# Exploitation-Guided Exploration for Semantic Embodied Navigation

**Justin Wasserman**[1,2] **Girish Chowdhary**[1] **Abhinav Gupta**[2] **Unnat Jain**[2,3]
[1] University of Illinois at Urbana-Champaign
[2] Carnegie Mellon University [3] FAIR at Meta

## Abstract

In the recent progress in embodied navigation, modular policies have emerged as a de facto framework. However, there is more to compositionality beyond the decomposition of the learning load into modular components. In this work, we investigate a principled way to syntactically combine these components. Particularly, we propose Exploitation-Guided Exploration (XGX) where separate modules for exploration and exploitation come together in a novel and intuitive manner. We configure the exploitation module to take over in the deterministic final steps of navigation *i.e.* when the goal becomes visible. Crucially, an exploitation module teacher-forces the exploration module and continues driving an overridden policy optimization. XGX, with effective decomposition and novel guidance, improves the state-of-the-art performance on the challenging object navigation task from 70% to 73%. Finally, we show sim-to-real transfer to robot hardware and XGX performs over two-fold better than the best baseline from simulation benchmarking. Project page: XGXvisnav.github.io

## 1 Introduction

Consider the 'planning' problem you underwent when finalizing your last vacation. There is usually plenty of uncertainty in the decision-making – where to go, is the weather good, managing the budget, booking ground transport, *etc*. We tackle this uncertainty by breaking the problem down into sub-parts and then offload some of these sub-parts to specialists or expert websites. However, there is an 'additional order' that goes beyond offloading sub-parts to these reliable and modular solutions. The very fact that you know these modular solutions exist helps you make more ambitious plans, be more creative, and likely travel more often and better than without this information. Generally speaking, division of work or *modularity is only one aspect of a compositional approach*. The very awareness of our repertoire of skills or (associated modules) helps manage resources to plan uncertain aspects of planning better. This intuition of our day-to-day intelligence guides our alternate take on visual robot navigation or *embodied navigation*. We believe there is more to be tapped in compositionality beyond policy decomposition. To this end, we investigate the research question: *"Can modular navigation agents learn better exploration when guided by their exploitation abilities?"*

Given that we as a research community have already built most of the blocks, the investigation of this research question is rather intuitive and straightforward! For this study, we focus on the semantic visual navigation task of object-goal navigation [3] where the agent's objective is to navigate to a goal category (akin to "find a chair"), which is a standardized and reproducible benchmark [39]. Our policy *decomposition* employs a state-of-the-art neural policy [31] for the exploration module. For the exploitation module, we devise a simple-and-effective geometric policy for visuo-motor servoing the exploitation phase (steps after the goal is in view). Contrary to prior works, in XGX, the exploitation module provides *guidance* to the exploration module (implemented as teacher-forced [44] variation

NeurIPS 2023 Workshop on Robot Learning: Pretraining, Fine-Tuning, and Generalization with Large Scale Models, New Orleans, United States of America

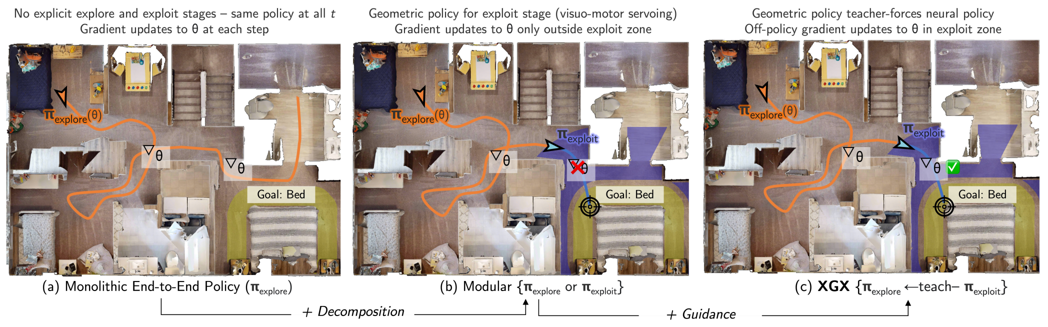

No explicit explore and exploit stages – same policy at all $t$
Gradient updates to θ at each step
Geometric policy for exploit stage (visuo-motor servoing)
Gradient updates to θ only outside exploit zone
Geometric policy teacher-forces neural policy
Off-policy gradient updates to θ in exploit zone
$\pi_{\text{explore}}(\theta)$
$\nabla\theta$
$\pi_{\text{exploit}}$
Goal: Bed
(a) Monolithic End-to-End Policy ($\pi_{\text{explore}}$)
(b) Modular {$\pi_{\text{explore}}$ or $\pi_{\text{exploit}}$}
(c) **XGX** {$\pi_{\text{explore}}$ ←teach– $\pi_{\text{exploit}}$}
*+ Decomposition*
*+ Guidance*

