of-View*: if no future waypoint is visible, p is assigned to the left or right regions according to the mean egocentric direction of the next five waypoints, indicating `Turn_Left` or `Turn_Right`.

6.3.2 ViKeyMem: Visibility-Aware Keyframe Memory

For long-horizon navigation, the agent requires memory of instruction-relevant landmarks and visited regions. However, existing VLM-based navigators typically feed 8–32 fixed-interval or uniformly sampled frames [17, 65, 69, 77, 90, 91, 94, 96, 107, 108], most of which carry little new information while increasing computational cost. Goal2Pixel introduces ViKeyMem, a visibility-aware keyframe memory that adds a frame only when the set of visible waypoints changes substantially.

A candidate frame is selected as a keyframe if it satisfies three conditions: (1) it is not visible from the most recently selected keyframe; (2) at least one subsequent waypoint is visible in the candidate's egocentric image; and (3) at least two distinct subsequent waypoints fall within the candidate's 45° forward field of view. The first condition is the core visibility criterion, ensuring that a new keyframe is added only when the candidate provides information not covered by the previous keyframe. The latter two conditions ensure that the selected keyframe faces the upcoming trajectory.

Because the selected keyframes are sparse (yielding 4–5 keyframes per 100 timesteps while covering every meaningful viewpoint transition), the agent's past trajectory is overlaid onto each keyframe by drawing blue dots along the trajectory, making the

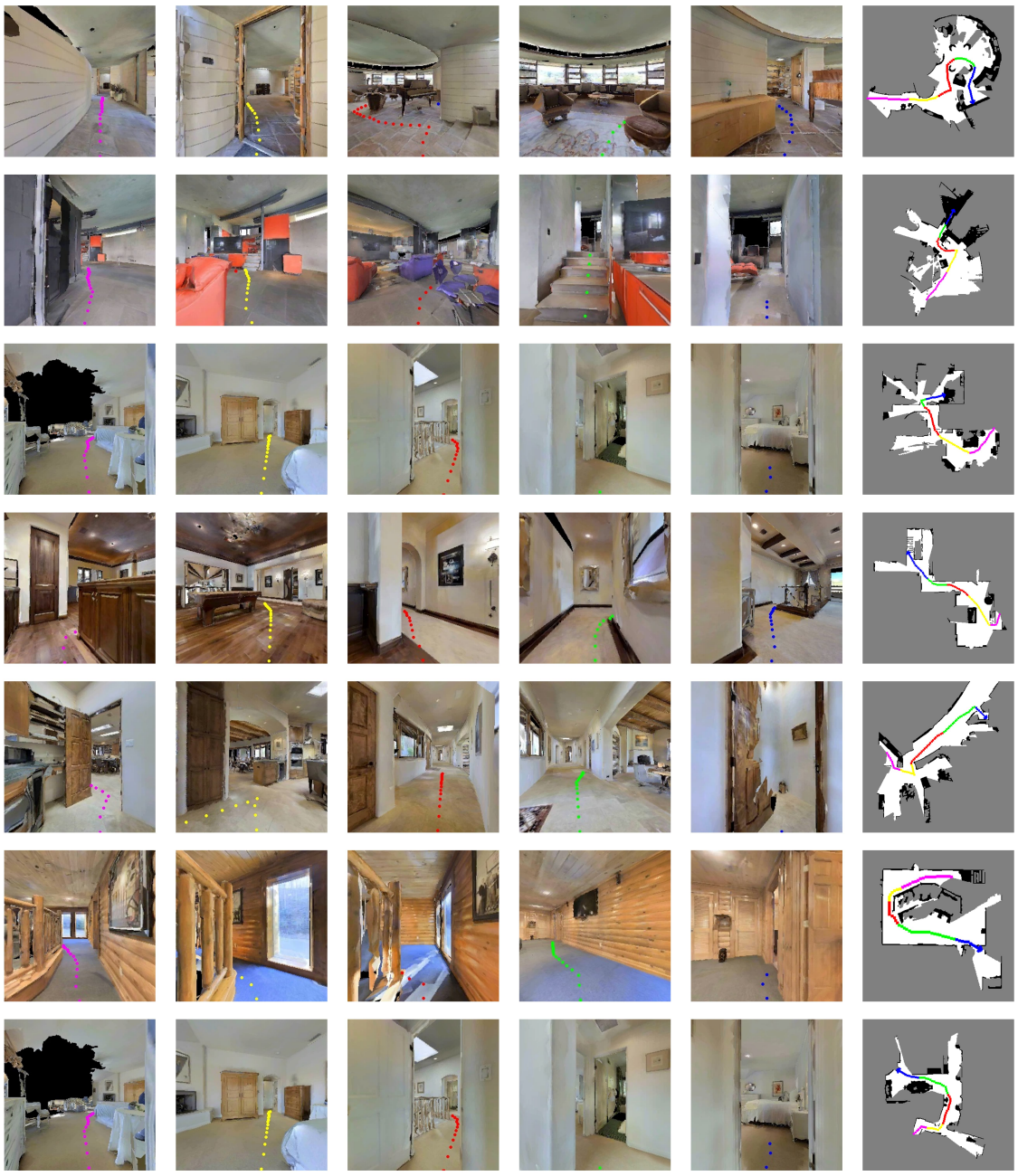

Figure 6.2: **ViKeyMem keyframe selection.** ViKeyMem selects keyframes only when the set of visible waypoints changes substantially. Blue dots overlaid on each keyframe show the agent's past trajectory, making motion connections between distant observations explicit.

motion connection between distant observations explicit (Figure 6.2). ViKeyMem requires no new training architecture or additional memory model, and can serve as a general history representation module for other navigation systems.

The sparsity of ViKeyMem is not merely a computational advantage; it also improves the quality of the VLM's reasoning by ensuring that each history image provides unique and decision-relevant information. The trajectory overlay further enhances this by providing geometric context. A detailed analysis of why ViKeyMem's sparsity improves reasoning quality is provided in Appendix D.

6.4 Goal2Pixel

The overall pipeline of Goal2Pixel, illustrated in Figure 6.1, consists of a three-stage execution pipeline and a VLM-based prediction module. Given the instruction, padded current observation, and navigation history, Goal2Pixel autoregressively outputs a pixel coordinate as text in the format "XXX, YYY," where the two three-digit numbers denote the horizontal and vertical coordinates in the augmented egocentric image plane. This predicted pixel serves as a spatially grounded intermediate representation for navigation.

In the first stage, the VLM predicts a goal pixel (u, v) on the image plane. In the second stage, if the predicted pixel lies in the regular RGB region, it is back-projected via camera geometry to a 3D waypoint (x, y, z) in the agent coordinate frame using the depth value at the predicted pixel location. If the predicted pixel falls into one of the predefined auxiliary directive regions, it is directly interpreted as `Turn_Left`, `Turn_Right`, or `Stop`. In the third stage, 3D waypoints are converted into executable low-level actions by a lightweight local planner that performs waypoint following and collision avoidance. This three-stage pipeline decouples high-level VLM reasoning from low-level motion execution, and since the VLM always outputs a single pixel coordinate regardless of the navigation decision, the output interface remains uniform across all action types.

6.4.1 Visual Semantic Embeddings

Goal2Pixel introduces navigation-specific visual patterns beyond ordinary RGB content, including auxiliary directive regions in the current observation and trajectory overlays in the history memory. Since these regions encode explicit navigation semantics but may not be reliably distinguished by a pretrained vision encoder, lightweight learnable visual semantic embeddings are introduced for the corresponding special visual tokens. Specifically, a learnable directive embedding is added to current-image tokens that overlap with auxiliary directive regions, and a learnable trajectory embedding is added to history-image tokens that contain trajectory overlays. All regular RGB tokens and non-trajectory history tokens remain unchanged.

6.4.2 Training Objective

Goal2Pixel predicts the navigation target as a coordinate string in the format "XXX, YYY" and is primarily trained with the standard token-level cross-entropy loss $\mathcal{L}_{\mathrm{CE}}$. However, token-level supervision treats each digit as an independent categorical label and therefore does not explicitly capture the geometric and metric structure of pixel coordinates. To make the supervision better aligned with pixel navigation, two coordinate-aware auxiliary losses are introduced.

Soft Predicted Pixel Coordinate. Since the model outputs discrete coordinate tokens, the predicted pixel is not directly available as a continuous value during training. A differentiable soft coordinate is derived from the generation logits. Let $z_k \in \mathbb{R}^{10}$ denote the logits at the k-th digit position over digit tokens $\{0, \ldots, 9\}$. The digit distribution and its expected value are computed as

$$\pi_k(d) = \frac{\exp(z_{k,d})}{\sum_{j=0}^{9} \exp(z_{k,j})}, \qquad \hat{d}_k = \sum_{d=0}^{9} d\, \pi_k(d). \tag{6.1}$$

For the coordinate format "$\hat{d}_1\hat{d}_2\hat{d}_3, \hat{d}_4\hat{d}_5\hat{d}_6$", the soft predicted coordinate is

$$\hat{u} = 100\hat{d}_1 + 10\hat{d}_2 + \hat{d}_3, \qquad \hat{v} = 100\hat{d}_4 + 10\hat{d}_5 + \hat{d}_6. \tag{6.2}$$

Both predicted and ground-truth coordinates are normalized to $[0, 1]$ by dividing by the maximum coordinate value 999.

Numeric and Angular Losses. Given the normalized predicted pixel $(\hat{u}_n, \hat{v}_n)$ and ground-truth pixel (u_n, v_n), the coordinate-aware losses are defined as

$$\mathcal{L}_{\text{num}} = \|(\hat{u}_n, \hat{v}_n) - (u_n, v_n)\|_1, \quad \mathcal{L}_{\text{ang}} = 1 - \cos\left(\theta(\hat{u}_n, \hat{v}_n) - \theta(u_n, v_n)\right), \tag{6.3}$$

where $\theta(u_n, v_n) = \text{atan2}(v_0 - v_n, u_n - u_0)$ is the egocentric direction from the bottom-center anchor $\mathbf{p}_0 = (0.5, 1.0)$ to the pixel. The numeric loss penalizes coordinate-level distance, while the angular loss encourages directionally consistent pixel grounding.

The final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{num}}\mathcal{L}_{\text{num}} + \lambda_{\text{ang}}\mathcal{L}_{\text{ang}}, \tag{6.4}$$

where $\lambda_{\text{num}} = 0.3$ and $\lambda_{\text{ang}} = 0.03$ control the contributions of the numeric and angular losses.

6.4.3 Why Pixel Prediction Works

A key question raised by Goal2Pixel is why pixel prediction should outperform direct action prediction. The answer lies in the fundamental ambiguity of action-level supervision in VLN-CE. Multiple distinct action sequences can reach the same spatial goal: an agent might turn left then move forward, or move forward then turn slightly right, and both sequences arrive at roughly the same location. When the model is forced to imitate one specific oracle action sequence, it receives unnecessarily restrictive supervision that ignores the spatial equivalence of alternative action sequences. Pixel prediction removes this ambiguity by specifying *where* to go rather than *how* to go, giving the local planner freedom to choose the best motion execution.

This reframing also changes the effective horizon of each VLM decision. With action prediction, each call advances the agent only 25 cm or 15°, requiring 46.62 VLM calls per episode on average. With pixel prediction, each call specifies a visible navigable target that may be several meters away, requiring only 7.75 calls per episode. The pixel-based interface therefore allows the VLM to operate at a more appropriate

temporal scale, making longer-range spatial reasoning possible without the overhead of step-by-step motion control.

The farthest-visible-waypoint target construction further encourages long-range decisions. By defining the ground-truth pixel as the farthest visible future waypoint along the oracle trajectory, verified by the project–reproject consistency check, the model is directly trained to look ahead and select distant navigable targets rather than making myopic short-range predictions. This is in contrast to action prediction, where the oracle action at each timestep only encodes the next immediate movement.

6.4.4 Pixel Distribution Analysis

An informative aspect of Goal2Pixel is the distribution of predicted pixels across the augmented image plane. Analysis of the training data reveals that the majority of ground-truth pixels fall in the regular RGB region (forward navigation), with a smaller proportion in the auxiliary directive regions (turning and stopping). Specifically, the forward navigation pixels tend to concentrate in the lower-center portion of the image, corresponding to navigable floor regions at medium range. This distribution reflects the fact that the farthest visible waypoint is typically several meters ahead on the floor, which projects to the lower-center region of the egocentric image. Figure 6.3 visualizes this distribution.

The auxiliary directive regions are activated in two geometric situations. The task completion region (bottom) is activated when the agent is within 1.0 m of the final destination, which occurs relatively infrequently—typically once per episode. The out-of-view regions (left and right) are activated when no future waypoint is visible in the current field of view, which occurs when the agent needs to turn before proceeding forward. The frequency of these activations depends on the complexity of the environment and the instruction: multi-turn instructions in cluttered environments trigger more turning decisions than straight-line navigation in open corridors.

This distribution analysis also reveals why the pure pixel paradigm outperforms hybrid action-pixel designs. In the parallel action-pixel design, the VLM must simultaneously learn to predict actions (for turning and stopping) and pixel coordinates (for forward navigation) within a single output space. This creates a multimodal prediction problem that the VLM struggles to solve, resulting in only 2.8% SR. In the

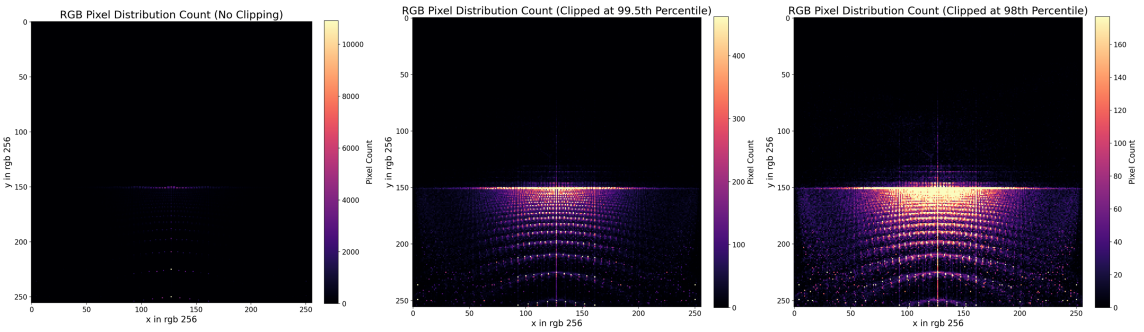

Figure 6.3: **Ground-truth pixel distribution for Goal2Pixel supervision.** Each regular RGB pixel is counted independently. Left: raw count distribution without clipping. Middle and right: distributions clipped at the 99.5th and 98th percentiles, respectively, to improve the visibility of low-frequency regions. The structured lower-image patterns are mainly induced by the discrete motion primitives in Habitat, while scattered points reflect more diverse navigation geometries such as stairs, height changes, doorways, and partial visibility.

sequential design, the VLM first decides whether to select a pixel or execute a discrete action, then predicts a pixel if needed. This two-stage process introduces additional error at the first stage and requires more VLM calls. The pure pixel paradigm avoids both problems by encoding all decisions in the same coordinate space, making the prediction task simpler and more consistent for the VLM.

