Foundation Models for Embodied Navigation: A Survey

Longxi Gao[1*], **Weikai Xie**[1*], **Haoze Qian**[1], **Rongjie Yi**[1], **Shihe Wang**[1], **Jiaye Song**[1], **Dongqi Cai**[2], **Jinliang Yuan**[3], **Yunhao Liu**[3], **Xuanzhe Liu**[4], **Shangguang Wang**[1] **and Mengwei Xu**[1†]
[1]Beijing University of Posts and Telecommunications, [2]Nanjing University, [3]Tsinghua University, [4]Peking University

Abstract: The remarkable advancements of embodied navigation, the process of physically situated agents perceiving and reasoning through egocentric observations to reach target locations, have recently been reshaped by the emergence of foundation models. Departing from traditional, task-specific policies trained from scratch on limited datasets, this new paradigm leverages the reasoning and multimodal capabilities of Large Language Models (LLMs), Vision-Language Models (VLMs), and Video Generation Models (VGMs), achieving superior generalization and flexible decision-making in unseen environments. For the first time, this survey systematically reviews the landscape of foundation models for embodied navigation, focusing on systems where these models play a central role in perception, long-horizon memory management, and action generation. The survey aims to categorize and interpret existing embodied navigation research through the lens of design paradigms, data sources, and training strategies. Through this analysis, we offer a synthesized outlook on the evolution of navigation brains, highlight the bottlenecks of dataset bias, and contribute guidance for future research, hoping to bring the field closer to robust, general-purpose embodied intelligence.
Project Page: https://MEmbodied.github.io/embodied-navigation-survey

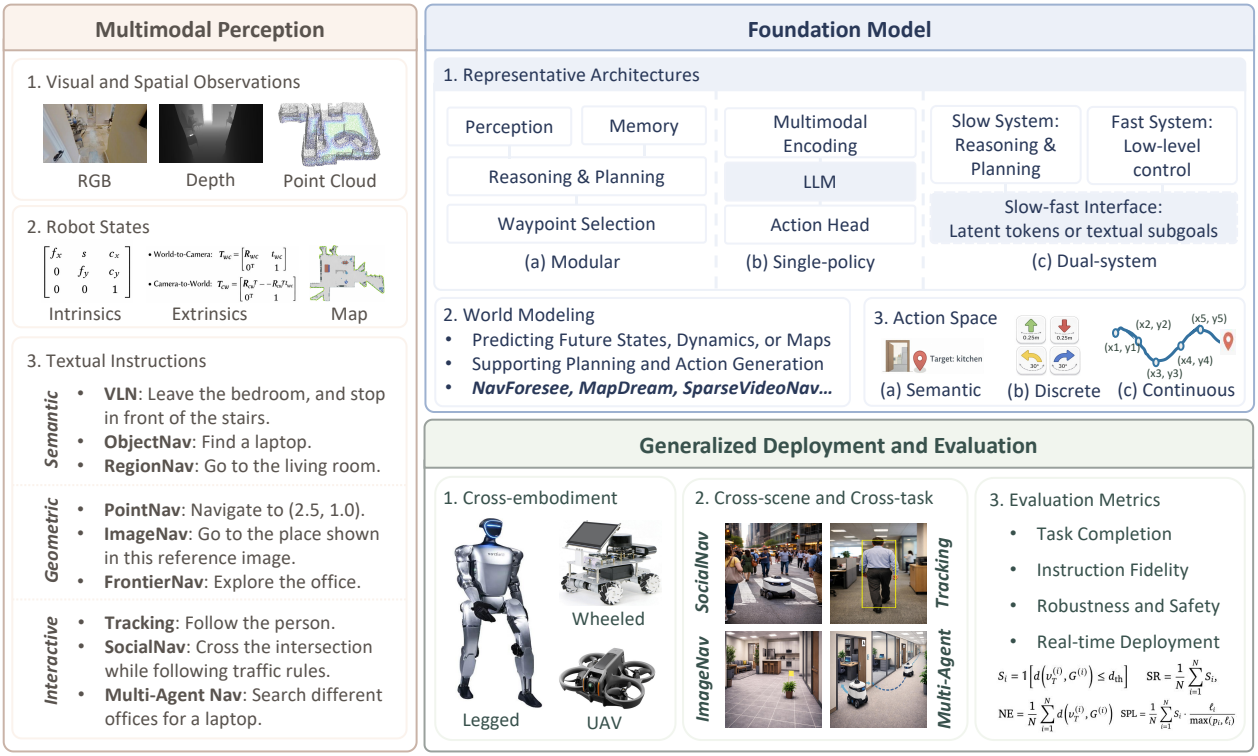

Figure 1 | Embodied navigation systems allow robots autonomously navigate to desired physical locations by understanding task instructions and comprehending environmental information. These systems are typically supported by architectural designs that integrate perception, memory, reasoning, and control, and are deployed across diverse embodiments, scenes, and task settings, with performance evaluated along multiple dimensions.

*Equal contribution: Longxi Gao {glx@bupt.edu.cn}, Weikai Xie {weikaixie@bupt.edu.cn}.
†Corresponding author: Mengwei Xu {mwx@bupt.edu.cn}.

Table of Contents

1. Introduction

Embodied navigation, the process by which an autonomous agent moves through a physical environment to reach a designated target while perceiving its surroundings through on-board sensors, is a fundamental building block of physical intelligence. Embodied navigation enables agents to transition from digital environments to the real world, facilitating critical applications in daily life and industry. These range from domestic service robots assisting in households and logistics robots managing warehouse inventories, to unmanned aerial vehicles (UAVs) performing search-and-rescue missions in hazardous terrains. Mastering it is essential for achieving truly autonomous systems that can interact intelligently and safely with human-centric spaces.

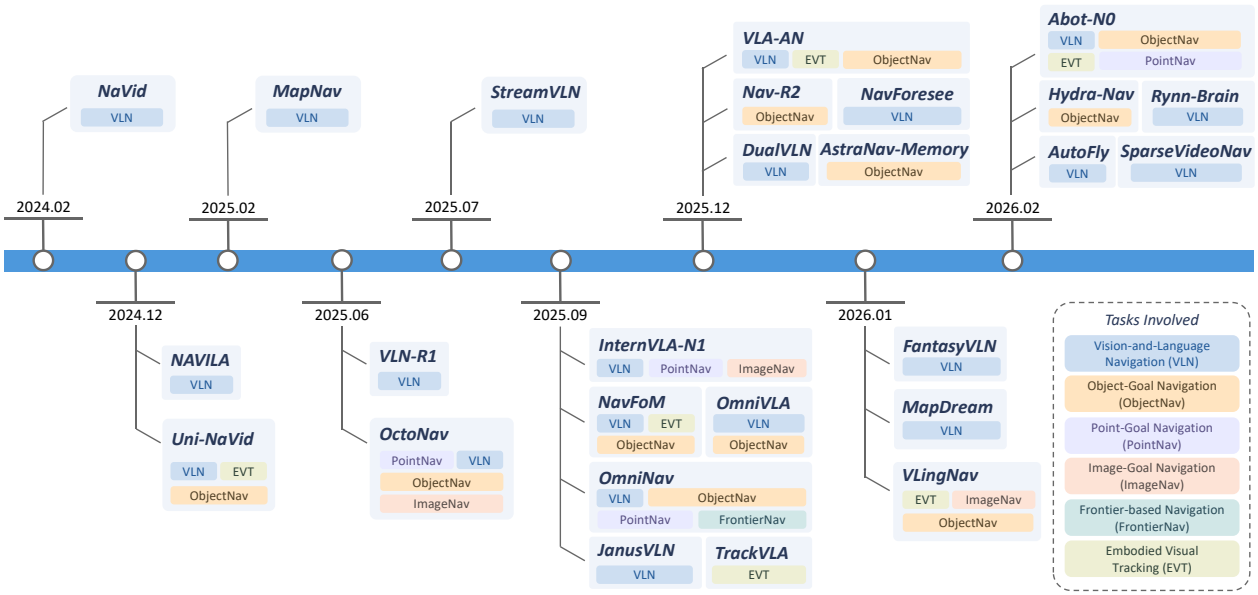

Figure 2 | Timeline of representative foundation models for embodied navigation and their associated tasks.

Despite decades of intensive research [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11], embodied navigation remains an unsolved and challenging task due to several inherent complexities. First, it demands a seamless integration of "low-level" reactive control, such as immediate obstacle avoidance and motion stability, with "high-level" cognitive reasoning, such as semantic path planning and mission decomposition. Second, navigation operates in vast, large-scale geometric spaces with nearly infinite variations, making it exceptionally difficult to collect the diverse, high-quality data necessary for robust training. Meanwhile, using simulation data struggles with the "sim-to-real" gap, where models trained in controlled simulations often fail to generalize to the noise, dynamic changes, and physical unpredictability of real-world environments. Third, real-world applications require a sophisticated balance of both temporal and spatial memory: an agent must not only remember where it has been (spatial) but also understand how the environment changes over time (temporal) to avoid getting stuck or repeating errors. Finally, the field is fragmented into an array of different sub-tasks, such as Point-Goal Navigation (PointNav), Object-Goal Navigation (ObjectNav), and Vision-and-Language Navigation (VLN), which has historically led to ad-hoc, specialized solutions that fail to generalize across different navigation challenges.

Recently, a transformative shift has occurred with the rise of foundation models for embodied navigation. These models are typically built atop large-scale pre-trained systems, i.e., Large Language Models (LLMs), Vision-Language Models (VLMs), and World Models (WMs), enabling agents to leverage vast prior knowledge to generalize across diverse scenarios [12, 13, 14, 15, 16]. The primary advantage of these foundation models lies in their ability to inject vast world knowledge and advanced logical reasoning into the navigation process, allowing agents to "think" through complex multi-step missions. Beyond simple zero-shot generalization to novel objects and environments, these models provide a shared semantic understanding that bridges the gap between various navigation sub-tasks. By moving away from brittle, hand-crafted pipelines toward these generalist architectures, embodied agents can achieve a level of autonomy and cognitive adaptability that was previously unattainable.

The Scope Compared with prior studies that focus on specific tasks or settings [17, 18, 19], this work provides

a more comprehensive and systematic overview of foundation models for embodied navigation. We explicitly define foundation models for embodied navigation as *those that leverage pre-trained language or multimodal foundation models as the core backbone of their decision-making pipeline*. Consequently, our scope is defined by two key exclusions: (1) we omit methods that treat LLMs/VLMs merely as external, modular tools for auxiliary tasks (such as captioning or object detection) without integrating them into the central control logic [20, 21, 22]; and (2) we exclude the field of autonomous driving, as it represents a mature, distinct domain with environmental constraints and safety requirements that differ significantly from the general embodied navigation tasks focused on in this survey [23, 24, 25].

The Outline We begin by establishing a clear taxonomy of navigation tasks, including semantic, geometric, and interactive navigation, and the diverse robotic embodiments they support, such as wheeled robots, legged robots, and UAVs (§ 2). We then dive into the core design paradigms, covering state representations, memory mechanisms, and decision-making architectures (§ 3). Finally, the survey explores critical practical aspects, including data collection pipelines, training strategies (§ 4), and the challenges of efficient deployment (§ 5) and standardized evaluation in real-world benchmarks (§ 6).

The Takeaways Through our in-depth investigation of existing literature, we obtain many valuable insights that could guide future research on this domain. Here, we present a few of them.

- Foundation models are shifting embodied navigation from task-specific policy learning toward a more generalist multimodal decision-making paradigm, where a shared backbone can support transfer across diverse navigation tasks, goals, and robotic embodiments.
- The central bottleneck is no longer semantic understanding alone, but the joint management of long-horizon memory, future anticipation, and real-time embodied execution under partial observability and dynamic environments.
- Strong benchmark performance should not be conflated with real-world capability, because current progress is still constrained by dataset biases, static-world assumptions, embodiment mismatch, and evaluation protocols that often under-emphasize safety, robustness, and deployment latency.
- Future advances will likely depend on joint progress in data engines, memory- and prediction-aware architectures, and hardware-aware deployment, so that embodied navigation systems can preserve the reasoning capacity of foundation models while remaining reliable and efficient on physical agents.

2. Problem Formulation and Taxonomy

In this section, we first formalize the embodied navigation problem and distinguish it from other embodied intelligence tasks, such as locomotion and manipulation. We then present taxonomies of navigation tasks and robotic embodiments, which serve as the basis for the design paradigms, training strategies, and evaluation settings discussed in the following sections.

2.1. Problem Formulation

Embodied navigation studies how an agent physically situated in an environment perceives its surroundings from an egocentric perspective, reasons about a navigation objective, makes sequential decisions, and executes actions to reach a target location. Specifically, at each timestep t the agent receives an observation o_t and selects an action a_t according to a policy $\pi(a_t \mid o_{\leq t}, g)$ to progress toward a specified goal g. After executing the selected action, the environment transitions from the current state s_t to a new state s_{t+1}. This interaction continues until the agent reaches the desired destination or a termination condition is triggered. The formulation of embodied navigation problems depends on several key factors, including how the navigation goal is specified, what observations are available to the agent, and what actions the agent can execute. Different forms of goal specification give rise to different navigation tasks, while the observation modalities and action spaces are closely related to the sensing configurations and motion capabilities of the robotic platform.

From a broader perspective, embodied navigation is closely related to other core sub-tasks of embodied intelligence, especially locomotion and manipulation [26, 27, 28, 29, 30], but it operates at a different level of abstraction. *Locomotion* focuses on how a robot moves through the physical world, emphasizing low-level motor control problems such as balance maintenance, gait generation, terrain adaptation, and collision-free motion. *Manipulation* focuses on how a robot interacts with objects, requiring fine-grained perception, grasp planning, contact reasoning, and force control. Compared with these two skills, embodied navigation is centered on goal-directed movement at the scene level: the agent must interpret the task objective, explore a partially

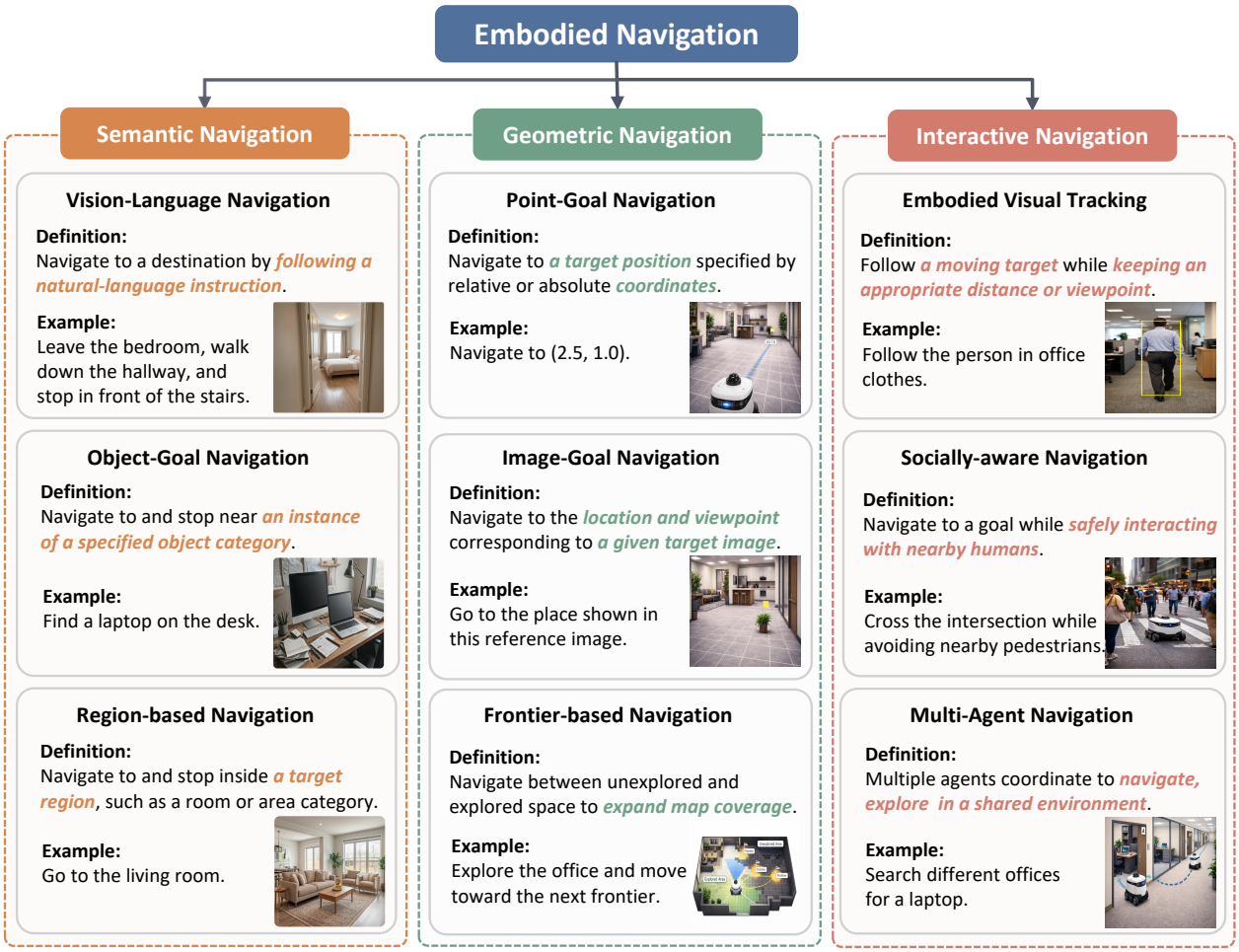

Figure 3 | Taxonomy of embodied navigation tasks organized by goal specification. **Semantic navigation** defines goals through high-level semantic concepts. **Geometric navigation** specifies goals through geometric or perceptual signals. **Interactive navigation** considers environments with dynamic entities or multiple agents.

observed environment, construct spatial representations, and plan long-horizon action sequences toward a potentially distant target. In real-world systems, these capabilities are tightly coupled rather than isolated, because effective navigation depends on reliable locomotion, and many practical tasks require the agent to first navigate to a target before performing manipulation. As a result, embodied navigation serves as the bridge between low-level physical execution and high-level goal-directed interaction in physical intelligence.

2.2. Task Taxonomy

Embodied navigation tasks can be categorized according to the type of navigation goal, which determines the form of instruction that an agent must interpret. Based on this criterion, navigation tasks can be broadly grouped into three categories: *semantic navigation*, *geometric navigation*, and *interactive navigation*. Figure 3 illustrates representative examples of each task category.

2.2.1. Semantic Navigation

Semantic navigation refers to tasks in which the navigation goal is specified through semantic information, including natural language instructions, object categories, and region categories. In these tasks, an agent must interpret high-level semantic concepts, ground them in the perceived environment, plan a feasible navigation trajectory, and ultimately reach the target goal.

- **Vision-and-Language Navigation (VLN)** [31] requires an agent to interpret natural language instructions (*e. g. "Walk through the hallway, go to the bedroom, and stop near the bed."*), perceive the environment from an egocentric perspective, and navigate to a destination described in the instruction. Early VLN settings

assume navigation on a predefined topological graph with accurate localization [31, 32]. Subsequent work extends this formulation to continuous environments, referred to as **VLN-CE** [33], where the agent navigates in a three-dimensional scene without relying on a predefined graph. Further extensions consider physically realistic deployment with robot-controlled locomotion, known as **VLN-PE** [34]. Success is typically defined as stopping within a predefined distance of the target location while following a trajectory consistent with the instruction.

- **Object-Goal Navigation (ObjectNav)** [35, 36, 37] requires an agent to navigate within an unseen environment to locate an instance of a specified object category and stop within a predefined distance of it (*e. g. "Find a laptop."*). Any instance of the specified category is considered a valid goal.
- **Region-based Navigation (RegionNav)** [35, 38] requires an agent to navigate to a region or area category (*e. g. "Go to the living room."*). The task is considered successful when the agent stops within the target region.

2.2.2. Geometric Navigation

Geometric navigation refers to tasks in which the navigation goal is specified through geometric or perceptual signals rather than semantic descriptions. In these tasks, the agent must reason about spatial relationships and navigate toward targets defined by coordinates or visual observations.

- **Point-Goal Navigation (PointNav)** [35, 39] specifies the navigation goal using metric coordinates relative to the current position of the agent. Given an instruction (*e. g. "Go to* (x, y)*."*), the agent must navigate to the specified point in the environment. Success is typically defined as stopping within a predefined distance of the target coordinates.
- **Image-Goal Navigation (ImageNav)** [40] specifies the navigation goal using a target image captured at the desired destination. Given an instruction (*e. g. "Go to the place shown in this reference image."*), the agent must navigate to the location and viewpoint from which the target image was originally taken. A related variant, **Instance-specific Image Goal Navigation (InstanceImageNav)** [41], requires the agent to navigate to the location where a particular object instance shown in the target image can be observed. Compared with ObjectNav, InstanceImageNav provides detailed visual context of the goal location and constrains the goal to a specific object instance rather than any instance of a category.
- **Frontier-based Navigation (FrontierNav)** [42] considers navigation in previously unknown environments, where the frontier refers to the boundary between explored space and unexplored space. Given only the current observations and an incrementally constructed map, the agent selects a frontier region as the next exploration target and navigates toward it at each step, so as to expand the explored area and gradually build a more complete map until no unexplored frontiers remain.

