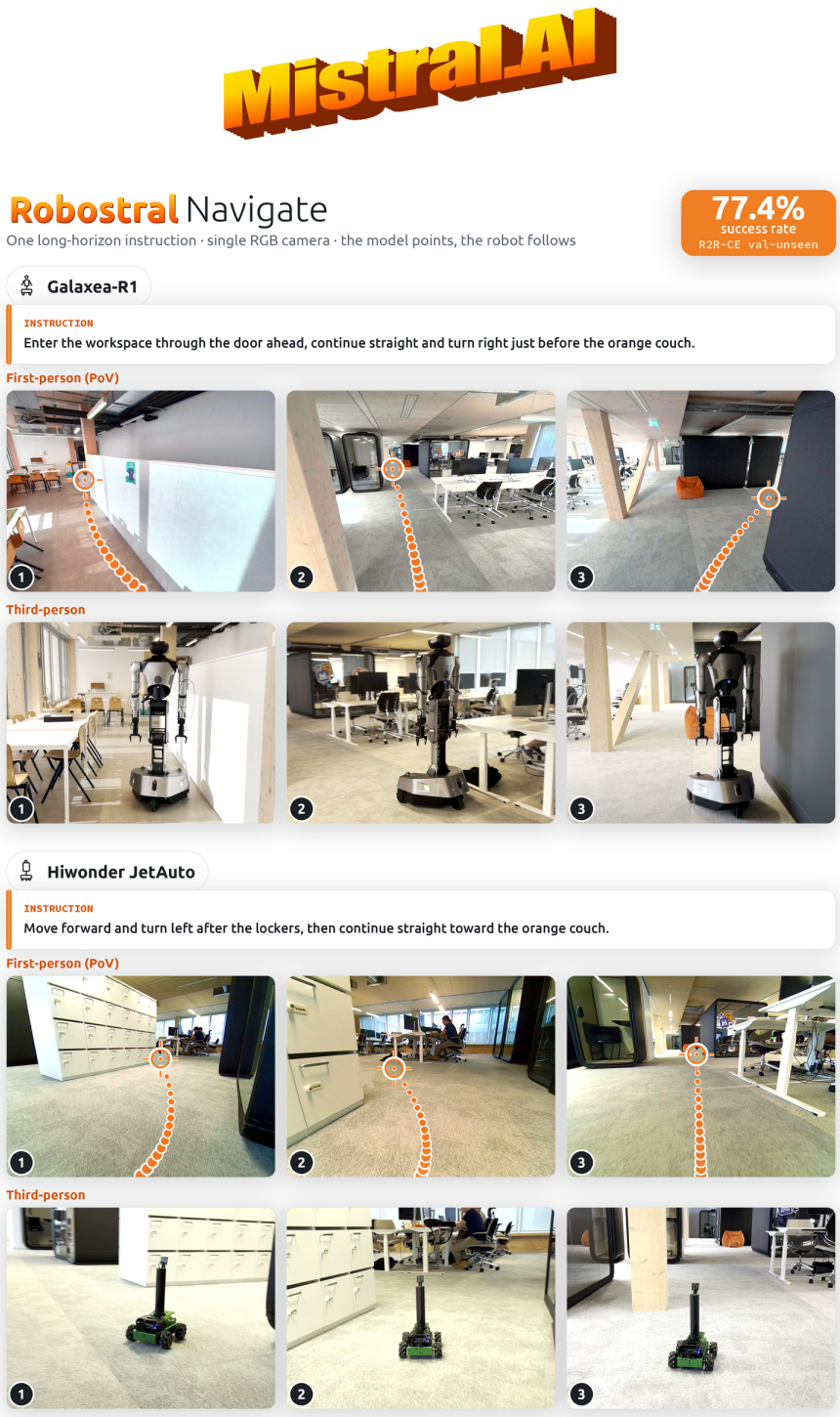
Robostral Navigate



Robostral Navigate — an 8B VLM that navigates using only a single RGB camera, pointing to target pixels. One instruction drives the full long-horizon episode; the first-person path shows the model steering around obstacles.