6.5 Experiments

Setup. Goal2Pixel is initialized from InternVL3 [16] and fine-tuned for one epoch on the 2.64M-sample VLN-CE corpus. The LLM backbone and projection layers are fully fine-tuned while the vision encoder is frozen. The learning rate is 2×10^{-5}, with at most 8 history images. Justification for full fine-tuning and loss weight selection is provided in Appendix D. The 2B and 7B models require 80 and 176 H100 GPU hours, respectively. Evaluation reports standard VLN-CE metrics: Navigation Error (NE), Oracle Success Rate (OS), Success Rate (SR), Success-weighted Path Length (SPL), and normalized Dynamic Time Warping (nDTW).

The experiments are designed to answer three questions. First, does the pure pixel-based paradigm outperform action-based and hybrid action-pixel-based output designs? Second, does ViKeyMem provide a more effective and efficient history

representation than conventional alternatives? Third, do the lightweight adaptations—semantic embeddings and coordinate-aware losses—contribute to pixel-grounded navigation?

Training Computation. Data packing is employed to improve training efficiency, yielding an average of 20.9 and 7.71 packed samples per H100 GPU hour for the 2B and 7B models, respectively. Despite the 2.64M-sample training corpus, the 2B model requires only 80 H100 GPU hours and the 7B model requires 176 H100 GPU hours for one epoch of training. This computational efficiency is partly due to the compact ViKeyMem history representation, which yields only 4–5 keyframes per 100 timesteps, and the data packing strategy that maximizes GPU utilization.

Main Results on R2R-CE and RxR-CE. Goal2Pixel achieves competitive state-of-the-art performance on both benchmarks (Table 6.1), compared with prior methods that do not use external training data. On R2R-CE, the 2B model reaches 54.1% SR and 52.5% SPL with just 7.75 VLM calls per episode, outperforming all prior methods including those with larger 7B backbones. The 7B model achieves the best NE of 4.80 and 52.7% SPL, improving +3.5% over JanusVLN [91]. On RxR-CE, the 7B model achieves 44.7% SPL and 63.0 nDTW, improving +0.4% and +3.9% over the strongest prior results, respectively. These improvements in SPL and nDTW indicate that Goal2Pixel follows more efficient and better-aligned trajectories. Notably, Goal2Pixel demonstrates strong trajectory quality and path efficiency across both benchmarks, with the 2B model already achieving the best SR on R2R-CE despite using a smaller backbone than most competitors. The gains in path efficiency are attributed to the pixel-based interface, which allows the agent to execute more direct local motion toward visible navigable targets, reducing unnecessary oscillations and improving trajectory efficiency.

Real-World Deployment. Goal2Pixel is deployed on a Mecanum wheeled robot [92] with an NVIDIA RTX-5060 GPU. Across 16 real-world trials, the model successfully grounds language instructions to navigable pixel targets. Since the model outputs a 2D pixel coordinate rather than a robot-specific action, the high-level prediction is decoupled from the embodiment, suggesting potential for cross-embodied navigation.

Table 6.1: **Comparison with state-of-the-art methods on R2R-CE and RxR-CE Val-Unseen splits.** For VLM methods, model size is reported. External data includes EnvDrop, DAgger, and general VQA. Best in **bold**, second-best underlined.

Method	Size	R2R-CE				RxR-CE				Ext. Data
		NE↓	OS↑	SR↑	SPL↑	NE↓	SR↑	SPL↑	nDTW↑	
InstructNav [39]	-	6.89	-	31.0	24.0	-	-	-	-	-
AO-Planner [12]	-	5.55	59.0	47.0	33.0	7.06	43.3	30.5	50.1	-
LaViRA [22]	-	6.54	48.7	38.3	28.3	-	-	-	-	-
CA-Nav [14]	-	7.58	48.0	25.3	10.8	-	-	-	-	-
CMA [26]	-	6.20	52.0	41.0	36.0	8.76	26.5	22.1	47.0	-
VLN↻BERT [26]	-	5.74	53.0	44.0	39.0	8.98	27.0	22.6	46.7	-
Ego^2-Map [27]	-	5.54	56.0	47.0	41.0	-	-	-	-	-
g3D-LF [66]	-	5.70	59.5	47.2	34.6	-	-	-	-	-
Sim2Real [67]	-	5.95	55.8	44.9	30.4	8.79	36.7	25.5	18.1	-
NavMorph [81]	-	5.75	56.9	47.9	33.2	8.85	30.8	22.8	44.2	-
MapNav [101]	7B	4.93	53.0	39.7	37.2	7.62	32.6	27.7	43.5	0K
NaVid [96]	7B	5.47	49.1	37.4	35.9	8.41	23.8	21.2	-	953K
NaVid-4D [37]	7B	5.99	55.7	43.8	37.1	-	-	-	-	-
StreamVLN [70]	7B	6.05	53.8	45.5	41.6	6.72	48.6	42.5	60.2	0K/10033K
Uni-NaVid [94]	7B	5.58	53.3	47.0	42.7	**6.24**	48.7	40.9	-	3577K
NaVILA [17]	7B	5.37	57.6	49.7	45.5	6.77	49.3	44.0	58.8	13132K
Efficient-VLN [108]	7B	6.41	54.5	45.9	41.9	6.51	49.8	41.5	59.4	0K
JanusVLN [91]	7B	5.17	58.0	52.8	49.2	6.46	**51.4**	44.3	59.1	0K
Goal2Pixel	**2B**	4.85	**59.9**	**54.1**	52.5	7.50	43.8	40.4	61.1	0K
Goal2Pixel	7B	**4.80**	58.3	53.9	**52.7**	6.91	48.1	**44.7**	**63.0**	0K

Real-world failure analysis is provided in Appendix D.

Real-World Failure Analysis. Analysis of unsuccessful trials reveals two primary failure modes: depth inaccuracy at ranges beyond 3 meters, and auxiliary directive confusion in visually cluttered environments. Detailed failure analysis is provided in Appendix D.

6.5.1 Ablation Studies

Output Paradigm. The pure pixel paradigm is compared against four alternative output designs (Table 6.2): one-action (predicting a single action per step), four-action

(predicting four actions per step), action-pixel parallel (predicting either an action or a pixel in one call), and action-pixel sequential (two-stage decision: first predict whether to select a pixel, then predict the pixel if needed).

The pure pixel paradigm substantially outperforms all alternatives. Compared with direct one-action prediction, pixel prediction improves SR by +21.2% (from 32.9% to 54.1%) while reducing VLM calls from 46.62 to 7.75 per episode—approximately 6× fewer calls. The four-action baseline achieves 37.0% SR with 15.77 calls, still significantly worse than pixel prediction. The parallel action-pixel design, which predicts either an action or a pixel in a single call, achieves only 2.8% SR, suggesting that the VLM struggles to jointly handle both output types. The sequential action-pixel design achieves 43.7% SR but requires 16.85 VLM calls, still substantially more than the pure pixel paradigm.

Compared with the hybrid action-pixel sequential design, the pure pixel paradigm improves SR by +10.4 points on R2R-CE and +13.7 points on RxR-CE. In terms of inference time, the pixel paradigm achieves a favorable efficiency–performance trade-off with 0.120s average inference time per episode, compared to 0.067s for one-action (which achieves much lower SR) and 0.161s for the sequential design. These results demonstrate that directly predicting low-level actions provides weak supervision for VLN-CE, since multiple feasible action sequences can reach the same spatial goal but the model is forced to imitate one specific sequence. The unified pixel output format not only enforces consistent supervision across different navigation decisions but also reduces the task burden on the VLM, enabling it to learn more accurate navigation targets.

Table 6.2: **Output paradigm ablation on R2R-CE and RxR-CE.** # Calls = average VLM invocations per episode; Inf. T = average inference time (s) per episode.

Output Paradigm	R2R-CE				RxR-CE			
	SR↑	SPL↑	# Calls↓	Inf. T↓	SR↑	SPL↑	# Calls↓	Inf. T↓
One Action	32.9	31.5	46.62	**0.067**	32.9	31.5	46.62	**0.075**
Four Actions	37.0	36.0	15.77	0.149	28.3	25.2	24.05	0.157
Action-Pixel (Parallel)	2.8	2.1	**4.82**	0.175	8.5	7.8	**4.03**	0.179
Action-Pixel (Sequential)	43.7	41.9	16.85	0.161	30.1	28.3	17.76	0.158
Pixel (Ours)	**54.1**	**52.5**	7.75	0.120	**43.8**	**40.4**	11.76	0.134

History Image Representation. ViKeyMem is compared against no history, fixed-interval sampling, uniform sampling, and multi-frequency sampling (Table 6.3).

Table 6.3: **History representation ablation.** Inf. T = inference time (s) per episode; Train T = H100 GPU hours.

Configuration	R2R-CE		RxR-CE		Time	
	SR↑	SPL↑	SR↑	SPL↑	Inf. T↓	Train T↓
w/o History	34.7	33.2	30.6	28.3	**0.116**	**36**
Fixed (1-step)	37.2	35.5	34.1	30.2	0.210	183
Fixed (5-step)	46.1	45.1	38.7	35.4	0.243	173
Uniform	**56.1**	**55.1**	41.6	38.5	0.224	156
Different Frequency	25.0	19.0	24.9	21.1	0.242	146
ViKeyMem	54.1	52.5	**43.8**	**40.4**	0.121	70
w/o Trajectory Overlay	48.7	47.0	41.1	38.1	0.123	72

Compared with 5-step fixed-interval sampling, ViKeyMem improves SR/SPL by 8.0/7.4 points on R2R-CE while reducing inference time from 0.243s to 0.121s and training time from 173 to 70 H100 GPU hours.

Lightweight Adaptations. Removing the angular or numeric loss consistently reduces SR on both benchmarks, while removing the visual semantic embeddings also leads to slight degradation in most cases. Detailed ablation results are provided in Appendix D.

6.6 Conclusion

Within the overall thesis, Goal2Pixel addresses the output interface layer. STRIVE showed that semantic reasoning needs structured context. SysNav showed that structured reasoning needs a proper system architecture. IntentNav showed that navigation decisions should be learned rather than prompted. Goal2Pixel completes the picture by showing that the output interface—how VLM reasoning becomes executable motion—is itself a fundamental design choice.

The central lesson of this chapter is that the output interface between VLM reasoning and robot motion is not merely an implementation detail but a fundamental design choice. By reformulating VLN-CE as navigable pixel grounding, Goal2Pixel provides a spatial interface that is native to the VLM, avoids the ambiguity of action-level supervision, encourages longer-range decisions through farthest-visible-waypoint targets, and unifies all navigation decisions (forward, turn, stop) in a single pixel-prediction space. ViKeyMem demonstrates that compact, visibility-aware history representation can match or exceed denser alternatives at a fraction of the computational cost. The coordinate-aware losses and semantic embeddings show that lightweight adaptations can effectively bridge the gap between pretrained VLMs and navigation-specific output requirements.

The progression across the four research chapters can now be seen in full. STRIVE showed that VLM reasoning needs structured context. SysNav showed that structured reasoning needs a proper system architecture. IntentNav showed that navigation decisions should be learned rather than prompted. Goal2Pixel shows that the output interface itself should be redesigned to match the geometry of the navigation problem. Together, these four contributions argue that effective embodied navigation requires careful design at every level: representation, system, learning, and output interface.

Chapter 7

Conclusion and Future Work

This thesis studied embodied navigation through four complementary perspectives: structured representation, system-level coordination, learned decision making, and output interface design. Across STRIVE, SysNav, IntentNav, and Goal2Pixel, the central theme has been that semantic reasoning becomes most effective when it is supported by the right structure, applied at the right level of abstraction, trained with the right objectives, and expressed through the right output interface.

The starting point of the thesis was a practical observation: semantic foundation models are powerful, but embodied navigation is not solved by inserting them directly into the control loop. An agent must reason over partial observations, choose efficient long-horizon routes, learn stable behavior from data, and remain robust under sensing noise and embodiment constraints. These demands expose several kinds of mismatch. The semantic model may be given poorly structured context. High-level semantic reasoning may be disconnected from the operational requirements of real robots. The training objective may ignore the directional structure of navigation choices. The output interface may force the VLM to predict in a format that is ambiguous, myopic, and expensive. The four technical contributions of this thesis each address one of these mismatches.

STRIVE showed that VLM-guided navigation benefits from a multi-layer representation of objects, viewpoints, and rooms, together with room-level reasoning and efficient in-room exploration. Its key contribution is to demonstrate that semantic reasoning becomes substantially more useful once the environment is summarized in

a structured form that captures both scene semantics and navigable topology.

SysNav showed that these ideas can be integrated into a deployable three-level system that supports robust real-world object navigation across multiple embodiments. Its key contribution is to elevate room-level semantic reasoning from a method choice to a system principle, coordinating high-level reasoning, mid-level planning, and low-level execution in a way that remains practical outside simulation.

IntentNav showed that navigation decisions themselves should be learned from human demonstrations rather than prompted from scratch. Its key contribution is to design both the decision representation and the training objective around the structure of human search intent, using Frontier-based Human-Intent Labeling and an Intent-Aligned Objective that respects the directional geometry of exploration decisions.

Goal2Pixel showed that the output interface between VLM reasoning and robot motion should be redesigned as navigable pixel grounding. Its key contribution is to demonstrate that the image plane itself can serve as a unified, efficient, and natural interface for VLM-based navigation, eliminating the ambiguity of action-level prediction and reducing VLM inference calls by approximately 6×.

Taken together, these contributions support the main thesis claim: robust and efficient embodied navigation requires careful design across representation, system architecture, decision learning, and output interface. The quantitative evidence across all four chapters demonstrates consistent improvements in success rate, path efficiency, and inference cost, with each contribution addressing a different dimension of the problem.

7.1 Future Work

Several promising directions remain for future research.

Toward an Integrated Navigation System. The four contributions in this thesis were developed as complementary but separate systems. A natural next step is to build an integrated architecture that combines room-level reasoning, embodiment-aware deployment, learned candidate-level decisions, and pixel-grounded action interfaces. A particularly promising avenue is to combine IntentNav's learning from human

demonstrations with Goal2Pixel's pixel grounding, yielding navigation policies trained with intent-aligned supervision adapted to pixel-level targets. The main challenge is designing interfaces between components so that each retains its strengths without introducing new bottlenecks.

Richer Tasks and Larger-Scale Environments. The first three chapters focus on category-level object goals, and Goal2Pixel addresses language instructions in indoor settings. Extending these design principles to richer task formulations—compositional instructions, object attributes, spatial relations, and long-horizon constraints—would require additional forms of memory and higher-level task structure. Scaling to outdoor and multi-floor settings would similarly demand different forms of spatial abstraction, where functional zones could replace rooms and floor plans could serve as high-level planning abstractions for vertical navigation.

Robustness and Generalization Beyond Navigation. Overall performance still depends on reliable depth sensing, object detection, and map maintenance under clutter and sensor noise. Better integration between uncertainty-aware perception and the navigation policy would strengthen practical impact. More broadly, the design principles developed in this thesis—structured representation, system-level coordination, learned decision making, and output interface design—may generalize to mobile manipulation, long-horizon assistance, and human-robot collaboration. Testing whether these principles transfer to other embodied domains would validate their generality.