2.2.3. Interactive Navigation

Interactive navigation considers navigation tasks in environments that contain dynamic entities or interactive agents, such as humans, vehicles, or other robots. Compared with static navigation settings, this category requires the agent to continuously adapt its motion in response to moving obstacles and interactions with other agents in the environment.

- **Embodied Visual Tracking (EVT)** [43, 44, 45] requires an agent to follow a moving target using egocentric visual observations while maintaining a desired distance or viewpoint. Given a tracking goal (*e. g. "Follow the person in the red coat."*), the agent must continuously perceive the target, estimate its motion, and plan actions to keep the target within view while maintaining an appropriate distance.
- **Socially-aware Navigation (SocialNav)** [46, 47, 48, 49] studies navigation in environments densely populated by humans, such as hospitals or urban streets. Given an instruction (*e. g. "Go to the reception desk through a crowded hallway."*), the agent must reach its destination while respecting social conventions, such as maintaining appropriate interpersonal distance and following traffic regulations.
- **Multi-Agent Navigation** [50] considers scenarios in which multiple agents operate simultaneously in a shared environment to accomplish a common task. Given a collaboration instruction (*e. g. "Search different rooms for a fire extinguisher."*), agents must coordinate their behaviors through communication or implicit interaction while jointly exploring the environment or searching for targets. Although such cooperation can improve robustness compared with navigation performed by a single agent, it also introduces additional challenges in communication and coordination among agents.

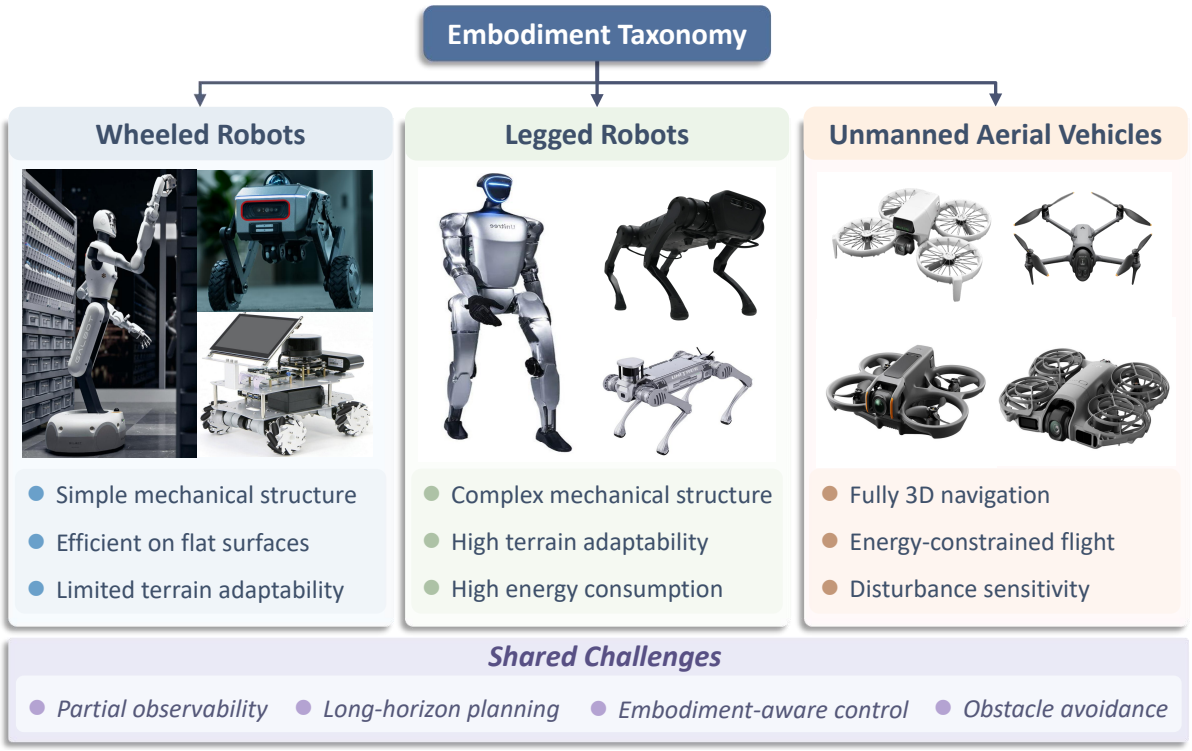

Figure 4 | **Embodiment taxonomy in embodied navigation.** Wheeled robots, legged robots, and UAVs differ in locomotion mechanism, terrain accessibility, operating space, and control complexity, leading to different requirements for perception, action design, and navigation planning.

2.2.4. Composite and Generalist Navigation

Although existing benchmarks typically isolate specific tasks to facilitate evaluation and comparison, real-world embodied navigation scenarios are often more complex and require an agent to integrate multiple navigation capabilities within a single episode. For example, under a VLN instruction such as *"Walk along the hallway, enter the living room, and find my keys."*, the agent may interpret entering the living room as a RegionNav objective, reformulate the route from the hallway to the living room as a PointNav problem, follow the planned waypoints to reach the living room, and then use ObjectNav to locate the keys. This process may also involve *Interactive Navigation* to avoid nearby people and, in the end, deliver the keys to the user.

More broadly, many practical instructions are high-level and complex, so the agent must determine how to decompose them into appropriate sub-tasks rather than follow a single fixed task formulation. This requires the continuous integration of spatial planning, semantic grounding, and context-aware action adaptation as the environment and the objective evolve. Recent developments in foundation models [12, 13, 14, 15, 16] make this broader setting increasingly feasible, because large-scale multimodal pre-training provides stronger semantic understanding, more flexible instruction following, and better transfer across different task formats. As a result, embodied navigation is gradually moving from narrowly defined benchmark tasks toward more open-ended and realistic scenarios that require seamless integration of multiple navigation capabilities.

2.3. Embodiment Taxonomy

Embodied navigation systems are deployed on robotic platforms with diverse physical embodiments to interact with the real world. Different embodiments introduce distinct sensing configurations, mobility capabilities, and forms of interaction with the environment, which further determine observation modalities, the design of the action space, and the physical constraints of motion. According to the locomotion mechanisms of the robotic platform, this section categorizes the embodiments commonly adopted in embodied navigation research into three groups: wheeled robots, legged robots, and UAVs. Figure 4 illustrates representative examples of each type of embodiment.

2.3.1. Wheeled Robots

Wheeled robots [51] are mobile platforms that rely on wheels for locomotion and are primarily designed to operate in flat or structured environments. A variety of wheeled configurations exist, including differential-drive, skid-steer, and omnidirectional designs, each providing different trade-offs in maneuverability, terrain adaptability, and motion precision. Such platforms generally have a simple mechanical structure and typically provide stable motion, low energy consumption, and efficient traversal on planar surfaces. These characteristics collectively contribute to the widespread adoption of wheeled robots not only in embodied navigation research but also in industrial automation, logistics, and service robotics. However, reliance on wheels constrains mobility in environments that contain stairs, obstacles, or highly uneven terrain, which limits the applications in complex outdoor scenarios.

2.3.2. Legged Robots

Legged robots [52, 53] employ articulated limbs for locomotion and are designed for environments that are difficult for wheeled systems to traverse. By adjusting footholds and body posture, these platforms can move across irregular terrain, climb obstacles, and maintain stability on uneven surfaces. Such capabilities provide greater terrain adaptability than wheeled robots, but they also introduce increased mechanical complexity, higher energy consumption, and more demanding control requirements. Legged robots are commonly categorized according to the number of legs used for locomotion. Most research in embodied navigation focuses on two representative forms: **quadruped** robots with four legs and **humanoid** robots with two legs. Quadruped robots typically maintain a stable support polygon during locomotion by moving one leg at a time while the remaining legs support the body, which enables stable gait patterns and robust movement on uneven terrain. Humanoid robots resemble the kinematic structure of the human body and are designed to operate in environments originally built for humans while interacting with human tools and infrastructure. However, such systems involve many degrees of freedom and require continuous balance control and coordinated body motion, which introduces significant challenges for perception, planning, and control.

2.3.3. Unmanned Aerial Vehicles (UAVs)

UAVs [54, 55], commonly known as drones, are aerial robots that generate lift through aerodynamic forces and operate without onboard human pilots. Compared with ground-based robotic platforms, UAVs navigate in fully 3D space and can move freely in both vertical and horizontal directions. This capability enables applications such as infrastructure inspection, search-and-rescue operations, and aerial exploration in both indoor and outdoor environments. At the same time, aerial navigation introduces additional challenges, including altitude-sensitive perception, energy-constrained flight, and precise control under aerodynamic disturbances.

2.3.4. Shared Objectives and Embodiment-Specific Constraints

Despite their different physical forms, wheeled robots, legged robots, and UAVs address the same high-level navigation problem: an embodied agent must perceive a partially observed environment, maintain spatial awareness, plan a feasible route toward a goal, and execute actions safely and efficiently. From this perspective, all embodiments require the integration of perception, mapping, memory, decision making, and control. They also share several core challenges, including partial observability, long-horizon planning, obstacle avoidance, and the need to align high-level task objectives with low-level physical execution.

Beyond these commonalities, the practical form of navigation differs substantially across embodiments because each platform is subject to different motion constraints, sensing configurations, and operating spaces. Wheeled robots usually navigate on planar surfaces with relatively stable and efficient motion, but their mobility is limited by stairs, gaps, and uneven terrain. Legged robots offer stronger terrain adaptability and can access more complex environments, yet they require more sophisticated whole-body coordination and balance control. UAVs extend navigation into fully 3D space and can bypass many ground-level obstacles, but this advantage comes with stricter energy constraints, stronger sensitivity to control errors, and greater demands on 3D perception and trajectory planning. Therefore, while these embodiments share a common navigation objective, they differ in observation design, action space definition, and planning complexity, which should be taken into account when analyzing embodied navigation systems.

3. Key Design Dimensions

Based on the problem formulations and taxonomy in §2, this section examines the core design dimensions of embodied navigation systems, from sensory input to final action execution. We first analyze how agents encode

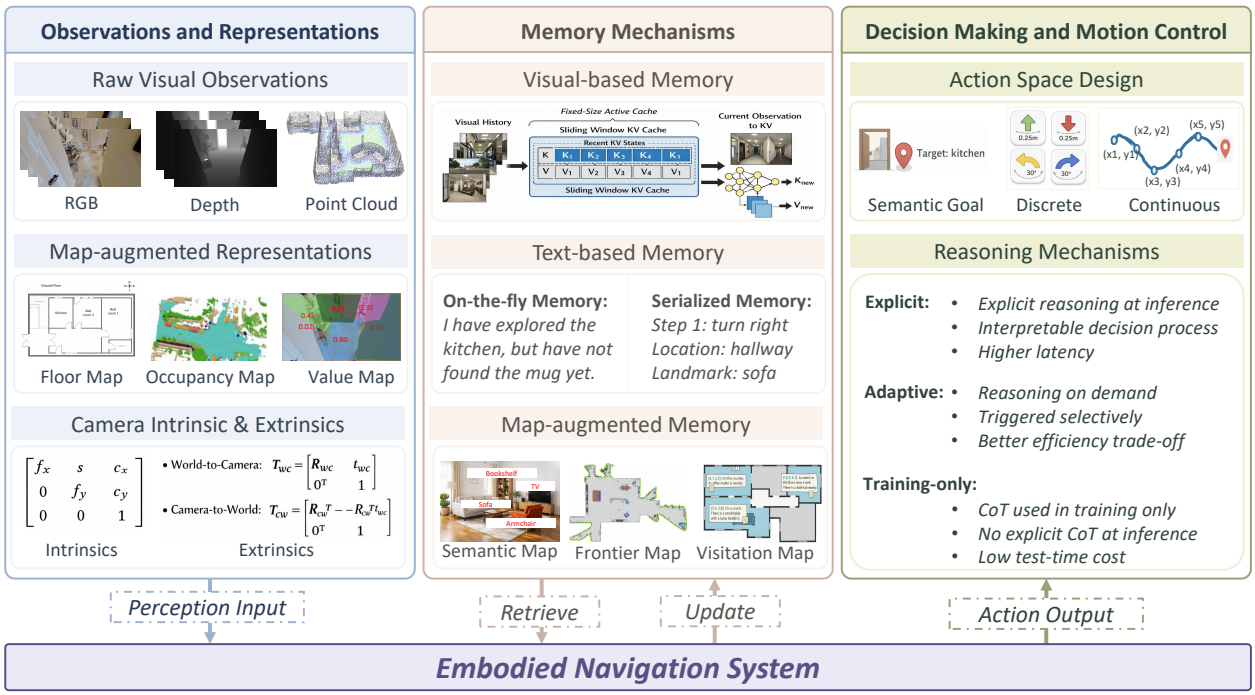

Figure 5 | Key design choices in embodied navigation from perception to action, including observations and representations, memory mechanisms, decision making and motion control, and their interactions within the embodied navigation system.

observations and spatial structure (§3.1), how they maintain and update memory under partial observability (§3.2), and how they translate accumulated information into navigation decisions and executable motion (§3.3). We then extend the discussion to the system level and examine what architectural designs are used to coordinate these input, memory, and output components, including modular systems, single-policy systems, dual-system designs, and world-model-based variants (§3.4). Figure 5 provides an overview of the main input, memory, and output forms, and Table 1 further summarizes how different embodied navigation systems instantiate these design choices.

3.1. Observations and Representations

An embodied agent's ability to navigate depends fundamentally on what it perceives and how that perception is represented. We organize observation and representation designs along three axes. First, we consider *raw visual observations*, including the egocentric RGB and depth signals that serve as the primary sensory input. Second, we discuss *map-augmented representations* that lift these per-step observations into structured spatial maps for explicit geometric and semantic reasoning. Third, we describe the role of *camera intrinsics and extrinsics*, the geometric metadata that enables spatially consistent fusion across viewpoints and time steps.

3.1.1. Raw Visual Observations

RGB-based observations. Embodied navigation typically begins with egocentric visual input. The most common choice is the current *RGB frame*, which provides direct information about the current state. To incorporate temporal context, some methods use a short sequence of recent RGB frames or an egocentric video clip [1, 4, 5, 45, 62, 75]. In addition, some methods use *multi-view RGB observations* [9], such as directional images or panoramic views, to enlarge the field of view and provide broader context for action selection.

Depth-aware observations. While RGB captures appearance, it often lacks explicit geometric cues, which can limit the model's ability to estimate free space and reason about spatial relationships. Many methods therefore incorporate *depth maps* or *point clouds* to represent scene structure more explicitly [5, 7, 57, 58, 61, 69, 74]. These geometric signals may come from RGB-D cameras, stereo vision, LiDAR, or monocular depth estimation. Compared with RGB alone, they offer stronger support for free-space estimation, obstacle reasoning, geometric grounding, and downstream planning and control.

Table 1 | Comparison of representative embodied navigation systems across observation representations, memory mechanisms, decision making strategies, and task coverage. For Map, fixed global maps are categorized as Observation, while maps updated during navigation are categorized as Memory. **Cam.Params** denotes camera intrinsics and extrinsics. **D.** and **C.** denote discrete and continuous actions, respectively.

Model	Observation				Memory			Decision		Task
	2D	Depth	Map	Cam. Params	Visual	Text	Map	Reasoning	Action	
Modular Systems										
CogNav[56]	✓						✓		D.	ObjectNav
InstructNav[57]	✓	✓	✓	✓					D.	VLN, ObjectNav
WMNav[58]	✓	✓		✓					D.	ObjectNav
Single-policy systems										
NaVid[59]	✓				✓				D.	VLN
Uni-NaVid[60]	✓				✓				D.	VLN, EVT, ObjectNav
NaVid-4D[61]	✓	✓			✓				D.	VLN
VLN-R1[62]	✓				✓				D.	VLN
JanusVLN[5]	✓	✓			✓				D.	VLN
OctoNav[63]	✓				✓			✓	D.	VLN, Object-, Point-, ImageNav
Hydra-Nav[10]	✓				✓	✓		✓	D.	ObjectNav
Nav-R2[64]	✓				✓			✓	D.	ObjectNav
NavForesee[65]	✓			✓				✓	D.	VLN
MapDream[66]	✓				✓				D.	VLN
FloorPlan-VLN[38]	✓		✓		✓				D.	VLN
StreamVLN[67]	✓				✓				D.	VLN
MapNav[68]	✓						✓		D.	VLN
NaVILA[1]	✓				✓				D.	VLN
FOM-Nav[69]	✓	✓		✓		✓	✓		D.	ObjectNav
ABot-N0[2]	✓				✓			✓	C.	VLN, EVT, Object-, PointNav
TrackVLA[45]	✓				✓				C.	EVT
NavFoM[9]	✓				✓				C.	VLN, EVT, ObjectNav
VLingNav[11]	✓				✓	✓		✓	C.	EVT, Object-, ImageNav
AstraNav-Memory[3]	✓			✓	✓				C.	ObjectNav
RoboTron-Nav[70]	✓			✓	✓				C.	ObjectNav
FantasyVLN[8]	✓				✓			✓	C.	VLN
UrbanVLA[71]	✓				✓				C.	Point-, SocialNav
SocialNav[72]	✓				✓			✓	C.	Point-, SocialNav
AutoFly[73]	✓								C.	VLN
VLA-AN[7]	✓	✓			✓				C.	VLN, EVT, ObjectNav
SparseVideoNav[6]	✓				✓				C.	VLN
Dual-system Architectures										
Nav-R1[74]	✓	✓			✓			✓	D.	VLN
InternVLA-N1[4]	✓				✓				C.	VLN, Point-, ImageNav
DualVLN[75]	✓				✓				C.	VLN
OmniNav[76]	✓				✓		✓	✓	C.	VLN, Object-, Point-, FrontierNav

3.1.2. Map-augmented Representations

Several works transform raw visual observations into structured spatial maps to provide more explicit geometric and semantic support for navigation. These maps are often represented in a *bird's-eye-view (BEV)* layout, a top-down projection that naturally encodes spatial relationships among obstacles, free space, landmarks, and the agent itself. Common map forms include *floor maps*, *occupancy maps* and *value maps*. A floor map provides a global structural prior of the environment, such as room layouts, walls, and connectivity [38]. An occupancy map records whether each spatial region is free, occupied, or unknown, supporting traversability estimation and FrontierNav [56]. It is typically constructed from depth, RGB-D, LiDAR, stereo, or fused multi-view observations. A value map instead assigns task-related utility scores to candidate locations, guiding the agent toward promising areas [57]. It is typically derived from observations, goals, semantic cues, exploration status, or estimated future reward, and provides more task-specific semantic guidance than floor maps or occupancy maps.

3.1.3. Camera Intrinsics and Extrinsics

Beyond semantic content, embodied navigation also relies on geometric metadata that describes the imaging geometry and the spatial configuration of each observation. *Camera intrinsics* (focal length, principal point, and distortion coefficients) govern the projection from 3D scene points to 2D pixel coordinates, and are essential for recovering metric depth and building spatially consistent maps. *Camera extrinsics* specify the 6-DoF pose (position and orientation) of each camera in a common coordinate frame, enabling observations captured at different viewpoints and time steps to be registered into a shared 3D space.

Recent embodied navigation systems exploit such geometric information in different ways. Some methods use camera parameters as explicit geometric conditions for visual encoding. For example, `RoboTron-Nav` [70] feeds camera parameters, including intrinsic and extrinsic matrices, into its 3D visual encoder to align 2D image features from different viewpoints and project them into a unified 3D representation. Other methods incorporate pose information to support temporal-spatial reasoning. `NavForesee` [65] encodes the agent's relative pose, including relative position and orientation, with an additional position encoder, and combines it with visual observations and dream queries in the world-model branch, helping the model capture temporal spatial relations and environment dynamics. Geometric information can also support online map construction and memory organization. For instance, `FOM-Nav` [69] uses camera extrinsics to back-project the current depth map and segmented objects from the camera view into the world coordinate frame, enabling online map building during navigation. `AstraNav-Memory` [3] organizes historical observations as camera-pose and image pairs, where the pose of each frame is serialized as text tokens and inserted before the corresponding visual tokens, enabling the model to recover spatial structure and reason over multi-view relations across long-horizon visual histories.

Overall, these examples show that camera intrinsics and extrinsics are important geometric cues for visual encoding, spatial memory, and world modeling in embodied navigation.

3.2. Memory Mechanisms

While observation and representation define what information is available to the agent at each step, memory mechanisms determine which information from current and past observations should be retained, how it should be compressed or structured over time, and how it should be recalled to support long-horizon navigation under partial observability. Existing methods differ primarily in the form in which historical information is preserved. Some methods directly retain visual observations or compress them into textual summary. Others externalizes history into maps or graphs to support explicit spatial reasoning. These directions reflect different design choices in the representation, maintenance, and use of historical information.

3.2.1. Visual-based Memory

Visual-based memory treats past observations as an extended visual context, retaining historical visual inputs or their compressed representations so that the agent can combine past perceptual evidence with the current view. The main design questions are therefore *which visual history to keep*, *how to compress it efficiently*, and *how to reuse it without excessive computational cost*.