Abstract

Deploying navigation systems at scale requires a recipe that minimizes sensor assumptions, generalizes across robot embodiments, and trains efficiently. Yet, today's best systems depend on depth sensors, multi-camera rigs, or pre-built maps, limiting the hardware they support and increasing deployment cost. We introduce Robostral Navigate, an 8B vision-language model built around this scalability objective. The model consumes only a stream of monocular RGB images—the most ubiquitous sensor across robotic platforms—and predicts waypoints by pointing to the next target location in the current camera view. Operating purely in image space, rather than robot-specific coordinates, makes the policy naturally robust to changes in camera intrinsics and scene scale, enabling deployment across wheeled, legged, and aerial robots without recalibration. We generate 2.4 million trajectories across 350k simulated scenes to reduce the reliance on real-world data collection and scale easily. We further introduce a prefix-caching training recipe that packs entire episodes into single training sequences, reducing training tokens by 22× and cutting training time from months to days. A tree-based attention mask prevents conditioning on previous ground-truth actions, encouraging visually grounded action prediction, and reinforcement learning is used to further improve exploration and recovery capabilities. On the Room-to-Room and Room-Across-Room in Continuous Environments (R2R-CE and RxR-CE) benchmarks, Robostral Navigate sets a new state of the art. On R2R-CE, it achieves a 77.4% success rate, surpassing the best monocular method by 10.5 points and the strongest depth- or multi-camera system by 5.3 points despite using only a single RGB camera. On RxR-CE, it reaches 75.1% success rate, outperforming all monocular baselines.

Webpage: https://mistral.ai/news/robostral-navigate

1 Introduction

Embodied navigation—the ability of a robot to move through an environment following natural language instructions—is a foundational capability for general-purpose robotics. Recent vision-language models have made impressive strides on this task [Zhang et al., 2024b,a, Cheng et al., 2025, Wei et al., 2025, Wang et al., 2026], but the best-performing systems depend on depth sensors, LiDAR, multi-camera rigs, or pre-built environment maps [Zhang et al., 2025a,b, Xue et al., 2026, Chu et al., 2026, Zhang et al., 2026]. Each additional sensing requirement narrows the set of compatible robots, increases per-unit cost, and demands environment-specific calibration—all of which limit how broadly a navigation policy can be deployed. As the field moves toward general-purpose robots that must operate across diverse hardware and novel environments [Zitkovich et al., 2023, Shah et al., 2023], the challenge is not just building a more accurate navigator, but finding a *scalable recipe* for navigation: one that minimizes sensor assumptions, generalizes across embodiments, and can be trained efficiently from simulation alone.

This scalability lens motivates every design decision in Robostral Navigate, an 8B vision-language model that achieves state-of-the-art performance on the Room-to-Room (R2R-CE) [Krantz et al., 2020] and Room-Across-Room (RxR-CE) [Ku et al., 2020] in Continuous Environments benchmarks using only a single RGB camera. A monocular RGB stream is the most universally available sensor across robotic platforms—from warehouse wheels to delivery drones—making it the natural input for a policy intended to scale. Rather than predicting metric displacements tied to a specific robot's geometry [Krantz et al., 2021, An et al., 2024], Robostral Navigate produces waypoints via *pointing*: it infers the image coordinates of the next target location in the robot's current camera view, together with the desired orientation upon arrival. Because waypoints are expressed in image space rather than metric coordinates, the policy avoids dependence on robot-specific geometry and transfers more naturally across camera configurations and embodiments.

The training recipe is designed with the same scalability objective. We train entirely in simulation [Savva et al., 2019, Ramakrishnan et al., 2022] on approximately 2.4 million trajectories across 350k scenes, eliminating the need for costly real-world data collection. A prefix-caching technique based on attention masking compresses entire episodes into single training sequences, reducing training tokens by 22× while preserving all learning signals and transforming month-long training

runs into days. After supervised training, we apply online reinforcement learning via CISPO [Chen et al., 2025], training on a curated hard subset of trajectories to improve exploration and recovery from failures, yielding an additional 4% improvement in success rate.

On R2R-CE validation unseen, Robostral Navigate achieves a 77.4% success rate, surpassing the best single-camera method (Qwen-RobotNav-4B, 66.9%) by 10.5 points and the best system using depth or multiple cameras (Qwen-RobotNav-8B, 72.1%) by 5.3 points—despite relying on neither. Similarly, on the english-only RxR-CE validation unseen benchmark, it establishes a new state of the art with a 75.1% success rate and a 68.7% SPL, outperforming all single-camera baselines and demonstrating superior path efficiency over depth-assisted models, despite using only a single monocular RGB camera. This result demonstrates that a scalable, minimal-sensor recipe, when combined with efficient training and reinforcement learning, can outperform approaches that leverage privileged sensing.