The broader implication of this thesis is both technical and conceptual. Technically, it provides four concrete methods that improve embodied navigation from different but complementary angles. Conceptually, it argues that embodied navigation should be understood as a layered design problem—one that requires careful attention to representation, system, learning, and grounding interface. That perspective, rather than any single module in isolation, is a central contribution of this work.

Appendix A

STRIVE Technical Details

A.1 Penalized Distance Formula

The penalized distance mechanism discourages backtracking by weighting the geodesic travel cost with factors reflecting the episode progress and exploration history. The penalized distance to a candidate room v_j^{room} is defined as

$$d_{\text{pen}}(v_j^{room}) = \left(1 + \frac{t}{2T_{\max}}\right)^{n_j} \cdot d_{\text{geo}}(v_j^{room}), \tag{A.1}$$

where t is the current step number, $T_{\max}$ is the maximum allowed steps, n_j is the number of previously explored viewpoints along the shortest path to candidate room v_j^{room}, and $d_{\text{geo}}(v_j^{room})$ is the geodesic distance from the agent's current position to the nearest viewpoint in room v_j^{room}.

The time ratio $t/(2T_{\max})$ grows monotonically as the episode progresses, and the exponent n_j penalizes paths that traverse previously explored regions. In early navigation stages, when t is small, the penalty factor is close to unity and the VLM can freely select rooms based on semantic promise. In later stages, rooms reachable via shorter and less-explored paths are preferred, discouraging backtracking to distant rooms that were bypassed earlier in the trajectory.

The design encourages forward exploration momentum while preserving the VLM's ability to make semantically informed exceptions. For example, if the agent has

explored two rooms without finding the target and a third unexplored room is accessible via a short path that does not pass through previously explored viewpoints, the penalized distance will be low and the VLM will prefer this room over a distant room that requires backtracking through explored areas.

A.2 Room Segmentation Algorithm

Room nodes are constructed by identifying all walls in the environment through planar surface fitting on the 3D point cloud and analyzing the distribution of point clouds along the vertical (z) axis. The algorithm proceeds as follows:

1. Detect horizontal planar surfaces corresponding to walls using RANSAC-based plane fitting on the accumulated 3D point cloud.
2. Project detected wall regions onto a 2D top-down grid.
3. Iteratively dilate the wall regions in the 2D projection. As walls expand, the initially connected walkable space is progressively partitioned into disconnected components.
4. Each disconnected component is treated as an individual room and added as a room node.
5. Edges are added between each room node and the viewpoint nodes located within the corresponding room.

This wall-based segmentation is robust in typical indoor environments where rooms are separated by physical walls and doorways. It may produce less meaningful partitions in open-plan layouts where functional areas are not separated by walls, though the viewpoint-level representation still provides useful sub-room structure.

A.3 JSON Structure

The VLM receives a structured prompt containing the task-related context about the current navigation progress and the known environment. As shown in Figure A.1, this context consists of four components:

```
Prompt_info: Find the <toilet>.
You are now at node with position [-1.017, -0.126, -0.8] in the Room 1.
The robot history trajectory is: Position [ 0.0, 0.0, -0.8] --> Position [-1.017, -0.126, -0.8]
```


```
"objects":
[{
  "object_idx": 0,
  "class": "door",
  "position": [-1.274, -1.083, -0.8],
  "confidence": 0.391,
  "size": [0.078, 0.853, 2.053]
},
{
  "object_idx": 1,
  "class": "luggage",
  "position": [-0.05, -0.921, -0.8],
  "confidence": 0.704,
  "size": [0.631, 0.331, 0.178]
},
{
  "object_idx": 2,
  "class": "wardrobe",
  "position": [0.972, -0.839, -0.8],
  "confidence": 0.581,
  "size": [2.909, 0.637, 2.38]
}, ..... ]
```

```
"rooms":
[{
  "room_idx": 0,
  "state": 1,
  "distance": 100000.0,
  "viewpoints": [
    {
      "viewpoint_idx": 0,
      "position": [0.0, 0.0, -0.8],
      "has_frontier": false,
      "objects": [6, 9, 0, 1, 2, 4, 5, 7, 8, 10]
    },
    {
      "viewpoint_idx": 1,
      "position": [1.625, -0.225, -0.8],
      "has_frontier": false,
      "objects": [3, 4, 5, 6, 7, 9]
    }
  ]
}, ..... ]
```


Figure A.1: **Visualization of the JSON scene representation.**

- **Target Object:** The instruction begins by specifying the target, e.g., "`Find the <target object>`".
- **Current Viewpoint and Position:** The agent's current viewpoint ID and 3D coordinates, e.g., "`The robot is now at Viewpoint` v_i `with position` $[x, y, z]$ `in Room` r_i".
- **Navigation History:** The trajectory up to the current step, provided as an ordered sequence of positions, e.g., "`The robot history trajectory is` $[x_1, y_1, z_1] \rightarrow [x_2, y_2, z_2] \rightarrow \ldots$".
- **Scene Representation:** The scene representation $\mathcal{R}$ is formatted as a JSON file containing hierarchical information about rooms, viewpoints, and objects:
  - **Rooms**: `room_idx`, `state` (1 for fully explored, 0 for partially explored), `distance` (penalized travel distance from the agent's current position), and `viewpoints` (list of viewpoint IDs in the room).
  - **Viewpoints**: `viewpoint_idx`, `position`, `state` (1 for visited, 0 for unvisited), `neighbors` (list of connected viewpoint IDs), `has_frontier` (whether unknown regions exist around the viewpoint), and `objects` (list of observable object IDs).

- **Objects**: `object_idx`, `position`, `class` (category), `confidence` (detection confidence), and `size` (bounding box dimensions in meters).

Note that when translating the representation $\mathcal{R}$ into a JSON file, objects are listed first rather than nested under each viewpoint. This is because an object may be associated with multiple viewpoints; directly listing objects under each viewpoint would lead to redundancy and may exceed the prompt's length limit. The full heuristic prompt, including the explanation of each JSON field provided to the VLM, is reproduced in Section A.7.

A.4 Qualitative Analysis

Qualitative analysis of STRIVE's navigation traces reveals several important behaviors. In multi-room apartments, the VLM correctly identifies room types based on observed objects and prioritizes rooms with strong semantic affinity to the target. For example, when searching for a bed, the VLM first directs the agent to bedrooms (identified by the presence of nightstands and dressers) before exploring living rooms or kitchens.

The early-stop mechanism demonstrates effective judgment about when to terminate in-room exploration. In a representative episode, after exploring the main area of a living room, only inner frontiers behind a sofa and a bookshelf remain. The VLM correctly reasons that these occluded regions are unlikely to contain a refrigerator and directs the agent to exit the room, saving approximately 15 steps compared to exhaustive frontier exploration.

The penalized distance mechanism becomes most visible in later navigation stages. In episodes where the agent has explored several rooms without finding the target, the mechanism successfully prevents backtracking to distant rooms that were bypassed earlier, instead directing the agent toward nearby unexplored rooms. This produces more coherent exploration trajectories compared to methods without path-cost awareness.

A.5 Token Usage Analysis

The token usage comparison between room-level and viewpoint-level VLM planning reveals a substantial efficiency difference. Room-level planning, as used in STRIVE, requires an average of 8,068 tokens per episode because the VLM is invoked only for cross-room decisions and early-stop checks. In contrast, viewpoint-level planning—where the VLM evaluates all candidate viewpoints at each step—requires 22,935 tokens per episode, approximately 2.8× more.

This reduction comes from two sources. First, the number of rooms is typically much smaller than the number of frontiers or viewpoints, so each VLM call involves less context. Second, the VLM is invoked less frequently because room-level decisions cover larger spatial extents than viewpoint-level decisions. The token savings are particularly important for real-world deployment, where VLM inference time and cost are practical constraints.

A.6 True vs. Inner Frontier Distinction

STRIVE distinguishes between two types of frontiers based on their geometric location relative to room boundaries. **True frontiers** lie along the boundary between the current room and unexplored or adjacent regions, typically corresponding to doorways, hallways, or open passages. These frontiers represent genuine opportunities to discover new rooms and their contents. **Inner frontiers** arise within the room itself, caused by furniture such as sofas, bookshelves, or kitchen islands that occlude small regions of space behind them.

The distinction is made through geometric analysis of the frontier's location. If a frontier lies on the room boundary (identified during wall-based room segmentation), it is classified as a true frontier. If it lies within the room interior, it is classified as an inner frontier. The agent first resolves true frontiers to systematically cover the room. Only when true frontiers are exhausted does the agent consider inner frontiers, and the VLM-based early-stop mechanism determines whether additional exploration of these small occluded regions is worthwhile.

This distinction is important because it prevents a common failure mode of frontier-based exploration: spending excessive steps resolving small occluded regions

that have no semantic relevance to the task. In practice, the VLM correctly identifies that inner frontiers behind a couch are unlikely to contain a refrigerator, or that frontiers behind a bookshelf are unlikely to conceal a bed.

A.7 Exploration Heuristic Prompt

To guide the VLM's cross-room decisions, a general heuristic prompt is prepended to every VLM invocation. This prompt provides the VLM with an overall concept of the object navigation task, explains the meaning of each field in the JSON file, and explicitly requires the VLM to balance semantic likelihood against travel cost. The full prompt is reproduced below:

```
You are a wheeled mobile robot operating in an indoor environment. Your goal is
to efficiently find a target object based on a human-provided instruction in a
new house. The current room you are in has been fully explored. To achieve the
goal, you must select the next room to explore from the partially explored rooms
listed in a JSON file, aiming to complete the task as quickly as possible.

### Provided Information:
    1. A specific instruction describing the task.
    2. A description of your current position and previous trajectories.
    3. A JSON file containing details about the scene, including rooms,
       viewpoints, and objects.

The JSON file contains the following information:
- Objects:
    - object_idx: A unique identifier for the object.
    - position: The spatial coordinates of the object.
    - class: The category or type of the object.
    - confidence: The confidence level of the classification result.
    - size: The bounding box size of the object (in meters).

- Viewpoints:
    - viewpoint_idx: A unique identifier for the viewpoint.
    - position: The spatial coordinates of the viewpoint.
```

```
    - state: The state of the viewpoint (1 for visited, 0 for unvisited).
    - neighbors: A list of connected viewpoints.
    - has_frontier: Relevant only when the viewpoint is unvisited.
        True: The viewpoint has a frontier, meaning unknown regions exist.
        False: The area has been observed from distant viewpoints,
               but small objects may still be unclear.
    - objects: A list of objects observable from the viewpoint.

- Rooms:
    - room_idx: A unique identifier for the room.
    - state: The state of the room (1 for fully explored, 0 for partially).
    - distance: The distance (in meters) the robot needs to travel to reach
      this room.
    - viewpoints: A list of viewpoints in the room.

### Task:
    You must carefully analyze the JSON file, using logical reasoning and
    common sense, to select the next room to explore from the list of
    partially explored rooms. Consider the following factors:
    - Evaluate how closely each room's viewpoints align with the overall
      task objective.
    - Optimize the exploration path by leveraging the robot's current
      momentum and minimizing unnecessary backtracking or redundant movements.
    - Assess the likelihood that exploring the selected room will meaningfully
      advance or complete the overall task.

### Output Format:
    Your response should include:
    - steps: The chain of thought leading to the decision.
    - final_answer: The idx of the next room to explore.
    - reason: The rationale for selecting this room.

Note: The chosen room must be partially explored.
```

A.8 Examples of VLM Reasoning

Figures A.2 to A.4 provide additional examples of the VLM's chain-of-thought reasoning process during cross-room selection. These examples demonstrate that the structured representation enables the VLM to reason effectively over both spatial layout and semantic cues, leading to more accurate room selection. Moreover, the VLM is able to balance the likelihood of finding the target object with the travel distance cost when planning room-to-room exploration.

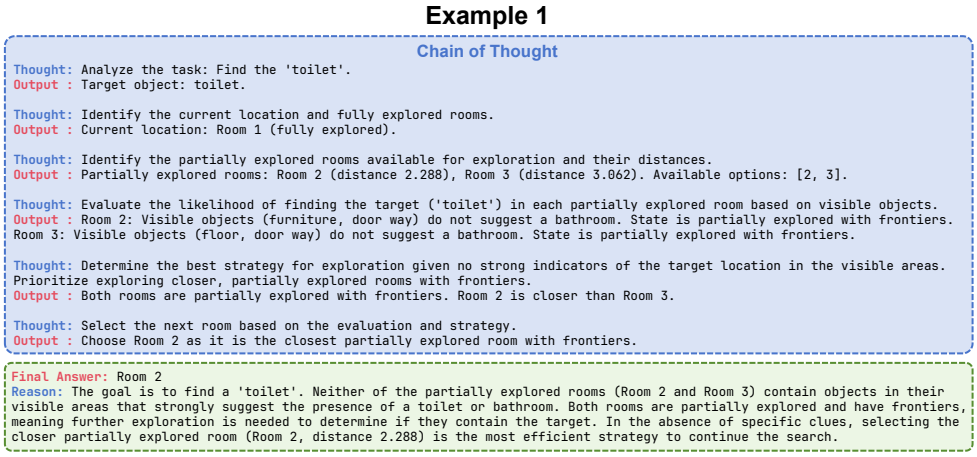

Figure A.2: **Example 1 of VLM reasoning.** The VLM evaluates semantic priors and penalized distance to select the next room.

A.9 Context-Aware Verification

When the object detector reports a candidate target, the VLM performs a context-aware verification to confirm or reject the detection. Rather than examining the target object in isolation, the VLM is provided with the surrounding scene context—neighboring objects, room type, and spatial layout—to assess whether the detection is plausible. For example, if the detector labels an object as a "toaster" but the surrounding context shows a bedroom with a bed and nightstand, the VLM can reject the false positive.

Figure A.5 shows representative examples. The VLM effectively utilizes the

Example 2

Chain of Thought

```
Thought: The task is to find a <potted_plant>.
Output : Analyze the task instruction.

Thought: Currently in Room 5, which is fully explored.
Output : Identify the current location and status.

Thought: Partially explored rooms available are 1, 2, 3, and 4.
Output : Identify the available rooms for exploration and their status.

Thought: Review the objects and viewpoints in each room: Room 1 (distance 4.612) has furniture, stuffed animal, lamp, box.
Room 2 (distance 1.553) has floor, doorway, furniture, sink. Room 3 (distance 3.18) has doorway, air duct, electrical outlet,
floor, wall. Room 4 (distance 2.9) has floor, furniture, blanket, window, doorway.
Output : Evaluate each partially explored room based on its content and distance.

Thought: Rooms 1, 2, and 3 don't show objects typically associated with potted plants. Room 2 (sink) seems least likely. Room
4 contains a window, which is a common location for potted plants.
Output : Assess the likelihood of finding a <potted_plant> in each room.

Thought: Despite Room 2 being closer, Room 4 is more likely to contain the target object (<potted_plant>) due to the presence
of a window. Prioritize likelihood over minimum distance in this case, assuming the distance difference is not prohibitive.
Output : Select the most promising room for exploration.

Thought: Room 4's state is 0, indicating it is partially explored.
Output : Confirm the selected room is partially explored.
```

```
Final Answer: Room 4
Reason: The goal is to find a <potted_plant>. I am in Room 5, which is fully explored. I need to select a partially explored
room to search. Evaluating the partially explored rooms (1, 2, 3, 4): Room 1 contains furniture, stuffed animal, lamp, and box.
Room 2 contains a sink, suggesting a bathroom or utility area, less likely for plants. Room 3 contains doorway, air duct,
electrical outlet, floor, and wall. Room 4 contains furniture, blanket, and crucially, windows, which are often associated
with potted plants. Although Room 2 is closer, the presence of a window in Room 4 makes it a more promising location to search
for a potted plant. Room 4 also has unexplored areas (frontiers). Therefore, Room 4 is the best choice to explore next.
```

Figure A.3: **Example 2 of VLM reasoning.** The VLM correctly prioritizes a semantically promising room despite a longer travel distance.

surrounding context to verify detected target objects and avoid false positives, demonstrating that scene-level reasoning complements per-object detection.

