Last-Mile Embodied Visual Navigation

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

Abstract: Realistic long-horizon tasks like image-goal navigation involve exploratory and exploitative phases. Assigned with an image of the goal, an embodied agent must explore to *discover the goal*, *i.e.*, search efficiently using learned priors. Once the goal is discovered, the agent must accurately calibrate the *last-mile of navigation* to the goal. As with any robust system, *switches* between exploratory goal discovery and exploitative last-mile navigation enable better recovery from errors. Following these intuitive guide rails, we propose SLING to improve the performance of existing image-goal navigation systems. Entirely complementing prior methods, we focus on last-mile navigation and leverage the underlying geometric structure of the problem with neural descriptors. With simple but effective switches, we can easily connect SLING with heuristic, reinforcement learning, and neural modular policies. On a standardized image-goal navigation benchmark [1], we improve performance across policies, scenes, and episode complexity, raising the state-of-the-art from 45% to 55% success rate. Beyond photorealistic simulation, we conduct real-robot experiments in three physical scenes and find these improvements to transfer well to real environments. Code and results: https://jbwasse2.github.io/portfolio/SLING

Keywords: Embodied AI, Robot Learning, Visual Navigation, Perspective-n-Point, AI Habitat, Sim-to-Real.

1 Introduction

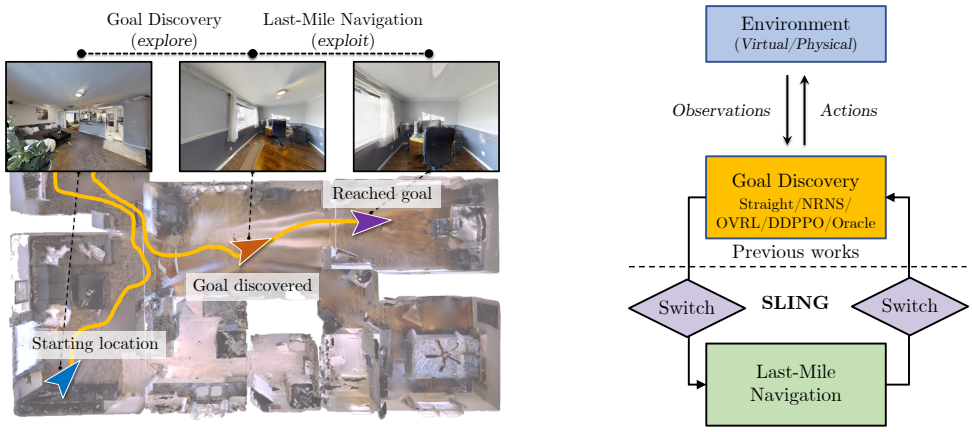
Imagine you are at a friend's home and you want to find the couch you have seen in your friend's photo. At first, you use semantic priors *i.e.* priors about the semantic structure of the world, to navigate to the living room (a likely place for the couch). But as soon as you get the first glimpse of the couch, you implicitly estimate the relative position of the couch, use intuitive geometry, and navigate towards it. We term the latter problem, of navigating to a visible object or region, as last-mile navigation.

The field of visual navigation has a rich history. Early approaches used hand-designed features with geometry for mapping followed by standard planning algorithms. But such an approach fails to capture the necessary semantic priors that could be learned from data. Therefore, in recent years, we have seen more efforts and significant advances in capturing these priors for semantic navigation tasks such as image-goal [2, 3, 4, 1, 5, 6] and object-goal navigation [7, 8, 9, 10]. The core idea is to train a navigation policy using reinforcement or imitation learning and capture semantics. But in an effort to capture the semantic priors, these approaches almost entirely bypass the underlying geometric structure of the problem, specifically when the object or view of interest has already been discovered.

One can argue that last-mile navigation can indeed be learned from data itself. We agree that, in principle, it can be. However, we argue and demonstrate that an unstructured local policy for last-mile navigation is either (a) sample inefficient (billions of frames in an RL framework [11]) or (b) biased and generalize poorly when learned from offline demonstrations (due to distributional shift [12, 13]). Therefore, our solution is to revisit the basics! We propose **S**witchable **L**ast-Mile

*equal technical contribution; †corresponding authors

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.