2 Modeling

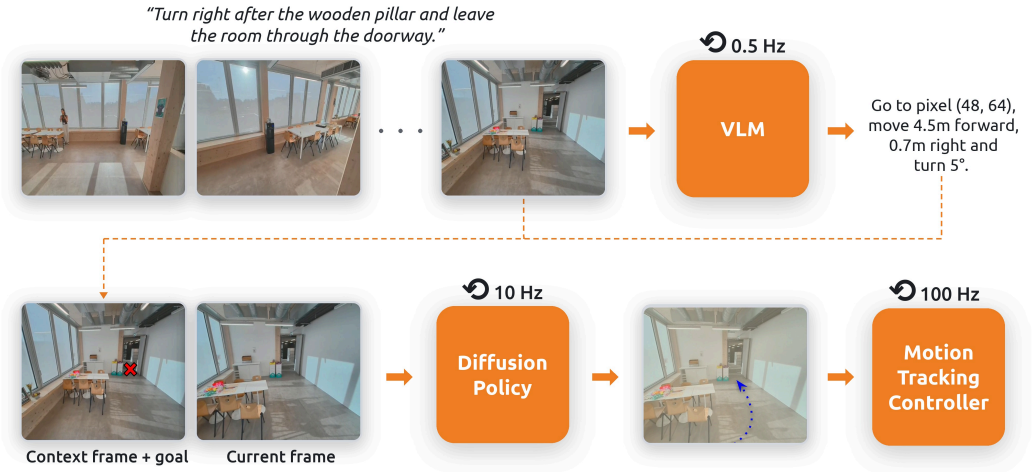


Figure 1: System Architecture. Given a natural language instruction and a stream of RGB observations, the VLM predicts the next waypoint at 0.5 hertz via pointing, metric displacement and orientation upon arrival. When the target lies outside the field of view, pointing is omitted. A diffusion policy takes the waypoint, the context frame (frame at which the waypoint has been inferred) and the current frame to produce a dense trajectory at 10 hertz. Finally, a motion-tracking controller turns this trajectory into high-frequency actuator torques.

In order to develop a scalable navigation system, we decompose the navigation task into two distinct challenges. First, the system must understand the navigation instruction, reason about the scene, and plan a coarse path for completing the task. Second, the coarse path must be translated into motor commands that efficiently move the robot towards the goal while avoiding obstacles. As illustrated in Figure 1, we built different models for each of these challenges. For the first, we trained a large 8B parameter vision-language model (VLM) called Robostral Navigate, which is needed for complex reasoning. For the second, we use a small 121M parameter diffusion policy, which has a sufficient capacity for the geometric understanding required for navigation. We combine these models in series and operate them at different inference frequencies.

2.1 Architecture Overview

Robostral Navigate takes a language instruction and history of monocular RGB frames as inputs, and outputs navigation waypoints in the form of pixel coordinates or displacement commands. The model is initialized from a dense 8B-parameter VLM designed for spatial grounding tasks such as pointing, counting, and object localization. Navigation is a natural extension of these grounding capabilities: once the model understands where things are in an image, it learns how to move toward them.

The frames are encoded by a vision encoder into a sequence of visual tokens, which are appended to the tokenized instruction to form the input sequence to the VLM. The history enables the model to reason about previously visited locations and track progress.

As illustrated in Figure 1, the inferred waypoints are converted to robot actions via a diffusion policy and a motion-tracking controller. Specifically, the diffusion policy outputs a low-level trajectory at 10 Hz in the form of an action chunk specifying the relative 2D displacements and 1D rotations for the next second. Then, an embodiement-specific motion-tracking controller converts the low-level trajectory into 100 Hz motor commands.

2.2 Navigation via Pointing with Displacement Fallback

Given a task instruction and a history of RGB frames, Robostral Navigate predicts the next destination using two modes: a preferred visual *pointing* mode and a *metric displacement* fallback.

In pointing mode, the model infers the image coordinates (u, v) of the furthest visible waypoint along the groundtruth trajectory, alongside the desired change of heading $\Delta\theta$ upon arrival, defined as the yaw rotation relative to the current frame. Pointing is highly favored because it operates in visual space, making the policy naturally robust to changes in camera intrinsics and world scale, facilitating deployment across diverse robotics platforms.

However, pointing alone cannot handle cases where the destination lies outside the current field of view (e.g. when the robot should turn around). To ensure the robot can always make progress, the system relies on a metric fallback mode consisting of local translational and rotational displacements $(\Delta x, \Delta y, \Delta\theta)$ relative to the current frame .

In practice, we formulate training as a joint prediction task: when the destination is visible, the model predicts all five quantities

$$a_{\text{vis}} = (u, v, \Delta x, \Delta y, \Delta\theta). \tag{1}$$

Because only approximately 10% of our training data features invisible destinations, predicting the metric displacements alongside pointing coordinates acts as an effective co-training task. When no future position is visible from the current view, the model simply omits the image coordinates and only predicts metric displacements

$$a_{\text{invis}} = (\Delta x, \Delta y, \Delta\theta). \tag{2}$$

The model also infers STOP when the task is completed.