A.10 Viewpoint-Optimized Re-Verification

Once a target object is detected, the system does not immediately accept the detection. Instead, it plans a path from the current position toward the target object and computes a better viewpoint for re-verification. Candidate viewpoints are sampled along this path at intervals of 0.25 m (the agent's forward step length). For each candidate, the width, height, and visible fraction of the target object in the camera frame are computed. The candidates are traversed in reverse order (from the target object back toward the current position), and the first viewpoint satisfying all of the following conditions is selected:

1. More than 95% of the target object's point cloud is visible.
2. The 2D bounding box area of the target object exceeds that at the original detection viewpoint.
3. The estimated object width and height at this viewpoint are both greater than

Example 3

Chain of Thought

```
Thought: Understanding the goal and current state.
Output : The goal is to find a 'chair'. I am currently in Room 0, which is fully explored.

Thought: Focusing on the available options for exploration.
Output : Evaluate the partially explored rooms (1, 2, 3) and the objects observed in each.

Thought: Assessing each room's potential.
Output : Room 1 is 1.314m away and its viewpoints only see 'rug'. Room 2 is 2.601m away and its viewpoints see 'column' or no
objects. Room 3 is 3.23m away and its viewpoints see 'table', 'rug', and 'column'.

Thought: Relating observed objects to the target object ('chair').
Output : A 'table' is often found in the same area as a 'chair'. Room 3 is the only room among the options where a 'table' has
been observed.

Thought: Prioritizing exploration based on likelihood.
Output : Based on the observed objects, Room 3 has the highest probability of containing a 'chair' compared to Room 1 and 2.

Thought: Considering distance alongside potential relevance.
Output : Although Room 3 is the furthest, the increased likelihood of finding the target object justifies exploring it next.
```

```
Final Answer: 3
Reason: Room 3 is the most promising room to explore next because a 'table' has been observed there, and 'chairs' are commonly
found with 'tables'. This suggests a higher probability of finding the target object ('chair') in Room 3 compared to Rooms 1
and 2, where only less relevant objects like 'rug' and 'column' were seen.
```

Figure A.4: **Example 3 of VLM reasoning.** The VLM avoids backtracking to a distant room and selects a nearby unexplored alternative.

80% of their original values, scaled by the square root of the bounding box area ratio.

The square root of the area ratio is adopted as a scaling factor to account for the expected increase in apparent object dimensions under improved viewing conditions, thereby ensuring a broader and clearer observation for more reliable VLM-based re-verification.

Figure A.6 shows representative examples. The re-verification viewpoint is not only closer to the target but also provides a clearer view, allowing the VLM to make a more confident verification decision.

Figure A.5: **Examples of context-aware verification.** The VLM uses surrounding scene context to confirm or reject candidate detections.

Figure A.6: **Examples of viewpoint-optimized re-verification.** The selected viewpoint provides a clearer and larger view of the target object, enabling more confident VLM verification.

Appendix B

SysNav Technical Details

B.1 Cross-Embodiment Platform Comparison

The three robot platforms used in SysNav's real-world evaluation exhibit different physical characteristics that affect navigation behavior:

Wheeled robot: Achieves the fastest traversal speed on flat surfaces but cannot cross door thresholds higher than 2 cm, limiting its accessible area in some buildings. Uses differential drive control and 2D LiDAR-based obstacle avoidance.

Unitree Go2 quadruped: Handles uneven terrain and stairs reliably but has a lower sensor mounting height, which slightly reduces its frontier detection range compared to the wheeled robot. Manages legged locomotion with body pose stabilization.

Unitree G1 humanoid: Provides the highest sensor vantage point and can traverse any terrain the quadruped can handle, but its walking speed is currently slower and its balance controller introduces slight oscillations that affect image stability.

Despite these platform-specific differences, the high-level navigation decisions remain consistent across all three embodiments, confirming that the semantic reasoning and room-based planning layers are genuinely embodiment-agnostic. The quadruped and humanoid platforms demonstrate that the system can handle challenging terrain such as door thresholds, carpet transitions, and uneven ground that would be impassable for a wheeled robot.

B.2 Viewpoint Coverage Parameter

The formal coverage criterion for viewpoint construction is defined in the main text (Chapter 4). For real-world deployment, the coverage distance d_{cover} is set to 1.5 m, meaning that semantic and geometric information within this radius is considered fully observed by a viewpoint node.

B.3 Cross-Embodiment Transfer Challenges

Cross-embodiment support is essential for real-world ObjectNav because the operating terrain is unknown ahead of time, while different embodiments differ substantially in what terrain they can negotiate. A system tied to a single platform may fail to reach the target purely for mobility reasons, even when its high-level navigation policy is sound. The specific challenges introduced by embodiment transfer fall along three axes:

Sensor extrinsics. Observations live in the sensor frame while motion and traversability checks live in the body frame, and the offset between the two varies per platform. Body-induced LiDAR occlusion falls in different directions and at different ranges on each platform, affecting near-range obstacle avoidance. Cross-modal fusion (e.g., projecting the LiDAR point cloud onto the camera image) depends on the inter-sensor relative pose, which must be re-calibrated per embodiment.

Physical footprint. Traversable space is body-specific: narrow doorways passable by the Go2 quadruped may be impassable for the wide-base wheeled robot. The system cannot use a single, embodiment-agnostic model to compute traversability and collision avoidance across platforms.

Motion dynamics. Different platforms have different motion limits. Wheeled robots and the quadruped can move and turn quickly, whereas the humanoid must move and turn relatively slowly to maintain balance, so velocity configuration must be tailored to each embodiment.

Thanks to its multi-level, zero-shot design, SysNav requires no per-embodiment training. Only a small set of parameters and the swappable low-level controller module need to change: the LiDAR–camera calibration matrix is updated in the high-level semantic reasoning module, and the sensor-to-body offset, blind-zone distribution,

traversable-space constraints, and motion dynamics are handled by tuning the low-level base autonomy parameters. The key contribution is not the use of off-the-shelf controllers, but the system-level *decoupling* of base autonomy into a standalone, swappable module.

B.4 VLM Runtime Analysis

To quantify the impact of VLM queries on system real-time performance, we analyze them from two perspectives: latency and frequency.

Latency. VLM query latency was measured over 1,000 trials, yielding an average of 1,153.7 ms with 95% of queries returning within 2,032 ms. The distribution is shown in Figure B.1.

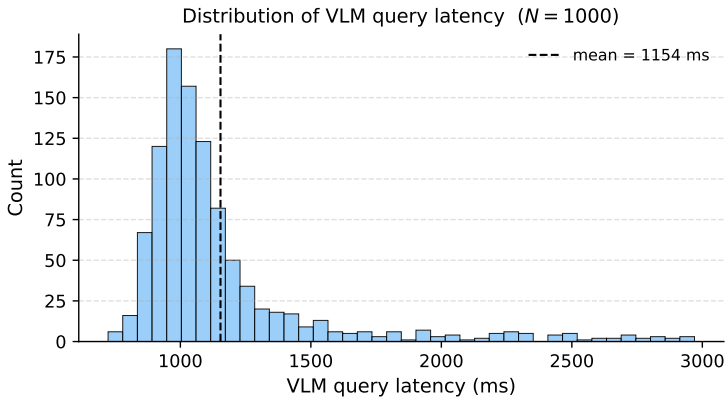

Figure B.1: **Distribution of VLM latency over 1,000 calls.** Mean latency 1,154 ms; 95% of queries return within 2,032 ms.

Frequency. SysNav issues three types of VLM queries: (1) *next-room queries*, which decide the next room to explore; (2) *early-stop queries*, which decide whether to abandon the current room in favor of a newly discovered one; and (3) *target-object verification queries*, which confirm whether a detected object matches the target and satisfies all specified constraints. Across 78 real-world trials with complete logs, the average query rates are 2.85/min (next-room), 0.94/min (early-stop), and 2.57/min (target verification). The per-type frequency distributions are shown in Figure B.2.

Prefetching design. To mitigate the latency of the most time-sensitive query type—the next-room query—SysNav adopts a prefetching strategy. Rather than

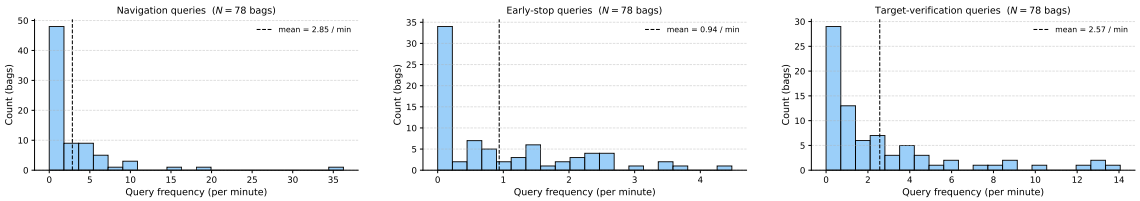

Figure B.2: **Per-episode frequency distributions of VLM queries.** Left: next-room queries (mean 2.85/min). Center: early-stop queries (mean 0.94/min). Right: target-object verification queries (mean 2.57/min).

waiting until the current room is fully explored, the system issues the VLM query as soon as it anticipates completing the current room. Since the prefetch is triggered only when the room is nearly finished, the scene representation sent to the VLM is already nearly identical to the final one. The prefetched decision is held in cache and committed only after the room has been fully explored without discovering substantial new regions. In the rare case that a large new region is uncovered after the prefetch, the cached decision is discarded and a fresh query is issued. This design allows VLM inference to overlap with the robot's residual exploration, effectively hiding the query latency without sacrificing decision accuracy. In real-world demonstrations, the robot rarely pauses for VLM responses, and most queries complete seamlessly while the robot is in motion.

B.5 Environment Scale and Floor Plans

The 190 real-world experiments span 15 distinct environments, including office rooms, laboratories, lounges, open-plan offices, and restaurants. These environments exhibit diverse architectural layouts, ranging from open spaces to enclosed areas with varying room counts and sizes. Table B.1 summarizes the scale statistics across simulation and real-world settings.

On average, real-world environments are roughly 4× larger in area than simulation benchmark scenes, rising to 10× for the four floor-scale large scenes. The geodesic and traversed paths are correspondingly longer: the four floor-scale scenes yield trajectories roughly 10× longer than the simulation average. As a zero-shot system, SysNav is inherently designed for generalization and does not suffer from overfitting

Table B.1: **Environment scale and path-length statistics across simulation benchmarks and real-world experiments.** "Area" denotes per-floor area for simulation and per-scene area for real-world settings.

Setting	Scenes	Episodes	Area (m^2)	GT Path Length (m)	Real Path Length (m)
Simulation (4 benchmarks)	47	8,195	98.38	6.78	10.72
Real-world (all scenes)	15	78	432.17	16.24	48.24
Real-world (4 floor-scale)	4	4	1,007.21	84.43	280.22

to specific training scenarios.

Figure B.3 shows the floor plans of the 11 real-world scenes used for quantitative and qualitative unit tests. Figure B.4 shows the four floor-scale large scenes used for long-range ObjectNav, each covering an entire floor with an average of 17.25 rooms.

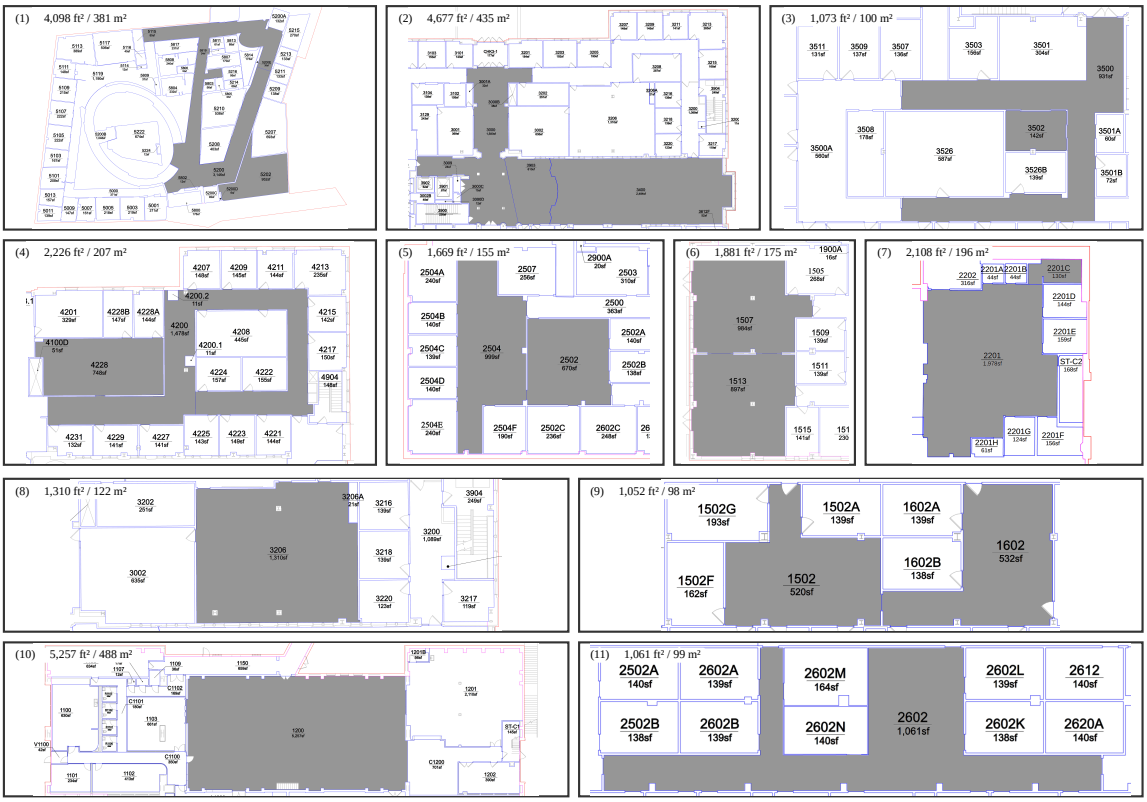

Figure B.3: **Floor plans of the 11 real-world scenes used for unit tests.** Dark-gray polygons denote explorable rooms, with the total area of each scene annotated in the top-left corner.