(a) Phases in Image-Goal Navigation

(b) Policy Overview

Figure 1: **Switchable Last-Mile Image-Goal Navigation**. (a) Long-horizon semantic tasks such as image-goal navigation involves exploratory discovery of goals and exploitative last-mile navigation, (b) An overview of SLING that allows for *switching* between policies from prior work and our last-mile navigation system.

Image-Goal **Navig**ation (SLING) – a simple yet very effective geometric navigation system and associated switches. Our approach can be combined with any off-the-shelf learned policy that uses semantic priors to explore the scene. As soon as the object or view of interest is detected, the SLING switches to the geometric navigation system. We observe that SLING provides significant performance gains across baselines, simulation datasets, episode difficulty, and real-world scenes.

Our key contributions are: (1) A general-purpose last-mile navigation system and switches, that we connect with five diverse goal discovery methods, leading to improvements across the board. (2) A new state-of the-art of 54.8% success *i.e.* a huge jump of 21.8% *vs.* published work [5] and 9.2% *vs.* a concurrent preprint [6], on the most widely-tested fold (Gibson-curved) of the AI Habitat image-goal navigation benchmark [1]; (3) Extensive robot experiments of image-goal navigation in challenging settings with improved performance over a neural, modular policy [1] trained on real-world data [14].

2 Related Work

Prior work in visual navigation and geometric 3D vision is pertinent to SLING.

Embodied navigation. Anderson *et al.* [15] formalized different goal definitions and metrics for the evaluation of embodied agents. In point-goal navigation, relative coordinates of the goal are available (either at all steps [16, 11, 17, 18, 19] or just at the start of an episode [9, 20, 21]). Successful navigation to a point-goal could be done without semantic scene understanding, as seen by competitive depth-only agents [16, 11]. Semantic navigation entails identifying the goal through an image (image-goal [1, 2, 22]), acoustic cues (audio-goal [23, 24]), or a category label (object-goal [8, 9]). Several extensions of navigation include language-conditioned navigation following [25, 26, 27, 28], social navigation [29, 30, 31, 32, 33], and multi-agent tasks [34, 35, 36, 37, 38, 39]. However, each of these build-off single-agent navigation and benefit from associated advancements. For more embodied tasks and paradigms, we refer the reader to a recent survey [40]. In this work, we focus on image-goal navigation in visually rich environments.

Image-goal navigation. Chaplot *et al.* [3] introduced a modular and hierarchical method for navigating to an image-goal that utilizes a topological map memory. Kwon *et al.* [41] introduced a memory representation based on image similarity, which in turn is learned in an unsupervised fashion from unlabeled data and the agent's observed images. Following up on [3], NRNS [1] improves the topological-graph-based architecture and open-sourced a public dataset and IL and RL baselines [11, 3] within AI Habitat. This dataset has been adopted for standardized evaluation [5, 6]. ZER [5] focuses on transferring an image-goal navigation policy to other navigation tasks. In a concurrent preprint, Yadav *et al.* [6] utilize self-supervised pretraining [42] to improve an end-to-end visual RL policy [11] for the image-goal navigation benchmark. Our contributions are orthogonal to the above and can be easily combined with them, as we demonstrate in Sec. 4.

Beyond simulation, SLING finds relevance to the rich literature of navigating to an image-goal on physical robots. Meng *et al*. [4] utilize a neural reachability estimator and a local controller based on a Riemannian Motion Policy framework to navigate to image-goals. Hirose *et al*. [43] train a deep model predictive control policy to follow a trajectory given by a sequence of images while being robust to variations in the environment. Even in outdoor settings, meticulous studies have shown great promise, based on negative mining, graph pruning, and waypoint prediction [44] and utilizing geographic hints for kilometer-long navigations [45]. Complementing this body of work, SLING tackles image-goal navigation in challenging indoor settings, without needing any prior data of the test environment (similar to [1, 6, 3]) *i.e*. during evaluation no access to information (trajectories, GPS, or top-down maps) in the test scenes is assumed.