Figure 1: **Overview of XGX.** Within learning-based visual navigation, (a) most prior works adopt a neural policy, trained end-to-end. (b) Some works employ a modular policy, with a dedicated modules; we devise an effective decomposition in XGX. (c) Unlike prior work, XGX enables the exploitation module to guide exploration module via off-policy updates and teacher forcing.

of the Proximal Policy Optimization). Note, prior works tackling vision-based navigation via modular policy *independently* optimizes modules, with no guidance or interplay in loss objectives. Fig. 1 illustrates these two pillars of XGX– *decomposition*, and *guidance*.

## 2 Exploitation-Guided Exploration

As visualized in Fig. 1, the key components of Exploitation-Guided Exploration methodology are the exploration module ($\pi_{\text{explore}}$), phase transition to the exploitation module ($\pi_{\text{exploit}}$), and optimization of the exploration module via guidance from the exploitation module.

### 2.1 XGX Modules

**Exploration Module ($\pi_{\text{explore}}$):** The architecture choice for the exploration module $\pi_{\text{explore}}$ is orthogonal to our novelty of XGX (decomposition and guidance). To this end, we adopt the best-performing available design – PIRLNav by Ramrakhya *et al.* [31]. Please see Sec. 7.1 for more details on the architecture and training regime.

**Phase Transition** : Intuitively, we check if the agent can 'see' the goal and is close to it. If so, phase transition from exploration module to exploitation module should triggered. Specifically, if semantic segmentation contains the goal category and the agent is within a prescribed distance $\delta$ from the goal, the control transitions to the exploitation module. The set of states for which this transition will occur can be more mathematically described by the set:

$$\mathcal{S}^{j}_{\text{exploit}} \triangleq \{(s) \in \mathcal{S} : \text{dist}_j(s) \leq \delta, \;\; O_j \in f_{\text{semantic}}(s)\}, \tag{1}$$

where $j$ is an index corresponding to the object-goal category labels $\{O_1, \ldots, O_n\}$, and $f_{\text{semantic}}$ an off-the-shelf semantic segmentation model. In this set, which we call the 'exploitation states', the agent employs simple and effective visuo-motor servoing *i.e.* our exploitation module, detailed next.

**Exploitation Module ($\pi_{\text{exploit}}$):** Once the goal is in sight, *i.e.*, within exploitation state, visuo-motor servoing is both simple and effective. To this end, our exploitation module $\pi_{\text{exploit}}$ can triangulate the goal, navigating to it, and executing the 'stop' action. Further details of $\pi_{\text{exploit}}$ are given in Sec. 7.2.

### 2.2 Exploitation Module Teacher-Forcing (Guidance)

The objective of a policy-gradient approach like XGX with PPO is to learn a policy towards maximizing a discounted, cumulative reward. The surrogate loss objective from PPO [36] is as follows:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}}\Big[\min(p_t(\theta)\hat{A}_t(\theta), \text{clip}\big(p_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t(\theta)\Big], \tag{2}$$

where $\hat{A}_t(\theta)$ is the advantage function, $\hat{V}_t(\theta)$ is the value estimate from the critic head, $p_t(\theta)$ is ratio function to correct for stale policy used for rollouts, and `clip` is performed to prevents too large of a gradient. The replay buffer of rollouts is denoted by $\mathcal{D}$ consisting of $(s_t, a_t, r_t)$ tuples for state, action, and environment rewards. The modification that allows $\pi_{\text{exploit}}$ to guide $\pi_{\text{explore}}$ is rooted in the

| # | Method | Type | SR ↑ | SPL ↑ |
|---|---|---|---|---|
| 1 | DD-PPO [43] | RL | 27.9 | 14.2 |
| 2 | OVRL [47] | SSL→RL | 62.0 | 26.8 |
| 3 | OVRL-V2 [46] | SSL→RL | 64.7 | 28.1 |
| 4 | RRR [34] | Modular TL | 30.0 | 14.0 |
| 5 | Frontier Based Expl. [48, 16] | Classical | 26.0 | 15.2 |
| 6 | Habitat-Web [32] (paper) | IL | 57.6 | 23.8 |
| 7 | Habitat-Web [32] (our impl.) | IL | 64.1 | 27.1 |
| 8 | PIRLNav [31] (paper) | IL→RL | 61.9 | 27.9 |
| 9 | PIRLNav [31] (our impl.) | IL→RL | 70.4 | 34.1 |
| 10 | XGX (ours) | IL→RL | **72.9** | **35.7** |
| RL: reinforcement, IL: imitation, SSL: self-supervised, TL: transfer | | | | |

