NAVWORLD: SCALING LONG-HORIZON OUTDOOR LANGUAGE NAVIGATION WITH CROWDSOURCED GROUND-LEVEL IMAGERY

Anonymous authors
Paper under double-blind review

ABSTRACT

While language-conditioned navigation has made substantial progress in indoor environments and autonomous driving, long-horizon instruction following for ground robots in outdoor pedestrian environments remains challenging. We present `NavWorld`, an ecosystem for long-horizon, language-conditioned outdoor navigation built around a simple idea: automatically turning large-scale, geographically diverse street imagery into language-labeled trajectories. Our data engine sources candidate navigation trajectories from Mapillary, a large-scale crowdsourced platform for ground-level imagery spanning diverse geographic regions and capture conditions. It then filters sequences for geometric and perceptual quality and automatically generates grounded high-level instructions, intermediate subtasks, and stopping conditions from visual observations and trajectory geometry. We train a language-conditioned navigation policy that generates language subtasks from high-level instructions, tracks progress across subtasks, and predicts continuous motion from monocular egocentric observations. We evaluate `NavWorld` in Habitat-GS, a photorealistic outdoor navigation simulator, where it achieves a 43% success rate on `NavWorld`-labeled instructions, more than $9\times$ that of the navigation Vision-Language-Action (VLA) baselines.

1 INTRODUCTION

"Leave the building entrance, turn right onto the paved path, and follow it past the trees. Stop beside the yellow traffic cone." For a person, following such an instruction is simple. For a robot, however, this long-horizon execution introduces a series of challenges. It must identify the intended path and recognize relevant landmarks in a changing scene, while also breaking the route into ordered subgoals. As it moves, the robot must choose actions while the final destination may still be out of view. Simultaneously, it must decide when to continue navigating to the current subgoal, when to move on to the next subgoal, or when to come to a stop. Moreover, as routes grow longer, a missed landmark or premature transition can introduce compounding errors that throw off subsequent subgoals. We study these challenges in *long-horizon, language-conditioned, outdoor navigation*, where robots follow multi-step language instructions through sidewalks, campus paths, parks, and other outdoor spaces beyond the road-centric autonomous driving setting.

Prior work has developed navigation agents that follow language instructions inside homes and office buildings, making substantial advancements on indoor benchmarks (Wei et al., 2025b;c), as well as outdoor robots (Shah et al., 2023b; Hirose et al., 2024; Cheng et al., 2025) that navigate using destination images, language instructions, or goal pose. Our work aims to advance outdoor, ground-robot navigation by enabling robots to follow multi-stage route instructions across diverse unseen outdoor environments. This raises two challenges in training data and policy execution. First, outdoor environments vary widely in layout, appearance, and landmarks. Learning across this variation calls for geographically diverse trajectories with reliable language instruction labels. Second, identifying the current sub-task is challenging because similar visual observations can correspond to different stages of the overall task. For example, when instructed to "pass the building, then turn right at the intersection," the robot may see the same building both before and after passing it, making it unclear whether it should continue straight or begin the turn. The robot must therefore infer which stage of

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

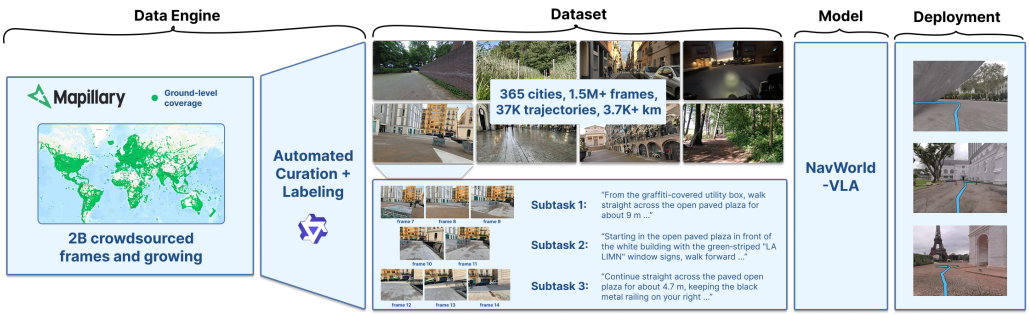


Figure 1: **Introducing `NavWorld`.** Our data engine curates geographically diverse Mapillary sequences, segments retained trajectories into navigation subtasks, and generates instructions from visual observations and trajectory geometry. The resulting dataset contains over 1.5 million frames from 37,000 trajectories spanning more than 3,700 km across 365 cities. We train a language-conditioned navigation policy on these labeled trajectories. The rightmost panels show zero-shot ground-robot navigation in an outdoor simulator.

the instruction is currently active, execute the corresponding actions, recognize when that stage is complete, and determine when to advance or stop.

We address the challenge of **long-horizon outdoor navigation by automatically turning large-scale, geographically diverse outdoor egocentric imagery into hierarchically language-labeled trajectories**. We leverage Mapillary (Neuhold et al., 2017), a large-scale crowdsourced platform of geotagged ground-level image sequences totalling more than two billion frames collected across diverse geographic regions and capture conditions. Mapillary has been a scalable source of data popular in enabling real-world perception tasks, such as, visual localization (Sarlin et al., 2023), place recognition (Warburg et al., 2020), and bird's-eye-view mapping (Ho et al., 2024). Specifically, we first filter out potentially noisy trajectories using criteria such as geometric consistency and perceptual quality. We then use trajectory geometry to decompose long trajectories into meaningful intermediate navigation subtasks. Finally, we generate a subtask instruction for each resulting image sequence by passing its visual observations and corresponding trajectory geometry to a VLM.

Our key contributions are:

- **`NavWorld` data engine:** An automated pipeline that converts raw Mapillary sequences into language-labeled navigation trajectories. It filters potentially noisy egocentric image sequences, divides long routes into intermediate subtasks, and generates language instructions grounded in the visual observations and trajectory geometry.
- **`NavWorld` dataset:** A geographically diverse outdoor language-conditioned navigation dataset generated by our data engine, comprising over 1.5M frames across more than 3,700 km of trajectories from 365 cities.
- **`NavWorld` model:** We train a language-conditioned policy for long-horizon outdoor navigation using monocular egocentric RGB observations and high-level instructions. Evaluated zero-shot in Habitat-GS (Xia et al., 2026b), a photorealistic simulator with outdoor environments, `NavWorld` achieves a 43% success rate on `NavWorld` labeled instructions, more than $9\times$ that of the evaluated navigation VLA baselines.

For more qualitative results, please visit our project website.

2 Related Work

Recent instruction-following navigation models, including Vision-Language-Navigation (VLN) and navigation VLA approaches such as StreamVLN (Wei et al., 2025c), NaVILA (Cheng et al., 2025), and InternVLA-N1-DualVLN (Wei et al., 2025a), leverage large-scale datasets such as R2R (Anderson et al., 2017), RxR (Ku et al., 2020), ScaleVLN (Wang et al., 2023), and InternData-N1 (InternData-N1 Dataset contributors, 2025) to achieve strong navigation performance. These

datasets leverage simulated environments to scale instruction–trajectory pairs to millions of examples and visual observations to tens of millions of frames. However, one key limitation is that all these datasets are indoors. These resources provide substantial supervision for indoor instruction following, motivating complementary data that capture geographically diverse outdoor routes and their associated language instructions.

Following language instructions outdoors requires a robot to identify the intended route by grounding references to paths, turns, and landmarks in its observations. Some outdoor navigation methods provide guidance through spatial waypoints or reference images: CityWalker (Liu et al., 2025) uses waypoint goals, while GNM (Shah et al., 2023a), ViNT (Shah et al., 2023b), and NoMaD (Sridhar et al., 2023) support visual-goal navigation.

