SCALING OBJECT NAVIGATION WITH CROSS-DOMAIN REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Object-goal navigation (ObjectNav) requires an agent to locate a specified object in an unfamiliar environment. Policies trained on limited 3D scenes generalize poorly to unseen layouts, yet collecting and annotating large-scale 3D training data remains expensive. To address this limitation, we propose a cross-domain RL framework that uses 2D semantic-map environments as an additional source of interactive training experience for scaling ObjectNav. First, TensorEnv turns static semantic maps into interactive 2D environments. Second, ParHab accelerates Habitat sampling and map construction. Third, MapAct takes partial semantic maps as input and predicts navigation actions. Fourth, NavPRL bridges the domain gap by transferring knowledge from a teacher model trained in TensorEnv to a student model trained in ParHab. Experiments show that incorporating 2D semantic maps improves navigation success and that increasing the number of training floorplans yields further gains at fixed policy capacity and per-stage budgets. The 12.08M-parameter MapAct supports local deployment on a 4 GB NVIDIA Jetson Nano and achieves 68.77%/37.60% SR/SPL on HM3D and 44.19%/19.57% on MP3D, competitive with methods based on large language and vision-language models.

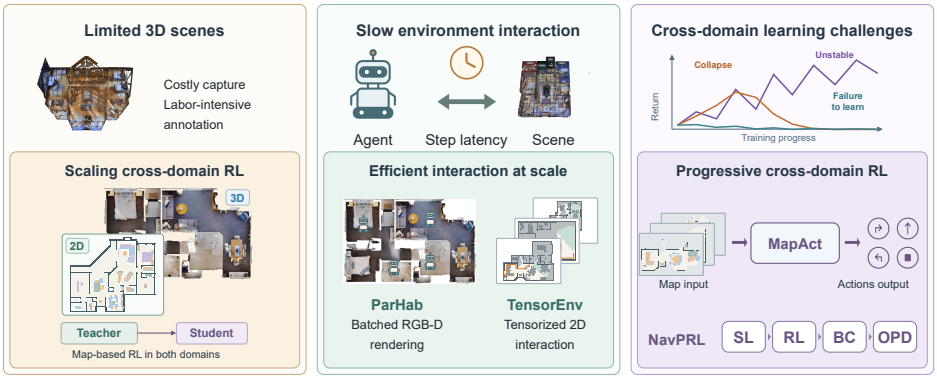


Figure 1: Motivation for cross-domain reinforcement learning in ObjectNav. The left panel shows the limited availability of scanned 3D scenes, motivating the use of 2D semantic maps and cross-domain RL to scale training. The middle panel shows the efficiency challenge in large-scale interaction: ParHab accelerates 3D sampling and map construction, while TensorEnv adapts 2D semantic maps to interactive tensorized environments; together, they make large-scale RL more efficient. The right panel illustrates the difficulty and instability of cross-domain training: MapAct adapts action prediction to map inputs, and NavPRL uses progressive RL to support reliable training performance.

1 INTRODUCTION

Object-goal navigation (ObjectNav) requires an agent to locate an object of a specified category in an unfamiliar environment. Reinforcement learning is well suited to this task because success is determined over an entire trajectory, allowing the policy to optimize a global navigation reward rather than isolated action labels. Previous work has used semantic exploration (Chaplot et al., 2020), auxiliary exploration objectives (Ye et al., 2021), imitation pretraining followed by RL fine-tuning (Ram-

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.

rakhya et al., 2023), and exploitation-guided exploration (Wasserman et al., 2024). However, these methods are generally trained on limited collections of scanned 3D scenes. Their interactive experience therefore remains constrained by the available scenes, which limits generalization to unseen layouts.

A direct way to scale this training process would be to collect and reconstruct more 3D scenes. Each additional scene requires scanning or reconstruction, semantic annotation, and interactive simulation, making large-scale 3D RL expensive. We therefore use 2D semantic maps as an additional source of RL experience while retaining 3D ObjectNav as the target task. NaviScale provides 192,000 maps composed from 24,000 residential floorplans and room-level semantic and obstacle maps extracted from real 3D scans (Lan et al., 2026). These maps expand the available layouts without requiring a new 3D reconstruction for every instance, but the 2D source domain differs from the 3D target domain, creating a domain gap.

To overcome the domain gap while scaling RL, we propose a cross-domain RL framework comprising two training environments, a map-based policy, and a teacher-supervised training procedure. TensorEnv and ParHab jointly improve training throughput: TensorEnv makes 2D semantic maps interactive through tensorized motion, visibility, and map updates, while ParHab accelerates 3D sampling through parallel execution and efficient semantic-map construction. These optimizations increase sampling speed and reduce the cost of collecting training trajectories, allowing more RL experience to be processed under the same training budget. MapAct adapts policy learning to 2D semantic-map RL by taking an accumulated semantic map as input and directly predicting an action. NavPRL transfers knowledge learned from 2D interaction to a student model trained in 3D. The student model is first initialized in ParHab with supervised learning and reinforcement learning, after which a teacher model trained in TensorEnv provides behavior-cloning supervision on successful trajectories and on-policy distillation at states visited by the student model.

With a fixed policy architecture and per-stage training budget, incorporating 2D semantic-map environments improves performance over training only on scanned 3D scenes, and increasing the number of training floorplans produces further gains. The resulting 12.08M-parameter MapAct achieves 68.77% SR and 37.60% SPL on HM3D, and 44.19% SR and 19.57% SPL on MP3D. The policy also runs locally on a 4 GB NVIDIA Jetson Nano.

Our contributions are as follows:

- We formulate RL scaling for ObjectNav as cross-domain training with additional interactive 2D semantic-map environments and adaptation to 3D navigation.
- We develop TensorEnv, ParHab, and MapAct to provide scalable map-based interaction, efficient 3D sampling, and a shared partial-map policy interface, together with NavPRL for teacher-supervised cross-domain transfer.
- We show that the framework improves HM3D performance over scanned-scene-only training, benefits from additional floorplans at fixed training budgets, and achieves strong simulation and local-deployment results with a compact policy.

2 Related Work

2.1 Navigation Data and Environments

Navigation training data fall into two categories: real-world scans and generated data. Matterport3D and HM3DSem provide scanned indoor geometry and semantic annotations (Chang et al., 2017; Yadav et al., 2023), but expanding this category requires additional scanning, reconstruction, and labeling. Generated and composed data reduce this dependence on collecting new scans. ProcTHOR constructs 3D indoor scenes through procedural generation (Deitke et al., 2022), while NaviScale combines residential floorplans with room-level semantic and obstacle maps from scanned scenes to produce a large collection of 2D semantic maps (Lan et al., 2026). We use the latter as an additional source of training environments and study whether interactive RL on these maps improves navigation in 3D scenes.

Training policies on these data also requires environments that support efficient interaction. The Habitat platform comprises Habitat-Sim for three-dimensional simulation and sensor rendering,

and Habitat-Lab for task definition, training, and evaluation (Savva et al., 2019). Habitat-Lab uses Habitat-Sim as its core simulator and provides interfaces for vectorized environment execution and batch rendering (Habitat-Lab Contributors, 2026b;a). For map-based policies, trajectory collection additionally requires semantic processing and incremental map construction. In our standard Habitat training pipeline, these observation-processing costs and per-environment execution constrain sampling throughput. ParHab coordinates parallel navigation, batched rendering, and GPU map construction to address these costs. TensorEnv complements this 3D environment by enabling policy interaction directly on static semantic maps, making the additional 2D data available for RL without rendering 3D observations.

2.2 Navigation Methods

End-to-end navigation methods encode observations, integrate information from previous steps, and directly predict actions. Within this formulation, prior work studies both the representation of observation history and the learning of navigation decisions. Ye et al. (2021) use auxiliary tasks to improve recurrent memory. PIRLNav initializes its policy from human demonstrations and then refines it through RL (Ramrakhya et al., 2023). These methods allow action selection to be optimized directly from demonstrations or trajectory returns. Their effectiveness nevertheless depends on the diversity of the scenes and trajectories available for training.

Modular navigation methods separate spatial representation, navigation-goal selection, and action execution. Observations update a map or another scene representation; a prediction module then selects a navigation goal, which a planner converts into actions. PONI learns potential functions on semantic maps to guide exploration (Ramakrishnan et al., 2022). PEANUT predicts target-object locations in unobserved regions from partial semantic maps (Zhai & Wang, 2023). NaviFormer incorporates spatial and temporal context into navigation-goal selection (Xie et al., 2025). More recent methods use pretrained models for this intermediate reasoning: SG-Nav reasons over scene graphs (Yin et al., 2024), VoroNav selects waypoints through language-model reasoning (Wu et al., 2024), and ImagineNav uses imagined scenes to guide navigation (Zhao et al., 2025). This decomposition makes intermediate predictions explicit and permits separate design of prediction and planning modules, although prediction errors can propagate to action execution. Our approach combines explicit spatial representation with direct action learning: MapAct retains observed spatial information in a semantic map, summarizes interaction history with a GRU, and predicts actions through an RL-trained policy. The shared map input allows the same architecture to learn from interaction in both 2D semantic maps and 3D scenes.