(a) Quantitative results for navigation on ObjectNav.

| # | Method | SR ↑ | SPL ↑ |
|---|---|---|---|
| 1 | Init. with IL on Human Demos + Fine-Tune [31] | 70.4 | 34.1 |
| 2 | 1 + Policy Decomposition | 71.5 | 34.3 |
| 3 | 1 + Policy Decomposition + Guidance (*i.e.* XGX) | **72.9** | **35.7** |
| 4 | XGX + GT Semantics *(upper bound)* | 73.5 | 39.8 |

(b) Head-on ablations for XGX.

Table 1: **(a)** XGX outperforms prior works based on IL, RL, and SSL. An equivalent jump in SPL is much harder than the same jump in success rate, indicating a significantly more efficient planning by XGX. **(b)** We observe gains in performance by including the exploitation module $\pi_{\text{exploit}}$ in our policy decomposition and adding *guidance* to neural exploration module. ↑ denotes higher is better.

ratio function. In standard PPO, the ratio function is $p_t(\theta) = \frac{\pi(a_t|s_t;\theta)}{\pi(a_t|s_t;\theta^-)}$, where $a_t \sim \pi(a_t|s_t;\theta^-)$ and $\theta^-$ is the stale parametrization of policy used to collect rollouts in $\mathcal{D}$. In XGX, actions in the replay buffer ($a_t$) are instead sampled from a teacher-forced policy. Concretely:

$$a_t = \begin{cases} a_t^{\text{exploit}} \sim \pi_{\text{exploit}}(a|s) & \text{if} \quad s \in \mathcal{S}_{\text{exploit}} \\ a_t^{\text{explore}} \sim \pi_{\text{explore}}(a|s;\theta^-) & \text{otherwise} \end{cases}, \tag{3}$$

where $\pi_{\text{explore}}$, $\mathcal{S}_{\text{exploit}}$, and $\pi_{\text{exploit}}$ were introduced in Sec. 2.1, Sec. 2.1, and Sec. 2.1, respectively. The value function relies on behavior policy as well. Since we employ the above teacher-forced policy, XGX value (and thereby advantage) estimates are also different. Notably, with $\pi_{\text{exploit}}$ employed in Eq. (3), the time horizon of the RL formulation shortens significantly. This allows more effective learning from sparse success cues to optimize the parameters $\theta$. Exploitation module $\pi_{\text{exploit}}$ is specialized for getting to the goal, it increases the reward the agent receives overall during training. Another advantage of the decomposition in Eq. (3), also allows for back-tracking, *i.e.*, if the agent leaves 'exploitation states' $\mathcal{S}_{\text{exploit}}$, the agent returns to following the exploration module.

## 3 Experiments

In this section, we test our XGX methodology across (1) large-scale photorealistic simulation across, (2) physical robot runs across three diverse real-world scenes. We also present results on the MiniGrid [11] environment in Sec. 8 where XGX outperforms PPO across multiple tasks. We experiment with Exploitation-Guided Exploration on the semantic visual navigation task of Object-Goal Navigation or ObjectNav [3, 2]. We adopt the protocol laid out for AIHabitat [3] which has been standardized as a public benchmark as well [39]. At the start of an ObjectNav episode, an embodied agent is initialized at a random location and is tasked to navigate to a goal category. At every time step the agent can choose an action from the action space $\mathcal{A} := \{$*move forward, turn right, turn left, stop, look down, look up*$\}$. The episode is considered a success if the agent can navigate within 1 meter of an instance of the goal category and execute the *stop* action. Consistent with prior work, we too adopt a sparse reward structure that returns a 1 on success and a 0 otherwise, at every time step. We include further details on the task, sensors, and metrics in Sec. 9.

### 3.1 Methods

We benchmark across a diverse set of baselines spanning purely reinforcement learning, imitation learning, self-supervised representation learning, transfer learning, and classical baselines:
• DDPPO [43]: RL baseline that employs proximal policy optimization in a distributed manner.
• OVRL [47]: OVRL pretrains a modified ResNet50 [19] head on the Omnidata Starter Dataset [14] with self-supervised learning. Then finetunes the encoder and the policy on top using 500M frames.
• OVRL-V2 [46]: Similar to OVRL, but the choice of the encoder is changed to vision transformers (ViT) [13] with a compression layer through a masked auto-encoding [18].
• Habitat-Web [32]: Purely IL on ∼80k human-collected episodes using a simulation-web interface.
• PIRLNav [31]: Going a step further from Habitat-Web baseline, this adopts a two-stage training regime – imitation learning for 500M steps followed by PPO updates for 300M frames.