Last-mile navigation. The works included above focus primarily on goal discovery. In contrast, recent works have also identified 'last-mile' errors that occur when the goal is in sight of or close to the agent. For multi-object navigation, Wani *et al*. [46, 47] observed a two-fold improvement when allowing an error budget for the final 'found' or 'stop' actions. Chattopadhyay *et al*. [48] found the last step of navigation to be brittle *i.e*. small perturbations lead to severe failures. Ye *et al*. [10] identified last-mile errors as a prominent error mode (10% of the failures) in object-goal navigation. However, none of these works address the problem with the last-mile of navigation. From a study inspired by [46], we infer that better (or more tolerant to error) last-mile navigation can indeed lead to better performance in the image-goal navigation task (details in Appendix H).

Connections to 3D vision. The objective of our last-mile navigation system is to predict the relative camera pose between two images *i.e*. agent's view and image-goal. To this end, pose estimation of a calibrated camera from 3D-to-2D point correspondences connects our embodied navigation task to geometric 3D computer vision. The Perspective-n-Point (PnP) formulation, with extensive research and efficient solvers [49, 50, 51], fits this use case perfectly. To find an accurate PnP solution, locating correspondences between the local features of the two images is critical. We utilize SuperGlue [52] which is based on correspondences learned via attention graph neural nets and partial assignments. We defer details of PnP and finding correspondences to Sec. 3.3, to make the approach self-sufficient. Notably, different from related works in 3D vision [53, 54, 55], we apply SLING to sequential decision-making in embodied settings, particularly, image-goal navigation. To take policies to the real world, we utilized robust SLAM methods [56, 57] for local odometry and pose estimation, which has also been found reliable by prior works in sim-to-real [58, 59, 60].

3 SLING

In this section, we begin with an overview of the task and the entire pipeline of SLING. We then discuss the implementations for goal discovery, our proposed system for last-mile navigation, and switches to easily combine it with prior works. While we explain key design choices in the main paper, a supplementary description and a list of hyperparameters, for effective reproducibility, is deferred to Appendix A.

3.1 Overview

We follow the image-goal navigation task benchmark by Hahn *et al*. [1] (similar to the prior formulations [2, 3]). The agent observes an RGB image $\mathbf{I}_a$, a depth map $\mathbf{D}_a$, and the image-goal $\mathbf{I}_g$. The agent can sample actions from $\mathcal{A} = \{\texttt{move forward}, \texttt{turn right}, \texttt{turn left}, \texttt{stop}\}$. The `stop` action terminates the episode.

As shown in Fig. 1a, we divide image-goal navigation into – a goal discovery and a last-mile navigation phase. In the goal discovery phase, the agent is responsible for discovering the goal *i.e*. navigating close enough for the goal to occupy a large portion of the egocentric observation ('goal discovered' image). Fig. 1b shows how the control flows between our system. If the *explore→exploit* switch isn't triggered, learning-based exploration will continue. Otherwise, if the *explore→exploit* switch triggers, the agent's observations now overlap with the image-goal and the control flows to the last-mile navigation system. We find that a one-sided flow (as attempted in [1, 3]) from *explore→exploit* is too optimistic. Therefore, we introduce symmetric switches, including one that flows control back to goal discovery.

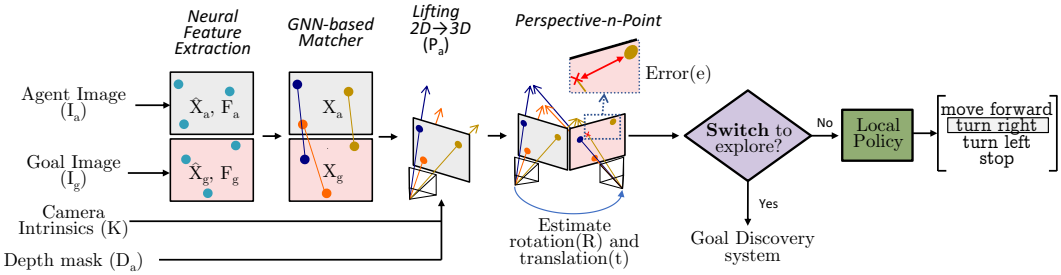


Figure 2: **Last-Mile Navigation system.** Neural keypoint feature descriptors are extracted and matched to obtain correspondences between the agent's view and the image-goal. The geometric problem of estimating the relative pose between the agent and goal view is solved using efficient perspective-n-point. A *exploit→explore* switch, if triggered, flows control back to the goal discovery phase. Else, the estimations are fed into a local policy head to decide the agent's actions.