3 Method

Our framework supplements training in scanned scenes with interaction on existing 2D semantic maps (Figure 2). TensorEnv trains a teacher model on the additional maps; its supervision is combined with RL updates to the student model in ParHab. We first describe how the two environments provide partial semantic maps (Section 3.1), then introduce MapAct (Section 3.2) and the NavPRL training procedure (Section 3.3).

3.1 Training Environments

ParHab: efficient interaction in 3D scenes. ParHab accelerates Habitat-Sim rendering and semantic map construction for policy training (Savva et al., 2019). At step t, Habitat-Sim provides an RGB image I_t, a depth image D_t, and the agent pose ξ_t, expressed by its position and heading in map coordinates. With predicted semantics, segmentation and map construction follow

$$\begin{aligned} S_t &= \mathrm{Segment}(I_t), \\ M_t^{\mathrm{3D}} &= \mathrm{MapUpdate}\left(M_{t-1}^{\mathrm{3D}}, D_t, S_t, \xi_t\right). \end{aligned} \tag{1}$$

Here $\mathrm{Segment}$ denotes YOLO26n and S_t its pixelwise semantic labels; exploration training uses simulator-provided labels instead. $\mathrm{MapUpdate}$ uses the depth, labels, and pose to integrate observations into the previous partial map M_{t-1}^{3D}. ParHab batches multiple views per scene and constructs maps on the GPU to increase sampling throughput. Appendix A.2 describes the rendering, synchronization, and mapping implementation.

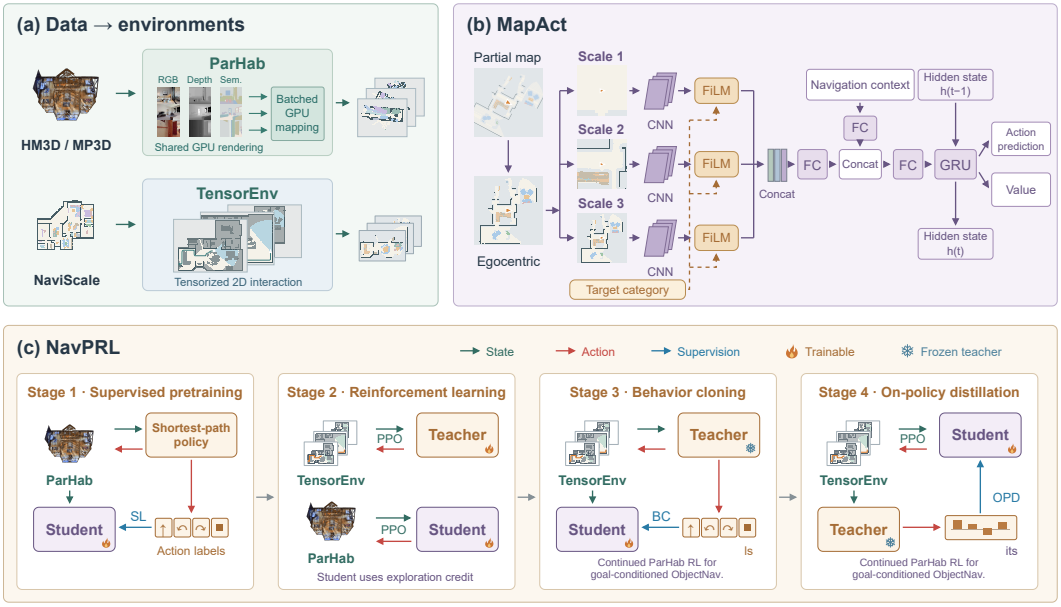


Figure 2: Training framework. (a) ParHab reconstructs maps from three-dimensional observations; TensorEnv supports interaction on 2D semantic maps. (b) MapAct combines target-conditioned map features, navigation context, and recurrent state. (c) SL initializes the student model in ParHab; RL trains exploration by the student model in ParHab and the teacher model in TensorEnv; BC uses successful teacher-model trajectories; OPD supervises states visited by the student model with teacher-model logits. Student-model ParHab PPO switches to goal-conditioned ObjectNav in BC and OPD; TensorEnv PPO also accompanies OPD. Action symbols denote the output space; STOP is masked during exploration.

TensorEnv: interaction on 2D semantic maps. TensorEnv makes existing NaviScale maps (Lan et al., 2026) available for interactive policy training. A complete map containing traversability, walls, and semantic categories defines each environment. After an action, TensorEnv updates the agent pose, checks collisions, and determines visible cells by ray casting. It combines newly observed cells with the previous partial map M_{t-1}^{2D} to obtain M_t^{2D}. Both environments expose accumulated partial maps through a shared channel interface, while differing in observation construction and noise. The policy receives this accumulated partial map; the complete map is used only by the environment. Batched GPU tensor operations implement motion, visibility queries, and map updates without rendering 3D observations. Appendix A.3 specifies the tensor layout and simulation settings.

Common policy input. Both M_t^{3D} and M_t^{2D} are two-dimensional maps; their superscripts identify the training environment. We write either as M_t below. At step t, the policy receives this partial semantic map, the agent pose ξ_t, and a target category g. Both maps use K semantic channels and four additional channels for obstacles, explored space, visited locations, and frontiers; frontiers indicate unexplored cells adjacent to explored free space. For a batch of B agents, the map shape is $B \times (K+4) \times H_M \times W_M$, where H_M and W_M are the map height and width in cells. Appendix A.1 gives the HM3D map configuration. Both environments support the action set $\mathcal{A} = \{\mathrm{stop}, \mathrm{forward}, \mathrm{left}, \mathrm{right}\}$.

3.2 MapAct: Semantic Maps to Actions

MapAct uses the accumulated map for spatial information and a recurrent state for navigation history. Let u_t denote recent interaction context and h_{t-1} the previous recurrent state. For one agent, the policy computation is

$$\begin{aligned} z_t &= \mathrm{MapEncoder}_\theta(M_t, \xi_t, g), \\ h_t &= \mathrm{GRU}_\theta\left(\mathrm{Fuse}_\theta(z_t, u_t), h_{t-1}\right), \\ (\ell_t, V_\theta(x_t)) &= \mathrm{Heads}_\theta(h_t), \qquad p_\theta(\cdot \mid x_t) = \mathrm{softmax}(\ell_t). \end{aligned} \tag{2}$$

Here θ denotes the policy parameters and $x_t = (M_t, \xi_t, g, u_t, h_{t-1})$ its input. $\mathrm{MapEncoder}$ summarizes local obstacles and broader semantic context in a 512-dimensional feature z_t. It uses agent-centered map crops at three scales and target-conditioned FiLM (Perez et al., 2018) to make the representation sensitive to the requested category. Fuse combines this feature with u_t to form the GRU input. The context contains the previous 16 interaction records, map coverage, and the remaining episode fraction. The GRU maintains a 512-dimensional state h_t, and Heads predicts four action logits ℓ_t and a scalar value estimate $V_\theta(x_t)$.

The distribution p_θ is the unmasked action distribution; training-specific action selection determines the executed action a_t. The environment then provides the next pose ξ_{t+1} and partial map M_{t+1} for the subsequent decision. Hidden states reset at episode boundaries. Appendix B.1 gives the encoder dimensions and context construction. MapAct has 12.08M parameters, including the value head and a two-output auxiliary distance head; this count excludes the separate YOLO26n semantic predictor. The auxiliary distance loss is disabled during exploration.

3.3 NavPRL: Progressive Policy Training

NavPRL trains the student model in four stages: supervised learning (SL), reinforcement learning (RL), behavior cloning (BC), and on-policy distillation (OPD). The student and teacher models use the MapAct architecture with parameters θ and θ_T, respectively, and maintain independent recurrent states.

Initialization and reinforcement learning. SL initializes the student model with shortest-path demonstrations from ParHab. For a demonstration dataset $\mathcal{D}$ containing policy inputs x and reference actions $a^\star$, the supervised loss is

$$\mathcal{L}_{\mathrm{CE}}(\theta; \mathcal{D}) = -\mathbb{E}_{(x,a^\star)\sim\mathcal{D}} \log p_\theta(a^\star \mid x). \tag{3}$$

In stage 2 (RL), the student model learns goal-free exploration in ParHab, while the teacher model is trained by RL in TensorEnv. Each policy optimizes the expected return of its trajectories,

$$J(\vartheta) = \mathbb{E}_{\tau\sim\pi_\vartheta^{\mathrm{beh}}} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right], \tag{4}$$

where ϑ denotes the parameters of the policy being trained, $\pi_\vartheta^{\mathrm{beh}}$ its behavior policy, τ a trajectory of length T, r_t the reward, and γ the discount factor. Student exploration masks STOP, retains the immediate reward for newly observed area, and adds exploration credit to eligible forward actions within the most recent 20 transitions. Appendix B.4 specifies the action-selection rules, rewards, and clipped policy objective.