• RRR [34]: Modular policy with a zero-shot transfer learning from an HM3D point-goal module.
• FBE [48, 16]: Frontier-Based exploration while incorporating semantics; adopted by Gervet *et al.* [16]. We report results on full HM3D-Val split (not just environments with a single floor [16]).

### 3.2 Results and Analysis (Photorealistic Simulation)

Metrics from empirical runs in AIHabitat are reported in Tab. 1a and Tab. 1b. We discuss these results and analysis in the following text. As detailed in Sec. 3, all results are reported on the HM3D validation scenes (never seen during training).

**XGX improves over all prior IL, RL, and SSL baselines (Tab. 1a).** Compared with diverse baselines (Sec. 3.1), XGX (in row 10) demonstrates significant gains, improving $70.4 \rightarrow 72.9$ in success rate (relative $4\% \uparrow$) over the previous best IL+RL method – PIRLNav [31] (in row 9). The best SSL method, OVRL-V2 [46] (in row 3), is fairly competitive in success rate (64.7 *vs*. 72.9) but significantly lags behind on SPL (28.1 *vs*. 35.7).

**Both the exploitation module and guidance are helpful (Tab. 1b).** Two key aspects of our proposed approach are (1) exploitation module: based on basic visuo-motor servoing and geometric computer vision and (2) guidance from this module to exploration. Here we undertake head-on ablations to quantitatively evaluate the utility of both in Tab. 1b. We observe improvements of $70.4 \rightarrow 71.5 \rightarrow 72.9\%$ (row $1 \rightarrow 2 \rightarrow 3$) in success rate by successively adding the policy decomposition and guidance to the initialization. These trends are more pronounced for the efficiency metric of SPL.

### 3.3 Results and Analysis (Physical Robot)

We sim-to-real transfer PIRLNav and our XGX on a wheeled navigation robot employed by navigation works [20, 15, 22, 42]. We navigate to five goal categories {couch, TV, chair, toilet, potted plant} across 14 trajectories per method. Further details about the environments are given in Sec. 11.

| | Method | SR ↑ | SPL ↑ |
|---|---|---|---|
| 1 | PIRLNav (released ckpt) | 0.0 | 0.0 |
| 2 | PIRLNav w/ our adaptation | 14.2 | 8.4 |
| 3 | XGX (ours) | **35.7** | **23.3** |

Table 2: **Real world object-goal navigation results.** In real world robotics experiments, XGX performs the best.

**Accurate embodiment reduces the sim-to-real gap (Tab. 2, rows 1 & 2).** The best-performing prior method, PIRLNav, does not demonstrate successful behaviors on the robot (see row 1, Tab. 2). We adapt PIRLNav (see row 2) by retraining it with a robot-specific configuration (height and FoV) as the default simulator configuration is different. This improved navigation behavior is consistant with a recent real world study [16].

**XGX improves the performance of the robot as well (Tab. 2, row 3).** In Tab. 2, we find that XGX empirically has the best performance in physical robot experiments as well. Comparing rows 2 and 3, we observe an improvement from $8.4\% \rightarrow 23.3\%$ in SPL.

## 4 Conclusion

In this work, we ask a fundamental question: "Can decomposed navigation agents learn better exploration when guided by their exploitation abilities?" To this end, we introduce XGX which utilizes teacher forcing implemented via off-policy updates. We deploy XGX with standard choices of parametric exploration and simple geometric exploitation modules, leading to state-of-the-art performance in simulation and an over two-fold improvement in physical robotics experiments.

## 5 Acknowledgments

We express our gratitude to the reviewers for their valuable suggestions that have significantly strengthened this work. JW and GC acknowledge the support provided by grant 1-473647-320010-191100-A01. Special thanks to Aaron Havens for his insightful feedback on our mathematical notation. We extend our appreciation to Mateus Valverde and EarthSense for their invaluable assistance with the robotics platform. Lastly, we want to express our sincere thanks to everyone who contributed feedback on our submission draft - Theophile Gervert and Sonia Raychaudhuri.

## References

[1] Ziad Al-Halah, Santhosh Kumar Ramakrishnan, and Kristen Grauman. Zero experience required: Plug & play modular transfer learning for semantic visual navigation. *CVPR*, 2022.

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

[3] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

[4] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Semantic visual navigation by watching youtube videos. In *NeurIPS*, 2020.

[5] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *NeurIPS*, 2020.