Historical frame selection. The most straightforward design retains a subset of historical frames and feeds them together with the current view, thereby providing temporal continuity and contextual cues. The main variation across methods lies in the strategy used to select historical frames. Some methods retain the most recent frames with a sliding window [4, 75], some sample frames at regular intervals from the beginning of

the episode [59, 60], and others adopt non-uniform sampling that preserves distant history more coarsely and recent history more densely [11, 62]. For example, VLingNav [11] follows the last strategy through a dynamic sampling scheme inspired by the Ebbinghaus forgetting curve, allowing the agent to maintain a compact record of earlier observations while preserving finer detail for more recent ones.

Visual memory compression. However, directly accumulating visual history quickly leads to excessive token length and substantial redundancy. To address this issue, many methods further compress historical frames, often assigning different compression levels to frames with different temporal roles. For example, NavFoM [9] allocates many more tokens to the current frame than to previous frames, so that the model preserves fine-grained information about the present state while still retaining a lightweight record of the past. Uni-NaVid [60] explicitly separates current observations, short-term history, and long-term history, and further reduces redundancy through online visual token merging. In contrast to NavFoM [9] and Uni-NaVid [60], which retain visual history by incrementally incorporating new observations and compressing them over time, Nav-R2 [64] extends visual observation memory to goal-aware maintenance by compressing, merging, and updating tokens of historical RGB observations according to their relevance to the current target. Instead of modifying the vision encoder, AstraNav-Memory [3] introduces an additional plug-and-play visual tokenizer before the VLM, which compresses visual tokens without changing the subsequent modules.

Efficient memory storage. To reduce the cost of repeatedly encoding long visual histories, some methods store visual memory in the form of KV cache. JanusVLN [5] encodes historical visual observations into KV states produced by a semantic visual encoder and a spatial geometric encoder, and incrementally maintains this memory through an initial window and a sliding window. In this way, visual history is represented as a fixed-size implicit neural memory that avoids repeatedly re-encoding historical frames. StreamVLN [67] adopts a related strategy through a fast-streaming dialogue context with a sliding-window KV cache, where only the KV states from a fixed-size active window of recent dialogue rounds are retained. This design serves as a reusable inference cache across interaction turns and avoids prefilling the entire historical context at every step. Similarly, Hydra-Nav [10] and OmniNav [76] use KV cache in their fast systems, allowing the model to encode only the latest observation at each step while continuously decoding low-level actions.

While visual-based memory is direct and intuitive, the retained memory remains a sequence of views rather than an explicit model of the environment, which makes long-horizon planning difficult.

3.2.2. Text-based Memory

Text-based memory builds on historical observations but represents retained history in natural language, often by summarizing past perceptions, actions, or intermediate conclusions. Existing approaches can be broadly divided into two groups: *methods that let the model generate textual memory during reasoning*, and *methods that explicitly serialize historical information into structured textual records*.

On-the-fly textual memory. One design allows the model to form textual memory as part of the reasoning process. OctoNav [63] follows this strategy through a Think-Before-Action mechanism, in which the model summarizes historical information and evaluates task progress before predicting the next action. This design encourages the model to make direct use of historical context during decision making, but its effectiveness depends on whether the model can accurately interpret and summarize past observations.

Serialized textual memory. Instead of relying on model-generated summaries, another design explicitly constructs textual memory through template-based serialization of historical information, which is then provided to the model as long-term memory for subsequent planning and decision making. Hydra-Nav [10] follows this strategy by organizing exploration history through serialized landmark nodes connected with executed actions. Although this design provides a more structured and objective form of memory, directly incorporating executed actions into textual memory often results in relatively low information density and may introduce additional noise or misleading signals.

Overall, text-based memory provides a flexible and interpretable way to exploit the language prior of foundation models. However, textual memory often omits fine-grained spatial and geometric details, which can reduce reliability in tasks that require precise geometric consistency.

3.2.3. Map-augmented Memory

Instead of retaining memory as a sequence of views or language summaries, map-augmented memory maintains a structured representation of the environment that is continuously updated during exploration to support

long-horizon decision making. Existing approaches mainly differ in *how the map is constructed, updated, and integrated into the policy model.*

Map memory as a parallel input branch. One common design maintains an explicit spatial map as a separate memory representation and encodes it jointly with current observations. `MapNav` [68] follows this strategy by building a top-down Annotated Semantic Map (ASM), which is initialized at the beginning of each episode and progressively updated during navigation with obstacle distributions, explored regions, and semantic object locations. During inference, the current observation frame and the ASM are encoded in parallel and projected into the embedding space of the language model through modality-specific projectors. The resulting embeddings are concatenated with instruction tokens to form a unified multimodal sequence for subsequent action prediction. `FOM-Nav` [69] adopts a related design for object-goal navigation, but represents memory through online frontier-object mapping. It maintains a scene point cloud for frontier extraction and an object-centric point cloud for persistent object memory. These elements are further converted into structured spatial embeddings that support high-level goal selection.

Relevant map memory injected into the current visual context. Another design integrates map-based memory more tightly with the current visual context, rather than encoding the entire map as an independent token sequence. `Mem2Ego` [77] follows this strategy by maintaining multi-level global memory, including a global frontier map, visitation memory, and top-k landmark semantic memory. This global memory is projected onto the four current egocentric views to form a memory-augmented observation with explicit markers of relevant memory elements. The augmented visual input is then provided to the model together with the task prompt, and the model selects the marker corresponding to the next navigation target. `GSMem` [78] takes a further step by constructing an online map based on the 3D Gaussian Splatting (3DGS) technique[79], allowing the agent to revisit candidate target regions by rendering novel views from selected viewpoints.

Although map construction and maintenance introduce additional computational cost and system complexity, explicit map-based memory provides a more stable and geometrically grounded representation of the environment, which is beneficial for long-horizon planning and spatial reasoning. Improving the efficiency of map construction, memory updating, and memory utilization therefore remains an important direction for future research.

3.3. Decision Making and Motion Control

Based on the observation and memory mechanisms described above, the model must decide what action to take and how that decision should be executed as robot motion. In existing embodied navigation systems, this process is mainly shaped by two design dimensions: *action space design, which defines the form of the predicted action*, and *reasoning mechanisms, which determine how the decision is produced.*

3.3.1. Action Space Design

Action space design defines the interface between high-level decision making and low-level robot execution, and directly affects control granularity, motion flexibility, and learning difficulty. Existing methods mainly adopt three formulations: *semantic target or direction selection*, *discrete action prediction*, and *continuous action prediction*.

Semantic target or direction selection. In this formulation, the backbone model, such as a VLM or LLM, does not directly predict executable robot actions. Instead, it identifies a semantic target or spatial direction in the environment based on visual observations, and the selected target is then mapped to coordinates in the global map and executed by a low-level control module. Methods such as `Mem2Ego` [77] and `MapNav` [68] follow this design, which places greater emphasis on semantic understanding than on direct motion generation.

Discrete action prediction. In this formulation, the backbone model directly predicts discrete navigation actions, including moving forward, moving backward, and turning left or right [5, 60, 67, 68]. Each action corresponds to a predefined motion primitive, such as moving 0.25 meters for one forward action and rotating 30 degrees for one turning action, thereby enabling discrete robot control. Some methods further extend this formulation by allowing the model to predict a discrete action together with a continuous parameter, including the distance of forward movement or the angle of rotation [1, 10, 38, 59, 63, 74]. This design is simple and naturally aligns with language-based action prediction. However, the limited action space often lacks the flexibility required for complex environments or precise control.

Continuous action prediction. In this formulation, the model predicts continuous actions for finer motion

control, typically represented in either 2D or 3D space [3, 4, 6, 7, 9, 11, 45, 75]. For example, DualVLN [75] converts 3D trajectories into pixel-level goal grounding samples and trains the model to predict 2D pixel coordinates, such as (x, y), corresponding to the next waypoint from egocentric RGB observations. In contrast, VLA-AN [7] adopts a 3D formulation in which the model predicts waypoints in 3D space together with desired yaw angles. Many methods further introduce a learnable action module after the backbone model, such as an multilayer perceptron (MLP) or a Diffusion Transformers (DiT), to convert predicted actions into executable trajectories. Continuous action representations provide greater flexibility, but they also require stronger spatial understanding and grounding, as well as carefully constructed training data to ensure meaningful waypoint prediction. In addition, because VLM inference is often time-consuming, many approaches predict multiple future waypoints in a single step to improve efficiency.

The choice of action space design depends on the specific task requirements, embodiment constraints, and model capabilities. Semantic target selection emphasizes interpretability and semantic understanding, discrete action prediction offers a balance between simplicity and control, and continuous action prediction provides the greatest flexibility at the cost of increased complexity.

3.3.2. Reasoning Mechanisms

Beyond action space design, reasoning mechanisms determine whether the model makes decisions through explicit intermediate reasoning or predicts actions more directly. Existing methods mainly differ in *when reasoning is invoked* and *whether explicit reasoning is retained during inference*.

Explicit reasoning with CoT. One line of work incorporates explicit chain-of-thought (CoT) reasoning into inference-time decision making [2, 63, 64, 65, 74, 76]. For example, Nav-R2 [64] explicitly decomposes decision making into target-environment and environment-action relational reasoning chains before predicting the next action, thereby linking semantic reasoning with action prediction. A similar idea appears in dual-system architectures such as OmniNav [76], which introduces CoT reasoning into the slow system built on VLMs to generate subgoals and plan paths.

Adaptive reasoning with CoT. While explicit CoT produces interpretable intermediate reasoning steps that can improve decision transparency and reliability, it also increases computational cost and decision latency, which can be problematic in navigation tasks that require rapid responses. To balance reasoning benefits and efficiency, some methods adopt adaptive reasoning, in which the model dynamically decides whether explicit reasoning is necessary according to current task demands and environmental complexity. Hydra-Nav [10] introduces iterative rejection fine-tuning that allows the model to detect stagnation points, such as completion of a subgoal in the high-level plan or invalidation of the current plan by new observations, and then trigger reasoning and replanning. Similarly, VLingNav [11] constructs training samples with and without CoT, allowing the model to learn a reasoning indicator token that determines whether reasoning should be activated.

CoT as training supervision. Another line of work uses reasoning only as an auxiliary signal during training, rather than as an explicit inference-time process. Aux-Think [80] adopts this strategy by using CoT during training to guide the internalization of reasoning patterns, while directly predicting actions during inference. This design reduces the risk that low-quality reasoning degrades navigation performance at test time. Following a similar idea, FantasyVLN [8] mixes training samples from four reasoning modes, including Non-CoT, Textual CoT, Visual CoT, and Multimodal CoT, and uses a gating mechanism to switch across these modes during training, while only the Non-CoT mode is activated during inference.

Reasoning mechanisms play a crucial role in shaping the decision-making process of embodied navigation systems. Explicit reasoning can enhance interpretability and decision quality, but it also introduces additional complexity and latency. Adaptive reasoning and auxiliary reasoning during training are promising strategies to balance these trade-offs.

3.4. System Architectures

This section reviews embodied navigation systems from the perspective of architectural design, focusing on how perception, memory, reasoning, and control are coordinated within the overall navigation pipeline. We organize existing methods into three main architectural categories: *modular systems*, which decompose navigation into explicit functional components and intermediate representations; *single-policy systems*, which use an end-to-end policy to directly map multimodal inputs to navigation actions; and *dual-system architectures*, which also follow the end-to-end paradigm but separate slow high-level reasoning from fast low-level control. Beyond

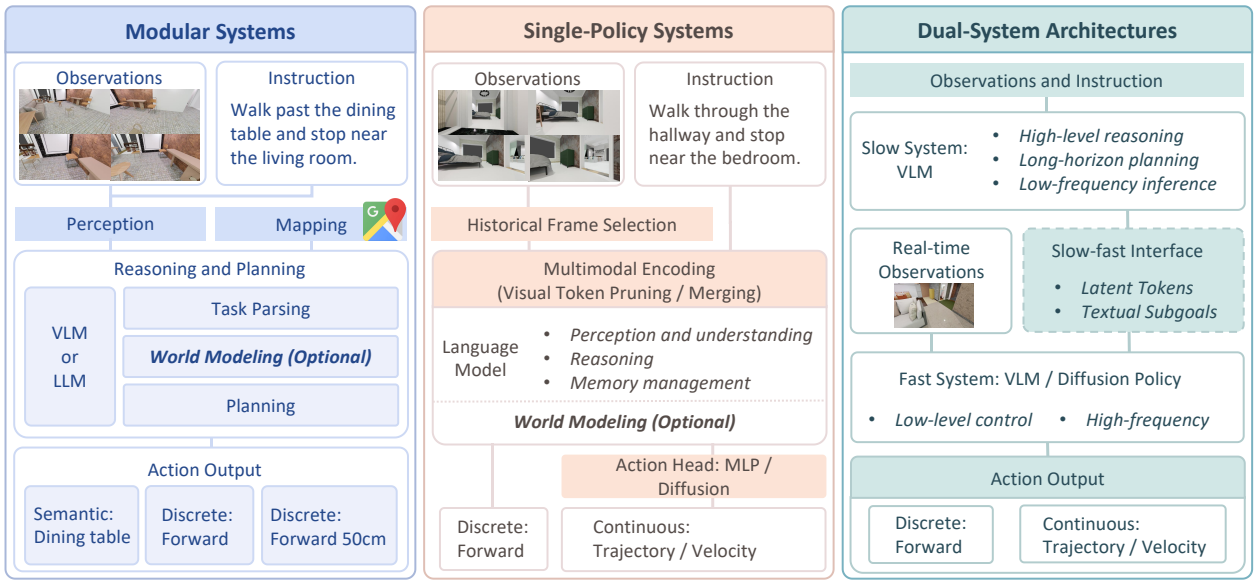

Figure 6 | **Representative architectures for embodied navigation.** *Modular systems* decompose navigation into explicit components for perception, mapping, reasoning, planning, and control. *Single-policy systems* use a unified backbone to map observations and instructions directly to actions. *Dual-system architectures* separate slow high-level reasoning from fast low-level execution and connect the two through intermediate representations such as latent tokens and textual subgoals.

this architectural grouping, we further discuss *world-model-based systems*, which use predictive modeling of future states, dynamics, or intermediate representations to support planning and action generation.

3.4.1. Modular Systems

Modular architectures decompose navigation into multiple functional components, such as perception, mapping, planning, and control, with the foundation model typically embedded in the reasoning and planning loop to coordinate these components.

A common modular pipeline first builds explicit intermediate states from sensory observations and then lets an LLM or VLM reason over these states for high-level decision making [56, 57, 58]. `CogNav` [56] follows this design by constructing an online cognitive map from posed RGB-D observations with a VLM and serializing this map into textual prompts for an LLM, which then schedules exploration strategies and selects navigation landmarks. `InstructNav` [57] adopts a similar idea: an LLM generates a Dynamic Chain-of-Navigation from instructions and scene object labels to predict the next action and landmark, and these high-level decisions are then grounded through RGB-D observations and semantic mapping to build multi-source value maps for waypoint selection.

Modular systems improve interpretability by explicitly separating perception, memory, planning, and control, making it easier to inspect failure sources and incorporate priors. Their main limitation is coordination overhead, since overall performance depends heavily on the quality of individual components and their interfaces.

3.4.2. Single-policy systems.

Compared with modular systems, end-to-end systems learn a direct policy that maps multimodal observations to navigation actions without relying on explicit intermediate modules. Single-policy systems represent the most direct form of this paradigm, in which a single foundation model jointly handles perception, context retention, and action generation within a unified policy. Within this setting, we first discuss methods that predict *discrete actions* and then turn to methods that predict *continuous controls*.

Single-policy systems with discrete action outputs. Some methods directly use a single policy to map observations to discrete navigation actions. For instance, `NaVid` [59] and `Uni-NaVid` [60] take RGB video streams together with language instructions as input and predict the next discrete action, whereas `NaVid-4D` [61] strengthens spatiotemporal reasoning by incorporating RGB-D video inputs. `VLN-R1` [62] follows the same

formulation but further improves the policy through reinforcement fine-tuning (RFT).

The main bottleneck of this design is long-horizon decision making under partial observability. Accordingly, several studies strengthen memory while preserving end-to-end training. StreamVLN [67] combines a fast sliding-window dialogue context with a slowly updated compressed visual memory, whereas JanusVLN [5] separates spatial-geometric and visual-semantic memories into two compact neural modules for long-horizon memory management. Other works instead strengthen explicit reasoning. For example, OctoNav [63] trains the model to reason before action prediction across diverse tasks and modalities, Nav-R2 [64] decomposes reasoning into target-environment and environment-action relational chains, and Hydra-Nav [10] introduces adaptive reasoning that switches between a fast action mode and a slow reasoning mode to reduce latency.

Single-policy systems with continuous action outputs. As discrete actions are often too coarse for precise control, a growing number of studies instead use a single policy to map observations to fine-grained continuous controls. These methods usually encode visual observations, language instructions, and historical context into a unified representation space, use an LLM or VLM backbone to produce high-level action representations, and then generate executable controls through different action decoding designs. According to the action decoding design, existing methods can be grouped into three categories:

- **Direct action prediction from backbone tokens.** The simplest design does not introduce an explicit trajectory decoder, but instead uses the backbone output itself as an action mapping space for continuous control prediction. For example, AutoFly [73] adopts a UAV-oriented design by augmenting RGB features with pseudo-depth features, mapping them together with language guidance into a unified multimodal representation, and using the final token set as an action space for predicting continuous velocity and direction commands.
- **MLP-based action decoding.** Some studies use MLP layers after the foundation model to map latent action tokens to executable controls [9, 11]. NavFoM [9] processes visual and textual inputs, produces an action hidden state through the LLM backbone, and decodes this state with a three-layer MLP planning head into a sequence of eight continuous trajectory points in either 2D or 3D. Similarly, VLingNav [11] and RoboTron-Nav [70] pass action tokens from the LLM to an MLP-based action module that converts them into robot trajectories. UrbanVLA [71] follows the same pattern and further applies Implicit Q-Learning (IQL) to fine-tune the action module.
- **Diffusion-based action decoding.** Other studies replace the simple MLP layers with more expressive trajectory generators based on a diffusion policy, which can model richer trajectory distributions and produce smoother action sequences [2, 45, 46, 76]. ABot-N0 [2] feeds VLM action tokens into a diffusion-based action head that generates a short-term trajectory plan of five waypoints in the local BEV frame, each with 2D position and yaw orientation. Similarly, SocialNav [72] uses conditional flow matching to translate VLM semantic priors into executable trajectories that respect social interaction constraints.

Overall, single-policy methods with discrete outputs are simpler and more naturally aligned with language-style action prediction, but they are less flexible for precise control. Continuous-control methods provide finer motion generation, yet they place higher demands on spatial grounding, action decoding, and training data quality.

3.4.3. Dual-system architectures.

Inspired by the separation between slow thinking and fast execution in human cognition, some recent end-to-end methods adopt a dual-system architecture [4, 74, 75, 76]. These approaches decompose the navigation policy into two coupled systems with complementary roles: a slow system responsible for high-level semantic reasoning and long-horizon planning, and a fast system responsible for low-level action generation and real-time control. Both systems can receive visual observations, but they usually operate at different frequencies and communicate through intermediate latent or textual representations to accomplish long-horizon navigation.

Existing methods mainly differ in the implementation of the two subsystems and, more importantly, *in the interface through which slow reasoning guides fast control*. In InternVLA-N1 [4], an LLM-based slow system processes long-horizon multimodal observations at 2 Hz and produces mid-term latent plan tokens for a diffusion-based fast system. The fast system operates at 20 Hz and generates continuous trajectories for real-time navigation control. DualVLN [75] follows a similar asynchronous dual-system design but introduces a different interaction interface between the two systems. Specifically, a set of learnable latent queries is appended to the contextual hidden states of the VLM to extract task-relevant semantic information that conditions the fast system.

Some methods further reduce the separation between the two subsystems by sharing parameters across them. `OmniNav` [76], for instance, employs a dual-system design in which the slow and fast subsystems share a VLM backbone and a diffusion policy, and communicate through textual subgoals generated by the slow system. `Nav-R1` [74] adopts a different form of coupling, in which the final few blocks of the slow system are reused as the fast system. In this design, the slow system performs long-horizon semantic reasoning through the full computation path, whereas the fast system directly generates navigation actions without explicit reasoning.

The key benefit of dual-system design is a cleaner division between high-level reasoning and real-time control. The main challenge is interface design: the two systems must exchange enough information to stay coordinated without making training unstable or inference too expensive.

3.4.4. World-model-based systems.

Beyond VLM-based architectural designs, another important direction introduces world modeling into embodied navigation [6, 58, 65, 66]. This design appears in both modular systems and end-to-end single-policy systems. Existing methods mainly differ in *what form of future information is modeled, such as dynamics, maps, or predicted visual observations, and how these predictions are coupled with action generation*.

In modular settings, world modeling is typically implemented through dedicated prediction modules that estimate environmental states or support high-level planning. For instance, `WMNav` [58] integrates multiple specialized VLM modules within a world-model-based navigation framework. PredictVLM estimates the current environmental state from panoramic RGB-D observations and updates a Curiosity Value Map, while PlanVLM and ReasonVLM use the selected view and memory signals to infer sub-tasks and generate the final navigation action. `Schrödinger's Navigator` [81] uses an occluder-aware trajectory sampler and a trajectory-conditioned 3D world model to imagine multiple plausible future observations, and then fuses these predictions into a Future-Aware Value Map for uncertainty-aware waypoint selection and action execution.