Figure B.4: **Floor plans of the 4 floor-scale large real-world scenes used for long-range ObjectNav.** On average, the four scenes span 1,007 m^2 and contain 17.25 rooms each.

Appendix C

IntentNav Technical Details

C.1 Case Study Analysis

Detailed comparison of navigation trajectories between IntentNav and zero-shot methods (STRIVE and SysNav) reveals complementary strengths.

In a representative MP3D episode where the target is a toaster in a large apartment with multiple rooms, the STRIVE-style zero-shot approach correctly identifies the kitchen as the most promising room based on semantic priors, but within the kitchen it spends several steps resolving inner frontiers caused by countertop occlusion before finding the target. IntentNav, by contrast, selects a frontier candidate near the kitchen entrance based on learned visual memory—the egocentric image from that candidate shows a partial view of countertop appliances—and navigates directly to it, finding the toaster in fewer steps.

However, in episodes where the target is in an unexpected location—for example, a pillow in a bathroom—IntentNav's learned policy initially favors the bedroom based on its training distribution, while the zero-shot VLM in STRIVE can reason about unusual placements using commonsense knowledge. This illustrates a fundamental trade-off: learned policies are more stable and efficient for typical scenarios but may lack the flexibility of zero-shot reasoning for atypical cases.

In episodes with many frontiers—for example, a large open office with multiple cubicle areas—the candidate set can grow to 20 or more candidates. IntentNav's reserved candidate-ID tokens handle this gracefully because the VLM performs

classification over a discrete set, and the agent-centric geometry features provide strong spatial grounding even when candidates are numerous. In contrast, text-coordinate prediction degrades because the VLM must generate free-form coordinate text, which becomes less reliable as spatial complexity increases.

C.2 BEV Map Construction

The mapping module maintains a global occupancy map with resolution $r = 0.05\,\mathrm{m}$ per cell covering $67.2\,\mathrm{m} \times 67.2\,\mathrm{m}$ (1344×1344 grid). The VLM receives a 448×448 egocentric crop covering $22.4\,\mathrm{m} \times 22.4\,\mathrm{m}$. At each timestep, valid depth pixels (filtered to $[0.51, 4.9]\,\mathrm{m}$) are back-projected into a camera-frame point cloud and transformed into the world frame. Points near the floor ($z < h_0 + 0.2\,\mathrm{m}$) support the explored/free-space estimate; points in the obstacle band ($h_0 + 0.2\,\mathrm{m} < z < 1.5\,\mathrm{m}$) support the occupied-cell estimate. The map maintains two complementary binary layers: an occupancy layer O_t for non-traversable cells and an explored layer E_t for cells observed by the sensor.

Depth-column completion. First-person depth images often contain vertical holes around thin structures (e.g., table legs), where background depth shows through and causes false free space in the BEV projection. Before back-projection, each column is scanned from the bottom row to the image midpoint. At each pixel j, a discontinuity is flagged when the pixel above is valid foreground ($d_{j-1} > 0.58\,\mathrm{m}$) but much closer than the current pixel ($d_{j-1} < d_j - 0.08\,\mathrm{m}$). All farther pixels in between are overwritten with d_{j-1}.

Near-field floor completion. The first-person camera leaves a near-field blind region on the floor. With 640×480 resolution, 79° horizontal FoV, and $0.88\,\mathrm{m}$ camera height, the lowest camera ray intersects the floor at approximately $1.42\,\mathrm{m}$, leaving roughly 28.5 BEV cells directly in front of the agent unobserved. Within the horizontal field of view, bearing is discretized into 3° bins; the nearest observed floor or obstacle point in each bin (up to $2.0\,\mathrm{m}$) is found, and the intervening BEV cells are interpolated as explored. This only affects the explored layer; obstacle cells are still determined by the obstacle-height point set.

Map channels. The local BEV crop contains four channels: occupancy (cells blocked by obstacles), explored area (cells observed by the sensor), current agent footprint, and trajectory history (accumulated past footprints).

Frontier extraction. Frontiers are extracted at the boundary between explored free space and unexplored regions, with morphological filtering and connected-component filtering to remove small noisy components. The remaining frontier centers form the frontier portion of the candidate set $\mathcal{C}_t$.

C.3 Waypoint Supervision Strategies

Human demonstrations are recorded as low-level action sequences rather than explicit waypoint decisions, so a label-recovery procedure is required to convert each replayed step into a candidate-level supervision label y_t. Three strategies are compared over the same candidate set $\mathcal{C}_t$:

Strategy 0: Pose-Heading Matching. A local heuristic that selects the frontier most aligned with the agent's instantaneous heading, without looking into the future trajectory. For each frontier candidate x_i, the angle between the agent heading $\mathbf{h}_a$ and the direction $(x_i - \mathbf{p}_a)/\|x_i - \mathbf{p}_a\|_2$ is computed, and the candidate with the smallest angular deviation is selected. Simple but noisy when the demonstrator is turning in place or temporarily facing away from the intended frontier.

Strategy 1: Future-Trajectory Matching. Compares each frontier candidate with the remaining demonstration path $\tau_{t:T}$ via a one-sided Chamfer distance: for each frontier x_i, a shortest path from $\mathbf{p}_a$ to x_i is computed via the simulator pathfinder, and the average minimum distance from the future trajectory to this path is measured. More robust to instantaneous orientation noise, but remains ambiguous when multiple frontiers lie along similar partial trajectories.

Strategy 2: Frontier-Collapse Supervision. The proposed strategy selects the frontier whose disappearance from the future frontier set is most pronounced. The replay is advanced to a pivot step t^+ (at least K=6 additional `move_forward` actions, extended if the frontier set is still nearly unchanged), then the current non-target frontier with the largest geodesic nearest-neighbor distance to $\mathcal{F}_{t^+}$ is selected. Two safeguards handle degenerate cases: a verified target option, if present, is used directly;

if all frontiers are unreachable, the method falls back to Strategy 1.

C.4 Target Proposal and Verification

The target option is treated as a verified waypoint hypothesis rather than a raw detection. A target candidate is appended to the frontier set only after it is semantically valid, temporally stable, geometrically localizable, and near a navigable position.

Semantic and temporal verification. During training, simulator semantic masks are used. During evaluation and deployment, online detectors (open-vocabulary detector-segmenter in simulation; YOLOE in physical deployment) are used. A binary detection buffer over the last three frames requires at least two positives before accepting a proposal.

3D localization. The accepted target mask is back-projected into a 3D point cloud, voxel-downsampled, and reduced to its largest connected component. Two 3D summaries are computed: the component median A and the median of the $K = \min(5, N)$ nearest-to-agent points B. The raw target waypoint is $\mathbf{p}_{\text{tar}} = 0.7B + 0.3A$, biased toward the visible side. A nearby navigable point on the same navigation island is found by retreating toward the agent in 0.1 m steps, then probing radial offsets if retreat fails. Maximum allowed shift is 1.0 m by default, 1.5 m for toilets, and 2.0 m for potted plants and TV monitors.

Persistence check. While following a target, a missed-detection counter is incremented only when the target is expected to be observable (inside the camera frustum and not occluded). When the counter reaches N_{miss}, the hypothesis is invalidated and removed, preventing commitment to a stale false positive.

C.5 Training Configuration

IntentNav is built on InternVL3-2B and trained on the Habitat-Web ObjectNav demonstrations [54] collected via AMT in MP3D scans. Crowd workers navigated in first-person RGB view without depth or GPS+Compass, producing demonstrations averaging ∼243 steps. After replay, filtering, candidate construction, and Frontier-based Human-Intent Labeling, the dataset contains 2,363,108 candidate-level training

samples from 28 MP3D training scenes and 21 goal categories.

Table C.1: **Training policy per module.** LoRA is applied to the frozen pretrained visual and language backbones; newly introduced projection and spatial modules are fully trainable.

Module	Base weights	Adaptation
BEV ViT	Frozen	LoRA rank 64
Ego ViT	Frozen	LoRA rank 16
MLP bridge	Fully trainable	—
LLM	Frozen	LoRA rank 64
`PositionEmbedding2D`	Fully trainable	—
`PairwiseSpatialEncoder`	Fully trainable	—
`<id_k>` token embeddings	Trainable via gradient hook	—

Table C.1 details the per-module training policy, and Table C.2 lists the full hyperparameters. The BEV ViT, Ego ViT, and Qwen2 LLM are frozen and adapted with LoRA; the MLP bridge and spatial modules are fully trainable. Thirty-two reserved candidate-ID tokens `<id_0>`–`<id_31>` are added to the vocabulary with embeddings updated via a gradient hook.

C.6 Prompt Template

The VLM input is serialized using a fixed prompt template that encodes the BEV map, candidate set, target goal, and candidate-associated egocentric RGB memories. The template opens with a system instruction:

```
Imagine you are an autonomous robot in an indoor habitat environment.
Inputs:
- BEV grid map <image_bev> showing free space (white), occupied space
(black), unexplored area (gray), frontier candidates (green dots),
robot pose/heading (red arrow), past trajectory (blue line), and egocentric
camera field of view (yellow cone).  An orange dot may appear indicating
the detected goal location.
- Each candidate is paired with an egocentric RGB memory image <image_ego>.
- Goal:  search for and navigate to **{goal}**.
```

Table C.2: **Hyperparameters for the released IntentNav checkpoint.**

Hyperparameter	Value
Model	
Base model	InternVL3-2B
LLM LoRA rank	64
BEV ViT LoRA rank	64
Ego ViT LoRA rank	16
BEV image resolution	448×448
Ego image resolution	224×224
BEV tokens per image	256
Ego tokens per image	32 (pooled from 256)
Maximum candidates per step	32
Optimization	
Optimizer	AdamW
Learning rate	1×10^{-4}
LR schedule	Cosine decay
Training epochs	3
Precision	BF16
Distributed training	DeepSpeed ZeRO Stage 2
Intent-Aligned Objective	
Angular bandwidth σ	25° (0.436 rad)
Loss blend λ	0.3
Target-safe hard CE	Enabled
Compute	
GPU	$4\times$ A100 (80 GB)

The agent state is serialized as `<state> <s> pos=(r, c) yaw_deg=`θ `<e_s>` in BEV-grid pixel coordinates. Each candidate is serialized as a `<cand>` block containing the ID token, type (frontier or target), BEV coordinate, and paired RGB memory image. The prompt closes with `Choose one candidate token. Output only one token in the form <id_k>.` At training time, candidate order is randomly permuted and the supervision target is remapped accordingly; at inference, the natural order is used (frontiers first, target last).

C.7 Physical-Robot Deployment Adaptations

The learned waypoint policy is kept unchanged in physical deployment; only simulator-specific perception and control interfaces are replaced. A ROS 2 bridge converts registered LiDAR scans, SLAM odometry, and egocentric camera images into the same BEV and candidate representation used during training, then publishes the selected waypoint to the robot's local planner.

LiDAR-based BEV construction. The robot subscribes to SLAM-registered LiDAR scans rather than dense depth images. The global BEV map preserves the training geometry (67.2 m×67.2 m, 0.05 m resolution, 448×448 crop). To reduce distribution shift, LiDAR returns are filtered to the 79° camera-facing sector. Obstacle evidence uses a robot-relative height band $[-0.4, 1.2]$ m; explored cells are marked by ray casting that stops at occupied cells.

Frontier stabilization for real scans. Additional filters handle noisy LiDAR maps: the extractor operates on the global map to avoid local-crop flicker, keeps only the robot-connected explored component, dilates obstacles before frontier masking, removes tiny components and candidates near the crop border or within 0.7 m of the robot, and passes at most 32 candidates to the VLM.

External target detector. Detection runs in a separate ROS node using a YOLOE segmentation model (TensorRT engine). Detection timestamps are aligned with SLAM poses via short pose-history interpolation, which is critical because even small delays can place the target bearing on the wrong side of the BEV map.

Target localization without metric depth. The physical detector provides no depth map, so the target is localized by ray casting along the bounding-box center bearing in the LiDAR BEV obstacle map. The ray stops at the first occupied cell, and the hit point is projected into the local BEV crop and appended after the frontier candidates.

Waypoint execution. Approaching within 0.5 m of a frontier waypoint triggers an immediate VLM re-query rather than a full stop, allowing inference to overlap with the final approach. A stop is issued only when the followed waypoint is the verified target.

C.8 Model Architecture Details

IntentNav is built on InternVL3-2B (InternViT 300M + Qwen2 1.5B, connected by an MLP bridge with GELU). The 448×448 BEV input yields 256 tokens after pixel shuffle (downsample ratio 0.5). The model is extended with dual visual routing, spatial embedding modules, and reserved candidate-ID tokens.

Dual-ViT routing. The BEV map and ego RGB memories have markedly different visual statistics, so they are processed by two separate ViT branches, both initialized from pretrained InternViT and adapted with independent LoRA modules. The BEV crop receives 256 tokens; each ego image receives 32 tokens (pooled from 256 via group-of-8 averaging), keeping the per-candidate token cost compact.

Spatial injection. For BEV visual tokens, a `PositionEmbedding2D` produces a 1536-d embedding (768-d sinusoidal position + 768-d learned heading projection) added at the agent cell and rendered candidate pixels. For text tokens, each candidate's relative-geometry feature $\mathbf{f}_i$ is passed through a `PairwiseSpatialEncoder` (3→384→1536→1536, GELU) and a dedicated `cand_pos_mlp`, then added to `<cand>` and `<e_cand>`. Separate adapters for agent-state and candidate-geometry embeddings avoid sharing projections across different input distributions.

Candidate-ID special tokens. Thirty-two reserved tokens `<id_0>`–`<id_31>` are added to the vocabulary. Their embeddings are initialized from the pretrained embedding mean and updated via a gradient hook while the rest of the LLM embedding table stays frozen. The angular soft loss is computed over the active ID token set at the answer position.

C.9 Real-World Trajectory Walkthroughs

The same MP3D-trained checkpoint is deployed on three physically distinct platforms—a wheeled Mecanum robot, a Unitree Go2 quadruped, and a Unitree G1 humanoid—without any platform-specific fine-tuning. Figure C.1 shows a wheeled robot trajectory and Figure C.2 shows a humanoid trajectory. Despite differences in locomotion, sensing, and embodiment morphology, the policy produces consistent exploration and target-approach behavior across all three platforms.