2.3 Pointing to Actions

The Robotstral Navigate waypoints (specified via pointing or displacement) are grounded in the robot's camera frame. To convert these waypoints into robot actions, we use a lightweight diffusion transformer model [Peebles and Xie, 2023] with approximately 121M parameters. The inputs to the model are the VLM prediction $a \in \{a_{\text{vis}}, a_{\text{invis}}\}$, the robot's height and radius, the latest RGB frame used by the VLM $o_{t_{\text{vlm}}}$, and the current RGB frame o_t. Note that due to the VLM inference time, there is a latency between the context frame $o_{t_{\text{vlm}}}$ and current frame o_t. The output from the policy is an action chunk consisting of a sequence of 30 delta coordinates $(dx, dy, d\theta)$ relative to the robot base (same as the displacement waypoints) for the next second. Finally, a motion controller translates this action chunk into high-frequency motor commands.

2.4 Cross-Robot Generalization

Formulating navigation as a pointing task naturally enables cross-robot generalization. To further enhance cross-platform transfer, we randomize robot configurations when generating training data for both the VLM and diffusion policies. Specifically, we randomize the robot height between 0.4 - 1.8m, robot radius between 0.15 - 0.45m, camera placement height between 70 - 100% of the robot height, and camera pitch between 0 - 25°. These design choices reduce dependence on any particular camera setup, physical scale, or robot morphology, allowing the system to transfer across embodiments without retraining the learned policies. To demonstrate cross-robot generalization, we deploy the navigation system on two substantially different mobile robot platforms: the Galaxea R1 [Galaxea AI, 2026] and the Hiwonder JetAuto [Hiwonder, 2026]. Notably, the same vision-language model and diffusion policy weights are used on both of the platforms illustrated in Figure 2.

Figure 2: Cross-robot deployment of Robostral Navigate on the Galaxea R1 (left) and Hiwonder JetAuto (right). The two platforms differ substantially in robot morphology, camera height, camera configuration, and base geometry. Both deployments use the same Robostral Navigate vision-language model and diffusion policy weights, with only the low-level platform controller differing between robots.

3 Training

3.1 Data Generation

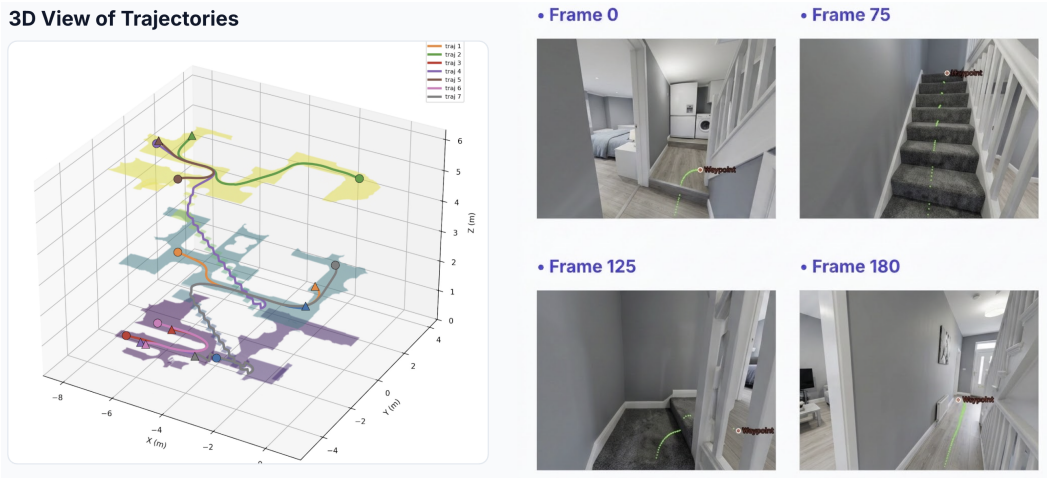


Figure 3: Example of training data. Left: 3D navigable area and sampled trajectories for a given scene. Right: Four frames from a cross-floor trajectory with future positions in-painted in green and waypoint in red.

To enable rapid iteration and mitigate the reliance on costly real-world data collection, we built an efficient simulation-based data generation pipeline. The pipeline generates expert navigation trajectories across a diverse set of indoor and outdoor environments, producing approximately 2.4 million trajectories across 350k scenes. Each sample consists of a natural language instruction paired with a sequence of RGB observations and corresponding navigation actions.

As illustrated in Figure 3, we use farthest point sampling to ensure diverse start and goal positions, resulting in trajectories of various length, sometimes spanning multiple floors. The diversity of scenes (offices, residential buildings, commercial spaces, and outdoors) is also a key factor for good

generalization. We ensure variation in layout complexity, object density, lighting conditions, and architectural style.