In end-to-end single-policy systems, world modeling is incorporated more tightly into policy learning. In these methods, the model not only predicts the next action, but also models future states, scene dynamics, or intermediate spatial representations to guide action generation. `NavForesee` [65] integrates hierarchical language planning and predictive world modeling within a unified VLM backbone, where the model autoregressively generates milestone-level subgoals while predicting short- and long-horizon environmental dynamics through dream queries to guide the final navigation action. `MapDream` [66] adopts a learned map-in-the-loop design that jointly generates a multi-channel BEV map and predicts actions, thereby incorporating a spatial map representation into end-to-end training. `SparseVideoNav` [6] introduces a video generation backbone to produce sparse video latents that represent predicted environmental changes, and these latents are combined with language instructions and fed into a DiT-based action head to generate navigation actions.

Overall, world-model-based systems strengthen anticipation by incorporating future-oriented prediction into navigation. Their main challenge lies in the coupling between prediction and control: when predicted futures are inaccurate, unstable, or poorly aligned with downstream decision making, navigation performance can degrade accordingly.

3.5. Takeaway Insights

The design choices discussed in this section reveal a persistent trade-off among *expressiveness*, *efficiency*, and *control reliability*. Richer observations and structured representations, such as depth, maps, and calibrated multi-view geometry, provide stronger support for spatial reasoning, but they also increase perception and fusion cost. More powerful memory mechanisms improve long-horizon consistency under partial observability, but must balance information retention against redundancy, compression overhead, and retrieval difficulty. At the decision layer, simpler action spaces and direct policies are easier to train and deploy, but they often sacrifice flexibility or precision, whereas more expressive action formulations and explicit reasoning can improve planning quality at the cost of additional latency and system complexity.

These trade-offs extend naturally to the architectural level. Modular systems improve interpretability by separating perception, memory, planning, and control through explicit intermediate representations, but their performance is highly dependent on the coordination of multiple components. End-to-end systems offer a more direct policy interface, yet place a greater burden on the foundation model to absorb perception, memory, reasoning, and control within a unified policy. Within this paradigm, single-policy systems emphasize simplicity, whereas dual-system designs introduce functional separation between slow reasoning and fast control. World-

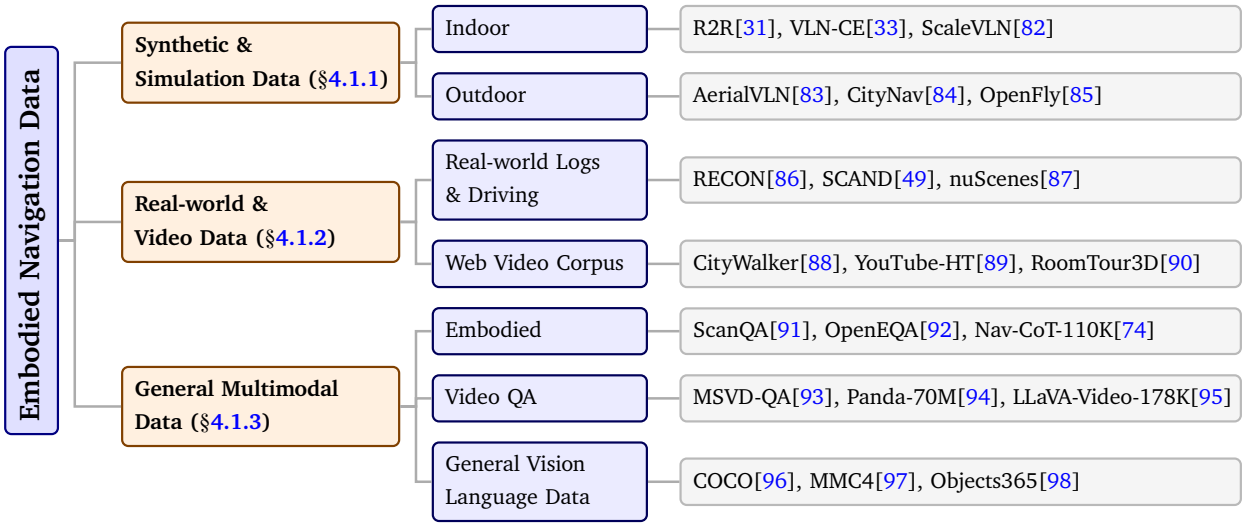

Figure 7 | Summary of embodied navigation data and representative datasets.

model-based designs further strengthen anticipation through prediction of future states, dynamics, or spatial representations.

Taken together, recent progress suggests that embodied navigation is moving away from purely reactive policies toward more selective use of geometry, memory, reasoning, and prediction. The central challenge is no longer simply to add more capability, but to introduce the right amount of structure at the right level, so that planning and long-horizon consistency improve without undermining efficiency and real-time control.

4. Data Collection and Training Strategies

Navigation foundation models require large-scale and diverse data to acquire robust embodied navigation capabilities, and, once such data are available, a variety of training strategies can be used to turn them into effective policies. Accordingly, this section reviews the topic from two complementary perspectives: *data collection* (§4.1) and *training* (§4.2). We first summarize the major data sources that support navigation models, then discuss representative learning paradigms built on them, finally highlighting dataset biases and leakage issues that remain key obstacles to real-world generalization.

4.1. Data Source

Navigation data for foundation models mainly fall into three categories, as summarized in Figure 7. The following subsections review synthetic and simulation data, real-world and web video data, and general multimodal data, respectively. Together, these sources provide scalable trajectory supervision, realistic embodied experience, and auxiliary semantic and reasoning knowledge for navigation foundation models.

4.1.1. Synthetic and Simulation Data

Synthetic and simulation data are generated in virtual environments for navigation learning. They are mainly defined by three aspects: scene assets, simulation backends, and trajectory generation pipelines. Representative indoor and outdoor synthetic/simulation datasets are summarized in Table 2 and Table 3, respectively.

Scene assets. Synthetic and simulation data for embodied navigation are fundamentally built on top of a diverse bank of *scene assets*, which can be obtained from several distinct sources. For indoor navigation, a major class of environments comes from *real-world 3D reconstruction*, where scanned buildings are converted into navigable digital scenes with realistic geometry and appearance. Representative examples include `MP3D`[103], `HM3D`[104], `Gibson`[105], and `Replica`[106], which provide reconstructed homes, offices, and other indoor spaces. A complementary class consists of *synthetic indoor environments*, such as `HSSD`[107] and `3D-FRONT`[108], which are generated from high-quality 3D assets or procedural interior layouts, offering better scalability, controllability, and diversity of object arrangements. More recently, some works have further introduced *high-fidelity reconstructed scenes* based on modern rendering techniques such as `3D Gaussian Splatting (3DGS)`[79] to narrow the visual gap between simulation and reality. For outdoor and aerial navigation,

Table 2 | Summary of representative **indoor** datasets for embodied navigation. Here, *inst.* denotes instruction, and *traj.* denotes trajectory.

Name	Description	Quantity	Data Structure	Scene	Backend	Used by
Vision-and-Language Navigation						
R2R[31]	Canonical indoor VLN benchmark	7,189 traj. / 21,567 inst. / 90 scenes	Panoramic RGB, inst., traj.	MP3D	Matterport Simulator	Nav-R1[74]
RxR[32]	Multilingual long-horizon VLN benchmark	16.5K traj. / 126K inst. / 90 scenes	Panoramic RGB, inst., traj.	MP3D	Matterport Simulator, Habitat	NaVILA[1], Uni-NaVid[60], StreamVLN[67], NavFoM[9], InternVLA-N1[4], ABot-N0[2], MapDream[66], Nav-R1[74]
VLN-CE R2R[33]	Continuous VLN version of R2R	4,475 episodes / 13.4K inst.	RGB, inst., traj.	MP3D	Habitat	NaVILA[1], Uni-NaVid[60], StreamVLN[67], NavFoM[9], InternVLA-N1[4], ABot-N0[2], MapDream[66], Nav-R1[74]
R2R-EnvDrop[99]	Augmented R2R with synthetic inst. and dropout	60 scenes / 14K inst.	RGB, inst., traj.	MP3D	Habitat	NaVILA[1], StreamVLN[67], ABot-N0[2], InternVLA-N1[4]
ScaleVLN[82]	Large synthetic VLN pretraining corpus	4.9M inst.–traj. pairs / 1200+ scenes	RGB, inst., traj.	HM3D, Gibson	Habitat	StreamVLN[67], InternVLA-N1[4]
InternData-N1[4]	Large synthetic navigation corpus with VLN-N1, VLN-CE, and VLN-PE subsets	53.5M images / 0.8M inst. / 3,154 scenes	RGB-D, inst., traj.	Replica, MP3D, Gibson, 3D-FRONT, HSSD, HM3D	BlenderProc Habitat, InternUtopia	InternVLA-N1[4], DualVLN[75]
Goal-Driven and Target-Centric Navigation						
SOON[100]	Scenario-oriented object navigation benchmark from arbitrary starts	40K traj. / 4K inst. / 90 scenes	Panoramic RGB, target description, traj.	MP3D	Matterport Simulator	Nav-R1[74]
HM3D-OVON[37]	Open-vocabulary object-goal benchmark	145 scenes / 280 categories	RGB-D, target description, traj.	HM3DSem	Habitat	AstraNav-Memory[3], NavFoM[9], ABot-N0[2], VLingNav[11], Hydra-Nav[10], Nav-R1[74], Nav-R2[64]
GOAT-Bench[101]	Lifelong multi-goal navigation benchmark with persistent memory	725K episodes / 145 scenes / 50–500 step memory	RGB-D, multimodal goals, traj.	HM3D	Habitat	AstraNav-Memory[3]
Habitat-Web[102]	Large teleoperated simulator corpus for embodied tasks	80K demonstrations	RGB, traj., inst.	MP3D, Gibson	Habitat	VLingNav[11]
EVT-Bench[45]	Language-described embodied visual tracking benchmark	804 scenes / 26K episodes / 100+ avatars	RGB video, target description, traj.	Indoor avatar-populated scenes	Habitat 3.0	NavFoM[9], ABot-N0[2], VLingNav[11], TrackVLA[45]
Nav-AdaCoT-2.9M[11]	Multi-task embodied navigation corpus with adaptive CoT labels	2.9M traj. / 718 scenes / 472K CoT samples	RGB video, inst., CoT, traj.	HM3D, MP3D	Habitat	VLingNav[11]

scene sources are broader and often city-scale: they include handcrafted or asset-based urban environments in `Unreal Engine` [109], large-scale semantic urban reconstructions such as `SensatUrban`[110], open-world game environments such as `Grand Theft Auto V (GTA V)` [111], and geographically grounded visual assets from `Google Earth` [112]. Compared with indoor scene banks, these outdoor assets emphasize long-range topology, large navigable areas, building-scale landmarks, and altitude-dependent viewpoints, making them particularly suitable for urban and UAV navigation.

Simulation backends. Once scene assets are available, they must be instantiated in a *simulation backend* that allows an embodied agent to move, sense, and interact with the environment so that navigation trajectories can be collected at scale. For indoor embodied navigation, `Habitat`[114] is the most widely used backend, as it provides fast rendering and supports embodied-AI simulation in rich indoor scenes, while recent versions further extend to humanoids, avatars, and collaborative human–robot settings. When stronger physical realism and robot-centered control are required, `Isaac Sim`[115] is increasingly adopted, since it is designed as a physically based robotics simulator and provides physics engines, realistic sensor simulation, and robot testing

Table 3 | Summary of representative **outdoor** datasets for embodied navigation. Here, *inst.* denotes instruction, and *traj.* denotes trajectory.

Name	Description	Quantity	Data Structure	Scene	Backend	Used by
AerialVLN[83]	Human-piloted aerial VLN benchmark	8,446 traj. / 25,338 inst. / 25 scenes	RGB, inst., traj.	UE city, factory, park, and village scenes	AirSim+UE4	OpenFly[85]; mentioned by OpenUAV[113], AutoFly[73]
CityNav[84]	Aerial VLN benchmark over real-world scanned cities	32,637 traj. / 2 cities	RGB, inst., traj.	SensatUrban point clouds	CityFlight / Potree	CityNav[84]
OpenUAV[113]	Realistic UAV object-search VLN dataset with 6-DoF dynamics	12,149 traj. / 22 scenes / 89 objects	Multi-view RGB, target description, inst., traj.	UE4 scenes	AirSim+UE4	NavFoM[9]
OpenFly[85]	Multi-engine aerial VLN corpus	100K traj. / 18 scenes / 15.6K vocabulary	RGB, inst., traj.	UE4/UE5, GTA V, Google Earth, and 3DGS scenes	UE4/UE5, GTA V, Google Earth Studio, 3DGS toolchain	OpenFly[85]
AutoFly[73]	Hybrid sim-to-real UAV autonomy dataset with coarse instructions	13K+ episodes / 2.5M image–language–action triplets / 1K real episodes	RGB-D, inst., traj.	12 AirSim scenes + real outdoor flights	AirSim+UE4	AutoFly[73]

capabilities that are useful for collecting control-aware navigation rollouts. For outdoor and aerial navigation, `Unreal Engine` and `AirSim`-style stacks are commonly used as they support large-scale 3D worlds, flexible camera viewpoints, and programmable vehicle control, making them well suited for UAV trajectory generation and aerial visual observations [83, 85, 113]. Finally, `Google Earth` and related geospatial 3D renderers are often used as large-scale geographic backends, providing photorealistic 3D city views and broad real-world coverage for collecting urban or aerial navigation data in geographically grounded environments [85].

Trajectory synthetic engines. Given scene assets and simulation backends, a trajectory data engine typically needs to construct three core components for navigation learning: the *trajectory*, the *observations* collected along that trajectory, and the associated *instruction or task annotation*. In practice, the pipeline usually starts by selecting several task-relevant key locations, such as a start point, a goal point, intermediate rooms, object instances, or semantic regions. The exact sampling strategy depends on the target task. For `VLN-CE R2R`[31, 33] and `VLN-CE RxR`[32, 33], each episode is primarily defined by a start location, a goal location, and an associated reference path, which is then paired with route instructions. For `HM3D ObjectNav`[104] and `HM3D-OVON`[37], the goal is selected around a target object category or an open-vocabulary object instance. In aerial settings such as `AerialVLN`[83], `UAV-Need-Help`[113], `OpenFly`[85], and `CityNav`[84], key locations are often defined by landmarks, regions, or target waypoints in large outdoor scenes. Once these key locations are fixed, trajectory construction can proceed in different ways: benchmark-style engines such as `R2R`[31], `RxR`[32], `VLN-CE`, and large-scale synthetic pipelines such as `ScaleVLN`[82] mainly rely on shortest-path search or graph-based planning to densify them into executable trajectories, whereas aerial datasets such as `AerialVLN`, `UAV-Need-Help`, and `CityNav` rely more heavily on human piloting or human demonstration trajectories collected in simulation environments. `ScaleVLN`[82], for example, reuses HM3D and Gibson scenes to synthesize 4.9M instruction–trajectory pairs over 1200+ photorealistic environments. More recent engines push this formulation further. `R2R-EnvDrop`[99] applies environmental dropout and back-translation to synthesize additional navigation training triplets; `InternData-N1`[4] unifies `VLN-N1`[4], `VLN-CE`, and `VLN-PE`[34] into a large simulated navigation corpus with over 50M egocentric images from 3000+ scenes and 4839 km of navigation experience; and `ABot-N0`[2] scales this idea into a unified trajectory engine spanning Point-Goal, Object-Goal, Instruction- Following, POI-Goal, and Person-Following, yielding 16.9M expert trajectories over 7802 high-fidelity scenes. After trajectories are defined, the backend is used to roll out the agent and record observations at each step, including RGB images, camera pose, depth, segmentation, and other embodiment-specific signals. Finally, annotations are attached either manually or automatically: human annotation remains central in `R2R`, `RxR`, `AerialVLN`, and `CityNav`, whereas synthetic engines such as `ScaleVLN`, `R2R-EnvDrop`, `InternData-N1`, and `OpenFly` rely more heavily on automated instruction

generation, rewriting, filtering, and augmentation. Overall, despite substantial variation across datasets, most synthetic navigation data engines follow the same underlying workflow: task-aware key-location sampling, trajectory construction through planning or human demonstration, simulator-driven observation rollout, and finally human or automated annotation.

4.1.2. Real-world and Web Video Data

Real-world and web video data provide navigation supervision beyond simulation. They mainly include directly recorded real-world trajectories and large-scale web videos from which navigation-relevant signals can be recovered. Representative datasets in these two categories are summarized in Table 4.

Trajectory data directly collected in the real world. Besides simulation-generated supervision, recent navigation models also benefit from real-world trajectory data in which the trajectories are directly observed rather than synthesized. One important source is *robot or vehicle logs*, where a physical platform moves through the world and records egocentric observations, actions, poses, and sometimes human guidance or interaction signals. Representative examples include autonomous driving datasets such as `nuScenes`[87] and `OpenScene`[116], which provide real-world driving trajectories together with rich multi-sensor observations, as well as robot-navigation corpora such as `HuRoN`[117], `RECON`[86] and `SCAND`[49]. This type of data is especially valuable because it contains genuine embodiment noise, actuation errors, sensor artifacts, and environment dynamics that are difficult to model faithfully in simulation. Compared with synthetic corpora, the key advantage of this class is not annotation scale but *behavioral realism*: the recorded trajectories reflect how real embodied agents actually move, recover, and interact under deployment constraints.

Web-scale video as implicit navigation supervision. A second source of real-world data comes from *web videos*, which provide massive visual coverage but do not directly expose executable trajectories. In this case, the core challenge is to recover navigation-relevant supervision from passive video. Recent works therefore convert video into pseudo-trajectories using 3D reconstruction, camera pose estimation, SLAM-style processing, or inverse dynamics. `NaVILA`[1] mines 2K YouTube room-tour videos and extracts 20K trajectories using entropy-based sampling and `MASt3R`[118]-based pose estimation. `RoomTour3D`[90] reconstructs 3D geometry from room-tour videos and derives trajectory-instruction pairs from the resulting camera motion. `LeLaN`[119] learns from a mixed real-world corpus with synthetically generated action labels and language prompts, including both robot data and YouTube videos, while `CityWalker`[88] turns large-scale urban walking and driving videos into outdoor navigation supervision. `NavFoM`[9] further scales this paradigm with `Sekai`[120], a large web-video corpus used for navigation-oriented pretraining. Conceptually, these methods do not provide native simulator trajectories, but they recover enough motion, geometry, and scene structure to supply useful navigation priors at web scale.

4.1.3. General Multimodal Data

General multimodal data are used as auxiliary supervision beyond navigation trajectories. They mainly contribute general visual-language knowledge, semantic priors, and reasoning ability. Representative embodied, video-QA, and general multimodal datasets are summarized in Table 5.

Not all data used by navigation foundation models are trajectory supervision. A growing fraction of recent systems are trained with *auxiliary multimodal data* whose purpose is to preserve general visual-linguistic competence, improve reasoning, or align behavior with social constraints. These data include general VQA, image captioning, OCR, grounding, interleaved image-text corpora, video QA, and explicit reasoning annotations, and they are typically mixed with navigation data rather than replacing it. For example, `Uni-NaVid`[60] adds 2.3M public video question-answering samples on top of its navigation corpus; `NaVILA` [1] supplements simulated and web-derived trajectories with auxiliary navigation datasets such as `ScanQA`[91], as well as general VQA; `StreamVLN`[67] mixes video QA from `LLaVA-Video-178K`[95] and `ScanQA` with `MMC4`[97]-style interleaved image-text data; and `NavFoM` combines 8.02M navigation samples with 3.15M image QA and 1.61M video QA samples to maintain broader multimodal knowledge. A newer trend is to make reasoning itself part of the supervision recipe: `ABot-N0` introduces 5.0M cognitive reasoning samples for navigable-area analysis, social navigation CoT, instruction reasoning, object-goal reasoning, and POI grounding; and `SocialNav`[72] builds a socially-aware data mixture that combines cognitive activation data with trajectory supervision from simulation, internet videos, and real robots. Unlike synthetic or real trajectory corpora, these datasets are not primarily used to teach low-level navigation trajectories, but to endow the model with richer semantic priors, better instruction understanding, stronger decision-making ability, and more socially aligned behavior.

Table 4 | Summary of representative **real-world** and **web-video** datasets for embodied navigation. Here, *inst.* denotes instruction, and *traj.* denotes trajectory.