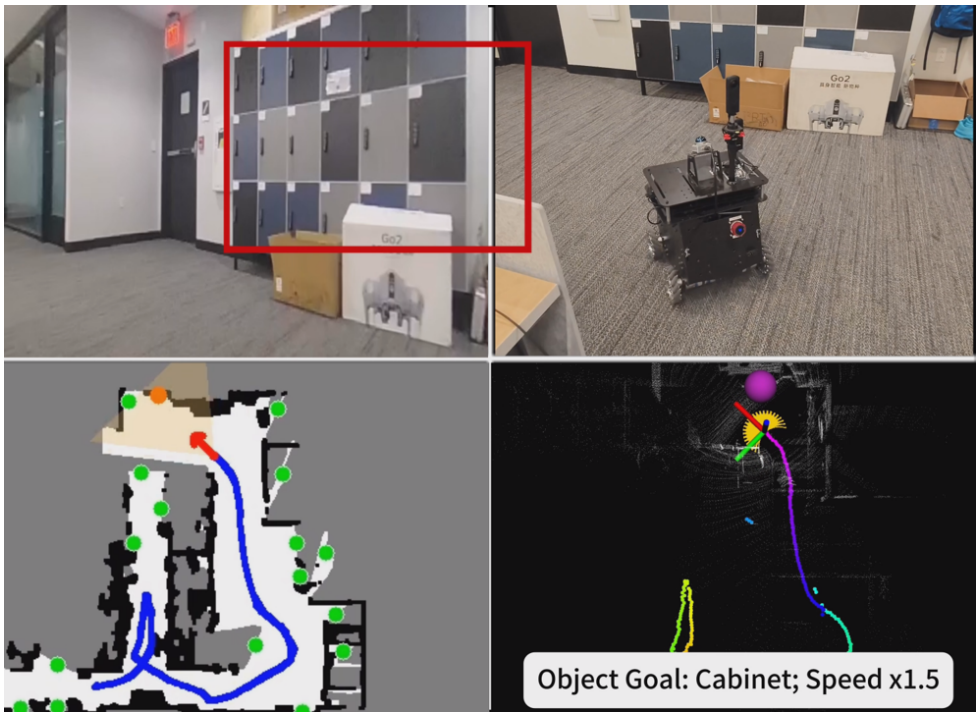

Figure C.1: **Wheeled robot, object goal: cabinet.** The agent navigates through a corridor lined with metal lockers, making a brief exploratory detour into a side passage before promptly reversing and committing to the main corridor.

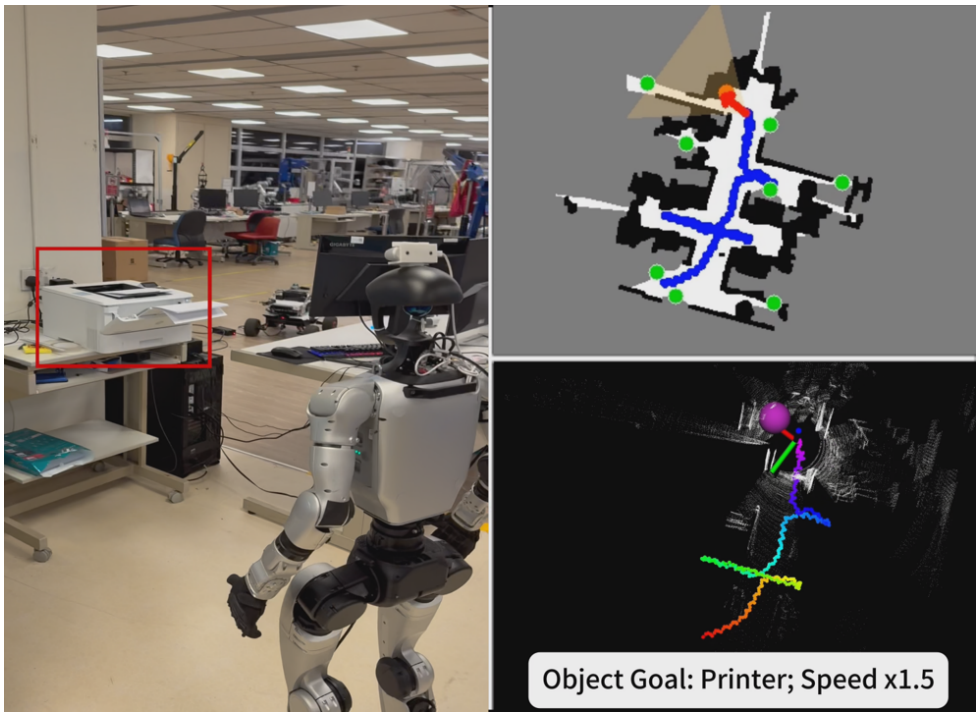

Figure C.2: **Unitree G1 humanoid, object goal: printer.** The humanoid navigates a densely furnished office workspace, systematically probing multiple side corridors before committing to the correct direction and locating the printer on a rolling desk.

Appendix D

Goal2Pixel Technical Details

D.1 Real-World Failure Analysis

As noted in the main text (Chapter 6), analysis of the 16 real-world navigation trials reveals two primary failure modes: depth inaccuracy at ranges beyond 3 meters, and auxiliary directive confusion in visually cluttered environments. Here we provide additional detail on each failure mode.

Depth inaccuracy at longer ranges: When the predicted pixel corresponds to a navigable region more than 3 meters away, the depth value at that pixel may be unreliable due to the limited range of consumer-grade depth sensors. The resulting 3D waypoint can be significantly displaced from the intended location, causing the local planner to navigate to an incorrect position. In two of the 16 trials, this caused the agent to drift toward a wall instead of the intended corridor entrance. The project–reproject consistency check catches some of these errors during training data construction but cannot prevent them at inference time.

Auxiliary directive confusion: In visually cluttered environments where the boundary between the RGB region and the auxiliary directive regions is not clearly delineated, the model occasionally predicts a pixel near the boundary that is ambiguous. This occurs more frequently when the current observation contains strong visual features near the image edges, which compete with the learnable directive embeddings for the model's attention.

These failures suggest two directions for improvement: (1) incorporating learned

depth estimation or depth completion to improve back-projection quality, and (2) strengthening the visual distinction between directive regions and the main image through more explicit visual markers.

D.2 Training Details

D.2.1 Training Setup

Goal2Pixel is initialized from InternVL3 [16], whose language backbone is built upon Qwen2.5 [3]. The training corpus contains 2.64M samples constructed from R2R-CE [32] and RxR-CE [33]: specifically, 653,820 samples from R2R-CE, 664,502 from RxR-CE English-US, and 1,331,460 from RxR-CE English-IN. Images are padded to a square shape and resized to 448×448 following the InternVL preprocessing pipeline. The maximum number of history images is set to 8.

The LLM backbone and projection layers are fully updated while the vision encoder is frozen. The model is optimized using AdamW [43] with a learning rate of 2×10^{-5}, a cosine decay schedule [42], and a 3% linear warm-up. Training uses bfloat16 precision, FlashAttention [20], and DeepSpeed ZeRO-3 [52, 56] for memory-efficient distributed training. The 2B model is trained on 2 H100 GPUs for 80 GPU hours (gradient accumulation 4, max sequence length 30K tokens, 15,500 optimization steps). The 7B model is trained on 4 H100 GPUs for 176 GPU hours (gradient accumulation 6, max sequence length 16K tokens, 14,300 optimization steps).

D.2.2 Training Prompt Template

The training prompt is shown below. The agent receives the current egocentric image and ViKeyMem history images, and is asked to predict a single pixel coordinate as the navigation target. Special coordinates are reserved for turning and stopping decisions.

```
You are an indoor navigation agent following a language instruction.
Given the current egocentric image, predict the next navigation target as a
2D pixel coordinate in the image. Output exactly one coordinate pair (X, Y),
where X and Y are integers in the range [000, 999]. The coordinate represents
```

```
the normalized horizontal and vertical position in the image.

Use the following exact coordinates for special actions:
  TURN_LEFT  = (000, 500)
  TURN_RIGHT = (999, 500)
  STOP       = (500, 999)

For forward navigation, output the pixel that best indicates the next
navigable direction or goal in the current view. Prefer predicting the
farthest navigable pixel in the visible image region whenever a navigable
target is visible. If no clear forward navigable target is visible, output a
pixel on the left or right auxiliary directive region to indicate turning.
If the agent is within 1.5 meters of the destination, output the STOP
coordinate on the bottom auxiliary directive region.

Return only the coordinate pair in the format XXX, YYY.

Instruction: <Current Episode Instruction>

Input:
  - Current-step egocentric RGB image: <image>, where the left/right gray
    padding indicates left/right turns, and the bottom padding indicates stop.
  - History images: Previous <N> egocentric RGB keyframes from the past
    trajectory, with blue trajectory dots showing the movement trajectory:
    <image> (latest), <image>, ..., <image> (oldest).

Output: A pixel coordinate pair in the format XXX, YYY.
```

D.2.3 Simulation Setup

The agent operates in indoor 3D environments and receives an egocentric RGB observation at each timestep. Following the common VLN-CE protocol, the RGB camera has a 90° horizontal field of view and produces observations with a resolution of 256×256 pixels. The simulator uses a discrete low-level action space: move forward by 25 cm, turn left by 15°, turn right by 15°, or stop. Although Goal2Pixel

predicts a pixel-grounded target rather than directly predicting one of these actions, all predictions are ultimately converted into executable commands via the built-in *ShortestPathFollower*.

To avoid overly long execution for a single model prediction, the number of low-level actions executed per VLM decision is capped at $t = 5$. A simple oscillation fallback is also employed: if the agent exhibits continuous left–right oscillation for 20 consecutive low-level actions, the current decision is overridden with a fixed forward-biased pixel prediction at $(500, 970)$ to encourage forward motion.

Evaluation is performed on the unseen validation splits of R2R-CE (1,836 episodes) and RxR-CE (3,669 episodes). Standard VLN-CE metrics are reported: Navigation Error (NE), Success Rate (SR), Oracle Success Rate (OS), Success-weighted Path Length (SPL), and normalized Dynamic Time Warping (nDTW).

D.2.4 Real-World Setup

Goal2Pixel is deployed on a Mecanum-wheeled robot [92] equipped with a Livox Mid-360 LiDAR and a Ricoh Theta Z1 panoramic camera. The model runs onboard on a laptop with an AMD Ryzen 9 8940HX CPU and an NVIDIA RTX-5060 Laptop GPU (8 GB VRAM).

The egocentric RGB observation is constructed by cropping a 90° field-of-view image from the panoramic camera. The LiDAR point cloud is projected into the camera view to obtain a depth map; since the projected depth can be sparse, interpolation is applied to fill missing values. A short ground-plane prior is added before depth projection to alleviate the near-field blind region. Due to the limited GPU memory, the maximum number of ViKeyMem history images is reduced to 3 during real-world deployment.

D.3 Why ViKeyMem Sparsity Improves Reasoning

The main text (Chapter 6) notes that ViKeyMem's sparsity improves not only computational efficiency but also the quality of the VLM's reasoning. This section elaborates on the underlying mechanisms.

When the history contains many similar frames—for example, 15 consecutive frames from a long corridor—the VLM's attention is diluted across redundant information, and the distinctive features of each frame become harder to identify. This attention dilution effect is particularly problematic for pretrained VLMs, which have limited context windows and may struggle to identify the most relevant frames among many similar ones. By retaining only frames where the visible waypoint set changes substantially, ViKeyMem ensures that each history image provides unique and decision-relevant information.

The trajectory overlay further enhances information content. By drawing blue dots along the agent's past trajectory, each keyframe encodes not only visual appearance but also the direction from which the agent arrived and the path it took. This motion information helps the VLM understand spatial relationships between distant observations, which is particularly important when the agent revisits a region and needs to connect the current view with previously stored keyframes.

D.4 Training Objective Details

Full fine-tuning: The language backbone is fully fine-tuned (rather than using LoRA) because pixel coordinate prediction requires the model to learn a new output format—three-digit numbers separated by commas—that is not present in its pretraining distribution. LoRA adapters alone are insufficient for learning this fundamentally new output modality. The vision encoder is frozen because its pretrained visual features are already well-suited for navigation, and fine-tuning on the relatively small VLN-CE dataset risks overfitting to MP3D visual styles.

Loss weight selection: The numeric loss weight $\lambda_{\text{num}} = 0.3$ provides a strong coordinate-level signal without overwhelming the primary cross-entropy loss. The angular loss weight $\lambda_{\text{ang}} = 0.03$ provides a gentle directional regularizer that encourages the model to predict pixels in the correct direction even when the exact coordinate is wrong. Larger values of λ_{ang} were found to degrade performance by over-constraining the prediction.

Limitation of soft coordinate approximation: One limitation of the soft coordinate formulation is that the expected digit value may be less reliable under a *multi-modal categorical distribution over digit tokens*. For example, if a digit position

assigns high probabilities to two distant digits, the expectation may produce an intermediate value that does not correspond to either plausible discrete prediction. This is a general limitation of expectation-based continuous relaxation for discrete token generation. However, the auxiliary losses are used only as regularization terms, while the primary supervision remains the token-level cross-entropy loss. The cross-entropy objective directly encourages the model to concentrate probability mass on the ground-truth digit tokens, which helps sharpen the predicted token distribution and mitigate multi-modality during training. Therefore, the soft coordinate losses complement, rather than replace, the standard text-generation objective.

D.5 Component Ablation

Beyond the history representation and output paradigm ablations in the main text, additional ablations examine the visual semantic embeddings and coordinate-aware losses. Removing the angular loss reduces SR from 54.1 to 53.4 on R2R-CE and from 43.8 to 42.3 on RxR-CE. Removing the numeric loss reduces SR to 52.3 on R2R-CE and 42.9 on RxR-CE. Removing the directive embedding slightly improves R2R-CE SR to 54.2 but degrades RxR-CE to 42.2. Removing the trajectory embedding reduces SR to 53.2 on R2R-CE and 43.3 on RxR-CE.

These results confirm that coordinate-aware auxiliary losses strengthen pixel grounding by helping the model capture the geometric structure of coordinate outputs, and that distinguishing directive regions and trajectory overlays from regular RGB tokens facilitates adaptation to navigable pixel grounding. An ablation of the maximum number of low-level execution steps per VLM prediction is provided in Section D.7.

D.6 ViKeyMem Algorithm and Visualization

Algorithm 2 summarizes the keyframe selection procedure of ViKeyMem. The three selection criteria are defined in the main text (Chapter 6); the algorithm below provides the complete procedural description. Because the selected keyframes are sparse, the agent's past trajectory is overlaid onto each keyframe by drawing blue dots along the

trajectory, making the motion connection between distant observations explicit. In practice, ViKeyMem typically retains only 4–5 keyframes per 100 timesteps.

Algorithm 2 ViKeyMem History Construction

Require: Trajectory $\tau = \{O_1, \ldots, O_T\}$
Ensure: History memory O_{ViKeyMem}
Initialize $K \leftarrow \{O_1\}$ and set $last\_key \leftarrow O_1$
for $t = 2$ to T **do**
    if O_t is not visible from $last\_key$ **then**
        if O_t contains at least one visible future waypoint **then**
            if the next two distinct future waypoints lie within 45° forward FOV **then**
                Add O_t to K
                $last\_key \leftarrow O_t$
            end if
        end if
    end if
end for
for each selected keyframe $O_k \in K$ **do**
    Overlay the past trajectory onto O_k
end for
return selected keyframes with trajectory overlays

D.7 Low-Level Execution Steps Ablation

Table D.1 studies how many low-level actions the agent should execute after each VLM prediction. When t is small, the agent queries the VLM more frequently but each prediction only induces very short-horizon motion. Setting $t = 1$ requires 31.90 VLM calls per episode on R2R-CE but only achieves 36.1% SR.