Behavior cloning and on-policy distillation. In stage 3 (BC), student-model RL in ParHab switches from goal-free exploration to goal-conditioned ObjectNav. The teacher model is frozen, and Equation 3 additionally supervises the student model on successful teacher-model trajectories from TensorEnv. In stage 4 (OPD), teacher-model queries along trajectories generated by the student model in TensorEnv provide supervision on states visited by the student model (Ross et al., 2011). For shared observations $o_t = (M_t, \xi_t, g, u_t)$, the teacher-model and student-model inputs are $x_t^T = (o_t, h_{t-1}^T)$ and $x_t^S = (o_t, h_{t-1}^S)$, respectively. The teacher model updates its own recurrent state from the student's observations and executed-action history. With p_T denoting the teacher model's action distribution at temperature one, OPD minimizes

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = -\mathbb{E}_{(x^T,x^S)\sim d_\theta} \sum_{a\in\mathcal{A}} p_T(a \mid x^T) \log p_\theta(a \mid x^S), \tag{5}$$

where d_θ is the distribution of paired teacher-model and student-model inputs along trajectories generated by the student model. Teacher-model probabilities remain fixed during each optimization update. ParHab PPO remains active in both stages, and TensorEnv PPO accompanies OPD. Teacher-model supervision thus transfers experience from the expanded map dataset while goal-conditioned 3D interaction adapts the student model to ObjectNav. Appendix B.2 gives the stage budgets and synchronized update procedure.

4 EXPERIMENTS

We first compare MapAct with prior ObjectNav methods, then test whether the cross-domain procedure improves three-dimensional navigation over ParHab-only training. Teacher-transfer and training ablations examine how two-dimensional experience is used; the following floorplan-diversity study tests its scaling at fixed optimization-sample budgets. We then analyze semantic context, report sampling profiles, and evaluate physical deployment.

4.1 EXPERIMENTAL SETUP

Data and evaluation. Training combines scanned scenes from HM3D or MP3D (Yadav et al., 2023; Chang et al., 2017) with existing NaviScale 2D semantic maps (Lan et al., 2026). We use HM3DSem v0.1, with 80 training and 20 validation scenes, and evaluate six target categories on 2,000 validation episodes. MP3D has 56 training and 11 validation scenes, with 2,195 evaluation episodes and 21 target categories. The MP3D model is trained separately using the same procedure with dataset-specific settings; hyperparameter development and ablations use HM3D. Episodes allow at most 500 actions. The agent receives RGB-D, planar GPS, and compass observations, predicts semantics with dataset-specific YOLO26n models, and constructs its map incrementally. Formal navigation evaluation enables STOP and follows Habitat's standard ObjectNav success and SPL criteria.

Metrics and repeated runs. For N episodes, let S_i denote success, ℓ_i the shortest path to a valid goal, and d_i the executed path length. We report success rate (SR) and success weighted by path length (SPL) (Anderson et al., 2018):

$$\mathrm{SR} = \frac{1}{N}\sum_{i=1}^{N} S_i, \qquad \mathrm{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{\ell_i}{\max(\ell_i, d_i)}. \tag{6}$$

Our MapAct and TensorEnv teacher-model results are means $\pm$ sample standard deviations across three independently trained models, including the ablations and teacher-model evaluations. Standard deviations use the $n-1$ denominator with $n = 3$. Metrics are percentages; differences are percentage points computed from unrounded means.

Training. NavPRL uses goal-free ParHab exploration in stage 2 (RL) and goal-conditioned ObjectNav in stages 3–4 (BC/OPD), with a 20-transition exploration-credit window in stage 2. Training uses A800 GPUs. Checkpoints are selected at fixed optimization-sample budgets, which include transition reuse and do not equate newly collected interactions or GPU time. Appendix B.2 reports the schedule and budget accounting.

4.2 COMPARISON WITH PRIOR METHODS

MapAct achieves 68.77% SR and 37.60% SPL on HM3D, improving over NaviScale by 4.47 SR and 2.80 SPL points. On MP3D, MapAct achieves 44.19% SR and 19.57% SPL, compared with NaviScale's 43.1% SR and 16.8% SPL. Table 1 distinguishes original-paper scores from our reproductions. Appendix E details the evaluation settings, PIRLNav retraining, and independent GitHub reports.

4.3 CROSS-DOMAIN TRAINING AND TRANSFER

Cross-domain training versus ParHab only. Table 2 compares final policies at the same total optimization-sample budget. The full framework reaches 68.77% SR and 37.60% SPL, compared with 58.20% SR and 29.03% SPL for ParHab-only training: gains of 10.57 and 8.57 percentage points. This tests the complete cross-domain procedure against 3D-only training.

Reverse-direction transfer variants. The TensorEnv teacher achieves 88.27% SR on 1,000 held-out 2D maps, but only 20.73% when evaluated on HM3D (Table 3). The two approximately 61% conditions are reverse-direction ablations in which a ParHab 3D teacher supervises a TensorEnv 2D

Table 1: ObjectNav comparison (%). Bold marks the highest value in each column. MapAct results are mean ± sample standard deviation over three runs.

		HM3D		MP3D	
Method	Venue	SR ↑	SPL ↑	SR ↑	SPL ↑
PEANUT	ICCV 2023	58.8*	29.1*	40.5	15.8
PIRLNav	CVPR 2023	61.9	27.9	–	–
XGX	ICRA 2024	57.1*	28.9*	–	–
SG-Nav-GPT	NeurIPS 2024	54.0	24.9	40.2	16.0
Uni-NaVid	RSS 2025	56.9*	27.5*	–	–
NaviFormer	AAAI 2025	61.3	29.6	40.1	15.1
g3D-LF	CVPR 2025	55.6	31.8	39.0	18.8
BeliefMapNav	NeurIPS 2025	61.4	30.6	37.3	17.6
TrajRAG	CVPR 2026	62.5	33.9	42.6	18.0
GLMap	CVPR 2026	62.7	33.7	42.5	18.3
NaviScale	arXiv 2026	64.3	34.8	43.1	16.8
MapAct (ours)	–	$\mathbf{68.77} \pm 0.12$	$\mathbf{37.60} \pm 0.10$	$\mathbf{44.19} \pm 0.27$	$\mathbf{19.57} \pm 0.21$

*Results reproduced by us; Appendix E gives settings, published scores, and independent GitHub reports. A dash denotes an unavailable result.

Table 2: HM3D ablations at equal total optimization-sample budgets (%).

Configuration	SR ↑	SPL ↑
Full framework	68.77 ± 0.12	37.60 ± 0.10
ParHab only	58.20 ± 0.36	29.03 ± 0.12
2D PPO without BC/OPD	58.37 ± 0.18	29.31 ± 0.11
w/o BC	59.40 ± 0.44	29.33 ± 0.12
w/o OPD supervision	62.07 ± 0.35	33.70 ± 0.10
w/o continued ParHab RL	22.80 ± 0.30	5.63 ± 0.38

Table 3: Reverse-direction transfer ablations (%). A ParHab 3D teacher supervises a TensorEnv 2D student; the two approximately 61% conditions differ in whether OPD is included.

Policy / domain	SR ↑	SPL ↑
TensorEnv teacher / 2D	88.27 ± 0.42	42.27 ± 0.35
TensorEnv teacher / HM3D	20.73 ± 0.49	5.07 ± 0.15
3D→2D (BC only)	60.90 ± 0.15	33.09 ± 0.12
3D→2D (BC + OPD)	61.02 ± 0.15	33.28 ± 0.10
Final student / HM3D	68.77 ± 0.12	37.60 ± 0.10

student; they test whether the 2D student can acquire 3D navigation behavior through 3D supervision. The BC-only condition reaches 60.90% SR and 33.09% SPL from three runs with SR values of 60.75%, 60.90%, and 61.05%. Adding OPD to BC gives 61.02% SR and 33.28% SPL, a 0.12-point SR increase. Both conditions remain below the 68.77% full framework, indicating that this reverse transfer is ineffective under the tested setup.

Training-component ablations. To compare the complete cross-domain procedure with direct student RL at the same budget, we replace BC/OPD supervision in TensorEnv with student PPO during stages 3–4. Stages 1–2 and ParHab PPO are unchanged, as are each stage's budget and its allocation between environments. This yields 58.37% SR and 29.31% SPL, close to ParHab-only training and 10.40/8.29 points below the full method. Under this budget, the teacher-supervised procedure therefore outperforms direct student PPO in TensorEnv.

For stage or stream removal, the budget is redistributed among retained components in their original proportions (Appendix B.2). Skipping BC reduces SR by 9.37 points; removing only OPD supervision, while retaining its RL updates, reduces SR by 6.70 points. Discontinuing ParHab RL during BC and OPD yields the largest degradation, to 22.80% SR and 5.63% SPL. These results support combining teacher supervision with goal-conditioned ParHab RL.

4.4 Scaling Floorplan Diversity

We vary the number of distinct training floorplans from 1K to 16K, keeping the policy architecture, every stage's processed optimization-sample budget, and all other settings fixed. TensorEnv uses the corresponding floorplan set during RL, BC, and OPD. Each configuration is evaluated on HM3D with three independently trained models; 16K is the full method. The scale counts distinct

floorplans, not room recompositions, episodes, or transitions. Unlike the ParHab-only comparison above, this study retains the complete training procedure and changes only its floorplan set.