Name	Description	Quantity	Data Structure	Used by
Real-World Data				
GoStanford[121]	Early large-scale robot navigation logs	32 scenes / 24h / 256K images	RGB, traj.	InternVLA-N1[4]
RECON[86]	Autonomous real-world exploration and navigation logs	9 scenes	RGB, traj.	ABot-N0[2], SocialNav[72]
SCAND[49]	Socially compliant robot navigation demonstrations	8.7h / 138 traj.	RGB, pose, action traj.	ABot-N0[2], SocialNav[72]
HuRoN[117]	Human–robot navigation demonstrations in crowded scenes	5 scenes / 3 buildings / 75h / 58 km	RGB, traj.	ABot-N0[2], SocialNav[72]
nuScenes[87]	Large-scale autonomous driving sensor-log dataset	1K scenes / 1.4M images	Multi-view RGB, LiDAR, radar, ego pose, map	NavFoM[9]
OpenScene[116]	Large-scale real-world driving occupancy and planning corpus	654K samples	Multi-view RGB, vehicle state, traj.	NavFoM[9]
Web Video Data				
CityWalker[88]	Socially aware urban navigation from web-scale city videos	2000+h web videos	Egocentric video, traj., urban control	ABot-N0[2], SocialNav[72]
Youtube-HT[89]	Internet video navigation corpus with trajectories	1,387 scenes / 119h / 550K images	RGB video, traj.	InternVLA-N1[4]
Youtube-VLN[122]	Web-video VLN corpus with instructions	4,078 scenes / 433h / 587K images / 14K inst.	RGB video, inst., traj.	InternVLA-N1[4]
RoomTour3D[90]	Geometry-aware room-tour video dataset for VLN tuning	100K traj. / 200K inst. / 1,847 scenes	RGB video, 3D reconstruction, traj., inst.	RoomTour3D[90]
Sekai[120]	World-exploration web-video corpus for navigation pretraining	182K videos / 2.03M navigation samples	RGB video, traj., inst.	NavFoM[9]

4.2. Training Strategies

This subsection reviews training strategies for navigation foundation models from three perspectives. We first discuss how models learn navigation policies directly, then summarize auxiliary objectives that improve intermediate reasoning and spatial learning, and finally examine how general vision-language tasks are integrated to maintain broader multimodal capabilities.

4.2.1. Navigation Capability Acquisition

Discrete action learning. A large family of navigation models formulates action prediction as autoregressive generation over a small discrete action vocabulary, typically including primitives such as *FORWARD*, *TURN-LEFT*, *TURN-RIGHT*, and *STOP*. Since most recent navigation models are built on top of pretrained LLM/VLM backbones, these actions are usually represented directly in textual form or as a small set of dedicated action tokens, so that navigation can be naturally cast as standard next-token prediction (NTP) and directly reuse the language modeling interface of the backbone. In this setup, the model predicts either the next single action or a short future action sequence, and training reduces to language-model-style supervision on action tokens. This formulation is attractive because it is simple, stable, and easy to integrate with instruction-conditioned visual reasoning. Representative examples include `Uni-NaVid` and `StreamVLN`, which autoregress over discrete navigation actions from video observations and instructions; `VLN-R1`[62] and `Nav-R1`[74], which combine

Table 5 | Summary of representative **general multimodal** datasets for embodied navigation. Here, *QA* denotes question-answering pairs.

Name	Description	Quantity	Used by
Embodied Multimodal Data			
ScanQA[91]	3D scene-grounded QA dataset for spatial understanding	41K+ QA / 800 scans	NaVILA[1], StreamVLN[67], Uni-NaVid[60], VLingNav[11], ABot-N0[2]
SQA3D[123]	Situated 3D QA benchmark	33.4K QA / 6.8K situations	Nav-R1[74]
MP3D-EQA[124]	Embodied QA benchmark in MP3D scenes	1,136 QA / 83 environments	Uni-NaVid[60]
OpenEQA[92]	Embodied QA benchmark	1,600+ QA / 180+ real-world environments	Uni-NaVid[60]
Nav-CoT-110K[74]	Embodied CoT corpus synthesized from VLN and ObjectNav sources	110K CoT traj.	Nav-R1[74]
Nav-R2 CoT Dataset[64]	OVON-specific structured CoT corpus	300K CoT samples	Nav-R2[64]
SocNav CAD[72]	Social-alignment reasoning mixture for navigation models	1.2M traversability / 825K CoT / 1M VQA	SocialNav[72]
ABot-N0 Reasoning Dataset[2]	Large-scale embodied cognitive reasoning corpus	5.0M reasoning samples	ABot-N0[2]
Video QA and Video Reasoning Data			
LLaVA-Video-178K[95]	Synthetic video instruction-tuning corpus	178K videos / 1.3M video instructions	StreamVLN[67], VLingNav[11]
Panda-70M[94]	Large-scale video–text caption corpus	70M video–text pairs	Uni-NaVid[60], TrackVLA[45]
Video-R1[125]	Video reasoning corpus for CoT-style supervision	165K CoT SFT / 260K RL	VLingNav[11]
MSVD-QA[93]	Short-video QA benchmark	1,970 videos / 50.5K QA	Uni-NaVid[60]
MSRVTT-QA[93]	Video QA benchmark	10K videos / 243K QA	Uni-NaVid[60]
ActivityNet-QA[126]	Long-video QA benchmark	5.8K videos / 58K QA	Uni-NaVid[60]
General Vision–Language Data			
MMC4[97]	Interleaved image–text web corpus	101.2M docs / 571M images / 43B tokens	StreamVLN[67]
COCO2014 / COCO2017[96]	General image-caption and object-centric visual corpora	330K images / >200K labels	ABot-N0[2]
RefCOCO Series[127, 128]	Referring expression grounding benchmarks	379K referring expressions	ABot-N0[2]
Objects365[98]	Large-scale detection and grounding corpus	600K images / 365 categories	ABot-N0[2]
MAmmoTH-VL[129]	Multimodal reasoning and instruction corpus	12M QA pairs	ABot-N0[2]

discrete action prediction with subsequent reinforcement fine-tuning; and a number of later VLN-style models such as `MapNav`[68], `JanusVLN`[5], `RynnBrain`[130], `Nav-R`^{2}[64], `MapDream`[66], and `NaVid-4D`[61], all of which keep the action space close to language tokens rather than regressing continuous trajectories. Overall, discrete action learning remains a competitive design when one wants to maximize compatibility with pretrained LLM/VLM backbones and preserve a clean language-model-style training pipeline.

Continuous action learning. Recent navigation foundation models increasingly move beyond small discrete action vocabularies and instead learn to predict continuous actions or short-horizon trajectories. A common way to do this is to keep the overall training interface autoregressive: continuous motions are expressed in a *text-like* form, such as parameterized turning angles, movement distances, or waypoint coordinates, and are then learned with standard NTP, much like ordinary LLM/VLM decoding. This strategy is adopted by models such as NaVid[59], NaVILA, and Hydra-Nav[10], which represent motion continuously in textual form, as well as OctoNav[63], OpenVLN[131], AstraNav-Memory[3], and VLA-AN[7], whose outputs likewise remain close to an autoregressive token-prediction interface.

Some works retain this language-model-like interface while introducing an additional discretization step: instead of expressing actions directly as natural text, they first tokenize a short continuous trajectory into a sequence of action tokens and then train the model autoregressively over these tokens. This design still benefits from the simplicity of NTP, while allowing richer control than a small fixed action vocabulary. AutoFly[73] is a representative example of this tokenized continuous-action recipe.

Other models move away from token prediction more explicitly and predict continuous waypoints or trajectory chunks through a dedicated *regression head*. In these cases, the hidden states of the VLM/LLM backbone are passed to an MLP-style action head, and supervision is applied directly in continuous space, typically with regression losses such as MSE. This design appears in NavFoM, VLingNav, NavForesee[65], RoboTron-Nav[70], UrbanVLA[71], and OmniVLA[132], all of which learn waypoint sequences or short trajectory chunks without reducing the problem to pure token generation.

Beyond deterministic prediction, recent work increasingly formulates continuous navigation as a generative trajectory modeling problem. Diffusion-style policies are particularly appealing because navigation is inherently multi-modal: under the same observation and instruction, multiple future paths may all be plausible. In this setting, the policy learns to generate a short continuous trajectory, rather than making a single deterministic prediction. InternVLA-N1[4] employs a diffusion policy trained with a DDPM-style objective for continuous action chunks conditioned on high-level goals, while TrackVLA[45] follows the same DDPM-style training formulation but uses DDIM-style sampling at inference time for embodied visual tracking. More recently, several navigation foundation models have shifted toward *flow matching*, including DualVLN[75], OmniNav[76], ABot-N0, SocialNav, and SparseVideoNav[6], which generate continuous trajectory chunks under conditioning signals such as intermediate goals, task decomposition, or future prediction.

Reward-driven policy refinement. Although supervised fine-tuning and imitation learning provide a strong initialization for navigation policies, they optimize mainly stepwise imitation and therefore do not directly address rollout errors, recovery behavior, or long-horizon control quality. Many recent works consequently use *reward-driven policy refinement* as a post-training stage for action or trajectory generation itself. In discrete-action settings, VLN-R1 applies GRPO-based reinforcement fine-tuning with a Time-Decayed Reward over future action correctness, Nav-R1 optimizes navigation behavior with rewards tied to path fidelity and endpoint accuracy, and OctoNav uses reward-based post-training to refine action and magnitude predictions before a final online RL stage under environment feedback. Similar trends also appear in continuous-control models: VLingNav adds an expert-guided reinforcement stage on top of direct trajectory supervision, while OpenVLN and UrbanVLA use reinforcement-style post-training to improve long-horizon waypoint or trajectory generation. This pattern also extends to recent flow-matching models: ABot-N0 adopts SAFE-GRPO as a third-stage post-training alignment procedure, while SocialNav uses SAFE-GRPO to refine socially compliant trajectory generation after imitation learning. In this sense, reinforcement learning is best understood here as a later-stage mechanism for refining navigation policy quality.

4.2.2. Auxiliary Task Learning

Grounded intermediate targets and control-oriented auxiliaries. Many recent navigation models do not learn action generation alone, but introduce auxiliary targets that bridge high-level planning and low-level action generation. A representative example is explicit intermediate-goal grounding. InternVLA-N1 decouples planning and execution via pixel-goal planning, and the released training pipeline directly supports System 2 training for 2D pixel-goal prediction before joint training with System 1 for action generation. DualVLN follows the same overall philosophy: its slow system is trained for pixel-goal grounding and iterative view adjustment, while the grounded outputs are not directly fed as raw coordinates into the trajectory generator. Instead, they are transformed through learnable queries that extract the relevant information from the backbone representations and generation outputs, and these query features are then used as conditioning signals for the

diffusion-style trajectory model. More generally, this family of auxiliaries strengthens spatial alignment before end-to-end action learning, making the final policy easier to optimize and more interpretable.

Reasoning, planning, and task decomposition. A second major auxiliary direction is to teach the model to *think before acting*. However, recent work suggests that reasoning must be integrated carefully. `Aux-Think`[80] shows that directly forcing test-time CoT can hurt VLN performance, and instead proposes using CoT as an auxiliary supervision signal during training while keeping online inference action-centric. Building on this intuition, several recent models introduce *explicit reasoning supervision* during training. `Nav-R1` constructs a cold-start CoT dataset and then jointly optimizes reasoning and action with rewards over format, semantic understanding, and navigation quality. `OctoNav` introduces a Think-Before-Action CoT dataset and corresponding TBA-SFT stage, using reasoning traces to improve generalist navigation before later policy refinement. `VLingNav` proposes Adaptive CoT (AdaCoT), while `Hydra-Nav` adopts a slow-fast reasoning design that learns not only *how* to reason but also *when* reasoning should be triggered. Other works specialize reasoning to particular navigation regimes: `ABot-N0` builds large-scale task-specific reasoning supervision for instruction following, object search, social navigation, and POI grounding; `SocialNav` adds social reasoning signals such as CoT explanations and social traversability prediction to encourage norm-aware behavior; `Nav-R`^{2} uses structured CoT to model target-environment relations and environment-action planning for open-vocabulary object navigation; and `FantasyVLN`[8] extends reasoning supervision to unified textual, visual, and multimodal CoT, so that the model can learn grounded long-horizon reasoning beyond pure text. At a higher level of abstraction, some models use auxiliary supervision for *planning and task decomposition* rather than only stepwise CoT. `OmniNav` explicitly decomposes long-horizon navigation into subtasks across instruct-goal, object-goal, point-goal, and frontier exploration, while `NavForesee` trains the model to decompose instructions, track progress, and generate concise textual sub-plans before action execution. Together, these works indicate that reasoning auxiliaries are most effective when they are selective, structured, and grounded in navigation-relevant abstractions such as subgoals, social norms, object-context relations, or future plans, rather than simply increasing the amount of text emitted at inference time.

World models, future prediction, and map learning. A third auxiliary direction aims to improve navigation by explicitly modeling *what lies ahead* or *what intermediate spatial structure should be built*. One branch focuses on future prediction as an auxiliary signal for planning. `NavForesee` is a representative example: besides hierarchical planning, it trains a dual-horizon predictive mechanism that models both short-term environmental dynamics and long-term navigation milestones, effectively turning future prediction into an auxiliary signal for downstream action generation. Along a similar line, `SparseVideoNav` uses sparse future video generation to support beyond-the-view navigation, showing that imagined future observations can provide a long-horizon planning prior without requiring dense step-by-step supervision. `FantasyVLN` extends this idea further by introducing multimodal CoT supervision that includes not only textual reasoning but also image-level CoT, so that the model can explicitly generate visual future hypotheses as part of navigation reasoning. Another branch focuses on explicit spatial structure construction. `MapDream` treats map learning as an auxiliary objective by training a dedicated map module to generate occupancy-style or BEV-like spatial representations, which are then fed back to the navigation policy to improve action prediction. Although these methods differ in representation—future trajectories, future frames, visual hypotheses, or explicit maps—they share the same intuition: navigation can be improved by learning intermediate predictive or spatial structure, rather than forcing the policy to infer everything implicitly from short-horizon observations alone.

Reward-based auxiliary alignment. Reward optimization is also increasingly used to align *auxiliary objectives* that are hard to supervise with fixed labels alone. In reasoning-centric models, `Nav-R1` uses reward signals on output format and embodied understanding, so reinforcement learning directly regularizes the structure and semantic quality of its reasoning traces. In socially aware navigation, `SocialNav` and `ABot-N0` use socially grounded rewards to reinforce norm awareness and social compliance, encouraging the model to internalize social rules rather than merely imitate demonstrations. Reward-based alignment can also be applied to intermediate representations: `MapDream` jointly fine-tunes its map module under navigation rewards so that the learned spatial representation becomes more useful for downstream decision making. Overall, reinforcement learning is valuable here because it can refine auxiliary targets such as reasoning coherence, social alignment, and map usefulness, all of which are difficult to specify fully through static supervision.

4.2.3. Vision-Language Task Learning

Why navigation models still need general VLM tasks. Most recent navigation models are built on LLM/VLM backbones whose main advantage is broad visual-linguistic generalization. However, direct adaptation to

navigation introduces a clear capability gap: the model must preserve general semantics while also acquiring embodied skills such as spatial grounding, egocentric perception, temporal memory, and instruction-conditioned decision making. As a result, many works mix navigation training with additional vision-language tasks such as VQA, EQA, captioning, grounding, OCR, or interleaved image-text modeling. The objectives are typically twofold: first, to avoid degrading the pretrained VLM into a narrow action predictor; and second, to inject embodied-relevant semantic and spatial priors that make navigation learning easier and more robust.

Typical integration patterns. One common pattern is *joint multitask training* throughout navigation learning. `Uni-NaVid` supplements its navigation corpus with large-scale public VideoQA and embodied QA style data. `StreamVLN` mixes navigation data with LLaVA-Video-178K, ScanQA, and MMC4-style interleaved image-text data to preserve general multimodal competence while improving streaming navigation. `NavFoM` explicitly combines millions of navigation samples with image QA and video QA, reflecting a strong commitment to retaining broad VLM priors in a cross-task navigation model. `NaVILA` similarly mixes navigation supervision with general VQA-style data and navigation-related auxiliary corpora to improve spatial reasoning and maintain transferability across robots. Another common pattern is *stage-wise warm-up*. `VLA-AN` uses a progressive multi-stage recipe that first strengthens scene comprehension and reasoning before full navigation learning; `ABot-N0` also includes an explicit cognitive warm-up through general VLM-style supervision before specializing its action expert; and `OmniNav` argues that image captioning and visual recognition data are essential for improving object understanding and instruction robustness in unified navigation. In short, modern navigation models are rarely trained on trajectories alone. Instead, they are typically post-trained as *navigation-specialized VLMs*, where trajectory learning, auxiliary reasoning, and general vision-language tasks are interleaved to balance specialization and generalization.

4.3. Takeaway Insights

Despite the rapid growth of navigation datasets and data engines, current embodied navigation models still face substantial generalization bottlenecks caused by dataset biases and leakage. These issues are particularly important because many recent models are trained at scale and may achieve strong benchmark performance while still relying on shortcuts induced by data construction rather than learning robust embodied navigation skills.

Shortest-path bias. A large fraction of synthetic navigation data is generated from shortest-path or near-shortest-path planners. This makes data collection efficient and evaluation well-defined, but it also biases the training distribution toward overly optimal trajectories. In real environments, however, successful navigation often requires behavior that deviates from the shortest path, such as pausing to observe, recovering from execution errors, detouring around dynamic obstacles, or choosing safer and more socially acceptable routes. A model trained predominantly on shortest-path demonstrations may therefore overfit to path optimality rather than learning robust exploration and recovery strategies. This issue is especially relevant for benchmark-style datasets such as `R2R`, `RxR`, `VLN-CE`, and many planner-generated ObjectNav corpora, where the expert policy is strongly tied to shortest-path planning.

Static-world bias. Most current simulation-based datasets assume that the world is effectively static: objects remain fixed, doors are typically in predefined states, pedestrians are absent or simplified, and the environment does not change over time. Such assumptions make annotation and benchmarking easier, but they create a mismatch with deployment settings in which embodied agents must interact with moving people, dynamic obstacles, changing visibility, or environment state changes. Even when dynamic elements are added, they are often much simpler than real human behavior. As a result, models trained on largely static worlds may struggle with online adaptation, reactive avoidance, or socially compliant motion once transferred to real-world operation.

Action-frequency mismatch. Another common issue is imbalance in the empirical action distribution. In many navigation datasets, especially those derived from shortest paths, some actions such as moving forward occur much more frequently than others such as stopping, rotating in place, or taking corrective maneuvers. This induces a frequency bias in the learned policy: the model may become overly confident in frequent actions while undertraining rare but important behaviors. The mismatch becomes even more problematic when the deployment setting requires behaviors that are uncommon in the training data, such as frequent replanning, waiting, obstacle negotiation, or fine-grained control near the goal.

Split leakage and weak generalization protocols. Navigation datasets can also suffer from leakage between

training and evaluation splits. In the simplest case, the same or nearly identical scenes may appear across splits. But leakage can also be more subtle: trajectories may overlap heavily in geometry, instructions may reuse highly similar templates, or the same environment may appear under slightly different task instantiations. Because navigation is inherently structured around scene layout, even partial overlap at the scene or trajectory level can significantly inflate benchmark performance. In practice, scene-level leakage and trajectory-level leakage are often entangled: a model that repeatedly sees the same environment may implicitly memorize its layout, landmarks, or object priors, even if the exact test path is new. This weakens the reliability of reported generalization results.

Biases in language and annotation. Instruction-conditioned datasets are additionally affected by annotation biases. Human-written instructions often exhibit repeated templates, common reference patterns, and dataset-specific lexical shortcuts. Synthetic instruction generation pipelines may introduce their own regularities, such as overly literal descriptions, templated phrasing, or mismatches between language and embodied uncertainty. As a result, models may partially solve the task through language priors or annotation artifacts rather than grounded navigation. This issue is especially important for recent large VLM-based models, since they are powerful enough to exploit correlations in wording without necessarily grounding them in the environment.

Embodiment mismatch and cross-embodiment transfer bias. A further bottleneck arises in cross-embodiment settings. Navigation datasets are often collected under one embodiment assumption—for example, a wheeled indoor robot, a legged robot, a first-person human camera, a UAV, or a car-like platform. Even when trajectories are represented in a shared form, the underlying control constraints, sensing geometry, and feasible motion patterns can differ substantially across embodiments. This mismatch is not limited to action space alone: it also includes the sensing setup itself, such as the number of cameras, camera placement, field of view, height, intrinsics, extrinsics, and the degree to which the platform observes the world panoramically or egocentrically. As a result, a policy trained on one embodiment may inherit strong biases about what observations look like, what actions are physically feasible, and how motion unfolds over time. These issues are especially pronounced in recent generalist navigation models that attempt to unify VLN, ObjectNav, UAV navigation, tracking, and driving within a single backbone. In such cases, the broader the embodiment space, the greater the risk that the model learns shallow correlations tied to a particular embodiment or sensor configuration rather than transferable navigation principles.

Consequences for real-world generalization. Taken together, these biases and leakage issues mean that strong benchmark performance should not be interpreted as evidence of robust real-world navigation. A model may perform well because it has learned shortest-path imitation, static-scene regularities, skewed action statistics, partial scene memorization, dataset-specific annotation patterns, or embodiment-specific sensing shortcuts, while still failing under realistic deployment conditions. This gap is especially important for foundation-model-style navigation systems, since scaling data and model capacity does not automatically remove underlying dataset biases. Instead, larger models may exploit them even more efficiently. Improving real-world generalization therefore requires not only more data, but also better data construction protocols, stronger split design, richer dynamic environments, broader embodiment coverage, and evaluation settings that reduce shortcut learning.