3.2 Supervised Training with Prefix Trees

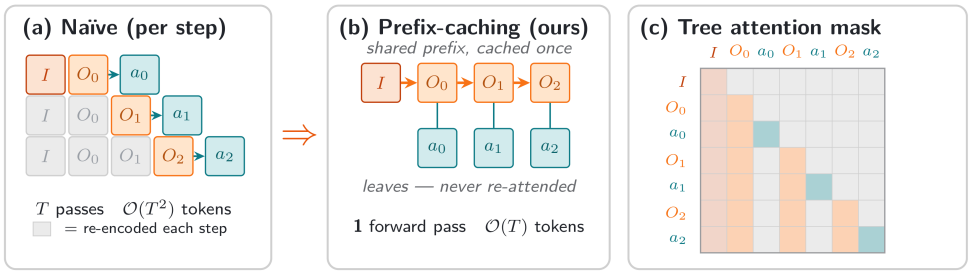


Figure 4: Comparison of (a) naive representation with our (b) prefix-caching. (c) demonstrates the tree attention mask with prefix-tree representation; I is the instruction, O_i is the i^{th} observation/image, a_i is the action at the i^{th} time step.

A key ingredient of Robostral Navigate is an efficient training algorithm based on prefix sharing, shown in Figure 4. Our method is inspired by the attention masking used in Molmo2 [Clark et al., 2026], which amortizes the encoding of a context (text, image or video) shared across multiple independent questions and answers. We push this concept further by exploiting the nested temporal structure of sequential decision-making: each iteration extends the history by concatenating the current observation, leading to significant redundancy between histories of the same episode.

In standard training, each time step is treated as an independent sample. To predict the action at time step t, the model processes the instruction plus all the frames accumulated up to t. A trajectory containing T steps therefore produces T separate training sequences, each containing an increasingly longer context. The total number of tokens processed scales as $\mathcal{O}(T^2)$.

Efficiency. Instead, we encode entire episodes as single training sequences consisting of an instruction followed by observations interleaved with actions

$$I|O_0|a_0|\dots|O_n|a_n.$$

With this formulation, each new element in the history is encoded only once, allowing the losses for all time steps to be computed in a single forward pass and reducing the token complexity from $\mathcal{O}(T^2)$ to $\mathcal{O}(T)$.

However, with such sequences, vanilla causal attention would allow the model to attend the groundtruth actions associated with previous time steps during training. This is an issue because we do not have access to groundtruth actions during deployment and the model would heavily rely on these actions as previous groundtruth actions are often very informative of future actions (notably because of the farthest-visible waypoint paradigm).

Attention masking. To mitigate this issue, we introduce an attention masking strategy based on *prefix trees*. A prefix tree is a compact representation of a set of sequences, where shared prefixes are merged together in a trunk, while each sequence's specific suffix is encoded in its own branch. For instance, if we have "AB" and "AC" then their flattened prefix-tree representation could be "ABC" where "A" would be the shared trunk, and "B" and "C" different branches. The tree structure can be represented by an attention mask following the rule that $token_i$ can attend to $token_j$ if (1) $j < i$ and (2) $token_j$ is in the trunk, or $token_i$ and $token_j$ are in the same branch. A prefix tree sample gives provably identical training signal than the per-time-step samples used to build it, while saving compute by avoiding redundant prefix encoding during training.

This approach is particularly vital for the RxR-CE benchmark, which features significantly longer instructions and trajectory lengths compared to R2R-CE. Compared to per-time-step samples, our approach reduces the number of training tokens by 22× without discarding any action-prediction targets. In practice, this reduces training time from months to days.