Increasing t from 1 to 5 substantially improves both accuracy and efficiency: on R2R-CE, SR increases from 36.1% to 54.1% while calls decrease from 31.90 to 7.75. These results indicate that a predicted pixel can effectively serve as a short-horizon spatial goal for multiple low-level actions.

However, further increasing t does not continue to improve performance. Although $t = 10$ reduces calls to 4.26, its SR drops to 53.6%. Under Unlimited execution, SR decreases to 40.5% despite using only 2.54 calls per episode. This suggests that executing for too long based on a single pixel prediction reduces the agent's ability to

Table D.1: **Ablation of the maximum number of low-level execution steps per VLM prediction.** Avg. # Calls = average VLM invocations per episode. Best in **bold**, second-best underlined.

Max Steps t	R2R-CE					RxR-CE				
	NE↓	OS↑	SR↑	SPL↑	Calls↓	NE↓	SR↑	SPL↑	nDTW↑	Calls↓
$t = 1$	5.70	38.0	36.1	35.7	31.90	8.95	25.0	24.2	49.1	33.48
$t = 2$	5.38	45.3	42.4	41.7	14.82	8.32	30.1	28.9	53.4	16.92
$t = 5$	**4.85**	**59.9**	**54.1**	**52.5**	7.75	7.50	**43.8**	**40.4**	**61.1**	11.76
$t = 10$	4.92	59.6	53.6	51.7	4.26	**7.48**	42.4	39.1	60.0	6.59
Unlimited	5.46	52.6	40.5	38.5	**2.54**	8.13	34.3	30.8	54.9	**3.87**

replan from updated visual observations. Overall, $t = 5$ provides the best trade-off between navigation accuracy and inference efficiency on both benchmarks.

D.8 Training Efficiency

As shown in Figure D.1, Goal2Pixel achieves competitive R2R-CE performance while requiring substantially less training cost than recent VLN-CE methods. Goal2Pixel-2B reaches comparable success rates to much larger or more expensive models, but only requires 70 H100 GPU hours for training. This computational efficiency mainly comes from the compact history representation introduced by ViKeyMem: instead of encoding dense observation histories or uniformly sampled frames, ViKeyMem selects a small number of visibility-aware keyframes that capture informative changes along the trajectory. As a result, most training samples require only a limited number of historical images, which reduces the visual-token budget and makes the overall fine-tuning process more efficient.

D.9 Ground-Truth Pixel Distribution

The spatial distribution of ground-truth pixels is analyzed in the main text (Chapter 6, Pixel Distribution Analysis). Here we provide additional discussion on the implications of this distribution for model training and generalization.

The highest-density regions appear near the lower center and along several horizontal bands, corresponding to common forward navigation targets on the floor. This

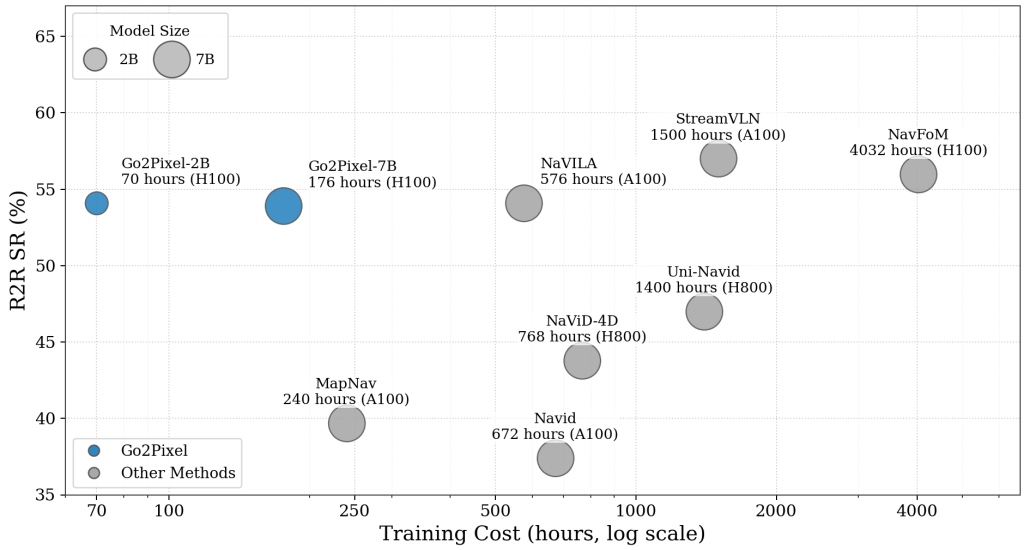

Figure D.1: **Comparison of R2R-CE success rate and training cost across recent VLN-CE methods.** The marker size indicates the model scale, and the x-axis is shown in log scale. Goal2Pixel achieves competitive navigation performance with substantially lower training cost than prior methods, largely enabled by ViKeyMem's compact history representation.

indicates that although the pixel output space is large, most forward supervision is concentrated in a relatively small subset of navigable image regions. Scattered low-frequency pixels outside the dominant pattern arise from more diverse geometric situations such as stairs, height changes, narrow doorways, or partially visible waypoints.

This analysis reveals an important property of pixel-space supervision constructed from Habitat-based VLN datasets. Although Goal2Pixel defines a large spatial action space, the empirical supervision is highly imbalanced: a small number of recurring pixels receive many labels, while most pixels are rarely selected. This concentration may make learning easier for common forward-navigation behaviors, but it can also bias the model toward frequent geometric patterns and provide limited supervision for rare pixel locations. More broadly, this suggests that a more compact variant could potentially restrict prediction to a set of representative pixel candidates, thereby reducing the effective output space while preserving the main advantage of spatial goal grounding.

D.10 Real-World Qualitative Results

Figure D.2 shows five representative real-world navigation trials. Each image corresponds to one VLM decision step, with predicted target pixels overlaid on the egocentric observations.

These examples demonstrate that Goal2Pixel can ground language instructions to meaningful pixel targets in real egocentric observations. The predicted pixels often correspond to instruction-relevant landmarks, and the robot converts these pixel predictions into executable motion through the local navigation controller. Since the model outputs a 2D target pixel as an intermediate spatial representation, this pixel-space output decouples high-level VLM reasoning from the robot's particular action space, making the same model output potentially reusable across different robot platforms.

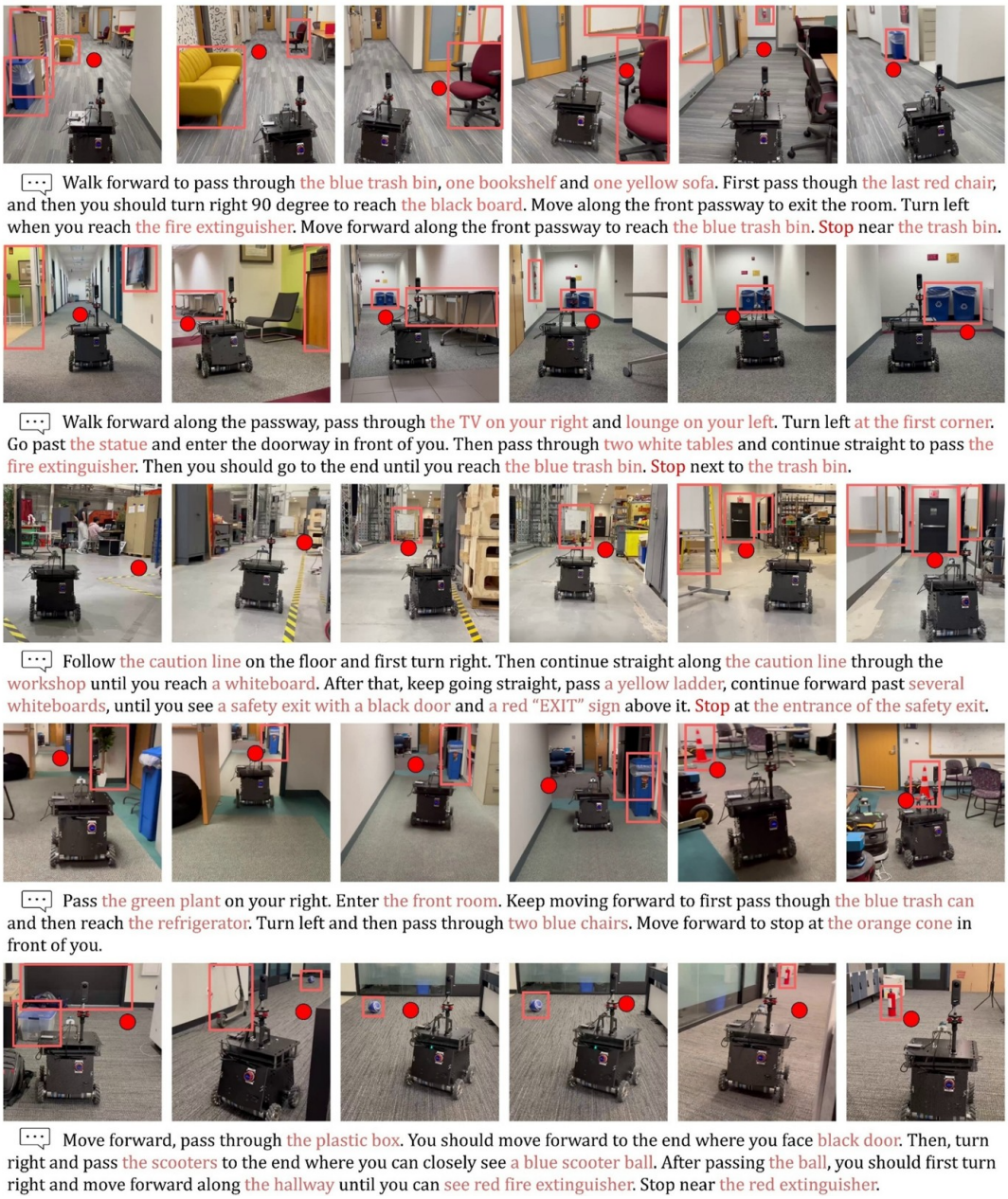

Figure D.2: **Real-world Goal2Pixel navigation.** Five representative trials following language instructions. Each image corresponds to one VLM decision step. Predicted pixels often correspond to instruction-relevant landmarks such as trash bins, sofas, chairs, doors, and hallway regions.

Bibliography

[1] Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2.1, 2.5

[2] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 2.2

[3] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 2.3, D.2.1

[4] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects. In *arXiv:2006.13171*, 2020. 2.1

[5] Wenzhe Cai, Siyuan Huang, Guangran Cheng, Yuxing Long, Peng Gao, Changyin Sun, and Hao Dong. Bridging zero-shot object navigation and foundation models through pixel-guided navigation skill. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5228–5234. IEEE, 2024. 5.1

[6] Chao Cao, Hongbiao Zhu, Howie Choset, and Ji Zhang. Tare: A hierarchical framework for efficiently exploring complex 3d environments. In *Robotics: Science and Systems*, volume 5, page 2, 2021. 2.1, 4.2.2

[7] Chao Cao, Hongbiao Zhu, Fan Yang, Yukun Xia, Howie Choset, Jean Oh, and Ji Zhang. Autonomous exploration development environment and the planning algorithms. In *2022 International Conference on Robotics and Automation*

(ICRA), pages 8921–8928. IEEE, 2022. 2.7, 4.2.3

[8] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 5.3

[9] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. 3.5, 4.4, 6.3

[10] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020. 2.1, 2.5, 3.1, 5.1

[11] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465, 2024. 1.1, 3.1, 5.1

[12] Jiaqi Chen, Bingqian Lin, Xinmin Liu, Lin Ma, Xiaodan Liang, and Kwan-Yee K Wong. Affordances-oriented planning using foundation models for continuous vision-language navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23568–23576, 2025. 2.2, 6.1

[13] Junting Chen, Guohao Li, Suryansh Kumar, Bernard Ghanem, and Fisher Yu. How to not train your dragon: Training-free embodied object goal navigation with semantic frontiers. *arXiv preprint arXiv:2305.16925*, 2023. 2.1

[14] Kehan Chen, Dong An, Yan Huang, Rongtao Xu, Yifei Su, Yonggen Ling, Ian Reid, and Liang Wang. Constraint-aware zero-shot vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 2.2, 6.1

[15] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198, 2024. 2.3

[16] Zhe Chen, Weiyun Wang, Yue Cao, Yang Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025. 2.3, 2.3, 5.2.4, 6.5, D.2.1

[17] An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024. 2.2, 6.3.2, 6.1

[18] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. Yolo-world: Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16901–16911, 2024. 4.2.1, 4.3

[19] Ronghao Dang, Liuyi Wang, Zongtao He, Shuai Su, Jiagui Tang, Chengju Liu, and Qijun Chen. Search for or navigate to? dual adaptive thinking for object navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8250–8259, 2023. 2.4

[20] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022. D.2.1

[21] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. Robothor: An open simulation-to-real embodied ai platform. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3.5

[22] Hongyu Ding, Ziming Xu, Yudong Fang, You Wu, Zixuan Chen, Jieqi Shi, Jing Huo, Yifan Zhang, and Yang Gao. Lavira: Language-vision-robot actions translation for zero-shot vision language navigation in continuous environments. *arXiv preprint arXiv:2510.19655*, 2025. 2.2, 6.1

[23] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181, 2023. 2.1

[24] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Devendra Singh Chaplot. Navigating to objects in the real world. *Science Robotics*, 8 (79):eadf6991, 2023. 2.1

[25] Tianjun Gu, Linfeng Li, Xuhong Wang, Chenghua Gong, Jingyu Gong, Zhizhong Zhang, Yuan Xie, Lizhuang Ma, and Xin Tan. Doraemon: Decentralized ontology-aware reliable agent with enhanced memory oriented navigation. *arXiv preprint arXiv:2505.21969*, 2025. 2.1, 3.1

[26] Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation. In *Proceedings of the IEEE/CVF conference on computer vision*

and pattern recognition, pages 15439–15449, 2022. 2.2, 6.1

[27] Yicong Hong, Yang Zhou, Ruiyi Zhang, Franck Dernoncourt, Trung Bui, Stephen Gould, and Hao Tan. Learning navigational visual representations with semantic map supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3055–3067, 2023. 2.2, 6.1

[28] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. 2022. 2.1, 2.5, 3.3.1, 4.2.1

[29] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 3.3.1, 3.4

[30] Jacob Krantz and Stefan Lee. Sim-2-sim transfer for vision-and-language navigation in continuous environments. In *European conference on computer vision*, pages 588–603. Springer, 2022. 2.7

[31] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 104–120. Springer, 2020. 2.2

[32] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pages 104–120. Springer, 2020. 6.3, D.2.1

[33] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4392–4412, 2020. 6.3, D.2.1

[34] Yuxuan Kuang, Hai Lin, and Meng Jiang. Openfmnav: Towards open-set zero-shot object navigation via vision-language foundation models. *arXiv preprint arXiv:2402.10670*, 2024. 2.1, 2.4, 3.1, 4.1, 5.1

[35] Boqi Li, Siyuan Li, Weiyi Wang, Anran Li, Zhong Cao, and Henry X Liu. Boosting zero-shot vln via abstract obstacle map-based waypoint prediction with topograph-and-visitinfo-aware prompting. *arXiv preprint arXiv:2509.20499*, 2025. 2.2