[6] Devendra Singh Chaplot, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural mapping. *ICLR*, 2020.

[7] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. *CVPR*, 2020.

[8] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *ICRA*, 2019.

[9] Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigation in 3d environments. *ECCV*, 2020.

[10] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. *ICLR*, 2019.

[11] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

[12] K. Cho, A. C. Courville, and Y. Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 2015.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[14] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *ICCV*, 2021.

[15] Mateus V Gasparino, Arun N Sivakumar, Yixiao Liu, Andres EB Velasquez, Vitor AH Higuti, John Rogers, Huy Tran, and Girish Chowdhary. Wayfast: Navigation with predictive traversability in the field. *RAL*, 2022.

[16] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Devendra Singh Chaplot. Navigating to objects in the real world. *arXiv preprint arXiv:2212.00922*, 2022.

[17] Meera Hahn, Devendra Singh Chaplot, Shubham Tulsiani, Mustafa Mukadam, James M Rehg, and Abhinav Gupta. No rl, no simulation: Learning to navigate without navigating. *NeurIPS*, 2021.

[18] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[20] Vitor AH Higuti, Andres EB Velasquez, Daniel Varela Magalhaes, Marcelo Becker, and Girish Chowdhary. Under canopy light detection and ranging-based autonomous navigation. *JFR*, 2019.

[21] Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G. Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *CVPR*, 2019.

[22] Tianchen Ji, Arun Narenthiran Sivakumar, Girish Chowdhary, and Katherine Driggs-Campbell. Proactive anomaly detection for robot navigation with multi-sensor fusion. *IEEE Robotics and Automation Letters*, 2022.

[23] Jindong Jiang, Lunan Zheng, Fei Luo, and Zhijun Zhang. Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation. *arXiv preprint arXiv:1806.01054*, 2018.

[24] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. In *CVPR*, 2022.

[25] Nuri Kim, Obin Kwon, Hwiyeon Yoo, Yunho Choi, Jeongho Park, and Songhwai Oh. Topological semantic graph memory for image-goal navigation. In *CoRL*, 2023.

[26] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[27] Arjun Majumdar, Gunjan Aggarwal, Bhavika Suresh Devnani, Judy Hoffman, and Dhruv Batra. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. In *NeurIPS*, 2022.

[28] 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 *CVPR*, 2022.

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

[30] Santhosh K Ramakrishnan, Dinesh Jayaraman, and Kristen Grauman. An exploration of embodied visual exploration. *arXiv preprint arXiv:2001.02192*, 2020.

[31] Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. Pirlnav: Pretraining with imitation and rl finetuning for objectnav. *arXiv preprint arXiv:2301.07302*, 2023.

[32] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *CVPR*, 2022.

[33] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *CVPR*, 2022.

[34] Sonia Raychaudhuri, Tommaso Campari, Unnat Jain, Manolis Savva, and Angel X Chang. Reduce, reuse, recycle: Modular multi-object navigation. *arXiv preprint arXiv:2304.03696*, 2023.

[35] 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. *ICCV*, 2019.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[37] Dhruv Shah and Sergey Levine. ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints. *RSS*, 2022.

[38] Habitat Team. Habitat CVPR challenge, 2020.

[39] Habitat Team. Habitat CVPR challenge, 2021.

[40] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.

[41] Joanne Truong, Sonia Chernova, and Dhruv Batra. Bi-directional domain adaptation for sim2real transfer of embodied navigation agents. *IEEE Robotics and Automation Letters*, 2021.

[42] Justin Wasserman, Karmesh Yadav, Girish Chowdhary, Abhinav Gupta, and Unnat Jain. Last-mile embodied visual navigation. In *CoRL*, 2022.

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

[44] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1989.

[45] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. *CVPR*, 2018.

[46] Karmesh Yadav, Arjun Majumdar, Ram Ramrakhya, Naoki Yokoyama, Alexei Baevski, Zsolt Kira, Oleksandr Maksymets, and Dhruv Batra. Ovrl-v2: A simple state-of-art baseline for imagenav and objectnav. *arXiv preprint arXiv:2303.07798*, 2023.

[47] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. *arXiv preprint arXiv:2204.13226*, 2022.

[48] Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*, 2023.

[49] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *ECCV*, 2022.