Outdoor language navigation has been studied with both route instructions and other forms of language supervision. Talk2Nav (Vasudevan et al., 2019) annotates street-view routes with landmark references and local directional descriptions. OmniVLA (Hirose et al., 2025) supports language alongside image and pose goals, while UrbanNav (Mei et al., 2026) learns navigation toward language-described landmarks from recorded walking trajectories. WILD-Nav (Xia et al., 2026a) supervises textual rationales alongside actions for point-goal navigation, while VAMOS (Castro et al., 2026) uses optional language preferences to guide navigation toward coordinate goals. VEGA (Seneviratne et al., 2026) uses reconstructed scene geometry to generate goal-conditioned local trajectories, including trajectories toward objects specified in language.

NavWorld constructs supervision that links route-level instructions to the execution of successive language subtasks along recorded outdoor trajectories. Its data engine curates crowdsourced Mapillary sequences and their camera metadata, identifies subtask boundaries from trajectory geometry, and generates subtask instructions from the corresponding visual observations and motion. The resulting dataset pairs geographically diverse outdoor routes with route-level instructions, estimated metric motion, and temporally aligned subtask annotations, providing supervision for local execution and transitions between subtasks.

Navigation models also differ in how they represent task progress and condition local actions. StreamVLN (Wei et al., 2025c) manages streaming observations through slow-fast context modeling. NaVILA (Cheng et al., 2025) generates language-formatted motion commands for a locomotion policy, while InternVLA-N1-DualVLN (Wei et al., 2025b) connects instruction understanding to trajectory generation through pixel goals and learned latent representations. NavWorld organizes execution around a persistent language subtask and explicitly decides whether to continue it, generate the next subtask, or terminate navigation. Within a shared VLM, hierarchical attention retains route-level context for subtask generation and progress control while restricting direct action conditioning to the active subtask and its execution history. The resulting representations provide layerwise conditioning to a flow-matching action expert.

3 METHOD

3.1 OUTDOOR NAVIGATION DATA ENGINE

We introduce the `NavWorld` Data Engine, a pipeline for constructing training data for outdoor language-conditioned navigation from crowdsourced street-view imagery. As illustrated in Fig. 2, the pipeline comprises sequence collection, geometric and perceptual filtering, trajectory segmentation, and instruction annotation. The resulting data pair visual observations and estimated metric motion with temporally aligned subtask instructions, providing supervision for local execution and progression along a route.

3.1.1 SCALABLE DATA ENGINE THAT UTILIZES MAPILLARY DATA

Collecting navigation data across geographic regions and capture conditions requires substantial effort. We therefore use Mapillary, a global crowdsourced street-view platform containing over 2 billion images (Neuhold et al., 2017), as our source of candidate sequences. Its imagery spans diverse locations, camera devices, environmental conditions, and modes of travel, providing a broad collection from which to curate navigation training data. We now describe the data sourcing and curation process, with more details found in Appendix B.

The collection pipeline accepts geographic bounding boxes or city names, which we resolve to bounding boxes. We enumerate the map tiles intersecting these regions and query Mapillary's vector-tile API to retrieve sequence IDs. Each sequence consists of a varying number of images captured by a contributor along a traveled path. For each sequence, we retrieve frame-level metadata, including RGB image URLs, timestamps, raw geographic coordinates, positions and orientations estimated through Structure from Motion (SfM), and camera intrinsics. This stage produces candidate sequences represented by image references and camera metadata, which we then choose from in the dataset curation step.

3.1.2 Geometric and Perceptual Filtering

Crowdsourced sequences may contain camera views that are misaligned with the direction of travel, as well as observations with severe occlusion or insufficient visible navigable ground. We therefore assess trajectory consistency through camera metadata and visual suitability through a vision-language model (VLM).

Geometric quality. We construct camera poses in a local metric coordinate frame using Mapillary's SfM-corrected positions and estimated orientations, with raw GPS coordinates used when corrected positions are unavailable. We reject sequences with excessively upward- or downward-facing camera orientations. We also measure the angle between the camera's forward axis and its inter-frame displacement over moving frames, retaining sequences whose median angular deviation falls below a predefined threshold.

Perceptual quality. Geometric consistency alone does not establish whether an image provides a usable view of the navigable environment. However, applying a VLM to every frame is computationally expensive at the scale of billions of images. Since poor viewing conditions often persist across adjacent frames, we evaluate a single sampled image from each candidate segment as an efficient approximation of its perceptual quality. The VLM rejects images with insufficient visible navigable ground, unsuitable viewpoints, severe occlusion, or visual degradation that prevents scene interpretation.

3.1.3 Automatic Grounded Data Labeling Pipeline

To provide explicit supervision for progress along a route, we partition each trajectory into temporally bounded segments. Each segment is annotated with a subtask instruction, linking a stage of the route to its corresponding visual observations and motion.

We determine subtask boundaries from the trajectory geometry. Candidate boundaries are placed immediately before and after detected turn events, isolating individual maneuvers from the surrounding motion. Long segments without a discrete turn are further subdivided based on cumulative heading change and cumulative arc length. A new boundary is introduced when either quantity exceeds a predefined threshold. This prevents extended straight or gradually curving portions of the trajectory from forming overly long subtasks while preserving locally coherent motion within each segment.

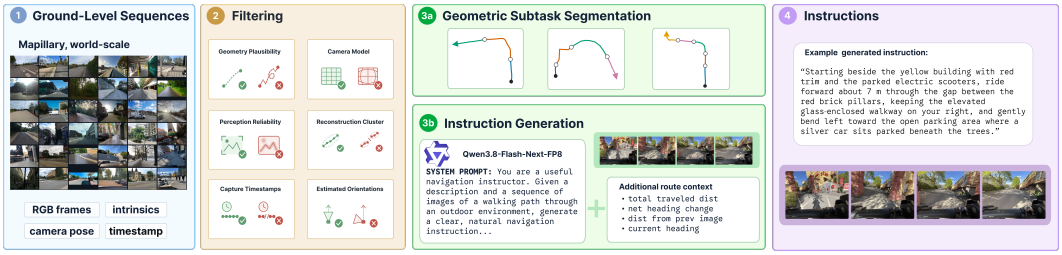


Figure 2: **Overview of `NavWorld` Data Engine.** We retrieve ground-level Mapillary image sequences and camera metadata (1), then clean and filter them for plausible motion, suitable camera views, temporal continuity, and visual quality (2). Using the retained trajectory geometry, we divide each route into subtasks (3a). A vision-language model receives each subtask's ordered images and motion context to generate a landmark-grounded navigation instruction (3b), producing trajectories with ordered subtask-level language labels (4).

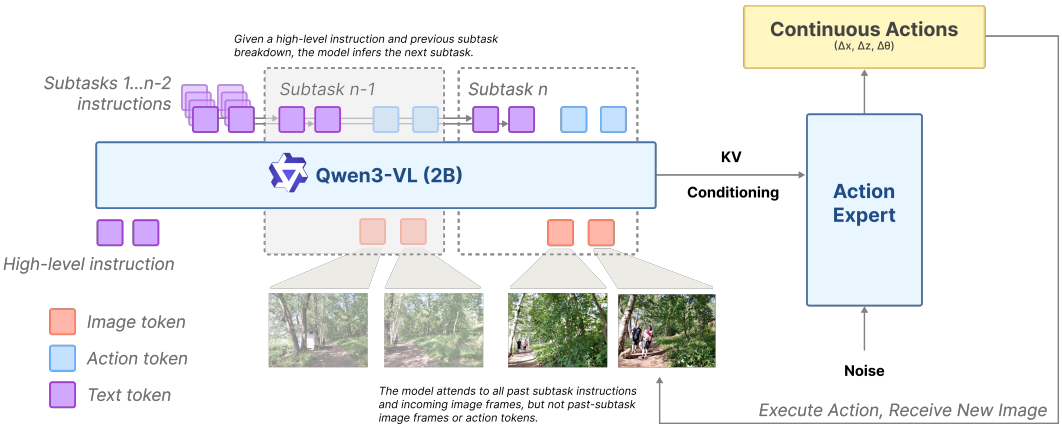


Figure 3: **Overview of `NavWorld` Architecture.** Our model decouples navigation into hierarchical planning and continuous execution. **Left (High-Level Planner):** A Qwen3-VL-2B-Instruct backbone takes the high-level navigation instruction alongside history and current RGB observations to decompose the route autoregressively into concise subtasks. **Right (Trajectory Generator):** A flow-matching action expert receives layerwise conditioning from the backbone context (active subtask text and subtask-relevant visual history) and predicts continuous, multi-step action chunks to generate the target trajectory.