[36] LinFeng Li, Jian Zhao, Yuan Xie, Xin Tan, and Xuelong Li. Compassnav: Steering from path imitation to decision understanding in navigation. *arXiv*

preprint arXiv:2510.10154, 2025. 2.6, 5.1, 5.1

[37] Haoran Liu, Weikang Wan, Xiqian Yu, Minghan Li, Jiazhao Zhang, Bo Zhao, Zhibo Chen, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Navid-4d: Unleashing spatial intelligence in egocentric rgb-d videos for vision-and-language navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10607–10615. IEEE, 2025. 6.1

[38] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023. 2.3

[39] Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. In *8th Annual Conference on Robot Learning*. 3.1, 4.1, 4.2, 6.1

[40] Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*, 2024. 2.2

[41] Joel Loo, Zhanxin Wu, and David Hsu. Open scene graphs for open-world object-goal navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*. 2.1, 2.5

[42] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. D.2.1

[43] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. D.2.1

[44] Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. *Advances in Neural Information Processing Systems*, 35:32340–32352, 2022. 3.1, 5.1

[45] Arsalan Mousavian, Alexander Toshev, Marek Fišer, Jana Košecká, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8846–8852. IEEE, 2019. 2.1

[46] Dujun Nie, Xianda Guo, Yiqun Duan, Ruijun Zhang, and Long Chen. Wmnav: Integrating vision-language models into world models for object goal navigation, 2025. URL https://arxiv.org/abs/2503.02247. 5.1

[47] Cheng Peng, Zhenzhe Zhang, Cheng Chi, Xiaobao Wei, Yanhao Zhang, Heng Wang, Pengwei Wang, Zhongyuan Wang, Jing Liu, and Shanghang Zhang. Pigeon: Vlm-driven object navigation via points of interest selection. *arXiv preprint arXiv:2511.13207*, 2025. 2.6, 5.1

[48] Xavi Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Ruslan Partsey, Jimmy Yang, Ruta Desai, Alexander William Clegg, Michal Hlavac, Tiffany Min, Theo Gervet, Vladimír Vondruš, Vincent-Pierre Berges, John Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023. 3.5, 4.4, 5.3

[49] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, et al. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*, 2023. 5.2.2

[50] Zhangyang Qi, Zhixiong Zhang, Ye Fang, Jiaqi Wang, and Hengshuang Zhao. Gpt4scene: Understand 3d scenes from videos with vision-language models. *arXiv preprint arXiv:2501.01428*, 2025. 1.1, 3.1

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 1.1

[52] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pages 1–16. IEEE, 2020. D.2.1

[53] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18890–18900, 2022. 2.1, 3.1, 5.1

[54] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5173–5183, 2022. 2.1, 5.2.2, 5.2.4, 5.1, C.5

[55] Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. Pirlnav: Pretraining with imitation and rl finetuning for objectnav. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17896–17906, 2023. 2.1

[56] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over

100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506, 2020. D.2.1

[57] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL https://arxiv.org/abs/2408.00714. 4.2.1, 4.3

[58] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 2.6

[59] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019. 2.1

[60] James A Sethian. A fast marching level set method for monotonically advancing fronts. *proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996. 2.5

[61] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1.1, 3.4, 4.3

[62] Yao-Hung Hubert Tsai, Vansh Dhar, Jialu Li, Bowen Zhang, and Jian Zhang. Multimodal large language model for visual navigation, 2023. URL https://arxiv.org/abs/2310.08669. 5.1

[63] Ao Wang, Lihao Liu, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yoloe: Real-time seeing anything. *arXiv preprint arXiv:2503.07465*, 2025. 4.2.1, 4.3

[64] Hanqing Wang, Wei Liang, Luc Van Gool, and Wenguan Wang. Dreamwalker: Mental planning for continuous vision-language navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10873–10883, 2023. 2.2

[65] Shuo Wang, Yongcai Wang, Zhaoxin Fan, Yucheng Wang, Maiyue Chen, Kaihui Wang, Zhizhong Su, Wanting Li, Xudong Cai, Yeying Jin, et al. Monodream: Monocular vision-language navigation with panoramic dreaming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 10074–10082,

2026. 6.3.2

[66] Zihan Wang and Gim Hee Lee. g3d-lf: Generalizable 3d-language feature fields for embodied tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14191–14202, 2025. 6.1

[67] Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. Sim-to-real transfer via 3d feature fields for vision-and-language navigation. *arXiv preprint arXiv:2406.09798*, 2024. 2.7, 6.1

[68] Zixuan Wang, Huang Fang, Shaoan Wang, Yuanfei Luo, Heng Dong, Wei Li, and Yiming Gan. Hydra-nav: Object navigation via adaptive dual-process reasoning. *arXiv preprint arXiv:2602.09972*, 2026. 5.1, 5.1

[69] Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, et al. Ground slow, move fast: A dual-system foundation model for generalizable vision-and-language navigation. *arXiv preprint arXiv:2512.08186*, 2025. 2.2, 6.3.2

[70] Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, et al. Streamvln: Streaming vision-and-language navigation via slowfast context modeling. *arXiv preprint arXiv:2507.05240*, 2025. 2.2, 6.1

[71] Congcong Wen, Yisiyuan Huang, Hao Huang, Yanjia Huang, Shuaihang Yuan, Yu Hao, Hui Lin, Yu-Shen Liu, and Yi Fang. Zero-shot object navigation with vision-language models reasoning. In *International Conference on Pattern Recognition*, pages 389–404. Springer, 2025. 2.4

[72] Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024. 2.5, 3.3.1, 4.2.1

[73] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019. 2.1, 5.1

[74] Pengying Wu, Yao Mu, Bingxian Wu, Yi Hou, Ji Ma, Shanghang Zhang, and Chang Liu. Voronav: Voronoi-based zero-shot object navigation with large language model. *arXiv preprint arXiv:2401.02695*, 2024. 2.1, 2.5, 3.1, 4.1

[75] Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. Gpt-4v (ision) is a human-aligned evaluator for text-to-3d generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22227–22238, 2024. 1.1, 2.3

[76] Xinda Xue, Junjun Hu, Minghua Luo, Xie Shichao, Jintao Chen, Zixun Xie, Quan Kuichen, Guo Wei, Mu Xu, and Zedong Chu. Omninav: A unified framework for prospective exploration and visual-language navigation. *arXiv preprint arXiv:2509.25687*, 2025. 5.1, 5.1

[77] Xinda Xue, Junjun Hu, Minghua Luo, Shichao Xie, Jintao Chen, Zixun Xie, Kuichen Quan, Wei Guo, Mu Xu, and Zedong Chu. Omninav: A unified framework for prospective exploration and visual-language navigation. *arXiv preprint arXiv:2509.25687*, 2025. 2.2, 6.3.2

[78] Karmesh Yadav, Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Oleksandr Maksymets, Rishabh Jain, Ram Ramrakhya, Angel X Chang, Alexander Clegg, Manolis Savva, Eric Undersander, Devendra Singh Chaplot, and Dhruv Batra. Habitat challenge 2022. https://aihabitat.org/challenge/2022/, 2022. 3.5, 4.4, 5.3

[79] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997. 2.1, 3.3.1

[80] Fan Yang, Dung-Han Lee, John Keller, and Sebastian Scherer. Graph-based topological exploration planning in large-scale 3d environments. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 12730–12736. IEEE, 2021. 2.1, 2.5, 3.3.1

[81] Xuan Yao, Junyu Gao, and Changsheng Xu. Navmorph: A self-evolving world model for vision-and-language navigation in continuous environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5536–5546, 2025. 2.2, 2.6, 6.1

[82] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16117–16126, 2021. 2.1

[83] Sriram Yenamandra, Arun Ramachandran, Mukul Khanna, Karmesh Yadav, Jay Vakil, Andrew Melnik, Michael Büttner, Leon Harz, Lyon Brown, Gora Chand Nandi, et al. Towards open-world mobile manipulation in homes: Lessons from the neurips 2023 homerobot open vocabulary mobile manipulation challenge. *arXiv preprint arXiv:2407.06939*, 2024. 2.7

[84] Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. *Advances in Neural Information Processing Systems*, 37:5285–5307, 2024. 2.1, 2.4, 2.5, 3.1, 3.3.2, 3.1, 4.1, 5.1

[85] Hang Yin, Xiuwei Xu, Linqing Zhao, Ziwei Wang, Jie Zhou, and Jiwen Lu.

Unigoal: Towards universal zero-shot goal-oriented navigation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19057–19066, 2025. 5.1

[86] Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–48. IEEE, 2024. 2.1, 2.4, 3.1, 3.1, 4.1, 4.2, 5.1

[87] Naoki Yokoyama, Ram Ramrakhya, Abhishek Das, Dhruv Batra, and Sehoon Ha. Hm3d-ovon: A dataset and benchmark for open-vocabulary object goal navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5543–5550. IEEE, 2024. 4.4

[88] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3mvn: Leveraging large language models for visual target navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 3554–3560. IEEE, October 2023. doi: 10.1109/iros55552.2023.10342512. URL http://dx.doi.org/10.1109/IROS55552.2023.10342512. 2.1, 2.4, 3.1, 3.1, 4.1

[89] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. Co-navgpt: Multi-robot cooperative visual semantic navigation using large language models. *arXiv preprint arXiv:2310.07937*, 2023. 2.6

[90] Zhuoyuan Yu, Yuxing Long, Zihan Yang, Chengyan Zeng, Hongwei Fan, Jiyao Zhang, and Hao Dong. Correctnav: Self-correction flywheel empowers vision-language-action navigation model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 18737–18745, 2026. 6.3.2

[91] Shuang Zeng, Dekang Qi, Xinyuan Chang, Feng Xiong, Shichao Xie, Xiaolong Wu, Shiyi Liang, Mu Xu, and Xing Wei. Janusvln: Decoupling semantics and spatiality with dual implicit memory for vision-language navigation. *arXiv preprint arXiv:2509.22548*, 2025. 2.2, 6.3.2, 6.5, 6.1

[92] Ji Zhang. Autonomy stack for mecanum wheel platform. https://github.com/jizhang-cmu/autonomy_stack_mecanum_wheel_platform, 2024. Accessed: 2025-04-29. 2.7, 3.4, 4.4, 5.3.4, 6.5, D.2.4

[93] Ji Zhang, Sanjiv Singh, et al. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014. 2.7

[94] Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*, 2024. 6.3.2, 6.1

[95] Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-navid: A video-

based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*, 2024. 2.1, 5.1, 5.1

[96] Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. *arXiv preprint arXiv:2402.15852*, 2024. 2.2, 6.3.2, 6.1

[97] Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation, 2024. URL https://arxiv.org/abs/2402.15852. 5.1, 5.1

[98] Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, et al. Embodied navigation foundation model. *arXiv preprint arXiv:2509.12129*, 2025. 2.2

[99] Liang Zhang, Leqi Wei, Peiyi Shen, Wei Wei, Guangming Zhu, and Juan Song. Semantic slam based on object detection and improved octomap. *IEEE Access*, 6:75545–75559, 2018. 2.1, 2.5

[100] Lingfeng Zhang, Qiang Zhang, Hao Wang, Erjia Xiao, Zixuan Jiang, Honglei Chen, and Renjing Xu. Trihelper: Zero-shot object navigation with dynamic assistance. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10035–10042. IEEE, 2024. 3.1, 4.1, 5.1

[101] Lingfeng Zhang, Xiaoshuai Hao, Qinwen Xu, Qiang Zhang, Xinyao Zhang, Pengwei Wang, Jing Zhang, Zhongyuan Wang, Shanghang Zhang, and Renjing Xu. Mapnav: A novel memory representation via annotated semantic maps for vlm-based vision-and-language navigation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13032–13056, 2025. 2.2, 6.1

[102] Mingjie Zhang, Yuheng Du, Chengkai Wu, Jinni Zhou, Zhenchao Qi, Jun Ma, and Boyu Zhou. Apexnav: An adaptive exploration strategy for zero-shot object navigation with target-centric semantic fusion. *arXiv preprint arXiv:2504.14478*, 2025. 2.4, 4.1, 5.1

[103] Siqi Zhang, Yanyuan Qiao, Qunbo Wang, Zike Yan, Qi Wu, Zhihua Wei, and Jing Liu. Cosmo: Combination of selective memorization for low-cost vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5511–5522, 2025. 2.2

[104] Zheyuan Zhang, Fengyuan Hu, Jayjun Lee, Freda Shi, Parisa Kordjamshidi, Joyce Chai, and Ziqiao Ma. Do vision-language models represent space and how? evaluating spatial frame of reference under ambiguities. *arXiv preprint arXiv:2410.17385*, 2024. 1.1, 2.1, 3.1, 5.1

[105] Qianfan Zhao, Lu Zhang, Bin He, and Zhiyong Liu. Semantic policy network for zero-shot object goal visual navigation. *IEEE Robotics and Automation Letters*, 2023. 2.1, 2.5

[106] Xiangyu Zhao, Yicheng Chen, Shilin Xu, Xiangtai Li, Xinjiang Wang, Yining Li, and Haian Huang. An open and comprehensive pipeline for unified object grounding and detection. *arXiv preprint arXiv:2401.02361*, 2024. 3.3.1, 3.4

[107] Duo Zheng, Shijia Huang, Lin Zhao, Yiwu Zhong, and Liwei Wang. Towards learning a generalist model for embodied navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13624–13634, 2024. 2.2, 2.6, 6.3.2

[108] Duo Zheng, Shijia Huang, Yanyang Li, and Liwei Wang. Efficient-vln: A training-efficient vision-language navigation model. *arXiv preprint arXiv:2512.10310*, 2025. 6.3.2, 6.1

[109] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation, 2023. URL https://arxiv.org/abs/2301.13166. 2.1, 2.4, 3.1, 4.1, 5.1

[110] Zibo Zhou, Yue Hu, Lingkai Zhang, Zonglin Li, and Siheng Chen. Beliefmapnav: 3d voxel-based belief map for zero-shot object navigation. *arXiv preprint arXiv:2506.06487*, 2025. 5.1

[111] Haokun Zhu, Zongtai Li, Zhixuan Liu, Wenshan Wang, Ji Zhang, Jonathan Francis, and Jean Oh. Strive: Structured representation integrating vlm reasoning for efficient object navigation. *arXiv preprint arXiv:2505.06729*, 2025. 5.1

[112] Haokun Zhu, Zongtai Li, Zihan Liu, Kevin Guo, Zhengzhi Lin, Yuxin Cai, Guofei Chen, Chen Lv, Wenshan Wang, Jean Oh, et al. Sysnav: Multi-level systematic cooperation enables real-world, cross-embodiment object navigation. *arXiv preprint arXiv:2603.06914*, 2026. 5.1

[113] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017. 2.1

[114] Ziyu Zhu, Xilin Wang, Yixuan Li, Zhuofan Zhang, Xiaojian Ma, Yixin Chen, Baoxiong Jia, Wei Liang, Qian Yu, Zhidong Deng, et al. Move to understand a 3d scene: Bridging visual grounding and exploration for efficient and versatile embodied navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8120–8132, 2025. 4.1