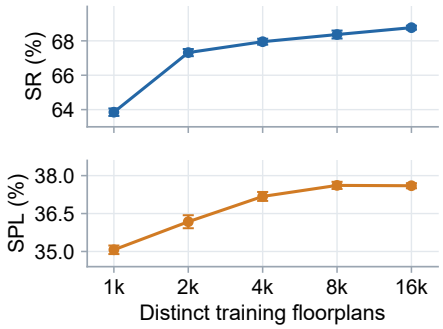


Figure 3: Floorplan scaling at fixed per-stage budgets. Error bars: sample SD over three runs.

Table 4: Semantic masking on HM3D (%).

Policy	Input	SR ↑	SPL ↑
Full	All	68.77 ± 0.12	37.60 ± 0.10
Full	Target	63.03 ± 0.57	34.43 ± 0.21
ParHab	All	58.20 ± 0.36	29.03 ± 0.12
ParHab	Target	58.10 ± 0.26	28.77 ± 0.15

SR increases across the tested sizes, from 63.85% to 68.77%, a gain of 4.92 percentage points computed from unrounded means (Figure 3). SPL rises from 35.07% at 1K to 37.61% at 8K and remains similar at 16K (37.60%). Thus, additional floorplan diversity improves three-dimensional navigation at fixed policy capacity and per-stage optimization-sample budgets, with diminishing SR gains and a plateau in SPL. These budgets do not fix newly collected interactions or GPU time.

4.5 Contribution of Non-Target Semantics

We zero non-target semantic channels at evaluation, retaining the target, geometry, exploration, and history channels and using the corresponding unmodified model checkpoints. Table 4 compares both the complete (Full) and ParHab-only (ParHab) policies using all semantic channels (All) or the target channel alone (Target).

For the complete policy, masking non-target categories reduces SR by 5.73 points and SPL by 3.17 points. The corresponding ParHab-only changes are 0.10 and 0.27 points. This contrast is consistent with greater dependence on non-target category context after the complete training procedure, although masking also changes the input distribution.

4.6 Sampling Efficiency

Table 5 reports sampling profiles on one A800 GPU for the original Habitat-based pipeline, ParHab, and TensorEnv. The original pipeline uses ten independent environment processes, ParHab uses 50 agents across ten scenes, and TensorEnv batches 100 agents. Both 3D pipelines are evaluated with simulator semantics and predicted semantics. In ParHab, semantic prediction accounts for 72.8% of sampling time; TensorEnv instead updates observations directly from stored semantic maps. The timing conventions and profiling protocols are specified in Appendices A.4–A.6.

4.7 Physical Deployment

We evaluate all four methods on the same 10 real-world tasks using a Hiwonder JetAuto robot with RGB-D sensing and RTAB-Map, following NaviScale (Lan et al., 2026). Our method runs locally on a 4 GB NVIDIA Jetson Nano and succeeds on 8 tasks. Released models of PEANUT, NaviScale, and PIRLNav succeed on 5, 6, and 0 tasks, respectively. PEANUT and NaviScale run on a remote NVIDIA RTX 3090 GPU connected to the robot over the network; PIRLNav runs locally on the edge device. Figure 4 illustrates an autonomous bed-search trajectory. Appendix D provides the comparison settings and additional visualization details.

Table 5: Sampling profiles on A800. Times are in milliseconds, using the convention indicated for each group. Pred. uses YOLO26n. Entries are means; a dash denotes an inapplicable stage. † denotes throughput calculated from summed stage times.

Setting	Act./render	Segm.	Map	Policy	Total	Steps/s
Original Habitat: 10 workers; time per concurrent worker step						
GT	11.720	–	316.814	16.522	345.056	28.53
Pred.	11.018	9.659	337.211	17.922	375.810	26.15
ParHab: 50 agents; amortized time per agent-step						
GT	0.428	–	0.306	0.102	0.838	1193.7
Pred.	0.339	2.479	0.457	0.128	3.404	293.8
TensorEnv: 100 agents; amortized time per agent-step						
Stored maps	–	–	0.407	0.824	1.231	812.1†

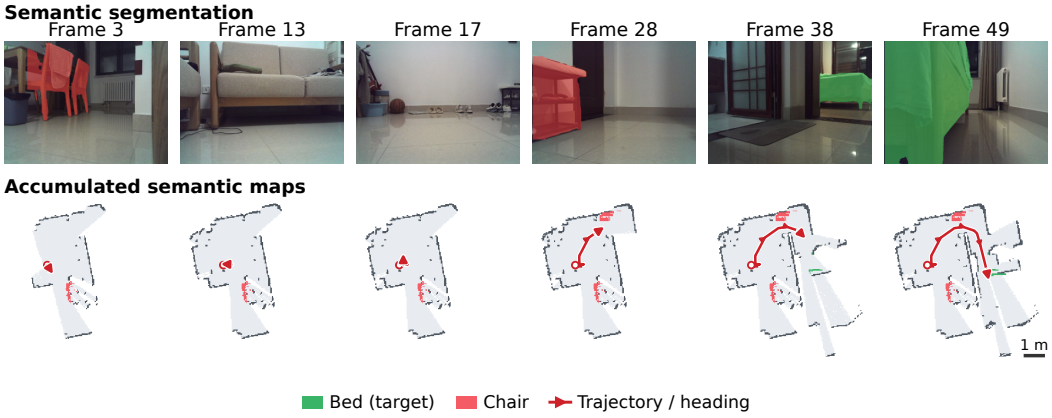


Figure 4: Real-world observations and semantic-map accumulation. Top: segmentation overlays. Bottom: maps with recorded trajectories and heading arrows in red. Early observations and heading changes (frames 3, 13, and 17) precede movement toward the corridor (frame 28) and bed observations (frames 38 and 49). All maps share the same spatial window.

5 Discussion and Limitations

TensorEnv omits three-dimensional occlusion, vertical structure, and sensor noise. Continued ParHab RL exposes the student model to maps reconstructed from sensor observations. Evaluation-time semantic masking also shifts the input distribution, limiting conclusions about the source of semantic-context benefits. Appendix C examines SL, exploration credit, and memory through single-run explored-area measurements; their effects on final ObjectNav success and training stability remain to be evaluated. The 20-transition credit rule excludes earlier actions and detours that initially increase distance to a discovery centroid, and its additional rewards modify the discounted training objective.

The scaling study establishes a diversity benefit over the tested floorplan range. Processed-sample budgets and configuration-specific sampling profiles do not determine total training cost; the profiles exclude gradient updates and use different observation workloads. The real-world comparison covers 10 tasks with different deployment configurations, demonstrating local execution but limiting broader transfer and efficiency conclusions. Semantic perception uses dataset-specific YOLO26n models, which are separate from the navigation policy.

6 Conclusion

We studied task-specific RL as an alternative to large language and vision-language models for navigation decisions. Our cross-domain framework uses existing 2D semantic maps to expand policy-training environments and transfers the teacher model's navigation knowledge to a student model that continues RL in 3D scenes. Additional floorplans improve HM3D success at fixed policy

capacity and per-stage budgets. The 12.08M-parameter MapAct achieves competitive HM3D and MP3D performance and runs locally on a 4 GB NVIDIA Jetson Nano, demonstrating the potential of task-specific experience for learning compact navigation policies.

REPRODUCIBILITY STATEMENT

Section 3 defines the policy and training procedure; the appendices describe environment operations, training settings, the exploration objective, and the real-world evaluation.

AI USE STATEMENT

AI tools were used to assist with software implementation and code editing, as well as language editing and polishing. All AI-assisted code and text were reviewed and verified by the authors, who take responsibility for the final implementation and manuscript.

REFERENCES

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. URL https://arxiv.org/abs/1807.06757.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017. URL https://arxiv.org/abs/1709.06158.

Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *Advances in Neural Information Processing Systems*, 2020. URL https://arxiv.org/abs/2007.00643.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied ai using procedural generation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/27c546ab1e4f1d7d638e6a8dfbad9a07-Abstract-Conference.html.

Habitat-Lab Contributors. Habitat-Lab: Environment batch renderer implementation. Source code snapshot, 2026a. URL https://github.com/facebookresearch/habitat-lab/blob/main/habitat-lab/habitat/core/batch_rendering/env_batch_renderer.py. Main-branch implementation examined on September 24, 2026.

Habitat-Lab Contributors. Habitat-Lab: Vectorized environment implementation. Source code snapshot, 2026b. URL https://github.com/facebookresearch/habitat-lab/blob/main/habitat-lab/habitat/core/vector_env.py. Main-branch implementation examined on September 24, 2026.

Chuanlin Lan, Yanwei Zheng, Weijian Liu, Zhitong Zhou, Xiao Zhang, Fuzhen Zhuang, and Dongxiao Yu. NaviScale: Generating large-scale semantic map datasets for object navigation. *arXiv preprint arXiv:2609.27218*, 2026. URL https://arxiv.org/abs/2609.27218.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1709.07871.

Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. PONI: Potential Functions for ObjectGoal Navigation with Interaction-free Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. PIRLNav: Pretraining with imitation and RL finetuning for ObjectNav. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17896–17906, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Ramrakhya_PIRLNav_Pretraining_With_Imitation_and_RL_Finetuning_for_ObjectNav_CVPR_2023_paper.html.

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*, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Manolis Savva et al. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. URL https://arxiv.org/abs/1904.01201.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Yiyao Wang, Sixian Zhang, Keming Zhang, Xinhang Song, Songjie Du, and Shuqiang Jiang. TrajRAG: Retrieving Geometric-Semantic Experience for Zero-Shot Object Navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15166–15176, 2026.