5. Efficient Deployment

The deployment of foundation models for embodied navigation on physical agents faces strict Size, Weight, and Power (SWaP) constraints. Unlike cloud-based methods that can tolerate high latency and rely on massive computing clusters, physical robots operate in dynamic, irreversible environments requiring low inference latency, high energy efficiency, and deterministic control. Therefore, bridging the gap between heavy foundation models and constrained edge hardware has become a critical research frontier. This section reviews efficient deployment for embodied navigation from two perspectives: real-world deployment across robotic embodiments (§5.1), and acceleration techniques spanning model design and software-system optimization (§5.2).

5.1. Embodiment-Specific Deployment

Different robotic embodiments provide distinct real-world testbeds for embodied navigation, with differences in payload, mobility, sensing, and control requirements. Existing work has been deployed across a range of physical platforms, including wheeled robots, legged robots, and UAVs. Table 6 summarizes representative deployment platforms and hardware setups reported in the literature.

Table 6 | Summary of representative embodiment-specific deployment settings for embodied navigation.

Platform	Computing Hardware	Deployed Models
Wheeled Robots		
WHEELTEC R550	Remote server	Nav-R1[74]
Galbot G1	Remote server (RTX 5090)	NavFoM[9]
Legged Robots		
Unitree Go2	Remote server (RTX 4090 / RTX 5090 / A6000)	InternVLA-N1[4], NaVILA[1], VLingNav[11], MapNav[68], OctoNav[63], UrbanVLA[71], NavFoM[9], TIC-VLA[133]
Unitree Go2	Remote server (RTX 4090), NVIDIA Jetson Orin NX	ABot-N0[2]
Unitree G1	Remote server (RTX 4090 / RTX 5090)	InternVLA-N1[4], DualVLN[75], NavFoM[9]
WR-2	Remote server	Quar-VLA[134]
UAVs		
Micro Quadrotor	NVIDIA Jetson Orin NX	VLA-AN[7]
Standard Quadrotor	Remote server (RTX 4090 / RTX 5090 / A40)	AutoFly[73], SkyVLN[135], OpenVLN[131], NavFoM[9], LongFly[136]

5.1.1. Wheeled Robots

Wheeled robots are common real-world platforms for embodied navigation deployment. `Nav-R1` [74] is deployed on the WHEELTEC R550 using a remote server for its Fast-in-Slow reasoning framework, while `X-Nav` [137] reports local deployment on wheeled platforms such as the TurtleBot 2 and Clearpath Jackal. In addition, `NavFoM` [9] is reported on the Galbot G1 platform with inference supported by a remote RTX 5090 server.

5.1.2. Legged Robots

Legged robots are another major platform category for real-world embodied navigation deployment. Among them, the Unitree Go2 is one of the most frequently used quadrupedal platforms in recent work. Representative Go2 deployments include models such as `InternVLA-N1`, `NaVILA`, and `NavFoM` [1, 4, 9], alongside many other recent systems reported on the same platform [5, 6, 10, 11, 45, 60, 63, 67, 68, 71, 72, 75, 133]. Most of these works are reported with remote-server support, while `ABot-N0` [2] adopts a hybrid setup that combines a remote RTX 4090 server with an onboard NVIDIA Jetson Orin NX.

Additional quadrupedal deployments are reported on other platforms. For example, `Quar-VLA` [134] is deployed on the WR-2 platform, and `OmniNav` [76] is also reported on a quadrupedal platform with cloud-side inference. Real-world legged deployment is also reported beyond quadrupeds. In particular, `InternVLA-N1` [4], `DualVLN` [75], and `NavFoM` [9] are reported on the Unitree G1 humanoid, while `MapDream` [66] demonstrates zero-shot transfer on the Unitree G1 platform. In addition, `NaVILA` [1] evaluates real-world navigation on the Booster T1 humanoid, and `OmniVLA` [132] reports deployment on the Unitree Go1.

5.1.3. UAVs

UAVs present the most extreme SWaP constraints, requiring a delicate balance between onboard real-time control and off-board heavy reasoning. To minimize latency for high-speed flight, `VLA-AN` [7] is deployed natively on a `Micro Quadrotor` powered by an `NVIDIA Jetson Orin NX`, achieving efficient closed-loop control without external dependencies.

Conversely, for tasks involving complex 3D environments and long-horizon planning, many frameworks utilize the `Standard Quadrotor` platform coupled with powerful ground stations. Models such as `AutoFly` [73], `SkyVLN` [135], `OpenVLN` [131], `NavFoM` [9], and `LongFly` [136] offload their perception and planning pipelines to remote servers equipped with RTX 4090, RTX 5090, or NVIDIA A40 GPUs. These setups enable the generation of feasible aerodynamic trajectories within millisecond constraints by streaming sensory data to

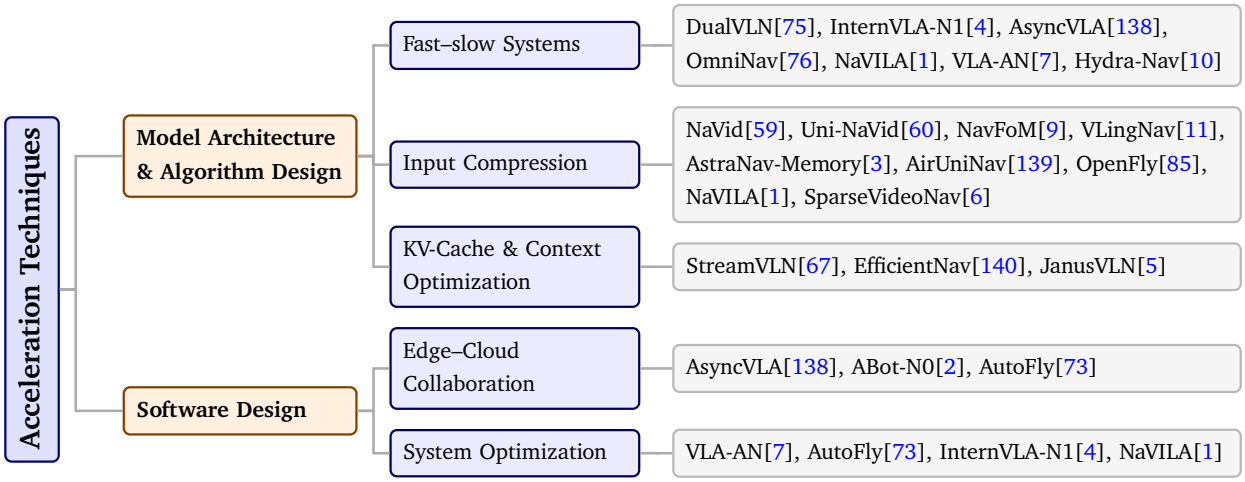

Figure 8 | Summary of acceleration techniques for embodied navigation systems and representative works.

high-compute clusters and returning velocity commands to the UAV's flight controller.

5.2. Acceleration Techniques

Deployment-oriented acceleration has become an increasingly important design axis in navigation foundation models, because embodied agents must maintain *high control frequency*, *long-context perception*, and *hardware feasibility* under constrained deployment resources. Existing efforts can be organized into two broad directions. The first accelerates inference through model architecture and algorithm design, including fast–slow system decomposition, input compression, and context-aware cache optimization. The second improves software-level deployment efficiency through edge–cloud collaboration and systems-level inference engineering. Figure 8 summarizes existing acceleration techniques and representative works.

5.2.1. Model Architecture & Algorithm Design

Fast–slow systems. The first line of work accelerates navigation through explicit separation between *slow deliberation* and *fast execution*. Instead of requiring a single large model to handle long-horizon reasoning and high-frequency control at every step, these methods decompose the system into a low-frequency planning module and a lightweight execution branch. Representative examples include `DualVLN` [75] and `InternVLA-N1` [4], which separate high-level grounding from fast trajectory generation, as well as `AsyncVLA` [138] and `OmniNav` [76], which further emphasize *asynchronous* or *fast–slow coordinated* control. Similar structural designs also appear in `NaVILA` [1], `VLA-AN` [7], and `Hydra-Nav` [10]. The common principle is to avoid invoking expensive reasoning at every control cycle, thereby improving online responsiveness.

Input compression. The second family accelerates inference by reducing the amount of visual information that enters the model. The core idea is to control the *input token budget* through *token compression*, *frame sampling*, *keyframe selection*, or other compact history representations. `NaVid` [59] and `Uni-NaVid` [60] are representative works on history compression, with the former using asymmetric token allocation for current and historical frames, and the latter introducing online token merging for long-horizon video observations. Later methods make budgeted compression more explicit: `NavFoM` [9] uses budget-aware temporal sampling, `VLingNav` [11] combines dynamic FPS sampling with adaptive pooling, and `AstraNav-Memory` [3] aggressively compresses each frame into a compact memory token set. Similar ideas also appear in `AirUniNav` [139], `OpenFly` [85], and `NaVILA` [1], which reduce history cost through dynamic frame selection or history sampling. `SparseVideoNav` [6] extends this line by sparsifying future modeling itself. Overall, these methods accelerate deployment by shrinking the visual context before or during model ingestion.

KV-cache and context optimization. The third category focuses on optimizing *how long context is maintained and reused* at inference time. Rather than directly reducing raw inputs, these methods improve *cache efficiency*, *context organization*, and *incremental update* so that long-horizon reasoning does not induce linearly increasing latency. `StreamVLN` [67] is the clearest example, combining slow–fast context modeling with sliding-window KV-cache reuse and pruning strategies for streaming navigation. `EfficientNav` [140] improves on-device planning through discrete memory caching, clustered memory organization, and semantics-aware retrieval.

JanusVLN [5] introduces dual implicit memory and a hybrid cache update mechanism for spatial and semantic history. The common objective of this line is to make *context reuse* efficient enough for long-horizon online deployment.

5.2.2. Software Level Design

Edge–cloud collaboration. At the software level, one important direction is *edge–cloud collaboration*, where heavy semantic reasoning or task planning is offloaded to a remote server while time-critical execution remains on the edge device. AsyncVLA [138] is the clearest example: the large VLA runs remotely, whereas the edge-side adapter executes fast local correction to mitigate both model and communication latency. ABot-N0 [2] follows a related *hybrid cloud-edge* pattern in real-world deployment: although the navigation model itself runs locally on Jetson Orin NX together with the neural controller, a cloud-side agentic planner performs high-level planning and dispatches executable sub-tasks to the onboard system. AutoFly [73] adopts distributed deployment for UAV navigation, offloading model inference to a remote server and combining it with multi-process pipelined inference for real-time control. Overall, these systems improve deployment efficiency by balancing semantic capacity, planning complexity, and control responsiveness across heterogeneous computing nodes.

Systems-level optimization. The second software-level direction improves efficiency through *systems engineering* rather than model redesign. Typical techniques include *accelerated kernels*, *operator fusion*, *pipeline scheduling*, and *low-bit deployment*. VLA-AN [7] is a representative example, reporting substantial onboard speedups through Flash-Attention, fused operators, KV-cache preloading, and CUDA graph scheduling. AutoFly [73] similarly combines ONNX conversion, TensorRT acceleration, custom CUDA operators, and multi-process pipelining to reduce end-to-end latency. InternVLA-N1 [4] applies TensorRT and asynchronous execution to speed up its fast control branch, while NaVILA [1] shows that *quantization* can significantly reduce memory usage and latency. Compared with architectural methods, this line of work is primarily concerned with transforming navigation models into efficient *real-world deployment systems*.

5.3. Takeaway Insights

Efficient deployment in embodied navigation is fundamentally a co-design problem across embodiment, model architecture, and inference systems. In practice, there is no single deployment recipe that works uniformly across embodiments. Wheeled and legged robots often rely on remote or hybrid execution to sustain the semantic capacity of large VLMs and VLAs, whereas UAVs face much tighter SWaP and control-frequency constraints, making onboard efficiency and tightly pipelined execution especially critical. This means that deployment choices are increasingly shaped by the physical properties of the robot itself, including payload budget, locomotion dynamics, sensing layout, and acceptable control latency.

At the algorithmic level, recent acceleration methods reveal a common principle: *computation should be allocated selectively across time, context, and hardware*. Fast–slow architectures reduce the frequency of expensive reasoning, input compression controls the visual token budget before it becomes a systems bottleneck, and KV-cache or memory optimization prevents long-horizon context from causing linearly growing inference cost. At the software level, edge–cloud collaboration and systems-level engineering further show that deployment efficiency is not determined only by model size, but also by how inference is partitioned, scheduled, quantized, and executed across heterogeneous hardware. In this sense, practical embodied deployment is increasingly shifting from monolithic inference toward *hierarchical, asynchronous, and hardware-aware execution*.

Taken together, these trends indicate that the central deployment trade-off is not simply between *large models* and *small devices*, but among *semantic capability*, *real-time responsiveness*, and *deployment autonomy*. Stronger reasoning and longer context generally improve navigation quality, but they also increase latency, energy consumption, and dependence on external compute. Future progress will therefore depend on embodiment-aware co-design of model architecture, memory management, and inference systems, so that navigation foundation models can preserve long-horizon intelligence while remaining feasible for reliable real-world deployment.

6. Benchmarks and Evaluation Metrics

Traditional navigation evaluation has largely emphasized whether the agent reaches the target and whether the resulting trajectory is efficient. For embodied navigation, however, evaluation must cover a broader set of capabilities. Beyond navigation success, a complete assessment should also examine whether the model

Table 7 | Representative embodied navigation benchmarks. *Embodi.* denotes the embodiment type.

Benchmark	Domain	Embodi.	Target Capability	Evaluation Metrics
Instruction-Conditioned Navigation				
R2R [31]	Indoor	Ground	Canonical indoor VLN with route following in scanned homes	SR, NE, SPL, nDTW, sDTW
RxR [32]	Indoor	Ground	Multilingual route following with dense spatiotemporal grounding	SR, NE, SPL, nDTW, sDTW
CityWalker [88]	Outdoor	Ground	Urban real-world instruction following from web-scale videos	navigation success rate, AOE, MAOE
TravelUAV [113]	Aerial	UAV	Assistant-guided aerial VLN with realistic flight dynamics and 6-DoF control	NE, SR, OSR, SPL
Goal-Conditioned Navigation				
Habitat ObjectNav v2 [36]	Indoor	Ground	Category-goal semantic search in unseen indoor scenes	SR, SPL, DTS
HM3D-OVON [37]	Indoor	Ground	Open-vocabulary object-goal navigation under semantic shift	SR, SPL
GOAT-Bench [101]	Indoor	Ground	Multimodal lifelong navigation across category, image, and language goals	SR, SPL, GcS
NAVSIM [141]	Outdoor	Ground	Driving-scale goal-directed planning under urban open-world variation	PDMS, NC, DAC, TTC, EP
Navigation-Related Embodied Reasoning				
OpenEQA [92]	Indoor	Ground	Active exploration for open-ended embodied question answering	LLM-Match, LLM-Match×SPL
MP3D-EQA [124]	Indoor	Ground	Embodied question answering with explicit exploration burden	answer accuracy, d_T, d_Δ, $d_{\min}$
Interactive and Dynamic Navigation				
SocNavBench [142]	Indoor & Outdoor	Ground	Crowd-aware motion under social-compliance constraints	success, collisions, path quality, interaction statistics
Social-VLN [75]	Indoor & Outdoor	Ground	Instruction following amid dynamic pedestrians and social rules	NE, OS, SR, SPL, HCR
Habitat 3.0 (Human Following) [143]	Indoor	Ground	Moving-target finding and following in human-populated scenes	find rate, follow success, tracking error
Cross-Embodiment Evaluation				
VLN-PE [34]	Indoor	Mixed	Physically realistic VLN under embodiment, controller, and lighting shift	TL, NE, OS, SR, SPL

follows instructions faithfully, grounds its decisions in relevant semantics, generalizes across environments and embodiments, operates safely under realistic constraints, and remains practical for real-time deployment. As a result, benchmark design and metric selection in this setting are closely tied to the specific capabilities that each task intends to test. This section first organizes existing benchmarks by task type and highlights the distinct evaluation objectives associated with each category (§6.1). It then summarizes the major families of evaluation metrics, including task success, trajectory quality, instruction or semantic alignment, generalization, safety, and system efficiency (§6.2).

6.1. Benchmark Categories

This subsection organizes embodied navigation benchmarks by *task type*, namely, by the primary capability that the agent is required to demonstrate. Specifically, the categories below cover *instruction-following navigation*, *goal-conditioned navigation*, *navigation-related embodied reasoning*, *interactive and dynamic navigation*, and a still-small set of *cross-embodiment evaluation* benchmarks. Table 7 summarizes representative benchmarks in each category, along with their domain, embodiment type, target capability, and evaluation metrics.

6.1.1. Instruction-Following Navigation

This category evaluates whether an agent can transform natural-language instructions into executable navigation behavior. The core challenge is not merely to reach a destination, but to interpret landmarks, sequential dependencies, and spatial constraints described in language, and to ground them into temporally coherent actions in the environment. As a result, these benchmarks test the joint ability of the model to understand linguistic intent, maintain alignment between language and perception over time, and decide when to continue, adjust, or terminate navigation.

Representative benchmarks in this family remain centered on indoor settings, including R2R [31], RxR [32], R2R-CE [33], REVERIE [144], CVDN [145], and SOON [100]. These benchmarks establish the canonical evaluation settings for instruction-following navigation, while VLN-PE [34] extends the evaluation toward physically realistic cross-embodiment transfer. More recent benchmarks preserve the same instruction-following objective while broadening the deployment context. For ground platforms, CityWalker [88] and BridgeNav [146] introduce weaker localization cues, greater appearance diversity, and longer real-world horizons. For aerial platforms, OpenUAV and its UAV-Need-Help benchmark [113], together with related UAV benchmarks such as AVDN, LHPR-VLN, and AerialVLN [83, 147, 148], extend the same task structure to full-3D flight and realistic control.

The interpretation of results in this category should therefore go beyond simple completion statistics. Task success remains important, but the more informative question is whether the executed trajectory faithfully reflects the instruction, including intermediate waypoint alignment, ordering of subgoals, and appropriate stopping behavior. In outdoor and aerial settings, this interpretation becomes even more demanding because longer horizons, weaker localization support, and real-time execution constraints can substantially affect performance. Consequently, instruction-following benchmarks are best understood through a combination of success-based metrics, path-sensitive measures, and deployment-aware considerations rather than through success rate alone.

6.1.2. Goal-Conditioned Navigation

This category evaluates whether an agent can search for and reach a target specified by a goal condition rather than by a step-by-step instruction sequence. The goal may be provided as an object category, a reference image, a region-level cue, or an open-vocabulary semantic description. Compared with instruction-following navigation, the central challenge here is not faithful execution of a prescribed route, but semantic search under partial observability. The agent must decide where to explore, what observations are relevant to the target, and when the accumulated evidence is sufficient to approach or declare success.

Representative benchmarks include HM3D ObjectNav [104], HM3D-OVON [37], Habitat ObjectNav v2 [36], and GOAT-Bench [101]. These benchmarks define the main evaluation settings for goal-conditioned embodied search in indoor environments, while related benchmark variants broaden the task family to earlier Habitat and Gibson protocols, image-goal formulations, and demand-conditioned object search [41, 103, 105, 149, 150]. The same task structure also extends beyond indoor scenes. In outdoor, driving-scale, and real-world settings, benchmarks such as MetaUrban [151], NAVSIM [141], nuScenes [87], and LeLaN [119] examine the same search-oriented capability under broader appearance variation, longer horizons, and more realistic deployment conditions. Related open-set outdoor evaluations further stress semantic shift and regional transfer [152].

The interpretation of results in this category should therefore emphasize more than completion and path efficiency alone. Reaching the target remains a primary objective, but the more revealing question is whether the agent explores in a semantically meaningful way, uses memory effectively to avoid redundant search, and generalizes to previously unseen targets or environments. For embodied navigation, these benchmarks are especially valuable because they test whether broad semantic priors acquired during pretraining can support general-purpose embodied search, including open-set target grounding and robust search behavior under distribution shift.

6.1.3. Navigation-Related Embodied Reasoning

This category evaluates tasks in which navigation is not the final objective by itself, but a means of acquiring the evidence required for higher-level reasoning. In these benchmarks, the agent must move through the environment to observe relevant regions, gather partial clues across time, and then complete downstream tasks such as question answering, scene judgment, or decision making. The key challenge is therefore not only to

arrive at a destination, but to navigate in a way that reveals the information needed for correct reasoning under partial observability.

Representative benchmarks in this family are currently led by MP3D-EQA [124] and OpenEQA [92]. Both require the agent to actively explore in order to gather the evidence needed for grounded question answering, rather than answering from a static scene representation alone. Although the benchmark inventory in this category remains smaller than in instruction following or object-goal navigation, the shared underlying structure is clear: navigation serves as a means of acquiring the observations needed for grounded reasoning and final-task completion.

The interpretation of results in this category should therefore not rely on navigation success alone. Reaching a relevant area is only one part of the problem. What matters more is whether the visited states actually reveal the critical evidence, whether memory preserves that evidence across the trajectory, and whether the final prediction is grounded in observations collected during interaction with the environment. As a result, these benchmarks are particularly useful for evaluating whether embodied navigation systems can coordinate exploration, evidence retention, and answer generation within a single embodied reasoning process.