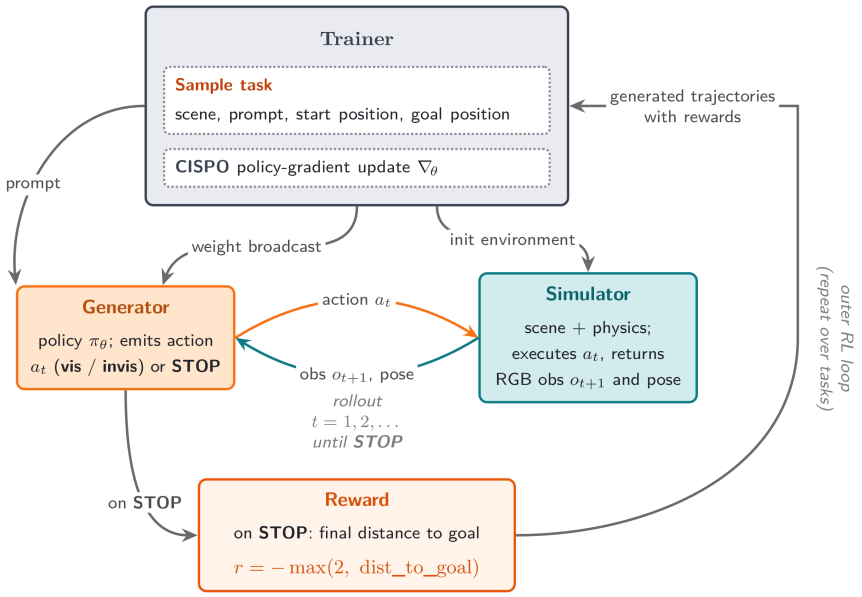


Figure 5: Overview of our online reinforcement learning pipeline. The trainer orchestrates the prompt sampling, physical simulator, and action generator (vLLM) in an asynchronous loop, updating the policy via CISPO using the $-\max(2, \text{dist\_to\_goal})$ reward signal upon episode termination.

3.3 Online Reinforcement Learning via CISPO

Although supervised fine-tuning (SFT) effectively instills a foundational visual-linguistic understanding and basic navigation skills, it fundamentally relies on behavioral cloning. This approach suffers from exposure bias [Ranzato et al., 2016] and covariate shift [Ross et al., 2011]: at inference time, compounding errors may lead the agent to out-of-distribution states. For instance, shortest-path trajectories do not demonstrate information-seeking and recovery behaviors, leading to poor performance when these abilities are needed.

To bridge this gap, we apply online reinforcement learning on top of the SFT model. By actively interacting with a simulator, the agent can experience the consequences of its own actions and learn in the states it actually visits. In this work, we use CISPO [Chen et al., 2025], leveraging group-relative advantage estimation to provide a stable, self-calibrated learning signal.

The reward is defined as $-\max(2, \text{dist\_to\_goal})$, where dist_to_goal is the geodesic distance between the last position and the goal in meters, derived with a path finding algorithm. By clipping the distance penalty at 2 meters, the reward flatlines once the agent is near the goal location. This prevents the agent from needlessly maneuvering near the target to accumulate rewards that are unrelated to task success, and incentivizes the agent to emit the STOP action to end the episode.

Scaling this online RL pipeline for a large VLM is a formidable systems engineering challenge due to the intense hardware requirements. Unlike text-only RL setups, our architecture requires orchestrating three massive GPU-bound workloads concurrently: 1) the physics and rendering engine simulating hundreds of parallel environments, 2) the distributed inference engine (vLLM) generating autoregressive actions, and 3) the training nodes executing distributed weight updates. Because the simulator itself consumes substantial GPU memory and compute to render high-fidelity observations, resource allocation had to be meticulously balanced across the cluster's nodes. This careful orchestration ensures that the high-throughput vLLM generators and the training ranks do not stall while waiting for simulator rendering, maintaining optimal GPU utilization end-to-end.

Training tasks and visual curriculum. We curate a *hard subset* of 35k tasks by rolling out the SFT policy and retaining only the trajectories on which it fails to reliably reach the goal. This focuses compute on the cases where the policy most needs improvement – complex layouts, ambiguous instructions, and long-horizon tasks, effectively filtering out episodes the model can already solve.

Table 1: Comparison on R2R-CE and RxR-CE validation unseen. Robostral Navigate achieves state-of-the-art results using only a single RGB camera, outperforming methods that rely on depth sensors or multiple cameras on both benchmarks. ✓ indicates single-camera RGB-only; ✗ indicates use of depth, LiDAR, or multiple cameras.

Model	Single Camera	R2R-CE Val Unseen				RxR-CE Val Unseen		
		NE ↓	OS ↑	SR ↑	SPL ↑	NE ↓	SR ↑	SPL ↑
Single RGB camera								
NaVid [Zhang et al., 2024b]	✓	5.47	0.490	0.370	0.350	5.72	0.457	0.382
Uni-NaVid [Zhang et al., 2024a]	✓	5.58	0.535	0.470	0.427	6.24	0.487	0.409
NaVILA [Cheng et al., 2025]	✓	5.22	0.625	0.540	0.490	6.77	0.493	0.440
InternVLA-N1 [Zhang et al., 2025a]	✓	4.89	0.606	0.554	0.521	6.41	0.495	0.418
Qwen-VLA-Base [Wang et al., 2026]	✓	5.20	0.617	0.538	0.494	6.40	0.451	0.458
StreamVLN [Wei et al., 2025]	✓	4.98	0.642	0.569	0.519	6.22	0.529	0.460
Qwen-VLA-Instruct [Wang et al., 2026]	✓	5.10	0.690	0.575	0.512	5.80	0.596	0.478
Qwen-RobotNav-4B [Zhang et al., 2026]	✓	4.22	0.736	0.669	0.605	4.15	0.713	0.615
Qwen-RobotNav-8B [Zhang et al., 2026]	✓	4.36	0.727	0.657	0.596	4.16	0.734	0.635
Depth / multi-camera								
InternVLA-N1 (S1+S2) [Zhang et al., 2025a]	✗	4.83	0.633	0.582	0.540	5.91	0.535	0.461
NavFoM [Zhang et al., 2025b]	✗	4.61	0.721	0.617	0.553	4.74	0.644	0.562
ABot-N0 [Chu et al., 2026]	✗	3.78	0.708	0.664	0.639	3.83	0.693	0.600
OmniNav [Xue et al., 2026]	✗	3.74	0.746	0.695	0.661	3.77	0.736	0.620
Qwen-RobotNav-4B [Zhang et al., 2026]	✗	3.80	0.772	0.695	0.636	3.80	0.752	0.650
Qwen-RobotNav-8B [Zhang et al., 2026]	✗	3.73	0.727	0.721	0.666	3.58	**0.765**	0.657
Robostral Navigate (ours)	✓	**3.20**	**0.813**	**0.774**	**0.742**	**3.47**	0.751	**0.687**