For each subtask, we use Qwen3.8-Flash-Next-FP8 (Qwen Team, 2026) with a segment's egocentric RGB observations and corresponding trajectory geometry. The image sequence provides the visual context of the demonstrated route, while the geometry describes its demonstrated motion; together, they condition the generation of a concise navigation instruction. This grounds each annotation in both the observed scene and the demonstrated motion. The resulting ordered sequence of subtask annotations provides structured supervision for local navigation behavior and transitions between subtasks in our hierarchical model.

3.2 NavWorld-VLA Model

We train a `NavWorld-VLA` model that leverages the hierarchical structure of our curated dataset, allowing it to take in task-level instructions, language subtasks, and local motion within a unified navigation policy. Inspired by the VLA architectures of π_0 and $\pi_{0.5}$ (Black et al., 2025b;a), our model combines a Qwen3-VL-2B-Instruct VLM backbone (Qwen Team, 2025) for subtask generation and control with a flow-matching (Lipman et al., 2022) action expert for continuous motion prediction as shown in Figure 3.

Related VLA architectures, such as GR00T N1 Bjorck et al. (2025), also couple a vision-language model with a flow-matching action module. NavWorld adapts this architectural pattern to long-horizon navigation with subtask-aware context.

Specifically, given a navigation instruction ℓ, the `NavWorld-VLA` model receives an egocentric RGB observation o_t at each decision step t. We denote the instruction associated with the k-th subtask by ℓ_k and the active subtask index at step t by $k(t)$. At the beginning of a navigation session, the model generates the first subtask instruction ℓ_1 conditioned on ℓ and the initial observation o_0, with $k(0) = 1$. To execute a subtask, the policy predicts an action chunk $\mathbf{A}_t = [\mathbf{a}_{t,1}, \ldots, \mathbf{a}_{t,H}]^\top \in \mathbb{R}^{H\times 3}$, where each $\mathbf{a}_{t,h} = (\Delta x_{t,h}, \Delta z_{t,h}, \Delta\psi_{t,h})^\top$ specifies a relative planar displacement and yaw change in degrees, expressed relative to the preceding pose. We use $H = 4$ motion increments per chunk. After executing each chunk, the robot receives a new observation, and the policy determines whether to continue the active subtask, generate the next subtask instruction, or terminate navigation. The following subsections describe how our modified attention mask assigns different context to subtask control and local execution, and how layerwise conditioning allows the action expert to translate the resulting backbone representations into continuous motion.

Subtask-aware Attention Masking: Within the shared VLM backbone (Fig.3), the attention masking mechanism determines which context is available to subtask generation, progress control, and local action prediction. Subtask generation and progress control require high-level information, whereas local action prediction primarily requires the active subtask and its execution history. Building on the language-subtask hierarchy of Black et al. (2025a), we implement this separation with a subtask-aware mask, where when executing a given subtask, only the language instructions from previous subtasks is shared. The resulting backbone representations provide layerwise conditioning for the continuous action expert described in Sec. 3.2.

Let b_k denote the decision step at which subtask k begins. When generating ℓ_k, the backbone can directly attend to the full instruction ℓ, previously generated subtask instructions $\ell_1, \ldots, \ell_{k-1}$, and the current observation o_{b_k}. During local execution at step t, action prediction can directly attend to the active subtask instruction $\ell_{k(t)}$, observations $o_{b_{k(t)}:t}$, and preceding action chunks $\mathbf{A}_{b_{k(t)}:t-1}$, represented as discrete tokens. Token generation within each subtask description or discrete action chunk follows causal order. Visual tokens attend only within their own observation, keeping their representations independent of task text.

A controller token at the end of each observation determines whether to continue the active subtask, begin a new subtask, or terminate navigation. Its context includes the full instruction, all previously generated subtask descriptions, the active subtask's observation and action history, and the current observation. This context enables the controller to relate local progress to the overall instruction while keeping action generation focused on the active subtask. The same attention rules apply during training and inference.

Continuous Action Generation with Layerwise Conditioning: Discrete action supervision and subtask-based attention mask adapt the pretrained backbone to represent task structure and navigation behavior. However, discretized actions have limited resolution, while precise navigation benefits from continuous motion predictions. Furthermore, the conditional action distribution can be multimodal, as multiple distinct trajectories may satisfy the same subtask. We therefore introduce a flow-matching action expert that generates continuous action chunks conditioned on the backbone's navigation representations.

The action expert uses a transformer architecture adapted from the diffusion transformer (DiT) (Peebles & Xie, 2022). Each expert block attends to keys and values from the corresponding backbone layer together with the noisy action tokens. The expert follows the same subtask-local visibility rules as action prediction, excluding the discrete tokens of the action chunk $\mathbf{A}_t$ currently being predicted.

We train the expert with flow matching between Gaussian noise and normalized action targets. At inference, we integrate its predicted velocity field from a Gaussian sample and denormalize the result to obtain $\mathbf{A}_t$. Following the principle of knowledge insulating Driess et al. (2025), we stop gradients through the backbone keys and values supplied to the expert. Consequently, the flow-matching objective updates the action expert, while the backbone continues to learn through cross-entropy supervision of subtask text, discrete actions, and transition and termination decisions.

3.2.1 Model Training

We train `NavWorld` in three stages. Stages 1 and 2 use training windows containing 1–3 consecutive subtasks, each paired with a window-specific instruction and an endpoint termination target. Stage 1 adapts the backbone using cross-entropy supervision of subtask text, discrete actions, and transition and termination decisions. Stage 2 initializes the backbone from Stage 1 and adds a flow-matching action expert, optimizing $\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_{\text{FM}}\mathcal{L}_{\text{FM}}$. Stage 3 continues training with a subtask-window curriculum, using windows of 1–6 subtasks for one epoch, followed by windows of 1–9 subtasks for one epoch, to enable our policy for longer-horizon control.

Following knowledge insulation (Driess et al., 2025), we detach the backbone keys and values supplied to the expert: cross-entropy updates the backbone, while flow matching updates only the action expert. Training uses eight NVIDIA H200 GPUs for approximately 24 hours on the full `NavWorld` dataset.

Table 1: **Zero-shot evaluation on held-out Habitat-GS outdoor scenes.** All methods are evaluated on the same 400 episodes under corrected Habitat-GS instructions and NavWorld instructions generated from the reference trajectories. NE is the geodesic distance in metres; SR, OSR, and SPL are percentages. SPL uses a 3 m success threshold. Bold indicates the best value within each instruction condition.