Zihan Wang and Gim Hee Lee. g3D-LF: Generalizable 3D-Language Feature Fields for Embodied Tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14191–14202, 2025.

Justin Wasserman, Girish Chowdhary, Abhinav Gupta, and Unnat Jain. Exploitation-guided exploration for semantic embodied navigation. In *IEEE International Conference on Robotics and Automation*, pp. 2901–2908, 2024. doi: 10.1109/ICRA57147.2024.10610117. URL https://arxiv.org/abs/2311.03357.

Pengying Wu, Yao Mu, Bingxian Wu, Yi Hou, Ji Ma, Shanghang Zhang, and Chang Liu. VoroNav: Voronoi-based Zero-shot Object Navigation with Large Language Model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 53757–53775, 2024.

Wei Xie, Haobo Jiang, Yun Zhu, Jianjun Qian, and Jin Xie. NaviFormer: A Spatio-Temporal Context-Aware Transformer for Object Navigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14708–14716, 2025. doi: 10.1609/aaai.v39i14.33612.

Karmesh Yadav et al. Habitat-matterport 3d semantics dataset. *arXiv preprint arXiv:2210.05633*, 2023. URL https://arxiv.org/abs/2210.05633.

Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. URL https://arxiv.org/abs/2104.04112.

Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. SG-Nav: Online 3D Scene Graph Prompting for LLM-based Zero-shot Object Navigation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0171.

Albert J. Zhai and Shenlong Wang. PEANUT: Predicting and Navigating to Unseen Targets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10926–10935, 2023.

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. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.013. URL https://www.roboticsproceedings.org/rss21/p013.html.

Sixian Zhang, Yiyao Wang, Xinhang Song, Keming Zhang, Zijian Xu, and Shuqiang Jiang. Multi-Scale Gaussian-Language Map for Zero-shot Embodied Navigation and Reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 37086–37097, 2026.

Xinxin Zhao, Wenzhe Cai, Likun Tang, and Teng Wang. ImagineNav: Prompting Vision-Language Models as Embodied Navigator through Scene Imagination. In *International Conference on Learning Representations*, volume 2025, pp. 94387–94401, 2025.

Zibo Zhou, Yue Hu, Lingkai Zhang, Zonglin Li, and Siheng Chen. BeliefMapNav: 3D Voxel-Based Belief Map for Zero-Shot Object Navigation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2782.

A TRAINING ENVIRONMENT DETAILS

This appendix expands the environment description in Section 3.1. ParHab retains three-dimensional scene interaction and reorganizes observation generation and processing. TensorEnv executes navigation directly on stored 2D semantic maps. Both feed the map interface used by MapAct, with environment-specific observation construction.

A.1 ENVIRONMENT STATE AND POLICY INPUTS

ParHab constructs observations from scene geometry, navigation meshes, and camera poses. TensorEnv uses complete semantic maps for collision and visibility queries. In both environments, the policy receives spatial information accumulated from observations, together with the target category and interaction context described in Section 3.2.

The recorded HM3D configuration uses a 960×960 canvas with 5-cm cells. Table A.1 specifies its 28 policy channels. For B parallel agents, the logical map tensor has shape $B \times 28 \times 960 \times 960$. Each episode maintains its own map and trajectory.

Table A.1: HM3D map channels supplied to MapAct.

Map component	Channels	Contents
Obstacles	1	Observed obstacles
Explored region	1	Accumulated visibility
Trajectory	1	Previously visited locations
Semantics	24	Observed category masks
Frontier	1	Unexplored cells near observed free space

The frontier channel identifies candidate regions for further exploration. The target category is supplied separately; the six HM3D goal categories are a subset of the 24 semantic channels.

A.2 PARHAB: PARALLEL NAVIGATION AND BATCHED OBSERVATION PROCESSING

ParHab retains Habitat-Sim's navigation meshes and rendering backend (Savva et al., 2019). CPU workers advance the agents and return camera poses; the main process then performs batched GPU rendering, mapping, and policy inference. Rendering waits for the current batch of poses. Multiple views of the same scene share rendering resources, allowing observations for several agents to be generated together. Action and pose metadata cross the CPU–GPU boundary, while rendered images remain on the GPU.

To connect rendering with tensor processing, ParHab extends the renderer's OpenGL–CUDA interface to copy sensor outputs into independently owned PyTorch tensors on the current CUDA stream. CUDA events coordinate access between streams so that processing waits for the required outputs rather than synchronizing the entire device at each handoff. The copied tensors allow subsequent map updates and policy inference to use the rendered observations without CPU image readback. Semantic map construction also runs on the GPU, batching observation processing and accumulation across agents.

RGB images have shape $B \times 480 \times 640 \times 3$; depth and semantic labels have shape $B \times 480 \times 640$. The camera has a 79° horizontal field of view and a height of 0.88 m. Depth measures distance along the optical axis over 0.5–5.0 m; mapping accepts normalized values greater than zero and at most one. Exploration uses rendered semantic labels, whereas ObjectNav evaluation uses YOLO26n predictions. Episode-relative position and heading come from the simulator state. On the physical robot, RTAB-Map supplies localization (Appendix D).

Mapping combines depth, semantic labels, and the supplied pose to locate observed surfaces in the episode frame. Height filtering separates obstacles from the floor, and semantic labels identify the observed categories. The HM3D mapper accumulates nine categories in their corresponding channels of the 24-channel policy representation; the remaining semantic channels are zero.

The map retains previously observed regions and the agent's trajectory as new observations arrive. Semantic information is exposed only in explored regions, while visited locations provide evidence

of traversable space. At episode boundaries, map, history, and recurrent states reset independently for each agent; terminal metadata is retained separately from the next episode's initial observation.

A.3 TensorEnv: Motion and Partial Observations on 2D Semantic Maps

Each episode selects a start and target and aligns a stored map to a 960×960 canvas. The logical static tensor has shape $B \times 26 \times 960 \times 960$: one traversability channel, one wall channel, and 24 semantic channels. A separate height map defines obstacles, including walls. Goal regions and distance fields support reward and success evaluation.

A forward action proposes a 0.25-m displacement, which is rejected if the checked path intersects non-traversable space. Turns change the heading by 30° without translation. STOP or the step limit ends the episode.

Ray casting determines the visible region after each action, retaining visible walls and excluding the cells they occlude. The field of view is 79° and the maximum sensing distance is 5 m. Visibility accumulates across steps, and newly observed area provides the exploration statistic. The initial observation establishes visibility without assigning discovery credit to an action.

Obstacle and category maps are masked by accumulated visibility before policy encoding. The dense environment interface has 29 channels, including current-location and target masks; MapAct excludes those two channels and adds the frontier band to obtain the 28 channels in Table A.1. Motion, visibility queries, and map construction operate across the environment batch without rendering RGB images.

A.4 ParHab Systems Profile

Measurement setting. The ParHab profiles in Table 5 use one A800 GPU and 50 navigation slots: ten scenes with five agents per scene, at 640×480 resolution. Each of ten independent measurements uses 30 warm-up batch steps followed by 300 timed batch steps. The timed loop includes action transfer, CPU pose updates, rendering, semantic processing, mapping, policy input construction, and policy inference. Gradient updates and trajectory or image saving are excluded. STOP is disabled during this fixed-length timing run.

The GT setting renders RGB, depth, and simulator semantics. The predicted-semantic setting renders RGB-D and uses YOLO26n Semantic with FP16 autocast; the navigation policy uses eager FP32 in both settings. These two profiles use the same policy checkpoint.

Timing convention. Act./render includes action transfer, CPU pose updates, and sensor rendering; Segm. includes semantic prediction and its preprocessing; Map includes map resets and updates; Policy includes input construction, inference, and action selection. Component and total batch times are divided by 50 to report amortized cost per agent-step. The total also includes minor loop overhead, so rounded component times need not sum exactly to the displayed total. Throughput counts all timed agent transitions and uses unrounded elapsed times. Stage boundaries include GPU synchronization. The reported standard deviations describe variation across the ten measurements; component entries are means.

A.5 TensorEnv Sampling Profile

The TensorEnv profile uses 100 parallel agents on one A800 GPU. After 30 warm-up batch steps, three rounds of 100 measured batch steps yield 30,000 agent transitions. Each stage boundary synchronizes the CUDA context. Mapping includes action execution, visibility computation, and observation updates; policy time includes input preparation, network inference, and action selection. TensorEnv operates on stored semantic maps and therefore requires neither RGB-D rendering nor image segmentation. Initialization, warm-up, gradient updates, and trajectory saving are excluded.

Each reported time is the synchronized batch duration divided by 100, averaged over 300 batches. The unrounded mapping and policy costs are 0.407329 and 0.824025 ms per agent-step, respectively, summing to 1.231354 ms. The corresponding rate, $1000/1.231354 = 812.1$ agent-steps/s, is calculated from these timed stages; it is not a separate measurement of full-loop throughput. The

available profile reports means without standard deviations. This 100-agent profile supplies both the total and the components in Table 5.