## 6 Related Work

**Embodied Navigation.** A popular paradigm for solving object-goal navigation has been to learn directly from simulation. This has been achieved through modular [17] approaches, end-to-end [43] approaches, and by fine-tuning to the environment [31]. DDPPO [43] has been used as a baseline for many experiments for its success in solving point-goal navigation. They train a policy via a decentralized and distributed update rule, allowing them to train a policy in the high-fidelity simulator, AIHabitat [**?**], with over 2.5 billion frames. Habitat-Web [32] was able to achieve competitive performance on the 2022 Habitat Challenge by utilizing imitation learning over trajectories collected from a human. In contrast to directly using the simulator for fully training a policy from scratch, several methods have also attempted to use pretraining [47, 46, 24] to improve the navigation results. OVRL-v2 has achieved the current SOTA performance on the image-goal and object-goal by first pretraining a transformer head via masked autoencoding on HM3D [29] and Gibson [45]. Moving further away from simulation, a number of methods have been proposed to solve visual navigation tasks without any training in simulation. These zero-shot methods include frontier-based exploration [16] or heuristics [42] in conjunction with a module to solve the navigation task, as well as learning from real-world demonstrations [17, 4].

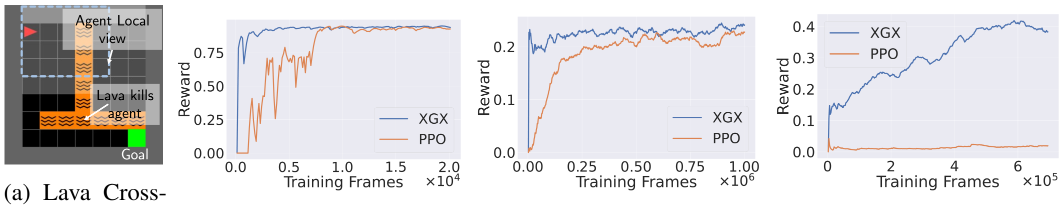
**Modular Policies in Visual Navigation.** Using modular policies is a popular paradigm for solving the visual-navigation task. Previous methods such as NRNS [17] use a modular strategy to build a topological map and with a Graph Neural Network predicts where on the topological map the agent should explore to find the image-goal. They propose a neural exploitation policy to complete the task. In Chaplot *et al.* [5] the authors optimize for exploration by training a semantic-map conditioned policy to predict a long-term goal of where in the map the agent should explore to. Once their agent is near the goal, they utilize a deterministic local policy to arrive at the goal. In NTS [7] the authors follow a similar paradigm by training a topological map to explore the environment and then using a local policy to reach a relative waypoint from the agent. Finally, in PONI [28], the authors propose a potential function network that predicts “where to look” in the environment by predicting the most promising area along the boundaries of a map. The authors also use an analytical planner to predict an action to solve how to navigate to a given boundary. In all of these previous works, the authors keep the exploration and exploitation modules separated. In contrast to these previous works, XGX does not learn to explore separately from the exploitation module. Instead, we learn to explore the environment with feedback from our exploitation module included during training.

**Goal-Conditioned Navigation on Physical Robots.** Recently, with the definition of the object-goal task in the embodied AI community, a number of papers have attempted to demonstrate their method for solving this task on a physical robot. This is an especially challenging task as real-world robots are susceptible to error modes not seen in simulation such as actuation and sensing noise, no access to ground truth data, and collecting trajectory examples is much slower [41, 8, 40]. Related works have studied image-goal and point-goal tasks and have shown transfer to a physical robot. ViKiNG [37] was proposed to solve the image-goal task at a kilometer scale by utilizing geographical hints through a top-down map. SLING [42] was able to improve over all previous baselines in simulation and real by utilizing a geometric-based solution for solving last-mile navigation. TGSM [25] utilizes a cross-graph mixer over a topological map that incorporates both image and object nodes to achieve SOTA performance on the image-goal task when a panoramic camera is utilized. However, in these tasks, an exact image or position of the goal is required. This does not allow for generalizing to our semantic visual navigation setting of navigating to a label such as “chair”.

## 7 Further Details of XGX

### 7.1 Further Detail of the Exploration Module (Supplements Sec. 2.1)

The architecture includes a ResNet [19] *i.e.* convolutional blocks for visual encoding and gated-recurrent unit [12] for connecting observation across time. A multi-layer perceptron policy head on top outputs a categorical action distribution. For further details, kindly refer to [31]. Taking lessons from their extensive benchmarking, we too warm-start the neural exploration module by imitation learning over offline demonstrations (from [32]) and then fine-tune with reinforcement learning via XGX. The decomposed policy transitions from the exploration module to the exploitation module, if certain criterion is met.



(a) Lava Crossing (b) Rewards for Empty (c) Rewards for 4-Rooms (d) Rewards for Lava Crossing

Figure 2: (a) one of the three navigation tasks XGX is tested on MiniGrid Platform. (b,c,d) Across these three tasks, XGX outperforms PPO and trains faster. As task complexity increases (b→c→d), the gains from XGX become increasly prominent.