	Habitat-GS Instructions						NavWorld Instructions					
Method	NE↓	SR_{3m} ↑	OSR_{3m} ↑	SR_{10m} ↑	OSR_{10m} ↑	SPL↑	NE↓	SR_{3m} ↑	OSR_{3m} ↑	SR_{10m} ↑	OSR_{10m} ↑	SPL↑
NavWorld (ours)	**8.45**	**20.2**	37.8	**73.0**	79.0	**18.7**	**5.57**	**43.0**	**50.0**	**86.0**	88.5	**41.3**
StreamVLN Wei et al. (2025c)	19.60	3.5	33.5	25.2	73.2	3.3	22.61	4.8	37.2	22.5	77.2	4.4
InternVLA-N1-DualVLN Wei et al. (2025a)	16.87	5.0	43.0	24.0	82.8	4.3	17.97	2.8	41.2	18.8	84.2	2.1
NaVILA Cheng et al. (2025)	26.33	0.5	**43.8**	8.2	**83.5**	0.4	24.93	2.5	46.8	12.5	**89.8**	2.2

4 EXPERIMENTS

We evaluate `NavWorld`'s generalization to unseen outdoor environments and investigate the contributions of training-data scale. Our experiments are designed to answer two questions:

RQ1 Can `NavWorld` enable stronger zero-shot outdoor language-conditioned navigation compared to baselines?

RQ2 How does the scale of the `NavWorld` dataset affect downstream navigation performance?

Setup. We evaluate on eight held-out outdoor scenes from *Habitat-GS* (Xia et al., 2026b), with 50 navigation episodes per scene for a total of 400 episodes. The routes span approximately 10–50 m and require long-horizon outdoor navigation. [1]

Metrics. Consistent with VLN tasks (Anderson et al., 2017; Savva et al., 2019), we report navigation error (NE), success rate (SR), oracle success rate (OSR), and success weighted by path length (SPL). NE measures the final geodesic distance to the goal; SR measures final goal-reaching success, whereas OSR measures whether the agent enters the goal region at any point during the rollout. SPL additionally accounts for path efficiency.[2]

Baselines. To capture both precise goal-reaching and meaningful progress toward the destination, we compare `NavWorld` against recent VLN approaches with different model architectures, including StreamVLN Wei et al. (2025c), InternVLA-N1-DualVLN Wei et al. (2025a), and NaVILA Cheng et al. (2025). Experiment-specific training protocols, and additional metrics are described in the corresponding subsections.

4.1 CAN NAVWORLD OUTPERFORM EXISTING VLN METHODS OUTDOORS?

Protocol. To answer RQ1, we evaluate each model under two instruction conditions: the corrected Habitat-GS instructions[1] and NavWorld instructions generated by applying our labeling pipeline to the same reference trajectories. The latter provide more detailed route descriptions. Both conditions use identical scenes, start states, goals, and reference trajectories. Reference trajectories are used for annotation and evaluation but are not provided to the policies during inference.

Results. Table 1 shows that `NavWorld` achieves the lowest NE under both instruction conditions. It also attains the highest SR at both thresholds and the highest SPL. Compared with the strongest baseline in each condition, SR at 3 m improves by 15.2 percentage points with corrected Habitat-GS instructions and by 38.2 points with NavWorld instructions. Some baselines achieve higher OSR despite lower SR, indicating that such baselines often fail to keep track of navigation progress and stop at the intended1 position. Figure 4 illustrates selected rollouts and their deviations from the reference paths.

[1] We corrected left/right inconsistencies in the supplied instructions and increased the rendering distance to improve sky rendering. All methods use the same corrected instructions and rendering settings for evaluations.

[2] The 3 m threshold follows indoor VLN evaluation such as R2R (Anderson et al., 2017); a 10 m threshold is used in the continuous outdoor setting of DaViNCi (Xie et al., 2026). DaViNCi reports Euclidean distance, whereas our NE is geodesic.

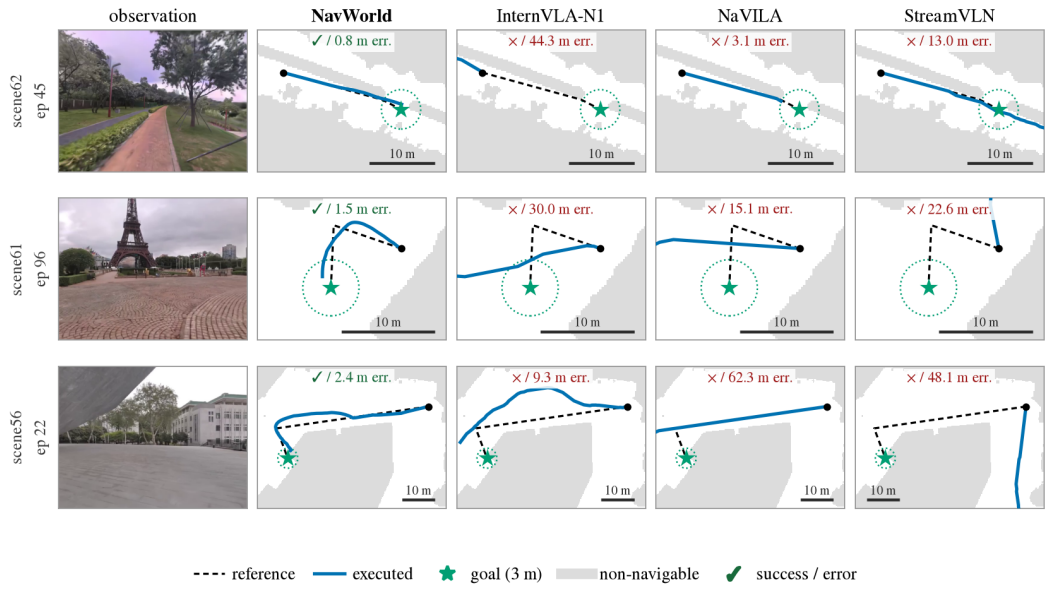


Figure 4: **Qualitative outdoor simulator navigation with `NavWorld` instructions.** All methods follow instructions generated by our grounded labeling pipeline from Habitat-GS reference trajectories. Examples are selected for clear instructions, identifiable goals, and rendering quality. Dashed black lines show reference paths, and blue lines show executed trajectories. Numerical labels report final geodesic navigation error; check marks and crosses indicate success and failure at the 3 m threshold.

4.2 Benefits of NavWorld Dataset Scale

Protocol. To answer RQ2, we train the same model architecture on different sizes of `NavWorld` dataset(0.25x, 0.5x, 1x) and evaluate all variants on the same held-out Habitat-GS episodes using our labels. The 3-stage training is kept the same for 3 models while we change the dataset size, including the fixed-epoch training recipe: larger datasets receive more optimizer updates.

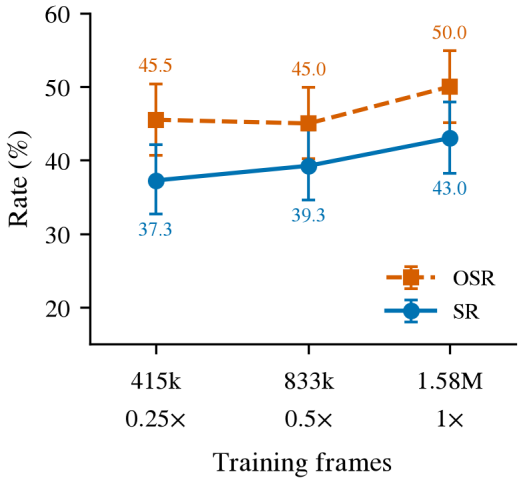


Figure 5: **Data scaling on Habitat-GS.** SR and OSR on 400 episodes with NavWorld instructions evaluated on 3 models trained on 0.25x, 0.5x, and 1x dataset scale. Error bars show 95% Wilson intervals.

Results. Larger NavWorld training sets yield higher success rates on held-out Habitat-GS episodes under our training recipe. As shown in Figure 5, increasing the number of training frames from 415k to 833k and 1.58M raises SR from 37.3% to 39.3% and 43.0%, respectively. The full dataset therefore improves SR by 5.7 percentage points over the smallest subset and by 3.7 points over the intermediate subset. It also achieves the highest OSR, at 50.0%, compared with 45.5% and 45.0% for the smaller subsets, although OSR does not increase monotonically across all three scales.