6.1.4. Interactive and Dynamic Navigation

This category evaluates navigation under conditions in which the environment cannot be treated as static during execution. The source of dynamics may come from moving obstacles, other agents, human interaction, social constraints, or targets that change position over time. In these settings, the agent cannot rely on a fixed route planned in advance. Instead, it must continually update its decisions on the basis of newly observed changes, maintain situational awareness, and replan online in order to remain effective and safe.

Representative benchmarks include Habitat 3.0 [143], SocNavBench [142], and Social-VLN [75]. Related benchmark protocols extend the same theme to human following, embodied tracking, social-navigation scenario design, and audio-visual navigation [45, 48, 107, 130, 153, 154]. Although these benchmarks differ in embodiment, sensing modality, and interaction structure, they share the same core requirement: navigation must remain adaptive under evolving environmental and social conditions.

The interpretation of results in this category should therefore extend well beyond overall task success. Reaching the target or maintaining task completion remains necessary, but the more informative questions concern whether the agent can replan effectively, recover from disrupted trajectories, maintain stable tracking, comply with social constraints, and avoid unsafe behavior during interaction. The main value of these benchmarks is that they expose failure modes that static benchmarks often hide, including brittle plans, delayed response to environmental change, unstable pursuit behavior, and violations of safety or social acceptability.

6.1.5. Generalist and Cross-Embodiment Evaluation

This category evaluates whether a model can preserve useful navigation capability across changes in embodiment, controller, sensing interface, and deployment regime. The central question is not performance within a single benchmark setting, but whether the same policy or architectural family remains effective when morphology, observation geometry, or low-level execution assumptions change.

A strict benchmark-only view reveals that this category is still relatively sparse. Among currently available resources, VLN-PE [34] is the clearest benchmark-style platform in this direction: it explicitly introduces a physically realistic cross-embodiment VLN benchmark spanning humanoid, quadruped, and wheeled robots, and evaluates transfer under controller engagement, lighting variation, and embodiment shift. Compared with the richer benchmark ecosystems already available for instruction following or object-goal navigation, cross-embodiment evaluation remains at an earlier stage of standardization.

The interpretation of results in this category should therefore emphasize transfer stability rather than isolated in-domain scores. What matters is whether performance is preserved under embodiment shift, whether the model degrades gracefully when observation geometry or locomotion changes, and whether the same navigation policy remains usable once physical control and realistic sensing are introduced. Even though this benchmark family is currently small, it provides the most direct test of the foundational claim behind embodied navigation systems: that useful navigation competence should survive beyond a single robot form or simulator abstraction.

6.2. Evaluation Metrics

Embodied navigation benchmarks do not necessarily emphasize the same evaluation priorities. Some focus on terminal success and efficiency, some place greater weight on trajectory fidelity and semantic grounding,

and others additionally require robustness, safety, or real-time deployability. In this section, we organize evaluation through four abstract metric layers: *task completion*, *trajectory fidelity and grounding*, *robustness, generalization, and safety*, and *real-time deployment*. This abstraction does not force all benchmarks into a single metric standard, but provides a common framework for analyzing which aspects of embodied performance each benchmark measures.

6.2.1. Task Completion

Task-completion metrics form the first layer of embodied navigation evaluation because they capture the minimum requirements that any navigation system must satisfy, namely, task completion, proximity to the goal. Even when benchmarks require instruction grounding, semantic search, or embodied reasoning, evaluation still begins with whether the agent reaches a valid terminal state under a reasonable motion budget. For this reason, completion metrics remain the common baseline across benchmark families and provide the most basic level of comparability.

Consider a benchmark with N evaluation episodes. For episode i, $v_0^{(i)}$ and $v_T^{(i)}$ denote the initial and terminal agent states, respectively, and $G^{(i)}$ denotes the goal set or success region. The shortest-path distance from a state v to the goal set is defined as $d(v, G) = \min_{g \in G} d(v, g)$, where d_{th} denotes the success tolerance. Let p_i denote the executed path length and $\ell_i = d(v_0^{(i)}, G^{(i)})$ denote the shortest-path distance from the initial state to the goal set. The episode-level success indicator can be defined as:

$$S_i = \mathbb{1}\left[d\left(v_T^{(i)}, G^{(i)}\right) \le d_{\text{th}}\right]. \tag{1}$$

The standard task completion metrics are then defined as [35]:

$$\text{SR} = \frac{1}{N}\sum_{i=1}^{N} S_i, \tag{2}$$

$$\text{NE} = \frac{1}{N}\sum_{i=1}^{N} d\left(v_T^{(i)}, G^{(i)}\right), \tag{3}$$

$$\text{OSR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\min_t d\left(v_t^{(i)}, G^{(i)}\right) \le d_{\text{th}}\right], \tag{4}$$

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \cdot \frac{\ell_i}{\max(p_i, \ell_i)}. \tag{5}$$

Here, *Success Rate (SR)* measures whether the agent terminates within the success region; *Navigation Error (NE)* measures the terminal distance to the goal region; *Oracle Success Rate (OSR)* measures whether the agent ever enters the success region under an oracle stopping rule; and *Success weighted by Path Length (SPL)* measures path efficiency relative to the shortest feasible route. This formulation is more general than point-goal notation: when the benchmark specifies a point target, one simply takes $G^{(i)} = \{v_*^{(i)}\}$.

These metrics provide the common baseline for embodied navigation evaluation by measuring whether the agent reaches the goal region, how close it gets to the goal, and how efficiently success is achieved. However, they summarize performance mainly through terminal outcome and path length, and therefore provide only a coarse account of navigation quality. In embodied navigation, they are necessary but not sufficient, because they do not reveal whether the trajectory faithfully follows an instruction, whether the behavior is semantically grounded, or whether the policy remains robust and usable under realistic execution constraints.

6.2.2. Trajectory Fidelity and Grounding

Task completion alone does not guarantee that navigation is semantically correct. In instruction-conditioned benchmarks, an agent may still reach the goal by exploiting dataset regularities, following a shortcut, or ignoring part of the instruction, even when its behavior does not faithfully reflect the intended route. For this reason, evaluation in this setting requires path-sensitive metrics that examine not only whether the agent arrives, but also whether the executed trajectory remains sufficiently consistent with the reference path implied by the task specification.

Let $P = (p_1, \ldots, p_m)$ denote the predicted trajectory and $R = (r_1, \ldots, r_n)$ the reference trajectory. A widely used path-alignment metric is normalized Dynamic Time Warping (*nDTW*):

$$\mathrm{nDTW}(P, R) = \exp\left(-\frac{\mathrm{DTW}(P, R)}{n\, d_{\mathrm{th}}}\right), \tag{6}$$

where $\mathrm{DTW}(P, R)$ is the dynamic time warping distance between the predicted and reference trajectories under the benchmark-specific geodesic distance, and d_{th} is the success tolerance. The success-weighted variant is defined as:

$$\mathrm{sDTW}(P, R) = S(P)\, \mathrm{nDTW}(P, R), \tag{7}$$

where

$$S(P) = \mathbb{1}[d(p_m, G) \le d_{\mathrm{th}}] \tag{8}$$

is the terminal success indicator for the predicted trajectory with respect to the benchmark goal set G. Intuitively, *nDTW* measures ordered trajectory similarity even when the endpoint is imperfect, whereas *sDTW* requires both successful completion and trajectory alignment.

Another representative metric is *Coverage weighted by Length Score* (*CLS*):

$$\mathrm{CLS}(P, R) = \mathrm{PC}(P, R) \cdot \mathrm{LS}(P, R), \tag{9}$$

where path coverage is

$$\mathrm{PC}(P, R) = \frac{1}{|R|} \sum_{r \in R} \exp\left(-\frac{d(r, P)}{d_{\mathrm{th}}}\right), \tag{10}$$

with

$$d(r, P) = \min_{p \in P} d(r, p), \tag{11}$$

and the length score is

$$\mathrm{LS}(P, R) = \frac{\mathrm{EPL}(P, R)}{\mathrm{EPL}(P, R) + |\mathrm{EPL}(P, R) - \mathrm{PL}(P)|}, \tag{12}$$

$$\mathrm{EPL}(P, R) = \mathrm{PC}(P, R) \cdot \mathrm{PL}(R). \tag{13}$$

Here, $\mathrm{PL}(P)$ and $\mathrm{PL}(R)$ denote the lengths of the predicted and reference paths, respectively. *CLS* therefore rewards both adequate coverage of the reference route and a plausible trajectory length.

These metrics are more informative than task completion metrics because they evaluate whether the executed trajectory remains close to the reference route in an ordered and length-aware sense. In this way, *nDTW*, *sDTW*, and *CLS* provide useful proxies for trajectory fidelity and are often effective for diagnosing shortcut behavior or incomplete route following.

6.2.3. Robustness, Generalization, and Safety

For embodied navigation systems, evaluation should extend beyond static task completion and trajectory fidelity to test whether the policy remains reliable under longer horizons, distribution shift, embodiment change, and dynamic interaction. This is especially important because embodied navigation systems are often motivated by claims of stronger memory, more general reasoning, and broader cross-embodiment transfer, none of which can be established by endpoint success alone. Therefore, they require evaluation protocols that test whether capability is preserved once the environment becomes less predictable, the task becomes less myopic, or the deployment condition departs from the training regime.

Long-horizon robustness is the first important dimension because binary success often becomes too sparse to diagnose behavior in extended tasks. A useful complementary quantity is average goal progress (*GP*), commonly used in dialogue-based or exploration-heavy settings:

$$\mathrm{GP} = \frac{1}{N} \sum_{i=1}^{N} \left[d\left(v_0^{(i)}, G^{(i)}\right) - d\left(v_T^{(i)}, G^{(i)}\right) \right]. \tag{14}$$

This metric is informative when the agent is not expected to reach the target in every episode but should nevertheless make meaningful progress toward it. More broadly, long-horizon evaluation should also examine

subgoal completion, re-localization success, perturbation recovery, backtracking ability, and the consistency of explicit or implicit spatial memory. In this sense, robustness under long horizons is not only a matter of whether the agent eventually succeeds, but also of whether it can maintain coherent behavior when the task requires memory, recovery, and continued progress over time.

The second dimension concerns transfer and generalization. Seen-to-unseen transfer within a fixed simulator remains useful, but it represents only a relatively weak form of robustness. Stronger protocols test whether performance is preserved across buildings, cities, weather conditions, object distributions, visual styles, embodiments, sensing modalities, and the sim-to-real boundary. Benchmarks such as `HM3D-OVON` [37], `GOAT-Bench` [101], `CityWalker` [88], and `VLN-PE` [34] illustrate why this broader notion of transfer matters. Related benchmark settings further stress outdoor regional transfer, driving-scale shift, and realistic aerial deployment [113, 141, 146, 151]. For embodied navigation systems, this family of metrics is especially consequential because it most directly tests whether the model has learned reusable embodied priors rather than benchmark-specific shortcuts.

Safety becomes equally central once navigation takes place in dynamic or socially constrained scenes. When the environment contains moving obstacles, crowds, or human interaction, success alone is no longer an adequate summary of policy quality. Relevant metrics include *collision count, collision rate, minimum clearance, near-miss frequency, time-to-collision, human-collision count, personal-space intrusion, and motion smoothness*. In target-following or tracking-oriented settings, target-following error and target reacquisition rate after temporary loss are also informative. Benchmarks such as `SocNavBench` [142], `Social-VLN` [75], and `Habitat 3.0` [143] make clear that in dynamic and social navigation, safety is not an auxiliary consideration but a primary metric family. More broadly, robustness, safety, and transfer metrics determine whether an embodied navigation system remains reliable once evaluation moves beyond static shortest-path settings, and therefore provide a necessary test of whether the model is genuinely deployable rather than merely successful under controlled conditions.

6.2.4. Real-Time Deployment

The final layer of evaluation concerns whether a navigation policy remains usable under the timing and resource constraints of real deployment. In embodied navigation, action quality and action timing cannot be separated: a policy that produces semantically strong decisions but reacts too slowly may still be ineffective, unstable, or even invalid once deployed on a physical agent. For this reason, real-time metrics should be treated as part of the evaluation itself rather than as peripheral systems statistics.

Let t_k^{obs} denote the time at which observation k is received and t_k^{act} the time at which the corresponding action is produced. The per-decision end-to-end latency is defined as:

$$L_k = t_k^{\text{act}} - t_k^{\text{obs}}, \tag{15}$$

and the average latency over K decisions is:

$$\bar{L}_{\text{e2e}} = \frac{1}{K} \sum_{k=1}^{K} L_k. \tag{16}$$

The corresponding policy frequency can be reported either as the measured average action rate over an episode or, in a synchronous pipeline, approximated by:

$$f_\pi \approx \frac{1}{\bar{L}_{\text{e2e}}}. \tag{17}$$

If the high-level policy is coupled to a lower-level controller with frequency f_c, it is also useful to report:

$$\rho = \frac{f_c}{f_\pi}, \tag{18}$$

which measures how many low-level control cycles elapse per high-level decision. This quantity is especially important for agile embodiments such as UAVs or legged robots, where delayed high-level actions can invalidate otherwise strong semantic reasoning.

Deployment-oriented evaluation should also report memory and compute constraints. A simple summary statistic is the peak memory footprint:

$$M_{\text{peak}} = \max_t M(t), \tag{19}$$

together with parameter count, VRAM usage, controller frequency, and, where relevant, sequence length or cache-related memory budgets. These quantities should not be treated as engineering afterthoughts. For embodied navigation, they are part of the practical meaning of benchmark results, because a navigation policy is only genuinely competitive if it can operate at the timescale and resource budget required by the target embodiment.

These metrics should therefore be interpreted as indicators of deployment validity rather than mere implementation details. High latency can reduce effective control responsiveness even when offline decision quality appears strong, and limited policy frequency can make a method unsuitable for fast or unstable platforms. Likewise, memory footprint and compute budget determine whether a reported result is feasible on the intended hardware rather than only on oversized experimental setups. This perspective is especially important when comparing large, reasoning-intensive models with lightweight reactive policies, since the comparison is otherwise incomplete. In benchmarks and benchmark platforms such as `OpenUAV/UAV-Need-Help` [113], `VLN-PE` [34], `CityWalker` [88], and `NAVSIM` [141], timing and system constraints are part of benchmark validity itself rather than secondary appendix material. Real-time deployability should therefore be treated as a core evaluation dimension whenever embodied navigation claims are intended to hold beyond controlled offline settings.

6.3. Takeaway Insights

Embodied navigation evaluation should be interpreted as a joint problem of *task type*, *metric choice*, and *deployment condition*, rather than as a comparison of isolated leaderboard numbers. A reported result only becomes meaningful when it is tied to the capability that the benchmark is designed to test and to the metric family that makes that capability visible. In this sense, benchmark taxonomy and metric taxonomy should be read together: different tasks stress different aspects of embodied competence, and therefore require different evidence for success.

This is why completion metrics alone cannot support a complete evaluation. In instruction-following navigation, success must be interpreted together with trajectory fidelity and, where possible, explicit grounding. In goal-conditioned navigation, the emphasis shifts toward semantic exploration, memory, and open-set generalization. In navigation-related embodied reasoning, the critical issue is whether movement exposes the evidence needed for grounded downstream inference. In dynamic, social, and cross-embodiment settings, the meaning of success depends even more strongly on robustness, safety, transfer, and system-level viability.

What ultimately matters is not only whether a model solves a benchmark, but what kind of competence that success actually demonstrates. A convincing embodied navigation evaluation should therefore show that success remains faithful to task semantics, robust under shift and interaction, and feasible under realistic deployment constraints. Progress is most credible when it is sustained across task families and under increasingly realistic conditions, rather than concentrated in a single static benchmark setting.

7. Conclusions and Future Directions

The remarkable advancements of vision, language, and video foundation models have fundamentally reshaped the landscape of embodied navigation. This survey has systematically categorized existing research through the lens of design paradigms, data sources, and training strategies, offering a unified framework to interpret how these models bridge high-level reasoning with low-level control. However, despite the rapid evolution of this domain and the proliferation of sophisticated architectures like dual-system and end-to-end models, a significant gap remains between benchmark performance and physical reality. As of the completion of this survey, our own experiences in leveraging and deploying state-of-the-art methods reveal that real-world task success rates remain strikingly low. These findings underscore that we are still far from truly "solving" the task of embodied navigation.

To move the field toward general-purpose embodied intelligence, we identify several critical directions for future research:

- **Establishing the scaling law for embodied navigation.** While scaling has proven transformative for language and beyond [155, 156, 157, 158], the path to a similar scaling law for navigation remains obscured by a severe data bottleneck. Existing synthetic and simulation data suffer from a persistent sim-to-real gap and the lack of data diversity, particularly in visual geometry and physical dynamics. Conversely, there is a distinct lack of large-scale, diverse, and high-quality real-world navigation data to provide the necessary supervision for multi-billion parameter models. Bridging this gap through better data engines or massive-scale robot logs is essential.
- **Converging VLM and VGM backbones.** A fundamental question for future navigation foundation models is whether to adopt VLMs or VGMs as the primary backbone. Theoretically, embodied navigation requires both the strong semantic reasoning and instruction-following capabilities of VLMs, as well as the world-modeling and physical-prediction capabilities of VGMs. Moving forward, the field should explore architectures that fuse these strengths, enabling agents that can both understand complex linguistic goals and predict the physical consequences of their movements.
- **Developing next-generation benchmarks.** Many existing benchmarks are increasingly outdated, failing to capture the requirements of modern navigation foundation models such as open-vocabulary reasoning and social compliance. Future benchmarks must move beyond terminal success rates to include rigorous evaluation of instruction fidelity, real-time latency, and robustness under dynamic disturbances.
- **Hardware-aware algorithmic optimization.** Given the strict hardware constraints of physical robots, future work must focus on bridging the gap between heavy foundation models and edge hardware. This includes advancing asynchronous inference, algorithmic compression, and intelligent device-cloud orchestration to ensure that reasoning does not come at the cost of reactive safety.

References

[1] An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Xueyan Zou, Jan Kautz, Erdem Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. In *RSS*, 2025.

[2] Zedong Chu, Shichao Xie, Xiaolong Wu, Yanfen Shen, Minghua Luo, Zhengbo Wang, Fei Liu, Xiaoxu Leng, Junjun Hu, Mingyang Yin, et al. Abot-n0: Technical report on the vla foundation model for versatile embodied navigation. *arXiv preprint arXiv:2602.11598*, 2026.

[3] Botao Ren, Junjun Hu, Xinda Xue, Minghua Luo, Jintao Chen, Haochen Bai, Liangliang You, and Mu Xu. Astranav-memory: Contexts compression for long memory. *arXiv preprint arXiv:2512.21627*, 2025.

[4] InternNav Team. InternVLA-N1: An open dual-system navigation foundation model with learned latent plans, 2025.

[5] 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.

[6] Hai Zhang, Siqi Liang, Li Chen, Yuxian Li, Yukuan Xu, Yichao Zhong, Fu Zhang, and Hongyang Li. Sparse video generation propels real-world beyond-the-view vision-language navigation. *arXiv preprint arXiv:2602.05827*, 2026.

[7] Yuze Wu, Mo Zhu, Xingxing Li, Yuheng Du, Yuxin Fan, Wenjun Li, Zhichao Han, Xin Zhou, and Fei Gao. Vla-an: An efficient and onboard vision-language-action framework for aerial navigation in complex environments. *arXiv preprint arXiv:2512.15258*, 2025.

[8] Jing Zuo, Lingzhou Mu, Fan Jiang, Chengcheng Ma, Mu Xu, and Yonggang Qi. Fantasyvln: Unified multimodal chain-of-thought reasoning for vision-language navigation. *arXiv preprint arXiv:2601.13976*, 2026.

[9] 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.

[10] 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.

[11] Shaoan Wang, Yuanfei Luo, Xingyu Chen, Aocheng Luo, Dongyue Li, Chang Liu, Sheng Chen, Yangang Zhang, and Junzhi Yu. Vlingnav: Embodied navigation with adaptive reasoning and visual-assisted linguistic memory. *arXiv preprint arXiv:2601.08665*, 2026.

[12] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[13] 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.

[14] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[15] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[16] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[17] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7606–7623, 2022.

[18] Yue Zhang, Ziqiao Ma, Jialu Li, Yanyuan Qiao, Zun Wang, Joyce Chai, Qi Wu, Mohit Bansal, and Parisa Kordjamshidi. Vision-and-language navigation today and tomorrow: A survey in the era of foundation models. *arXiv preprint arXiv:2407.07035*, 2024.

[19] Zecheng Li, Xiaolin Meng, Xu He, Youdong Zhang, and Wenxuan Yin. Large-scale model-enhanced vision-language navigation: Recent advances, practical applications, and future challenges. *Preprints*, 2026.

[20] Hongcheng Wang, Andy Guan Hong Chen, Xiaoqi Li, Mingdong Wu, and Hao Dong. Find what you want: Learning demand-conditioned object attribute space for demand-driven navigation. *Advances in Neural Information Processing Systems*, 36:16353–16366, 2023.

[21] 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.