### 7.2 Further Detail of the Exploitation Module (Supplements Sec. 2.1)

The exploitation module, utilizing the same off-the-shelf semantic segmentation model as phase transition, it transforms the RGB to a semantic mask of the goal object. Next, it lifts the 2D semantic mask to 3D by utilizing the depth mask. From this depth mask, and with knowledge of the camera intrinsics, a waypoint is calculated. To navigate to this waypoint, direct planning works well. For this, we utilize a local metric map (free from the depth sensor) and employ a fast-marching method [6, 17] for collision avoidance.

## 8 Diagnostic Task: Navigation in 2D MiniGrid (Supplements Sec. 3)

As a proof of concept, we investigate XGX in the fast MiniGrid environments [11]. The three tasks we benchmark on are: Empty, 4-Rooms, and Lava Crossing. These tasks require the navigation agent to explore the environment, avoid obstacles (particularly the detrimental lava) and stop at the goal represented as a green square. The standard reward structure used in these tasks is $1 - 0.9 * \frac{\text{step count}}{\text{max steps}}$ where max steps is the maximum number of steps the agent is allowed to take in a given episode. The agent observes a local, 'egocentric' (along the direction the agent is facing) patch of the 2D maze. An example Lava Crossing environment (mazes are procedurally generated and randomized) with helpful annotations is visualized in Fig. 2a. For XGX's exploitation module $\pi_{\text{exploit}}$, the agent uses a local planner to create an action that will navigate it toward the goal (only when goal is agent's egocentric view).

**Results.** As demonstrated in Fig. 2, XGX unanimously outperforms PPO across the three tasks. XGX demonstrates faster convergence over vanilla PPO. In the empty environment task, XGX achieves a reward of $0.9$ after just $5\%$ of training budget, while PPO needs $40\%$. In the Lava Crossing environment, XGX attains a reward of $0.41$ while PPO is stuck at $\sim 0.01$ despite training it for over $500k$ steps.

## 9 Object-Goal Task Definition (Supplements Sec. 3)

**Task Definition.** The goal definition, *i.e.*, the category of object to navigate to is sampled from a set of 6 categories [3] that are visually and physically well-defined. Consistent with several works on ObjectNav [3, 27, 47, 33, 1], we adopt the Habitat-Matterport3D (HM3D) scenes dataset. HM3D consists of high-quality photorealistic scans of 1000 real-world indoor scans. HM3D is an ideal choice for our study as it includes several scene types – homes, workspaces, eateries, and retail shops. Scenes span many geographical locations, floor area, and multiple floored scenes are also included. This diversity allows for a better chance at generalizing to real-robot runs. For evaluation, we adopt the standard and publicly available 2000 episodes HM3D-val split.

**Sensors.** We adopt the standard sensor suite for AIHabitat ObjectNav [38, 39]. Particularly, the agent has access to three sensor observations: (1) egocentric RGB image of 640×480, (2) corresponding depth mask, (3) relative localization obtained from a 'GPS+Compass sensor' [35, 3]. For off-the-shelf $f_{\text{semantic}}(\cdot)$, we utilize a RedNet [23] trained on the HM3D-training split to predict egocentric semantic information in simulation. In the real-world we utilize DETIC [49] to predict semantic segmentation.

| Method | % SCov ↑ | Cov |
|---|---|---|
| PIRLNav [31] | 36.3 | 63.5 |
| XGX (ours) | **39.1** | 59.8 |

Table 3: **Analysis for exploration efficiency.** We find that XGX outperforms PIRLNav (row 9 and row 10, Tab. 1a) and it does so with more efficient, goal-conditioned exploration.

**Metrics.** Following the literature in embodied navigation [35, 43, 21, 9, 3], we adopt metrics of success rate (SR) and success weighted by path length (SPL). SPL captures the policy's efficiency in path planning. For targeted evaluation of exploration modules, we also report a metric from learning-based exploration [10, 6, 7, 30], particularly, the percent of the environment seen ($\%$ Cov). A point in the environment is considered 'covered' if it is within 3.2m of the agent and its field of view. Note, less coverage is better only if success rate is the same. To this end, symmetric to the SPL metric introduced in [2], we devise a metric ($\%$ SCov) to balance task success with efficient exploration, defined as:

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{\%\ \texttt{Cov}_i^{\text{oracle}}}{\max(\%\ \texttt{Cov}_i, \%\ \texttt{Cov}_i^{\text{oracle}})} \tag{4}$$

Where N is the number of total episodes, $S_i$ specifies the success of episode i, and $\%\ \texttt{Cov}_i^{\text{oracle}}$ is the $\%$ Cov of a given episode when using an shortest-path policy.