A.6 Original Habitat Pipeline and Comparison Scope

The original Habitat-based application uses ten independent processes on one A800, each with its own environment, semantic mapper, and policy. It uses 640×480 observations, 960×960 maps, and policy inputs at three spatial scales (60, 240, and 960 cells), with a GRU and a 16-step interaction history. Each worker is assigned one CPU core. Thirty warm-up steps precede three rounds of 100 measured steps per worker, yielding 3,000 agent transitions for each semantic setting. No episode resets occur during measurement. Initialization, warm-up, learning updates, and saving are excluded.

For GT semantics, action/render time includes semantic rendering and instance-to-category conversion. With predicted semantics, the YOLO26n FP16 call is timed separately. Action/render also includes CPU movement, sensor readback, and environment bookkeeping. Mapping uses the application's original per-process mapper; policy time includes full-map preparation, multiscale and history inputs, inference, and action selection. The GT and predicted-semantic runs use the same policy checkpoint as the TensorEnv profile.

These component durations are means per concurrent worker step, including scheduling and synchronization waits; they are not divided by ten. Throughput instead counts all workers' transitions over their common elapsed time. ParHab and TensorEnv report amortized batch costs. Thus, the time columns characterize their respective execution arrangements, while the throughput column reports aggregate sampling rates, with the TensorEnv rate derived as noted above. Different agent counts, map-processing implementations, and observation workloads preclude attributing the throughput differences to an isolated optimization. These sampling profiles exclude gradient updates and do not measure complete NavPRL training time.

B Implementation and Training Details

B.1 Policy Configuration and Tensor Shapes

The map encoder in Equation 2 uses three agent-centered crops of 60, 240, and 960 cells, sampled with nearest-neighbor interpolation and zero padding. Adaptive maximum pooling reduces each crop to 32×32. For HM3D, concatenating the three 28-channel crops gives 84 input channels. Each convolution has three groups, with 64 output channels per scale. The encoder dimensions are listed below; spatial tensors use channel–height–width order and omit the common batch dimension B.

Operation	Output shape
Concatenate three map crops	$84 \times 32 \times 32$
2×2 convolution, stride 2	$192 \times 16 \times 16$
3×3 convolution	$192 \times 14 \times 14$
Target-conditioned FiLM	$192 \times 14 \times 14$
3×3 convolution	$192 \times 12 \times 12$
3×3 convolution	$192 \times 10 \times 10$
Flatten	19200
Linear projection and ReLU	512

The 3×3 convolutions have stride one and no padding. FiLM projects the 32-dimensional target embedding to 192 scale-specific multiplicative and additive coefficients, each broadcast over spatial positions. Each scale receives 64 multiplicative and 64 additive coefficients, conditioned on the target category.

History and scalar context. The context $u_t \in \mathbb{R}^{130}$ consists of two scalar features followed by 16 history records, ordered from oldest to newest. Each record contains eight values: a four-dimensional one-hot action, a collision indicator, two position coordinates relative to the current agent, and a validity indicator. The relative position is rotated into the current agent frame and

divided by 50 cells (2.5 m). Missing records are filled with zeros. The two scalar features are $20\ \mathrm{mean}(E_t)$, where E_t is the binary explored-space channel of M_t, and $1 - t/T_{\max}$, the remaining episode fraction; $T_{\max}$ is the episode step limit. The resulting dimension is $2 + 16 \times 8 = 130$. This 16-step input history is separate from the 20-transition reward-credit window.

The fusion module in Equation 2 maps the 130-dimensional context to 64 dimensions, concatenates it with the 512-dimensional map feature, and projects the resulting 576-dimensional vector to a 512-dimensional GRU input. The GRU state is also 512-dimensional. Its heads output four action logits, one value, and a two-component auxiliary goal displacement. The auxiliary distance head is counted in the 12.08M policy parameters, but its loss is disabled in the exploration configuration.

B.2 Training Settings and Budget Accounting

Training-sample counts. We count transitions processed during optimization, including repeated use of a transition. Thus, the training-sample budget differs from the number of newly collected environment interactions. Completed episodes are accumulated until a transition threshold is reached. Their transitions are shuffled into minibatches for a fixed number of optimization iterations. Because completed episodes can overshoot the sampling threshold, the effective ratio of processed to collected transitions varies. Table B.1 summarizes the counts and GPU assignments. Each BC/OPD stage processes 500M transitions per GPU: 500M from TensorEnv and 1,000M from ParHab, totaling 1,500M. Table B.1 describes the complete procedure. The component ablations in Table 2 use the same total training budget and evaluate the final policy of each configuration. The floorplan-count study in Figure 3 keeps each stage budget fixed and changes only the number of distinct TensorEnv training floorplans; all settings use the same MapAct architecture and evaluation protocol.

Table B.1: Training budgets on A800 GPUs. M denotes one million processed transitions, including reuse.

Stage	Training amount	GPU allocation
1: SL (ParHab)	12,360 updates	1 GPU; batch 17,408
2: Exploration (ParHab)	1,000M	2 GPUs; batch 512 per GPU
2: Teacher RL (TensorEnv)	1,000M	2 GPUs
3: BC + ObjectNav RL	1,500M	1 TensorEnv + 2 ParHab GPUs
4: OPD + ObjectNav RL	1,500M	1 TensorEnv + 2 ParHab GPUs

Budget allocation in component ablations. The ablations compare final policies at the same total optimization-sample budget. When a stage or environment-specific training stream is removed, its budget is redistributed among the retained components in proportion to their allocations in the complete procedure. Formally, let b_i denote the original allocation of component i, $\mathcal{I}$ the retained components, and B_{tot} the common total budget. The ablated allocation is

$$b_i^{\mathrm{abl}} = B_{\mathrm{tot}} \frac{b_i}{\sum_{j \in \mathcal{I}} b_j}, \qquad i \in \mathcal{I}. \tag{B.1}$$

This preserves the relative allocations of retained components while matching the total budget. Skipping BC removes its training stage; removing OPD supervision retains the stage's RL training; removing continued ParHab RL removes that stream during BC and OPD. The comparison therefore evaluates each modified training procedure after its full allocation, rather than comparing intermediate stage checkpoints. Equal sample budgets do not imply equal GPU time or newly collected interactions.

Direct student PPO in TensorEnv without BC/OPD. This ablation compares teacher-model supervision with direct student-model interaction on the same 2D maps under matched training budgets. Stages 1–2, including goal-free exploration by the student model in ParHab, remain unchanged. In stage 3, student-model PPO in TensorEnv replaces BC on teacher-model trajectories. In stage 4, OPD supervision is removed and the student model learns directly through TensorEnv PPO. The original goal-conditioned ParHab PPO remains active in both stages. Each stage retains its 1,500M processed-transition budget and original allocation: 500M from TensorEnv and 1,000M from ParHab. Thus, the student model uses the same budget to learn from 2D interaction without BC/OPD supervision; neither stage is removed or its budget redistributed. The final policy is evaluated on HM3D with the same protocol as the other ablations.

Reverse-direction transfer variants. Both approximately 61% conditions use a ParHab 3D teacher to supervise a TensorEnv 2D student, so that the 2D student can learn behavior relevant to 3D navigation. The BC-only condition removes OPD supervision while retaining the accompanying RL updates; its SR values are 60.75%, 60.90%, and 61.05%, with an aggregate of 60.90% SR and 33.09% SPL. The corresponding BC+OPD condition reaches 61.02% SR and 33.28% SPL. Both remain below the full framework's 68.77% SR, indicating that this reverse transfer is ineffective under the tested setup.

Scale-study aggregation. For the floorplan-count study, we aggregate per-run SR values after rounding to the nearest 0.05 percentage points, the resolution of 2,000 equally weighted binary-success episodes. SPL is aggregated without this rounding.

Initialization and collection. SL uses shortest-path supervision in ParHab. Its learning rate decreases linearly from 10^{-2} to 3×10^{-5} and then remains constant; the reported initialization takes approximately 95 minutes on one A800. For stage-2 ParHab exploration, the 80 HM3D training scenes are divided evenly across two GPUs. This stage uses no navigation goal, masks STOP, and collects 500-transition episodes. Each GPU separately accumulates at least 25,000 completed-episode transitions before the synchronized update. Each update uses 100 optimization iterations with an aggregate minibatch of 1,024 transitions, split as 512 per GPU. At exactly 25,000 collected transitions per GPU, the processed-to-collected ratio is $100 \times 512/25{,}000 = 2.048$; episode-level overshoot lowers this ratio. Exploration uses Adam at a constant learning rate of 10^{-6}, without warmup or decay, and clips the gradient ℓ_2 norm at 0.5.

Teacher supervision. In stages 3–4, student-model training in ParHab switches to goal-conditioned ObjectNav. BC uses action labels from successful teacher-model trajectories. OPD uses the soft-target objective in Appendix B.3 at states visited by the student model and also performs TensorEnv PPO. The student model continues ParHab PPO during both stages. After collection, the three GPUs combine their minibatch gradients in a synchronized student-model update. The teacher model remains frozen. The TensorEnv process uses BC supervision in the BC stage and combines OPD supervision with RL in the OPD stage; the two ParHab processes use RL in both stages. The gradients from the one TensorEnv process and two ParHab processes are averaged before the Adam update; this cross-process average is separate from the weighting of objectives within each process.