These results show that the largest training setting improves both entry into the goal region and successful episode completion, supporting the utility of expanding the automatically generated NavWorld corpus for navigation in held-out environments. Because the epoch budgets are fixed, larger datasets also receive more optimizer updates; the observed gains therefore reflect scaling the dataset under this training recipe.

5 Conclusion

We presented `NavWorld`, an ecosystem for long-horizon, language-conditioned outdoor navigation. At the core of our approach is a data engine that automatically converts large-scale, geographically diverse Mapillary ground-level egocentric image sequences into language-labeled navigation data. The data engine automatically sources from desired cities, filters potentially noisy trajectories, decomposes long trajectories into intermediate subtasks, and generate grounded language instructions. Using this engine, we construct a large-scale dataset comprising more than 1.5 million frames and over 3,700 km of trajectories across 365 cities. Using this data, we train a language-conditioned navigation policy that follows long route instructions from monocular egocentric observations and achieves a 43% success rate in zero-shot Habitat-GS evaluation on `NavWorld`-labeled instructions, substantially outperforming navigation VLA baselines. By transforming readily available crowdsourced imagery into navigation supervision, our approach offers an alternative to costly robot-specific data collection and labeling. We believe this direction can enable mobile robots that navigate reliably across increasingly broad and geographically diverse outdoor environments.

6 Other

AI use statement

In this work, we used generative AI tools for our data engine (data filtering, trajectory instruction labeling, and more), providing feedback on experiments, implementing methods, and interpreting results. Additionally, we used generative AI tools for editing and creating software code, brainstorming, identifying and summarizing relevant literature, editing research paper for readability, and drafting a summary of experimental setup for the appendix. We have reviewed all AI-assisted work. We verified and tested LLM-generated code, manually checked and verified relevant literature, and reviewed suggested edits to our paper and drafts of the appendix. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

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, 2017. URL `https://arxiv.org/abs/1711.07280`.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Lin-Xi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv.org*, 2025. doi: 10.48550/arXiv.2503.14734.

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, 2025a. URL `https://arxiv.org/abs/2504.16054`.

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, 2025b. URL https://arxiv.org/abs/2410.24164.

Mateo Guaman Castro, Sidharth Rajagopal, Daniel Gorbatov, Matt Schmittle, R. Baijal, Octi Zhang, Rosario Scalise, Sidharth Talia, Emma Romig, C. D. de Melo, et al. Vamos: A hierarchical vision-language-action model for capability-modulated and steerable navigation, 2026. URL https://arxiv.org/abs/2510.20818.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. NaVILA: Legged robot vision-language-action model for navigation, 2025. URL https://arxiv.org/abs/2412.04453.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better, 2025. URL https://arxiv.org/abs/2505.23705.

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. In *Conference on Robot Learning*, 2024. doi: 10.48550/arXiv.2410.03603.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Omnivla: An omni-modal vision-language-action model for robot navigation, 2025. URL https://arxiv.org/abs/2509.19480.

Cherie Ho, Jiaye Zou, Omar Alama, Sai Mitheran Jagadesh Kumar, Benjamin Chiang, Taneesh Gupta, Chen Wang, Nikhil Keetha, Katia Sycara, and Sebastian Scherer. Map it anywhere (mia): Empowering bird's eye view mapping using large-scale public data. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2407.08726.

InternData-N1 Dataset contributors. Interndata-n1 dataset. https://huggingface.co/datasets/InternRobotics/InternData-N1, 2025.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding, 2020. URL https://arxiv.org/abs/2010.07954.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2022. URL https://arxiv.org/abs/2210.02747.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6875–6885. IEEE, 2025. doi: 10.1109/cvpr52734.2025.00645.

Yanghong Mei, Yirong Yang, Longteng Guo, Qunbo Wang, Ming-Ming Yu, Xingjian He, Wenjun Wu, and Jing Liu. Urbannav: Learning language-guided embodied urban navigation from web-scale human trajectories. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18505–18513. Association for the Advancement of Artificial Intelligence (AAAI), 2026. doi: 10.1609/aaai.v40i22.38916.

Gerhard Neuhold, Tobias Ollmann, S. R. Bulò, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *IEEE International Conference on Computer Vision*, pp. 5000–5009. IEEE, 2017. doi: 10.1109/ICCV.2017.534.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2022. URL https://arxiv.org/abs/2212.09748.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qwen Team. Qwen3.8-Flash-Next: A new architecture, towards ultimate cost-efficiency, August 2026. URL https://qwen.ai/blog?id=qwen3.8-flash-next.

Paul-Edouard Sarlin, Daniel DeTone, Tsun-Yi Yang, Armen Avetisyan, Julian Straub, Tomasz Malisiewicz, Samuel Rota Bulo, Richard Newcombe, Peter Kontschieder, and Vasileios Balntas. OrienterNet: Visual Localization in 2D Public Maps with Neural Matching. In *CVPR*, 2023.

M. Savva, Jitendra Malik, Devi Parikh, Dhruv Batra, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, et al. Habitat: A platform for embodied ai research. *IEEE International Conference on Computer Vision*, pp. 9338–9346, 2019. doi: 10.1109/ICCV.2019.00943.

Gershom Seneviratne, Yohan Abeysinghe, Jianyu An, Vaibhav Shende, Rahul Kumar, and Dinesh Manocha. Vega: Learning navigation vlas from in-the-wild egocentric video with geometric trajectory supervision, 2026. URL https://arxiv.org/abs/2606.18426.

Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. GNM: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7226–7233. IEEE, 2023a. doi: 10.1109/icra48891.2023.10161227.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. ViNT: A foundation model for visual navigation. In *Conference on Robot Learning*, 2023b. doi: 10.48550/arXiv.2306.14846.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. NoMaD: Goal masked diffusion policies for navigation and exploration. In *IEEE International Conference on Robotics and Automation*, pp. 63–70. IEEE, 2023. doi: 10.1109/ICRA57147.2024.10610665.

A. Vasudevan, Deng-Xin Dai, and L. van Gool. Talk2nav: Long-range vision-and-language navigation with dual attention and spatial memory. *International Journal of Computer Vision*, 129(1): 246–266, 2019. doi: 10.1007/s11263-020-01374-3. URL https://doi.org/10.1007/s11263-020-01374-3.

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 *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11975–11986. IEEE, 2023. doi: 10.1109/iccv51070.2023.01103.

Frederik Warburg, Soren Hauberg, Manuel Lopez-Antequera, Pau Gargallo, Yubin Kuang, and Javier Civera. Mapillary street-level sequences: A dataset for lifelong place recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, and Xihui Liu. Ground slow, move fast: A dual-system foundation model for generalizable vision-and-language navigation, 2025a. URL https://arxiv.org/abs/2512.08186.

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, 2025b. URL https://arxiv.org/abs/2512.08186.

Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, et al. StreamVLN: Streaming vision-and-language navigation via slowfast context modeling, 2025c. URL https://arxiv.org/abs/2507.05240.

Bingyi Xia, Han Bao, Zhewei Chen, Hanjing Ye, Jingwen Yu, Yuhan Pang, Wenjun Xu, and Jiankun Wang. Exposing the long-tail in embodied urban navigation via scalable learning from in-the-wild videos, 2026a. URL https://arxiv.org/abs/2608.16476.

Ziyuan Xia, Jingyi Xu, Chong Cui, Yuanhong Yu, Jiazhao Zhang, Qingsong Yan, Tao Ni, Junbo Chen, Xiaowei Zhou, Hujun Bao, et al. Habitat-gs: A high-fidelity navigation simulator with dynamic gaussian splatting, 2026b. URL https://arxiv.org/abs/2604.12626.

Zihao Xie, Pingrui Lai, Yitong Wu, and Hua Yang. Davinci: A dataset towards outdoor vision-and-language navigation with continuous actions and dynamic elements, 2026. URL https://arxiv.org/abs/2608.11901.