3.2 Goal Discovery

We can combine our versatile last-mile navigation system and switching mechanism with any prior method. These prior methods are previously suggested solutions to image-goal navigation. We demonstrate this with five diverse goal discovery (GD) implementations.

Straight [61]. A simple, heuristic exploration where the agent moves forward and unblocks itself, if stuck, by turning right (similar to an effective exploration baseline in [61]).

Distance Prediction Network (NRNS-GD) [1]. Exploratory navigation is done by proposing waypoints in navigable areas (determined utilizing the agent's depth mask), history is maintained using a topological map, and processed using graph neural nets. The minimum cost waypoint is chosen utilizing outputs from a distance prediction network. More details are given in Appendix B and [1].

Decentralized Distributed PPO (DDPPO-GD) [11]. An implementation of PPO [62] for photorealistic simulators where rendering is the computational bottleneck. This has been a standard end-to-end deep RL baseline in prior works, across tasks [18, 1, 5, 6, 63].

Offline Visual Representation Learning (OVRL-GD) [6]. A DDPPO network, with its visual encoder pretrained using self-supervised pretext tasks [42] on images obtained from 3D scans [64].

Environment-State Distance Prediction (Oracle-GD). To quantify the effect of errors coming from the goal discovery phase, we devise an upper bound. This is a privileged variant of NRNS-GD that accesses the ground-truth distances from the environment, exclusively for the goal discovery phase. For fine details of its construction, particularly, how we curtail this to be an oracle explorer and not an oracle policy, see Appendix B.

3.3 Last-Mile Navigation

The proposed last-mile navigation module transforms the agent's observations and image-goal into actions that take the agent closer to the goal. The steps are shown in Fig. 2 and detailed next.

Neural Feature Extractor. We first transform the agent's RGB $\mathbf{I}_a$ to local features $(\hat{\mathbf{X}}_\mathbf{a}, \mathbf{F}_a)$, where $\hat{\mathbf{X}}_\mathbf{a} \in \mathbb{R}^{n_a \times 2}$ are the positions and $\mathbf{F}_a \in \mathbb{R}^{n_a \times k}$ are the visual descriptors in the agent's image. Here, n_a is the number of detected local features and k is the length of each descriptor. Similarly, $\mathbf{I}_g$ leads to features $(\hat{\mathbf{X}}_\mathbf{g}, \mathbf{F}_g)$, where $\hat{\mathbf{X}}_\mathbf{g} \in \mathbb{R}^{n_g \times 2}$ and $\mathbf{F}_g \in \mathbb{R}^{n_g \times k}$ with n_g local features in the image-goal. Following DeTone *et al*. [65], we adopt an interest-point detector, pretrained on synthetic data followed by cross-domain homography adaptation (here, $k = 256$).

Matching Module. From extracted features $(\hat{\mathbf{X}}_\mathbf{a}, \mathbf{F}_a)$ and $(\hat{\mathbf{X}}_\mathbf{g}, \mathbf{F}_g)$, we predict matched subsets $\mathbf{X}_\mathbf{a} \in \mathbb{R}^{n \times 2}$ and $\mathbf{X}_\mathbf{g} \in \mathbb{R}^{n \times 2}$. The matching is optimized to have $\mathbf{X}_\mathbf{a}$ and $\mathbf{X}_\mathbf{g}$ correspond to the same point. We utilize an attention-based graph neural net (GNN) that tackles partial matches and occlusions well using an optimal transport formulation, following Sarlin *et al*. [52]. The above neural feature extractor and GNN-based matcher help enjoy benefits of learning-based methods, particularly, those *pretrained* on large offline visual data without needing online, end-to-end finetuning. The geometric components, relying on these neural features, are described next.