[22] Shuaihang Yuan, Hao Huang, Yu Hao, Congcong Wen, Anthony Tzes, and Yi Fang. Gamap: Zero-shot object goal navigation with multi-scale geometric-affordance guidance. *Advances in Neural Information Processing Systems*, 37:39386–39408, 2024.

[23] Jingyuan Zhao, Wenyi Zhao, Bo Deng, Zhenghong Wang, Feng Zhang, Wenxiang Zheng, Wanke Cao, Jinrui Nan, Yubo Lian, and Andrew F Burke. Autonomous driving system: A comprehensive survey. *Expert Systems with Applications*, 242:122836, 2024.

[24] Tuo Feng, Wenguan Wang, and Yi Yang. A survey of world models for autonomous driving. *arXiv preprint arXiv:2501.11260*, 2025.

[25] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10164–10183, 2024.

[26] Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

[27] Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

[28] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[29] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[30] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[31] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[32] 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.

[33] 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.

[34] Liuyi Wang, Xinyuan Xia, Hui Zhao, Hanqing Wang, Tai Wang, Yilun Chen, Chengju Liu, Qijun Chen, and Jiangmiao Pang. Rethinking the embodied gap in vision-and-language navigation: A holistic study of physical and visual disparities. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9455–9465, 2025.

[35] 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.

[36] 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. *arXiv preprint arXiv:2006.13171*, 2020.

[37] 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.

[38] Kehan Chen, Yan Huang, Dong An, Jiawei He, Yifei Su, Jing Liu, Nianfeng Liu, and Liang Wang. Floorplan-vln: A new paradigm for floor plan guided vision-language navigation, 2026.

[39] Xiaoming Zhao, Harsh Agrawal, Dhruv Batra, and Alexander G Schwing. The surprising effectiveness of visual odometry techniques for embodied pointgoal navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16127–16136, 2021.

[40] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12875–12884, 2020.

[41] Jacob Krantz, Stefan Lee, Jitendra Malik, Dhruv Batra, and Devendra Singh Chaplot. Instance-specific image goal navigation: Training embodied agents to find object instances. *arXiv preprint arXiv:2211.15876*, 2022.

[42] 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.

[43] Fangwei Zhong, Peng Sun, Wenhan Luo, Tingyun Yan, and Yizhou Wang. Towards distraction-robust active visual tracking. In *International Conference on Machine Learning*, pages 12782–12792. PMLR, 2021.

[44] Fangwei Zhong, Kui Wu, Hai Ci, Churan Wang, and Hao Chen. Empowering embodied visual tracking with visual foundation models and offline rl. In *European Conference on Computer Vision*, pages 139–155. Springer, 2024.

[45] Shaoan Wang, Jiazhao Zhang, Minghan Li, Jiahang Liu, Anqi Li, Kui Wu, Fangwei Zhong, Junzhi Yu, Zhizheng Zhang, and He Wang. Trackvla: Embodied visual tracking in the wild. *arXiv preprint arXiv:2505.23189*, 2025.

[46] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

[47] Ross Mead and Maja J Matarić. Autonomous human–robot proxemics: socially aware navigation based on interaction potential. *Autonomous Robots*, 41(5):1189–1201, 2017.

[48] Daeun Song, Jing Liang, Amirreza Payandeh, Amir Hossain Raj, Xuesu Xiao, and Dinesh Manocha. Vlm-social-nav: Socially aware robot navigation through scoring using vision-language models. *IEEE Robotics and Automation Letters*, 10(1):508–515, 2024.

[49] Haresh Karnan, Anirudh Nair, Xuesu Xiao, Garrett Warnell, Sören Pirk, Alexander Toshev, Justin Hart, Joydeep Biswas, and Peter Stone. Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation. *IEEE Robotics and Automation Letters*, 7(4):11807–11814, 2022.

[50] Xinzhu Liu, Di Guo, Huaping Liu, and Fuchun Sun. Multi-agent embodied visual semantic navigation with scene prior knowledge. *IEEE Robotics and Automation Letters*, 7(2):3154–3161, 2022.

[51] Humanoid Robotics Technology. Wheeled robot: An overview, 2025. Accessed: 2026-03-24.

[52] Humanoid Robotics Technology. Legged robot basics: An introduction, 2025. Accessed: 2026-03-24.

[53] Wikipedia contributors. Legged robot — Wikipedia, the free encyclopedia, 2026. Accessed: 2026-03-24.

[54] Asif Ali Laghari, Awais Khan Jumani, Rashid Ali Laghari, and Haque Nawaz. Unmanned aerial vehicles: A review. *Cognitive robotics*, 3:8–22, 2023.

[55] Wikipedia contributors. Unmanned aerial vehicle — Wikipedia, the free encyclopedia, 2026. Accessed: 2026-03-24.

[56] Yihan Cao, Jiazhao Zhang, Zhinan Yu, Shuzhen Liu, Zheng Qin, Qin Zou, Bo Du, and Kai Xu. Cognav: Cognitive process modeling for object goal navigation with llms, 2025.

[57] 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.

[58] Dujun Nie, Xianda Guo, Yiqun Duan, Ruijun Zhang, and Long Chen. Wmnav: Integrating vision-language models into world models for object goal navigation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2392–2399. IEEE, 2025.

[59] 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.

[60] 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.

[61] Haoran Liu, Weikang Wan, Xiqian Yu, Minghan Li, Jiazhao Zhang, Bo Zhao, Zhibo Chen, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Na vid-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.

[62] Zhangyang Qi, Zhixiong Zhang, Yizhou Yu, Jiaqi Wang, and Hengshuang Zhao. Vln-r1: Vision-language navigation via reinforcement fine-tuning. *arXiv: 2506.17221*, 2025.

[63] Chen Gao, Liankai Jin, Xingyu Peng, Jiazhao Zhang, Yue Deng, Annan Li, He Wang, and Si Liu. Octonav: Towards generalist embodied navigation. *arXiv preprint arXiv:2506.09839*, 2025.

[64] Wentao Xiang, Haokang Zhang, Tianhang Yang, Zedong Chu, Ruihang Chu, Shichao Xie, Yujian Yuan, Jian Sun, Zhining Gu, Junjie Wang, et al. Nav-R^2 dual-relation reasoning for generalizable open-vocabulary object-goal navigation. *arXiv preprint arXiv:2512.02400*, 2025.

[65] Fei Liu, Shichao Xie, Minghua Luo, Zedong Chu, Junjun Hu, Xiaolong Wu, and Mu Xu. Navforesee: A unified vision-language world model for hierarchical planning and dual-horizon navigation prediction. *arXiv preprint arXiv:2512.01550*, 2025.

[66] Guoxin Lian, Shuo Wang, Yucheng Wang, Yongcai Wang, Maiyue Chen, Kaihui Wang, Bo Zhang, Zhizhong Su, Deying Li, and Zhaoxin Fan. Mapdream: Task-driven map learning for vision-language navigation. *arXiv preprint arXiv:2602.00222*, 2026.

[67] 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.

[68] 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.

[69] Thomas Chabal, Shizhe Chen, Jean Ponce, and Cordelia Schmid. Fom-nav: Frontier-object maps for object goal navigation. *arXiv preprint arXiv:2512.01009*, 2025.

[70] Yufeng Zhong, Chengjian Feng, Feng Yan, Fanfan Liu, Liming Zheng, and Lin Ma. Robotron-nav: A unified framework for embodied navigation integrating perception, planning, and prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6416–6425, 2025.

[71] Anqi Li, Zhiyong Wang, Jiazhao Zhang, Minghan Li, Yunpeng Qi, Zhibo Chen, Zhizheng Zhang, and He Wang. Urbanvla: A vision-language-action model for urban micromobility. *arXiv preprint arXiv:2510.23576*, 2025.

[72] Ziyi Chen, Yingnan Guo, Zedong Chu, Minghua Luo, Yanfen Shen, Mingchao Sun, Junjun Hu, Shichao Xie, Kuan Yang, Pei Shi, et al. Socialnav: Training human-inspired foundation model for socially-aware embodied navigation. *arXiv preprint arXiv:2511.21135*, 2025.

[73] Xiaolou Sun, Wufei Si, Wenhui Ni, Yuntian Li, Dongming Wu, Fei Xie, Runwei Guan, He-Yang Xu, Henghui Ding, Yuan Wu, et al. Autofly: Vision-language-action model for uav autonomous navigation in the wild. *arXiv preprint arXiv:2602.09657*, 2026.

[74] Qingxiang Liu, Ting Huang, Zeyu Zhang, and Hao Tang. Nav-r1: Reasoning and navigation in embodied scenes. *arXiv preprint arXiv:2509.10884*, 2025.

[75] 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.

[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.

[77] Lingfeng Zhang, Yuecheng Liu, Zhanguang Zhang, Matin Aghaei, Yaochen Hu, Hongjian Gu, Mohammad Ali Alomrani, David Gamaliel Arcos Bravo, Raika Karimi, Atia Hamidizadeh, et al. Mem2ego: Empowering vision-language models with global-to-ego memory for long-horizon embodied navigation. *arXiv preprint arXiv:2502.14254*, 2025.

[78] Yiren Lu, Yi Du, Disheng Liu, Yunlai Zhou, Chen Wang, and Yu Yin. Gsmem: 3d gaussian splatting as persistent spatial memory for zero-shot embodied exploration and reasoning, 2026.

[79] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[80] Shuo Wang, Yongcai Wang, Wanting Li, Xudong Cai, Yucheng Wang, Maiyue Chen, Kaihui Wang, Zhizhong Su, Deying Li, and Zhaoxin Fan. Aux-think: Exploring reasoning strategies for data-efficient vision-language navigation. In *Advances in Neural Information Processing Systems*, 2025.

[81] Yu He, Da Huang, Zhenyang Liu, Zixiao Gu, Qiang Sun, Guangnan Ye, and Yanwei Fu. Schrödinger's navigator: Imagining an ensemble of futures for zero-shot object navigation. *arXiv preprint arXiv:2512.21201*, 2025.

[82] Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12009–12020, 2023.

[83] Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. Aerialvln: Vision-and-language navigation for uavs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15384–15394, 2023.

[84] Jungdae Lee, Taiki Miyanishi, Shuhei Kurita, Koya Sakamoto, Daichi Azuma, Yutaka Matsuo, and Nakamasa Inoue. Citynav: A large-scale dataset for real-world aerial navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5912–5922, 2025.

[85] Yunpeng Gao, Chenhui Li, Zhongrui You, Junli Liu, Zhen Li, Pengan Chen, Qizhi Chen, Zhonghan Tang, Liansheng Wang, Penghui Yang, et al. Openfly: A comprehensive platform for aerial vision-language navigation. *arXiv preprint arXiv:2502.18041*, 2025.

[86] Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid exploration for open-world navigation with latent goal models. In *CoRL*, Proceedings of Machine Learning Research, pages 674–684. PMLR, 2021.

[87] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[88] Xinhao Liu, Jintong Li, Yicheng Jiang, Niranjan Sujay, Zhicheng Yang, Juexiao Zhang, John Abanes, Jing Zhang, and Chen Feng. Citywalker: Learning embodied urban navigation from web-scale videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6875–6885, 2025.

[89] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Semantic visual navigation by watching youtube videos. *Advances in Neural Information Processing Systems*, 33:4283–4294, 2020.

[90] Mingfei Han, Liang Ma, Kamila Zhumakhanova, Ekaterina Radionova, Jingyi Zhang, Xiaojun Chang, Xiaodan Liang, and Ivan Laptev. Roomtour3d: Geometry-aware video-instruction tuning for embodied navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27586–27596, 2025.

[91] Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19129–19139, 2022.

[92] Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, et al. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16488–16498, 2024.

[93] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1645–1653, 2017.

[94] Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang-wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, et al. Panda-70m: Captioning 70m videos with multiple cross-modality teachers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13320–13331, 2024.

[95] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Llava-video: Video instruction tuning with synthetic data. *Trans. Mach. Learn. Res.*, 2025.

[96] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[97] Wanrong Zhu, Jack Hessel, Anas Awadalla, Samir Yitzhak Gadre, Jesse Dodge, Alex Fang, Youngjae Yu, Ludwig Schmidt, William Yang Wang, and Yejin Choi. Multimodal c4: An open, billion-scale corpus of images interleaved with text. *Advances in Neural Information Processing Systems*, 36:8958–8974, 2023.

[98] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8430–8439, 2019.

[99] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2610–2621, 2019.

[100] Fengda Zhu, Xiwen Liang, Yi Zhu, Qizhi Yu, Xiaojun Chang, and Xiaodan Liang. Soon: Scenario oriented object navigation with graph-based exploration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12689–12699, 2021.

[101] Mukul Khanna, Ram Ramrakhya, Gunjan Chhablani, Sriram Yenamandra, Theophile Gervet, Matthew Chang, Zsolt Kira, Devendra Singh Chaplot, Dhruv Batra, and Roozbeh Mottaghi. Goat-bench: A benchmark for multi-modal lifelong navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16373–16383, 2024.

[102] 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.

[103] 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.

[104] Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.

[105] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9068–9079, 2018.

[106] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[107] Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X Chang, and Manolis Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16384–16393, 2024.

[108] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021.

[109] Epic Games. Unreal engine. Accessed: 2026-03-24.

[110] Qingyong Hu, Bo Yang, Sheikh Khalid, Wen Xiao, Niki Trigoni, and Andrew Markham. Sensaturban: Learning semantics from urban-scale photogrammetric point clouds. *International Journal of Computer Vision*, 130(2):316–343, 2022.

[111] Rockstar Games. Grand theft auto v. Accessed: 2026-03-24.

[112] Google. Google earth studio. Accessed: 2026-03-24.

[113] Xiangyu Wang, Donglin Yang, Ziqin Wang, Hohin Kwan, Jinyu Chen, Wenjun Wu, Hongsheng Li, Yue Liao, and Si Liu. Towards realistic uav vision-language navigation: Platform, benchmark, and methodology. *arXiv preprint arXiv:2410.07087*, 2024.

[114] 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.

[115] NVIDIA. Isaac Sim.

[116] OpenScene Dataset Contributors. OpenScene: The Largest Up-to-Date 3D Occupancy Prediction Benchmark in Autonomous Driving, August 2023.

[117] Noriaki Hirose, Dhruv Shah, Ajay Sridhar, and Sergey Levine. Sacson: Scalable autonomous control for social navigation. *IEEE Robotics and Automation Letters*, 9(1):49–56, 2023.

[118] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pages 71–91. Springer, 2024.

[119] Noriaki Hirose, Catherine Glossop, Ajay Sridhar, Dhruv Shah, Oier Mees, and Sergey Levine. Lelan: Learning a language-conditioned navigation policy from in-the-wild videos. *arXiv preprint arXiv:2410.03603*, 2024.

[120] Zhen Li, Chuanhao Li, Xiaofeng Mao, Shaoheng Lin, Ming Li, Shitian Zhao, Zhaopan Xu, Xinyue Li, Yukang Feng, Jianwen Sun, et al. Sekai: A video dataset towards world exploration. *arXiv preprint arXiv:2506.15675*, 2025.

[121] Noriaki Hirose, Amir Sadeghian, Marynel Vázquez, Patrick Goebel, and Silvio Savarese. Gonet: A semi-supervised deep learning approach for traversability estimation. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 3044–3051. IEEE, 2018.

[122] Kunyang Lin, Peihao Chen, Diwei Huang, Thomas H Li, Mingkui Tan, and Chuang Gan. Learning vision-and-language navigation from youtube videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8317–8326, 2023.

[123] Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. SQA3D: situated question answering in 3d scenes. In *ICLR*, 2023.

[124] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6659–6668, 2019.

[125] Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Junfei Wu, Xiaoying Zhang, Benyou Wang, and Xiangyu Yue. Video-r1: Reinforcing video reasoning in mllms. *arXiv preprint arXiv:2503.21776*, 2025.

[126] Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9127–9134, 2019.

[127] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European conference on computer vision*, pages 69–85. Springer, 2016.

[128] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 11–20, 2016.

[129] Jiawei Guo, Tianyu Zheng, Yizhi Li, Yuelin Bai, Bo Li, Yubo Wang, King Zhu, Graham Neubig, Wenhu Chen, and Xiang Yue. Mammoth-vl: Eliciting multimodal reasoning with instruction tuning at scale. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13869–13920, 2025.

[130] Ronghao Dang, Jiayan Guo, Bohan Hou, Sicong Leng, Kehan Li, Xin Li, Jiangpin Liu, Yunxuan Mao, Zhikai Wang, Yuqian Yuan, et al. Rynnbrain: Open embodied foundation models. *arXiv preprint arXiv:2602.14979*, 2026.

[131] Peican Lin, Gan Sun, Chenxi Liu, Fazeng Li, Weihong Ren, and Yang Cong. Openvln: Open-world aerial vision-language navigation. *arXiv preprint arXiv:2511.06182*, 2025.

[132] Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Omnivla: An omni-modal vision-language-action model for robot navigation. *arXiv preprint arXiv:2509.19480*, 2025.

[133] Zhiyu Huang, Yun Zhang, Johnson Liu, Rui Song, Chen Tang, and Jiaqi Ma. Tic-vla: A think-in-control vision-language-action model for robot navigation in dynamic environments. *arXiv preprint arXiv:2602.02459*, 2026.

[134] Pengxiang Ding, Han Zhao, Wenjie Zhang, Wenxuan Song, Min Zhang, Siteng Huang, Ningxi Yang, and Donglin Wang. Quar-vla: Vision-language-action model for quadruped robots. In *European Conference on Computer Vision*, pages 352–367. Springer, 2024.

[135] Tianshun Li, Tianyi Huai, Zhen Li, Yichun Gao, Haoang Li, and Xinhu Zheng. Skyvln: Vision-and-language navigation and nmpc control for uavs in urban environments. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 17199–17206. IEEE, 2025.

[136] Wen Jiang, Li Wang, Kangyao Huang, Wei Fan, Jinyuan Liu, Shaoyu Liu, Hongwei Duan, Bin Xu, and Xiangyang Ji. Longfly: Long-horizon uav vision-and-language navigation with spatiotemporal context integration. *arXiv preprint arXiv:2512.22010*, 2025.

[137] Haitong Wang, Aaron Hao Tan, Angus Fung, and Goldie Nejat. X-nav: Learning end-to-end cross-embodiment navigation for mobile robots. *IEEE Robotics and Automation Letters*, 11(1):698–705, 2025.

[138] Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Asyncvla: An asynchronous vla for fast and robust navigation on the edge. *arXiv preprint arXiv:2602.13476*, 2026.

[139] Yuzhao Luo, Ming Zhu, Xinghan Liu, Hengzhi Su, Yapeng Hu, Yuncheng Liu, and Tian Chen. Airuninav: Unified vision-language navigation for uavs in indoor and outdoor scenes.

[140] Zebin Yang, Sunjian Zheng, Tong Xie, Tianshi Xu, Bo Yu, Fan Wang, Jie Tang, Shaoshan Liu, and Meng Li. Efficientnav: Towards on-device object-goal navigation with navigation map caching and retrieval. *arXiv preprint arXiv:2510.18546*, 2025.

[141] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, et al. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *Advances in Neural Information Processing Systems*, 37:28706–28719, 2024.

[142] Abhijat Biswas, Allan Wang, Gustavo Silvera, Aaron Steinfeld, and Henny Admoni. Socnavbench: A grounded simulation testing framework for evaluating social navigation. *ACM Transactions on Human-Robot Interaction (THRI)*, 11(3):1–24, 2022.

[143] Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander Clegg, Michal Hlavac, So Yeon Min, Vladimir Vondrus, Théophile Gervet, Vincent-Pierre Berges, John M. 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. In *ICLR*. OpenReview.net, 2024.

[144] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9982–9991, 2020.

[145] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406. PMLR, 2020.

[146] Yuxiang Zhao, Yirong Yang, Yanqing Zhu, Yanfen Shen, Chiyu Wang, Zhining Gu, Pei Shi, Wei Guo, and Mu Xu. Bridging the indoor-outdoor gap: Vision-centric instruction-guided embodied navigation for the last meters. *arXiv preprint arXiv:2602.06427*, 2026.

[147] Yue Fan, Winson Chen, Tongzhou Jiang, Chun Zhou, Yi Zhang, and Xin Wang. Aerial vision-and-dialog navigation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3043–3061, 2023.

[148] Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-horizon vision-language navigation: Platform, benchmark and method. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12078–12088, 2025.

[149] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3164–3174, 2020.

[150] Hongcheng Wang, Andy Guan Hong Chen, Xiaoqi Li, Mingdong Wu, and Hao Dong. Find what you want: Learning demand-conditioned object attribute space for demand-driven navigation. *Advances in Neural Information Processing Systems*, 36:16353–16366, 2023.

[151] Wayne Wu, Honglin He, Jack He, Yiran Wang, Chenda Duan, Zhizheng Liu, Quanyi Li, and Bolei Zhou. Metaurban: An embodied ai simulation platform for urban micromobility. International Conference on Learning Representations (ICLR) 2025, 2025.

[152] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9297–9307, 2019.

[153] Changan Chen, Ziad Al-Halah, and Kristen Grauman. Semantic audio-visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15516–15525, 2021.

[154] Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigation in 3d environments. In *European conference on computer vision*, pages 17–36. Springer, 2020.

[155] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[156] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[157] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[158] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.