Table 2: **Supplementary dataset statistics.** Other than the "validation trajectories" row, all statistics refer to the training partition. Geographic coverage counts cities/regions with retained data. Distributions are reported as median [25th, 75th percentile].

Quantity	`NavWorld` dataset
Training trajectories	36,142
Validation trajectories	1,059
Cities/regions with retained data	365
Source sequences	27,340
Unique source frames	1,578,984
Annotated subtasks	494,563
Trajectory length (m)	79.3 [48.5, 143.9]
Subtask length (m)	7.3 [5.8, 9.1]
Subtasks per trajectory	10 [7, 19]

Table 3: **Metadata used during collection and filtering.**

Metadata	Use
Sequence and frame identifiers	Associate observations with source sequences and annotation records.
Image references	Retrieve RGB observations for filtering, labeling, and policy training.
Capture timestamps	Order observations and identify temporal gaps.
Raw geographic coordinates	Locate source observations and provide a position fallback when corrected coordinates are unavailable.
SfM-corrected positions	Construct a trajectory in a local metric coordinate system.
Estimated orientations	Evaluate camera pitch and viewing-direction alignment; derive relative motion.
Camera model, parameters, and image dimensions	Interpret the camera geometry and image coordinate system.
Reconstruction component identifier	Identify component boundaries for discontinuity checks (`merge_cc`).

A Dataset Statistics

Table 2 summarizes the collected `NavWorld` dataset.

B Data Engine and Annotation Details

The final pipeline constructs navigation trajectories from filtered RGB observations and camera-motion metadata, then generates geometric subtask boundaries and language instructions. This appendix specifies the filtering criteria, segmentation parameters, and labeling inputs used to produce these trajectories.

B.1 Sequence Retrieval and Metadata

The collection pipeline accepts geographic bounding boxes or city names. City names are resolved to query regions, which are covered by map tiles. We query these tiles for sequence identifiers and retrieve the associated frame metadata. Table 3 groups the metadata by its use in the pipeline. The end-to-end collection recipe uses zoom-14 tiles and a 1024-pixel image long-edge setting. Frames are ordered by capture time, with frame identifier used to break ties.

We construct camera poses in a sequence-local metric frame with axes pointing east, down, and north. SfM-corrected geographic coordinates are preferred per frame, with raw coordinates used as a fallback. We assume locally flat terrain and project camera positions onto a horizontal plane, discarding recorded altitude variation. Camera orientation is recovered from the supplied rotation vector; frames without a usable rotation use a yaw-only orientation from the computed compass heading, then the raw heading, with zero as the final fallback. The camera pitch filter rejects sequences with no usable rotation vectors, although individual retained frames may still use a heading fallback.

Table 4: **Geometric processing parameters for the filtering pipeline.**

Parameter	Description	Value
Moving-frame threshold	Minimum displacement for viewing/motion alignment	$0.2\,\mathrm{m}$
Pitch acceptance interval	Median optical-axis elevation	$(-45^\circ, 25^\circ)$
Alignment threshold	Median viewing/motion deviation must be below	25°
Retained sequence length	Minimum number of frames after thinning	25
Candidate range length	Number of retained observations	25–90
Candidate density limit	Traveled distance divided by frame count	$4\,\mathrm{m/frame}$
Turning-content threshold	Minimum number of turning-classified frames	3
Per-sequence range limit	Maximum number of selected ranges	4
Segmentation grid	Arc-length resampling interval	$0.5\,\mathrm{m}$
Heading chord	Extent before and after each grid location	$3.7\,\mathrm{m}$
Turn-rate threshold	Absolute heading-change rate	$4^\circ/\mathrm{m}$
Turn-angle threshold	Minimum net change across a detected interval	25°
Approach/exit padding	Offset outside detected turn intervals	$1\,\mathrm{m}$
Subtask-length targets	Minimum and maximum target lengths	3–10 m

B.2 Geometric and Perceptual Filtering

Geometric filtering removes sequences whose camera motion or viewpoint is unsuitable for navigation supervision. We first reject unsupported camera projection models, then check camera pitch and the alignment between the viewing direction and inter-frame motion. Because travel direction is unreliable when the camera is nearly stationary, alignment is computed only on moving frames. We use the median alignment angle over the sequence to reduce the effect of noisy frame-level estimates.

We also split sequences at clear spatial or temporal discontinuities. A split is triggered by an inter-frame hop larger than four times the median hop, a time gap over $300\,\mathrm{s}$, or a turn greater than 100° between consecutive moving hops. At SfM reconstruction-component boundaries, we use stricter thresholds of three times the median hop or 60°. To detect short-range reversals, we compare endpoint displacement with traveled distance over a $15\,\mathrm{m}$ forward window and split when the ratio falls below 0.55. These splits are used only for sequence cleaning and are separate from the language subtasks introduced below.

We subsample the native frame sequence before constructing candidate ranges. Sampling is based on accumulated viewpoint change, combining traveled distance normalized by $2\,\mathrm{m}$ with absolute camera yaw change normalized by 10°. We retain a frame whenever this combined budget reaches one. Sequences with fewer than 25 retained frames are removed.

Candidate ranges contain 25–90 frames and are required to have at most $4\,\mathrm{m}$ of traveled distance per frame on average. To improve coverage, we select windows within $300\,\mathrm{m}$ placement buckets, prioritize windows that contain maneuvers, and keep at most four ranges from each sequence. Each retained range must also contain at least three turning frames. If any image in a range cannot be retrieved, that range is discarded.

For perceptual filtering, we inspect one sampled RGB frame from each candidate segment. The filter checks whether the image contains enough visible navigable surface and scene context to support navigation supervision, accounting for visibility, occlusion, traversability, and image quality.

B.3 Geometric Subtask Segmentation

For each trajectory retained by filtering, we determine subtask boundaries from the ground-plane camera positions supplied by Mapillary. We first resample the trajectory at $0.5\,\mathrm{m}$ intervals and estimate heading using a chord extending $3.7\,\mathrm{m}$ in each direction, clipped at the trajectory endpoints. We unwrap the heading sequence and detect contiguous intervals where the absolute heading-change rate exceeds $4^\circ/\mathrm{m}$. Intervals whose net change reaches 25° are retained as turns, and candidate boundaries are placed $1\,\mathrm{m}$ before and after them.

For gaps longer than $10\,\mathrm{m}$, we choose a low-curvature split 3–10 m after the previous boundary while leaving at least $3\,\mathrm{m}$ before the next. Splits inside detected turns, including a $0.5\,\mathrm{m}$ buffer, are

Table 5: **Instruction-labeler parameters.**

Parameter	Value
Model identifier	Qwen/Qwen3.8-Flash-Next-FP8
Image selection	All retained segment frames, in travel order
Image maximum side / request limit	768 pixels / 72 images
Thinking / reasoning effort	Enabled / low
Temperature / top-p / top-k	1.0 / 0.95 / 20
Minimum probability	0
Presence / repetition penalty	0 / 1.0
Generation / context limit	2048 / 49152 tokens

excluded. We remove short segments, snap boundaries to the nearest original frames, and deduplicate them while preserving the endpoints. A final pass removes short segments and splits long ones when possible. The 3–10 m range is a target rather than a hard constraint, since long turns or sparse observations may leave no valid split. Adjacent subtasks share the boundary frame but not the motion interval.

B.4 Instruction Generation

For each subtask, Qwen3.8-Flash-Next-FP8 (Qwen Team, 2026) receives every retained RGB observation in travel order, together with geometry computed from that subtask's camera trajectory. The labeler uses these images and measured motion directly. Images are numbered and resized to a maximum side length of 768 pixels. The geometric context contains total traveled distance, net heading change, and a per-image table of distance from the previous image and heading relative to the first image.