During data collection, we pack tasks by scene, resulting in consecutive rollouts that are from the same scene. This induces an implicit visual curriculum: each batch contains multiple episodes sharing the same visual context. Empirically, this scene-contiguous ordering outperforms random shuffling.

4 Results

We evaluate Robostral Navigate on the R2R-CE and RxR-CE benchmarks, which are the standard instruction-following benchmarks for navigation. Both require an agent to follow a previously unseen natural language instructions to navigate from a start to a goal position in previously unseen photorealistic environments. For these evaluations, we use a pathfinder from Habitat Savva et al. [2019], Szot et al. [2021] for navigation between the waypoints predicted by Robostral Navigate.

Metrics: We report standard navigation metrics that capture both task completion and trajectory efficiency.

- **Navigation Error (NE ↓)** measures the geodesic distance between the agent's final position and the goal.
- **Success Rate (SR ↑)** is the proportion of episodes where the agent stopped within 3 m of the goal.
- **Oracle Success (OS ↑)** is the proportion of episodes where the agent came within 3 m of the goal at any point along its trajectory.
- **Success weighted by Path Length (SPL ↑)** evaluates navigation efficiency by weighting SR using the ratio of the shortest-path length to the actual path length, penalizing unnecessary detours.

4.1 Main Results

Table 1 presents a comparison of Robostral Navigate against recent methods on R2R-CE and RxR-CE validation unseen. We categorize methods by whether they operate from a single RGB camera or require additional sensing modalities (depth, LiDAR, or multiple cameras).

Robostral Navigate achieves a success rate of 77.4% and SPL of 74.2% on R2R-CE validation unseen, establishing a new state of the art across all metrics. It surpasses the best prior single-camera method (Qwen-RobotNav-4B at 66.9% SR) by 10.5 points, and the best method using depth or multiple cameras (Qwen-RobotNav-8B at 72.1% SR) by 5.3 points—despite using only a single RGB camera with no auxiliary sensors.

In the R2R-CE validation seen split, Robostral Navigate achieves a 79.4% success rate, demonstrating strong performance in both seen and unseen environments. This performance is directly unlocked by our online RL phase, which profoundly shifts the policy from brittle to robust navigation. Compared to our R2R-CE SFT baseline (75.96% seen, 73.40% unseen), our post-RL checkpoint achieves a success rate of 79.43% on seen environments (+3.47%) and 77.43% on unseen environments (+4.03%). Peak validation performance during the run reached 80.46% on seen and 77.43% on unseen environments, highlighting the robust generalization capabilities unlocked by our online RL pipeline. The high SPL (74.2%) shows that these successes are achieved via efficient paths, without unnecessary exploration.

On the more challenging RxR-CE benchmark, Robostral Navigate continues to demonstrate superior performance. It achieves a success rate of 75.1% and an SPL of 68.7%, while maintaining a low navigation error of only 3.47 meters. Compared to the prior state-of-the-art single-camera method, Qwen-RobotNav-8B (73.4% SR, 63.5% SPL), Robostral Navigate improves the success rate by 1.7 percentage points and significantly boosts path efficiency (SPL) by 5.2 points. Remarkably, our single-camera, RGB-only model even outperforms the best depth- and multi-camera-assisted baseline (Qwen-RobotNav-8B with depth) on both SPL (68.7% vs. 65.7%) and navigation error (3.47m vs. 3.58m), while remaining competitive on success rate (75.1% vs. 76.5%).

Similar to the trends on R2R-CE, on the RxR-CE benchmark, the online RL phase brings a substantial boost over the SFT baseline. Robostral Navigate improves from 71.2% to a state-of-the-art success rate of 75.1% (+3.9%), demonstrating that reinforcement learning generalizes well to long-horizon instruction-following tasks. These results underscore Robostral Navigate's exceptional ability to handle long, complex instructions and plan highly optimal trajectories without relying on auxiliary depth or panoramic sensors.