## 10 XGX is More Efficient at Exploration (Supplements Sec. 3.2)

**Efficiency Analysis – XGX can explore faster (Tab. 3).** Here we undertake a focused investigation of learned-based exploration i.e. quantifying exploration and not navigation success to the goal. We find that XGX explores an average of $59.8\%$ per scene (compared to PIRLNav's $63.5\%$), while still performing better on other navigation success metrics. This is because XGX's exploration module was *a priori* optimized to offload servoing to the goal of the exploitation module. XGX taps into compositionality, making exploration significantly more efficient while also improving performance. This intuition is also empirically supported by XGX outperforming PIRLNav on $\%$ SCov at 36.3% to $39.1\%$.

## 11 Real World Environment Description (Supplements Sec. 3.3)

We test our robotics setup in three diverse scenes (1) `Apartment` - as visualized in Fig. 3 (2) `Office` - environment with long hallways and many connecting rooms and (3) `Food Court` - containing many chairs and furniture that needs to be avoided. In order to acquire pose information, we employ localization using the robot's LiDAR sensor and SLAM [26].

## 12 Error Modes

**Error Analysis.** The failure modes of XGX are visualized in Fig. 4. The largest source of failure is missing annotations, where $f_{\text{semantic}}$ of RedNet would correctly classify a goal object, only for the environment's semantic mesh to not be correctly labeled for the given object. Major failure modes for the exploration module are due to not searching the environment well by either just looping over the same space ('Exploration in Loops') or not trying to go up/down stairs when it should ('Missed Staircases'). Furthermore, the exploration module will occasionally call the 'stop' action immediately at the start of an episode ('Stop Right Away'). Issues with the switching mechanism includes failing to correctly recognize the goal ('Recognition Errors') and failing to switch between the exploration and exploitation modules ('Switching Error'). Other simulator errors include the floor geometry not being connected between spaces ('Broken Floor Geometry') and incomplete meshes where if the agent tries to go up/down stairs it can not ('Stuck on Stairs').

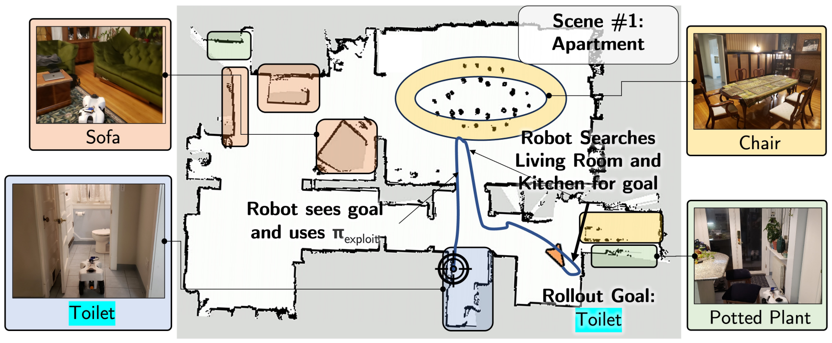

Sofa
Toilet
Scene #1: Apartment
Chair
Robot Searches Living Room and Kitchen for goal
Robot sees goal and uses $\pi_{\text{exploit}}$
Rollout Goal: Toilet
Potted Plant

Figure 3: **Physical robot navigation setup and rollout visualization in `Apartment` environment.** The agent navigates to one of 4 goal categories. Here, we visualize an actual rollout for the 'toilet' category and mark the phase transition to the exploit module. Note: The agent has not seen the scene or access the top-down map (the map is for visualization).

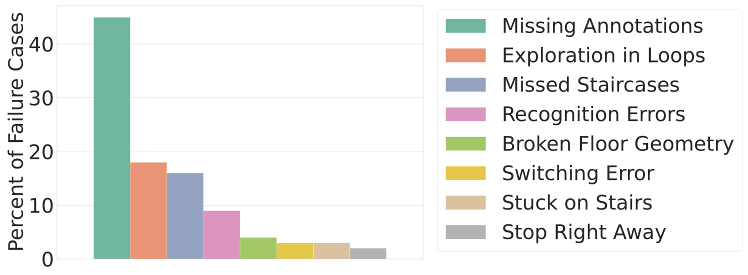


Figure 4: **Error Analysis of XGX *(simulation)*.** Beyond missing annotations in the dataset, the top three error modes of XGX are (1) $\pi_{\text{explore}}$ manifesting a looping behavior, (2) missing staircases, (3) and errors in semantic segmentation.