The system prompt asks for one English sentence using visible landmarks, turn directions, and distances, and explains how localized turns differ from continuous curves in the per-image geometry. Table 5 lists the parameters.

System prompt.

> You are a useful navigation instructor. Given a description and a sequence of images of a walking path through an outdoor environment, generate a clear, natural navigation instruction in 1 English sentence. Reference visible landmarks (trees, buildings, roads, fences, terrain), turn directions, and distances. Output ONLY the instruction text, nothing else.
>
> The per-image geometry gives you how far apart every two consecutive frames are, and the heading tells you the accumulated heading difference from the first frame. A turn concentrated between two images is a discrete turn at that point; a heading that changes steadily across all of them is a continuous curve, not a turn followed by a straight.

Each subtask annotation pairs a navigation instruction with its inclusive frame boundaries. Before export, we verify that every trajectory has the expected number of subtask annotations and that their indices are contiguous and correctly ordered.

C Model Architecture and Action Representation

C.1 Sequence Representation and Controller Targets

The backbone is Qwen3-VL-2B-Instruct (Qwen Team, 2025), augmented with tokens marking the overall instruction, subtask descriptions, action chunks, and episode termination. The input begins with the route instruction between `<task_instruction_start>` and `<task_instruction_end>`. Each observation is represented by the backbone's visual tokens. A new subtask is written between `<subtask_start>` and `<subtask_end>`; an action chunk is written between `<action_start>` and `<action_end>`.

The controller is implemented through next-token prediction at the observation's existing `<|vision_end|>` position. Its three outputs are `<action_start>` for continuing the active

Table 6: **Attention mask by token type.**

Query tokens	Tokens attended to
Route-instruction tokens	Causal prefix of the route instruction.
Visual tokens	Causal prefix of the same observation's visual-token block.
Subtask tokens	Route instruction, previously predicted subtask descriptions, boundary observation, and causal prefix of the current subtask description.
Controller position	Route instruction, predicted subtask descriptions, observations and actions of the active subtask, and the current observation.
Discrete action tokens	Active subtask description, its observations through the current step, previous action chunks within that subtask, and causal prefix of the current action chunk.
Action-expert queries	Active subtask description, its observations through the current step, previous same-subtask action chunks, the current action-start anchor, and all noisy action tokens.

subtask, <subtask_start> for beginning a subtask, and <episode_end> for terminating navigation. A subtask description is generated before the first action chunk. The final observation is followed by <episode_end> and has no outgoing action target.

C.2 Token-Level Attention Masking

Table 6 specifies the attention mask details. All backbone attention respects token order, and each token can attend to itself. Visual tokens attend only to the causal prefix within their own observation. The controller position uses a separate masking rule to combine task progress with local history; other token groups do not attend to controller keys in the configuration described here.

The action expert attends to the current <action_start> position, restricted to the prefix available at that position. Thus, it excludes the discrete coordinate tokens of the chunk currently being predicted and all future tokens.

The training and inference implementations share these masking rules. The configured training recipe additionally drops completed action chunks from subsequent attention context with probability 0.2.

C.3 Motion Targets and Four-Increment Chunks

For consecutive camera poses $(\mathbf{R}_i, \mathbf{p}_i)$ and $(\mathbf{R}_{i+1}, \mathbf{p}_{i+1})$, with rotations mapping camera coordinates to the local world frame, we compute

$$\Delta\mathbf{p}_i = \mathbf{R}_i^\top(\mathbf{p}_{i+1} - \mathbf{p}_i), \qquad \Delta\mathbf{R}_i = \mathbf{R}_i^\top\mathbf{R}_{i+1}. \tag{1}$$

Camera axes are right, down, and forward. We retain the rightward and forward components of translation and the yaw component of relative rotation. Translation is measured in meters and yaw in degrees, with positive yaw denoting a right turn. The implementation extracts yaw as the second component of the relative rotation's `xyz` Euler decomposition. We use the full camera rotations without gravity rectification and project the resulting relative translations onto the camera right–forward plane. This produces approximate supervision for planar robot motion: camera pitch and roll can affect displacement magnitudes and the extracted heading changes.

To obtain $H = 4$ increments per image interval, we first integrate the planar displacements and yaw changes into a trajectory, fit smoothing splines to planar position and accumulated yaw, and subdivide each original inter-frame interval into four equal intervals in the cumulative chord-length parameter of the original planar positions. Consecutive resampled poses are converted back into local increments. Each $\mathbf{a}_{t,h} = (\Delta x_{t,h}, \Delta z_{t,h}, \Delta\psi_{t,h})^\top$ is therefore relative to the preceding resampled pose.

The resampler uses a position smoothing scale of $0.2\,\mathrm{m}$, a yaw smoothing scale of 3°, endpoint weight 5, and additional turn weight 3 for absolute local yaw changes exceeding 8°. Spline degree is at most three and is reduced for short sequences. Rows with failed fits are excluded; fits are also

Table 7: **Layerwise action-expert configuration.** The table describes the action expert's configuration in our reported experiments; The expert uses one block per selected backbone text layer;

Setting	Value
Backbone	Qwen3-VL-2B-Instruct
Selected text layers	All 28, in order
Expert blocks	28
Expert residual width	512
Feed-forward expansion ratio	4
Attention heads and head dimensions	Inherited from the backbone configuration
Action input projection	Linear
Time conditioning	AdaLN-Zero
Noisy action tokens	4 tokens, each representing 3 coordinates
Backbone-conditioning gradients	Stopped
Inference solver	10 uniform Euler steps

rejected if they add more than 90° of total absolute rotation or increase path length beyond 1.2 times the original length plus 0.5 m.

C.4 Layerwise Action Expert

The action expert receives one noisy three-dimensional vector for each of the four increments. A linear input projection, learned positional embeddings, and flow-time conditioning provide its initial token representations. Each expert block attends jointly to the visible backbone keys and values from its corresponding layer and to the noisy action tokens. Attention geometry follows the backbone, while the expert maintains its own residual width and feed-forward blocks. Table 7 lists the configuration used by the current combined-corpus recipe.

During training, Gaussian noise is linearly interpolated with a normalized target chunk $\bar{\mathbf{A}}_t$ at a flow time sampled uniformly from $[0, 1]$. The expert is trained by mean squared error to predict the velocity $\mathbf{A}_t - \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ denotes the sampled noise. At inference, Euler integration transforms an initial Gaussian sample into a normalized action chunk, which is then denormalized to produce $\mathbf{A}_t$. Backbone keys and values are detached, so this objective updates the expert while cross-entropy trains the backbone.

C.5 Continuous Actions in the Autoregressive History

The configured inference mode samples $\mathbf{A}_t$ with the action expert, quantizes that sample using the same coordinate bins as training, and appends the resulting discrete tokens to the backbone history. It does not independently generate a second discrete action chunk for execution. The history records the sampled command rather than measured robot odometry. The policy interface assumes that the four increments are executed before the next observation is supplied; subsequent images provide visual feedback for the next controller and action decisions.

D Training and Inference Configuration

Unless otherwise stated, we train `NavWorld` using the following recipe. Experiment-specific data subsets and initialization are described in Appendix E.

D.1 Optimization Schedule

Table 8 summarizes the training schedule. Stage 1 adapts the backbone with cross-entropy supervision. Stage 2 initializes the backbone from Stage 1 and adds the flow-matching action expert. Stage 3 extends the subtask window in two successive one-epoch phases, retaining both backbone and expert weights. Each phase starts a new optimizer and learning-rate schedule.

We fine-tune the full backbone, including its vision encoder. Table 9 lists the shared settings. For the joint stages, the flow-matching coefficient is $\lambda_{\mathrm{FM}} = 1$. Cross-entropy uses all supervised positions,

Table 8: **Shared training schedule.** Window lengths count contiguous subtasks. An epoch is one pass through the training trajectories, with a window sampled at each access. Updates denote optimizer steps.