5 Conclusion

We introduced Robostral Navigate, an 8B vision-language model for embodied navigation that achieves state-of-the-art performance on the R2R-CE and RxR-CE benchmarks. Robostral Navigate is designed with scalability in mind to minimize sensor assumptions, generalize across embodiments, and train efficiently from simulation alone. The policy predicts navigation actions via pointing and only needs a single RGB camera to be deployed. The policy is trained with a prefix-caching technique that cuts training tokens by 22$\times$ while preserving all learning signals, resulting in a highly efficient supervised finetuning training recipe. Finally, the policy is finetuned using reinforcement learning with CISPO, resulting in an agent that is very efficient at solving navigation tasks (as measured on the R2R-CE and RxR-CE benchmarks).

Navigation is a foundational capability for general-purpose robotics. Accordingly, we view Robostral Navigate as a first step towards building a general-purpose embodied agent. Specifically, this work demonstrates that combining large-scale simulation, efficient training, and reinforcement learning can produce a state-of-the-art embodied navigation model that has minimal sensing requirements.

Core contributors

Arjun Majumdar, Avinash Sooriyarachchi, Benjamin Tibi, Chris Bamford, Elliot Chane-Sane, Guillaume Lample, Khyathi Raghavi Chandu, Ludovic Ho Fuh, Mathieu Poirée, Olivier Duchenne, Rosalie Millner, Srijan Mishra, Théo Cachet, Thomas Chabal.

References

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

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. MiniMax-M1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

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

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.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinuo Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Winson Han, Ali Farhadi, and Ranjay Krishna. Molmo2: Open weights and data for vision-language models with video understanding and grounding. In *Conference on Computer Vision and Pattern Recognition*, 2026.

Galaxea AI. Galaxea r1. https://userguide-galaxea.github.io/Product_User_Guide/Introducing_Galaxea_Robot/product_info/R1/, 2026. Accessed: 2026-07-15.

Hiwonder. Jetauto ros robot. https://developer.nvidia.com/embedded/community/jetson-projects/jetauto, 2026. Accessed: 2026-07-15.

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*, 2020. URL https://arxiv.org/abs/2004.02857.

Jacob Krantz, Aaron Gokaslan, Dhruv Batra, Stefan Lee, and Oleksandr Maksymets. Waypoint models for instruction-guided navigation in continuous environments. In *International Conference on Computer Vision*, 2021. URL https://arxiv.org/abs/2110.02207.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Conference on Empirical Methods in Natural Language Processing*, pages 4392–4412, 2020.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *International Conference on Computer Vision*, pages 4195–4205, 2023.

Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3D dataset (HM3D): 1000 large-scale 3D environments for embodied AI. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

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

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *International Conference on Computer Vision*, 2019. URL https://arxiv.org/abs/1904.01201.

Dhruv Shah, Blazej Osinski, Brian Ichter, and Sergey Levine. LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2207.04429.

Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Qiuyue Wang et al. Qwen-VLA: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026.

Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, Xihui Liu, and Jiangmiao Pang. StreamVLN: Streaming vision-and-language navigation via slowfast context modeling. *arXiv preprint arXiv:2507.05240*, 2025.

Xinda Xue, Junjun Hu, Minghua Luo, Shichao Xie, Jintao Chen, Zixun Xie, Kuichen Quan, Wei Guo, Mu Xu, and Zedong Chu. OmniNav: A unified framework for prospective exploration and visual-language navigation. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25687.

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*, 2024a.

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. In *Robotics: Science and Systems*, 2024b. URL https://arxiv.org/abs/2402.15852.

Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, Yuxin Fan, Wenjun Li, Zhibo Chen, Fei Gao, Qi Wu, Zhizheng Zhang, and He Wang. Ground Slow, Move Fast: A dual-system foundation model for generalizable vision-and-language navigation. *arXiv preprint arXiv:2512.08186*, 2025a.

Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, Yuxin Fan, Wenjun Li, Zhibo Chen, Fei Gao, Qi Wu, Zhizheng Zhang, and He Wang. Embodied navigation foundation model. *arXiv preprint arXiv:2509.12129*, 2025b.

Jiazhao Zhang, Gengze Zhou, Hale Yin, Yiyang Huang, Zixing Lei, Qihang Peng, Haoqi Yuan, Jie Zhang, Xudong Guo, Xiaoyue Chen, et al. Qwen-RobotNav Technical Report: A scalable navigation model designed for an agentic navigation system. *arXiv preprint arXiv:2606.18112*, 2026.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*. PMLR, 2023.