Teacher evaluation. The two-dimensional evaluation uses 1,000 held-out maps with ten tasks per map, totaling 10,000 tasks. The result in Table 3 aggregates this evaluation for three independently trained teachers. These 1,000 maps are held out from policy training. Teacher evaluation uses the corresponding ObjectNav goal-category set, a 500-action limit, and STOP-based success in the valid goal region, following the same task convention as the three-dimensional evaluation. Distances and path lengths are evaluated within the two-dimensional environment.

Dataset-specific semantic prediction. Separate YOLO26n models are trained for HM3D and MP3D using Habitat-rendered RGB views and simulator semantic labels, with dataset-specific category mappings. Camera positions and orientations are sampled exclusively from the corresponding training scenes, following NaviScale (Lan et al., 2026); validation and test scenes are excluded.

Inherited data and observation settings. We use the observation settings documented by NaviScale (Lan et al., 2026): a 79° field of view, a 5-m maximum perception range, and a camera height of 0.88 m. TensorEnv uses the same field of view and maximum range. The RGB and depth observations are 640×480 pixels, with depth configured over 0.5–5.0 m. Forward motion is 0.25 m and turns are 30°. Semantic maps have 5-cm cells. The MP3D environment maps 21 semantic categories, matching its 21 goal categories. NaviScale comprises 192,000 2D semantic maps from 24,000 floorplans, with eight room compositions per floorplan; its source room maps come exclusively from the HM3D and MP3D training splits. These corpus-level counts describe the source dataset, rather than the size of the training subset used in an individual run.

Recurrent-state optimization. During collection, the pre-action GRU state h_{t-1} is detached and stored in FP32 with its observation. Training shuffles these paired records together and supplies the

saved state as the GRU input. Gradients propagate through the current transition only; no backpropagation through a sequence of earlier transitions is performed. Recurrent memory is nevertheless maintained across decisions during rollout collection.

Three-dimensional success criterion. The Habitat-Lab 0.3.3 ObjectNav task measures geodesic distance to valid goal viewpoints. A successful episode requires STOP and a distance below 0.1 m to this viewpoint set. This threshold is a distance to valid viewpoints, not to the object's center. The same goal-distance definition determines the shortest-path term in SPL.

B.3 On-Policy Distillation Objective

Let $o_t = (M_t, \xi_t, g, u_t)$ denote the shared observable input along a trajectory generated by the student model in TensorEnv. The teacher model and student model maintain separate hidden states:

$$h_t^T = F_T(o_t, h_{t-1}^T), \qquad h_t^S = F_\theta(o_t, h_{t-1}^S), \tag{B.2}$$

where F_T and F_θ denote the teacher-model and student-model versions of the map encoding, fusion, and GRU update in Equation 2. Both recurrences follow observations and executed actions from the student-model trajectory and reset at episode boundaries. The teacher model is frozen, but its hidden state evolves along this trajectory; it does not use the student's hidden state. Define $x_t^T = (o_t, h_{t-1}^T)$ and $x_t^S = (o_t, h_{t-1}^S)$. With temperature one, the four-action distributions and distillation loss are

$$\begin{aligned} p_T(a \mid x^T) &= \mathrm{softmax}(\ell_{\theta_T}(x^T))_a, \\ p_\theta(a \mid x^S) &= \mathrm{softmax}(\ell_\theta(x^S))_a, \\ \mathcal{L}_{\mathrm{OPD}}(\theta) &= -\mathbb{E}_{(x^T, x^S)\sim d_\theta} \sum_{a=1}^{4} p_T(a \mid x^T) \log p_\theta(a \mid x^S). \end{aligned} \tag{B.3}$$

Here d_θ is the distribution of paired inputs collected along trajectories generated by the student model. Teacher-model probabilities and saved recurrent inputs are fixed during each optimization update. The student-model gradient equals that of forward KL divergence $D_{\mathrm{KL}}(p_T \| p_\theta)$, since teacher-model entropy is independent of student-model parameters. TensorEnv PPO and continued ParHab PPO accompany distillation (Section 3.3).

B.4 Exploration Policy Objective

The following objective and optimizer settings specify goal-free ParHab exploration in stage 2. Stages 3–4 instead train goal-conditioned ObjectNav policies.

Exploration-credit allocation. ParHab exploration retains an immediate reward for newly observed area. Earlier actions that move the agent toward a new region may receive no immediate exploration reward if the region becomes visible only later. We therefore add exploration credit to successful forward actions within a window of 20 transitions, including the discovery transition itself.

Let p_j and p_{j+1} be the positions before and after action a_j. A discovery at transition t reveals area ΔA_t with spatial centroid c_t. Its candidate window is $\mathcal{W}_t = \{\max(0, t-19), \ldots, t\}$. Let m_j indicate that a_j is a forward action and its displacement meets the minimum movement threshold. We measure its progress toward the newly discovered region by

$$w_{j,t} = m_j \left[\|p_j - c_t\|_2 - \|p_{j+1} - c_t\|_2 \right]_+, \qquad j \in \mathcal{W}_t, \tag{B.4}$$

where $[v]_+ = \max(v, 0)$. Thus, a rotation, a failed forward movement, or a movement away from the discovery centroid receives no additional credit for that discovery. The normalized allocation is

$$\alpha_{j,t} = \begin{cases} \dfrac{w_{j,t}}{\sum_{k\in\mathcal{W}_t} w_{k,t}}, & \displaystyle\sum_{k\in\mathcal{W}_t} w_{k,t} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{B.5}$$

Numerically negligible weight totals are treated as zero. With the precomputed teacher-exploration area for the floor, $A_{\text{ref}} > 0$, the additional reward for transition j is

$$b_j = \frac{\lambda_{\text{exp}}}{A_{\text{ref}}} \sum_{t=j}^{\min(j+19,T-1)} \Delta A_t \alpha_{j,t}, \qquad r_j = r_j^{\text{imm}} + b_j + r_j^{\text{other}}, \tag{B.6}$$

where λ_{exp} controls the additional credit, r_j^{imm} is the immediate exploration reward, and r_j^{other} contains the motion rewards and penalties specified below. The bonus b_j supplements the immediate reward without subtracting or replacing it. Area measurements and the reference use consistent units. If a discovery has no positively weighted eligible movement, its additional credit is zero. Rewards are assigned after rollout collection and before training returns are computed; the policy still selects each action using only the information available at that step. Exploration uses a PPO-style clipped surrogate (Schulman et al., 2017) weighted by scaled discounted returns, with the reward coefficients and optimization settings specified below.

Exploration rewards. The coefficient in Equation B.6 is $\lambda_{\text{exp}} = 6.40365$, normalized by the precomputed teacher-exploration area for the corresponding floor. The immediate exploration reward is $r_j^{\text{imm}} = 0.2\Delta N_j/2500$ for non-turn actions and zero for turns, where ΔN_j counts newly observed 5-cm cells at transition j. Successful forward motion receives $+0.002$, collision -0.08, left–right alternation -0.02, a triggered consecutive-turn penalty -0.02, and stagnation -0.05. The alternation and consecutive-turn penalties are separate terms and can both apply.

Action distribution. Let $\mathcal{A}_t$ be the allowed actions, excluding STOP during exploration and excluding forward motion when it is blocked. The masked policy distribution π_θ is combined with a fixed prior,

$$q_\theta(a \mid x_t) = 0.8\pi_\theta(a \mid x_t) + 0.2p_t(a), \qquad p_t(a) = \frac{w_a \mathbf{1}[a \in \mathcal{A}_t]}{\sum_{b \in \mathcal{A}_t} w_b}, \tag{B.7}$$

where $w = [0, 3, 1, 1]$ follows the order STOP, forward, left, right. Both components are normalized over allowed actions. The policy ratio and entropy are computed from q_θ, with probabilities clamped below at 10^{-12} before logarithms.

Returns and policy weights. For a completed trajectory, discounted returns satisfy

$$R_T = 0, \qquad R_t = r_t + 0.99R_{t+1}, \qquad \widehat{A}_t = \frac{R_t}{s_R + 10^{-8}}, \tag{B.8}$$

where s_R is the sample standard deviation, with denominator $n - 1$, over the returns participating in the local loss. The policy weight $\widehat{A}_t$ uses the return directly: the implementation neither subtracts the value prediction nor centers the returns, and does not use generalized advantage estimation in this configuration. Returns and policy weights are held constant during differentiation.

Clipped surrogate and auxiliary value prediction. For the recorded action a_t, define

$$\rho_t = \exp\{\text{clip}[\log q_\theta(a_t \mid x_t) - \log q_{\text{old}}(a_t \mid x_t), -20, 4]\}, \tag{B.9}$$

where q_{old} supplies the fixed reference probabilities. The minimized objective is

$$\begin{aligned} \mathcal{L}_{\text{expl}} = & -\text{mean}_t\Big[\min\Big(\rho_t \widehat{A}_t, \text{clip}(\rho_t, 0.8, 1.2)\widehat{A}_t\Big)\Big] \\ & + 0.25\,\text{mean}_t[(V_\theta(x_t) - R_t)^2] - 0.003H(q_\theta), \\ H(q_\theta) = & -\text{mean}_t \sum_{a \in \mathcal{A}_t} q_\theta(a \mid x_t) \log q_\theta(a \mid x_t). \end{aligned} \tag{B.10}$$

The value term is half the mean squared error weighted by 0.5, yielding 0.25 overall; value predictions are unclipped and do not provide a policy baseline. The log-ratio bound is a numerical safeguard separate from PPO surrogate clipping. The action-mixture weight and clipping parameter both equal 0.2 but serve distinct roles. No BC, distance-head, explicit KL, or parameter-norm penalty is added.