Phase	Subtasks/window	Budget	Peak learning rate
Stage 1	1–3	500 updates	10^{-4}
Stage 2	1–3	30 epochs	10^{-4}
Stage 3a	1–6	1 epoch	2×10^{-5}
Stage 3b	1–9	1 epoch	2×10^{-5}

Table 9: **Shared optimization settings.** Effective batch size assumes the reported eight-GPU launch. Adam parameters are inherited from the pinned training-library defaults unless overridden at launch.

Setting	Value
Optimizer	Fused AdamW
Adam (β_1, β_2); ϵ; weight decay	$(0.9, 0.999)$; 10^{-8}; 0
Learning-rate schedule; warmup	Cosine; 10 updates per phase
Gradient norm limit	1.0
Per-device batch; gradient accumulation	1 window; 8 steps
Effective batch on eight GPUs	64 windows per full update
Precision; gradient checkpointing	Bfloat16; non-reentrant
Image-processor pixel budget	307,200 pixels per image
Sample packing; auxiliary VQA training	Disabled; disabled

whereas the flow objective samples at most four action chunks per training window. Validation uses at most four deterministically spaced chunks. Action-history dropout and gradient isolation follow in Appendices C.2 and C.4.

D.2 Window Sampling and Augmentation

We sample a requested window length uniformly from the phase's integer range and cap it by the trajectory's available subtask count. For that length, a candidate start receives weight 10 if its window contains a subtask whose absolute summed camera-yaw change exceeds $15°$; other starts receive weight 1. Thus, turn emphasis changes where a window starts, rather than its requested length. Training resamples windows at each access. Validation uses a trajectory-identifier seed to select a fixed window for each phase's sampling configuration.

Each sampled window is paired with a navigation instruction describing only its selected subtasks. We treat the window as a complete training episode: its final observation is followed by ¡episode end¿ and has no outgoing action target, even when the source trajectory continues beyond the window. Termination supervision therefore corresponds to completion of the instruction supplied for that window.

Training images receive independent color jitter with brightness, contrast, saturation, and hue settings of 0.25, 0.25, 0.2, and 0.05, respectively. With probability 0.5, we horizontally reflect every image in the window, negate rightward displacement and yaw in the motion targets, and exchange left/right references in its text. These augmentations are disabled for validation.

D.3 Decoding and Run Provenance

Inference processes one navigation session at a time. The controller selects the highest-scoring of its three permitted decisions, and subtask text is decoded greedily with a limit of 64 generated tokens, stopping at the subtask-end or end-of-sequence token. A termination decision produces no action. Continuous actions remain stochastic because the expert starts from Gaussian noise; its integration and history construction are specified in Appendices C.4 and C.5.

Table 10: **Continuous-control evaluation settings.** All methods share the simulated-time budget and speed limits. Inference wall time is excluded from the motion budget.

Setting	Value
RGB/depth rendering resolution	640×480
Horizontal field of view; camera height	79°; 1.5 m
Near/far clipping distances	0.01 m / 10,000 m
Agent height; radius	1.5 m; 0.1 m
Motion constraints	Scene navigation mesh; sliding enabled
Integration interval	0.05 s (20 Hz)
Maximum forward speed; yaw speed	1 m/s; 60°/s
Discrete baseline command duration	0.25 s
Baseline forward/turn/tilt increments	0.25 m / 15° / 15°
Episode budget	125 s (2,500 control ticks)
Base evaluation seed	0

The configured curriculum starts from the Stage 2 epoch-30 checkpoint and then the final export of Stage 3a. The training driver does not automatically select the checkpoint with the best validation loss.

E Evaluation Protocols and Baseline Implementation

We evaluate using the continuous-control protocol with a 125 s simulated-time budget per episode.

E.1 Episodes and Instruction Variants

The corrected Habitat-GS split contains 50 episodes in each of `scene56`–`scene62` and `scene65`, giving 400 outdoor episodes. The evaluator selects episodes by numeric identifier within each scene. Scenes `scene63` and `scene64` are indoor captures and are excluded. The quantitative comparison uses all selected episodes; clear instructions, identifiable goals, and good rendering are selection criteria only for qualitative examples.

The corrected instruction condition uses `val_v2`. The repair addresses a reversed heading-sign convention in the original instruction generator: positive heading changes correspond to left turns. It changes turn directives while preserving start states, goals, reference paths, and landmark-side descriptions.

For the `NavWorld` instruction condition, we render the same reference routes, apply the geometric segmentation and labeling procedure in Appendix B, and concatenate subtask instructions in route order. The labeler receives images and geometric context, without the original benchmark instruction. Split construction replaces only instruction fields and requires nonempty text and contiguous subtask indices. Reference paths and goal coordinates are used for annotation and scoring, and are not supplied to the policies.

E.2 Rendering and Motion Execution

Table 10 summarizes the shared continuous execution settings. The rendering far plane is increased from 1,000 to 10,000 m to retain distant outdoor scene content. The repaired renderer defines protocol version 2; results obtained with the earlier renderer require separate provenance and cannot be treated as a matched rendering comparison.

Discrete baseline motion and camera commands execute over five control ticks; STOP and wait consume one tick. InternVLA's two downward and two upward camera tilts therefore consume 1 s. Collision-blocked motion also consumes simulated time. Navigation-mesh filtering constrains translation, with sliding enabled. Budget exhaustion can interrupt a command and prevents further inference or a subsequent STOP. The 125 s budget therefore limits simulated execution, not model calls or wall-clock runtime.

E.3 Metrics and Exceptional Outcomes

Navigation error is the final shortest-path distance on the scene navigation mesh. Success requires an executed STOP with final geodesic distance strictly below the specified radius. Oracle success tests whether the initial pose or any subsequent executed pose enters that radius, independently of STOP. Metrics are updated after every control tick, including intermediate motion within action chunks. SPL uses the initial geodesic distance and the sum of Euclidean distances between successive executed positions, with a 3 m success radius.

The shared scorer evaluates one success radius per run, defaulting to 3 m. Its saved final distance, minimum distance, and executed-STOP flag also permit computing SR and OSR at 10 m from the same rollouts.

Budget exhaustion and controller exhaustion do not count as STOP. Completed failures remain in the aggregate denominator.

E.4 Policy Inputs and Baseline Adapters

All policies receive the route instruction and egocentric imagery, but their local execution interfaces differ. StreamVLN and NaVILA use RGB observations. InternVLA-N1-DualVLN additionally supplies rendered depth to its local trajectory predictor, resized to 224×224 and clipped to $[0, 5]$ m. The `NavWorld` learned policy receives RGB; its waypoint follower uses simulator position and heading to execute model-predicted motion, without access to the reference route or evaluation goal.

StreamVLN retains its streaming cache, reset every 32 actions, with a default history-sampling parameter of 8 frames. NaVILA samples historical RGB frames and the current view to match its checkpoint's video length, padding short histories with blank frames. InternVLA conditions high-level predictions on up to 8 historical RGB frames and the current view. Its local completion signal requests replanning; only the high-level STOP ends navigation.

For `NavWorld`, the continuous follower tracks the predicted waypoint polyline with a 0.35 m lookahead and a 0.05 m endpoint tolerance, then aligns to the chunk's final heading within 2°. It exits position tracking after 2 s of stalled forward motion; this does not itself terminate navigation. A valid chunk requiring no motion incurs one wait tick before the next policy decision. The runner derives a deterministic seed from the scene identifier, episode identifier, and base seed, and seeds PyTorch before each rollout, including flow-matching noise sampling.

Invalid-output handling is adapter-specific. StreamVLN and InternVLA terminate unsuccessfully when no supported action can be parsed; NaVILA's current parser falls back to one forward action for unmatched text. Thus, identical invalid-response handling cannot be assumed.