C Exploration Ablations

We examine initialization, exploration-credit allocation, and memory in ParHab. All credit-allocation variants retain the immediate exploration reward and differ in how the additional area bonus is assigned. The no-redistribution variant assigns that bonus to the discovery transition. The metric is explored area in square meters, with training progress measured in millions of processed steps. Each configuration contributes one training run. Figure 5 plots all 1,001 recorded values per series without smoothing or averaging across runs.

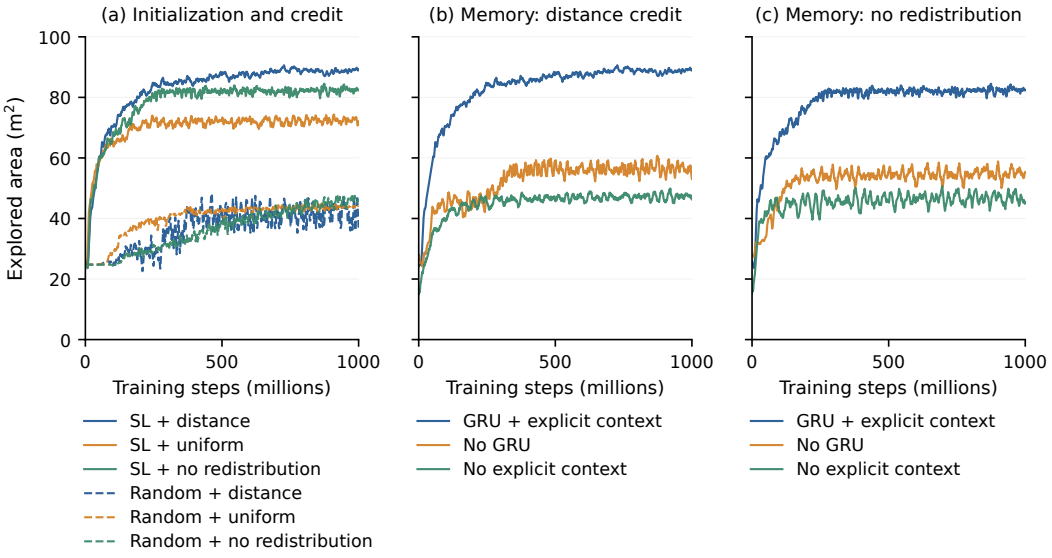


Figure 5: Exploration-specific ablations. (a) SL initialization versus random initialization under distance-based credit, normalized uniform allocation, and no redistribution. (b,c) Removal of the GRU or explicit interaction context under distance credit and no redistribution. Explicit context includes recent interaction history and scalar features. The complete-model traces in (b,c) repeat their corresponding traces from (a); there are ten distinct series in total.

With SL initialization, the final recorded explored areas are 89.28 m^2 for distance credit, 82.36 m^2 without redistribution, and 72.04 m^2 for normalized uniform allocation. Distance credit thus exceeds the latter two endpoints by 6.92 and 17.24 m^2, respectively. Its benefit depends on initialization in these records: with random initialization, distance credit ends at 36.96 m^2, below no redistribution (46.50 m^2) and uniform allocation (44.21 m^2).

Under distance credit, removing the GRU reduces the final recorded area from 89.28 to 53.05 m^2, and removing explicit context reduces it to 46.27 m^2. Without redistribution, the corresponding endpoints are 82.36, 55.34, and 45.46 m^2. Both forms of memory contribute to exploration in these runs.

D REAL-WORLD EVALUATION

We evaluate MapAct and the released models of PEANUT, NaviScale, and PIRLNav on the same 10 tasks. The robot platform is Hiwonder JetAuto with RGB-D sensing and RTAB-Map. MapAct and PIRLNav execute locally on the edge device, an NVIDIA Jetson Nano with 4 GB memory. PEANUT and NaviScale execute on a remote NVIDIA RTX 3090 GPU and communicate with the robot over the network. The goal categories are chair, bed, plant, toilet, television, and sofa, following the HM3D task setting.

Table D.1: Real-world success on the same 10 tasks. Baselines use their released models. Deployment configurations are reported alongside the task outcomes.

Method	Deployment	Successful tasks
PEANUT	RTX 3090, network connection	5/10
NaviScale	RTX 3090, network connection	6/10
PIRLNav	Local Jetson Nano (4 GB)	0/10
MapAct (ours)	Local Jetson Nano (4 GB)	**8/10**

MapAct succeeds on eight tasks under local execution. In the PIRLNav trials, movement showed no clear goal-directed pattern and was accompanied by premature stopping. These observations illustrate a transfer difficulty for the released policy in this physical setting; they do not isolate training distribution from perception, control, or deployment effects.

Qualitative example. Figure 4 shows six observations from a 49-frame autonomous bed-search recording. The sequence was recorded after resetting RTAB-Map; its first saved observation follows initialization and therefore already contains mapped geometry.

Semantic memory was accumulated chronologically from the recorded per-frame projections, while the original occupancy grids were retained. Map cells measure 5 cm, and the trajectories and heading arrows are drawn from recorded positions and orientations. The visualization illustrates how observations accumulate in the map; Table D.1 reports task success separately from the simulation benchmarks.

E Additional Comparisons and Reproduction Notes

Baseline scores in Table 1. The HM3D scores for PEANUT are from our reproduction on 2,000 validation episodes, reported in NaviScale (Lan et al., 2026). The original PEANUT paper reports 63.8/32.0 on 500 episodes; its MP3D scores in Table 1 are taken from the original paper. A dash in Table 1 denotes an unavailable result.

Table E.1 places published HM3D scores alongside our evaluation and independent GitHub reports. Dataset versions and evaluation status are specified because they affect the interpretation of these results. We discuss PIRLNav retraining separately below.

Table E.1: Published scores and reproduction reports for Uni-NaVid and XGX. Each pair denotes SR/SPL (%); a dash denotes an unavailable value, and NaN denotes undefined SPL. The independent XGX reports include an incomplete evaluation and a possible mismatch between scene and episode versions, as detailed below.

Method	Original paper	Our evaluation	Independent GitHub report
Uni-NaVid RSS 2025	73.7 / 37.1	56.9 / 27.5 HM3D v0.1	Issue #14: 56.9 / — (v0.1); 70.2 / — (v0.2)
XGX ICRA 2024	72.9 / 35.7	57.1 / 28.9	Issue #5: 35.48 / 14.03 (v0.2; 809/1,000 episodes); follow-up: 57.15 / NaN (v1 task episodes)

Uni-NaVid: our evaluation and an independent report. Our evaluation of the released policy on HM3D v0.1 yields 56.9% SR and 27.5% SPL. The SR matches the independent v0.1 result in Issue #14. The reporter also obtains 70.2% on v0.2 after configuration changes and a switch from an RTX 3090 to an A800, and considers the issue resolved. The authors explain that licensing restrictions required replacing the ObjectNav training trajectories in the public release. The HM3D version used for the published score remains unverified; these dataset and release differences limit direct comparison with that score.

PIRLNav: our retraining and independent reports. The original CVPR 2023 paper reports two training scales on HM3D VAL: $61.9 \pm 0.47\%$ SR and $27.9 \pm 0.56\%$ SPL with 20K human demonstrations, and 70.4% SR and 34.1% SPL with 77K demonstrations (Ramrakhya et al., 2023, Tables 1–2 and Sections 4.3, 5.3). Table 1 uses the 20K setting; the 77K result is also listed in the official repository. Using the public training code and released training data, our retraining attempt yielded an SR of 0–1%, never exceeding 1% during the training we conducted. Independently, Issue #10 reports near-zero success after self-training, and another user reports the same difficulty without a posted solution. These reports describe a retraining difficulty consistent with our observation. The original reporter also states that the provided checkpoints performed well, distinguishing retraining from checkpoint evaluation. However, both policy-weight links in the official README, including the RL-finetuned checkpoint, returned HTTP 404 (`NoSuchBucket`) when checked on September 26, 2026. Download failures are also reported in Issue #11 and subsequent comments on Issue #10.

XGX: our evaluation and independent reports. Our reproduction yields 57.1% SR and 28.9% SPL, compared with the published 72.9% SR and 35.7% SPL. Independently, the first result in Issue #5 was recorded after 809 of 1,000 HM3D v0.2 episodes; the run subsequently encountered infinite goal distances and undefined SPL. The follow-up uses v1 task episodes and again reports undefined SPL. Its author suspects that the episodes were paired with v2 scene assets, whereas the repository specifies HM3D-V1 scenes and episodes. No corrected complete evaluation is posted. These reports document difficulties with the public evaluation pipeline; the possible version mismatch and invalid metrics prevent treating them as matched evaluations of the published result.