Lifting Points from 2D→3D. Next, the agent's 2D local features are lifted to 3D with respect to the agent's coordinate frame *i.e*. $\mathbf{P}_a \in \mathbb{R}^{n \times 3}$. This is done by utilizing the camera intrinsic matrix $\mathbf{K}$

(particularly, principle point p_x, p_y and focal lengths f_x, f_y) and the corresponding depth values for each position in $\mathbf{X_a}$, say $\mathbf{d_a} \in \mathbb{R}^n$. The i^{th} row of $\mathbf{P}_a$ is calculated as

$$\mathbf{P}_a\,(i,:) = \left(\frac{\mathbf{X_a}(i,1) - p_x}{f_x} * \mathbf{d_a}(i), \frac{\mathbf{X_a}(i,2) - p_y}{f_y} * \mathbf{d_a}(i), \mathbf{d_a}(i) \right), \tag{1}$$

where $\mathbf{X_a}(i,1)$ and $\mathbf{X_a}(i,2)$ correspond to the x and y coordinate of i^{th} feature in $\mathbf{X_a}$, respectively. Formally, $\mathbf{d_a}(i) := \mathbf{D}_a\,(\mathbf{X_a}(i,1), \mathbf{X_a}(i,2))$.

Perspective-n-Point. The objective of the next step *i.e.* Perspective-n-Point (PnP) is to find the rotation and translation between the agent and goal camera pose that minimized reprojection error. Concretely, for a given rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ and translation vector $\mathbf{t} \in \mathbb{R}^3$, the 3D positions $\mathbf{P}_a$ of local features can be reprojected from the coordinate system of the agent to that of the goal camera:

$$\begin{bmatrix} \tilde{\mathbf{X}}_\mathbf{g} \\ 1 \end{bmatrix} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \begin{bmatrix} \mathbf{P}_a \\ 1 \end{bmatrix}; \qquad \text{Reprojection error } e = \|\tilde{\mathbf{X}}_\mathbf{g} - \mathbf{X_g}\|_2^2. \tag{2}$$

where $\tilde{\mathbf{X}}_\mathbf{g}$ are the reprojected positions. Minimizing the reprojection error e, via ePnP [51] and RANSAC [49] (to handle outliers), we obtain the predicted rotation and translation. The reprojection is visualized in Fig. 2, where the agent's amber point is lifted and reprojected in goal camera coordinates. The reprojection is different from its correspondence in the goal image.

Estimating Distance and Heading to Goal. The predicted translation $\mathbf{t}$ can help calculate the distance $\rho = \|\mathbf{t}\|_2$ from the agent to the goal. Similarly, the heading ϕ from the agent to the goal can be obtained from the dot product of the unit vectors along the optical axis (of the agent's view) and $\mathbf{t}$. Concretely, $\phi = \text{sgn}(t[1]) * \arccos\left(\mathbf{t} \cdot \mathbf{o}_a / \|\mathbf{t}\|_2 \|\mathbf{o}_a\|_2\right)$. The sign comes from $t[1]$ which points along the axis perpendicular to the agent's optical axis but parallel to the ground. The sign is particularly important when calculating the heading as it distinguishes between the agent turning right or left.

Local policy. Finally, the distance ρ and heading ϕ between the agent's current position to the estimated goal are utilized to estimate actions in the action space $\mathcal{A}$ to reach the goal. Following accurate implementations [66, 1], we adopt a local metric map to allow the agent to heuristically avoid obstacles and move towards the goal. For further details, see Appendix A.

3.4 Switches

We define simple but effective switches between the two phases of goal discovery (*explore*) and last-mile navigation (*exploit*). The *explore→exploit* switch is triggered if the number of correspondences $n > n_{\text{th}}$, where n_{th} is a set threshold. This indicates that the agent's image has significant overlap with the image-goal, so control can flow to the last-mile navigation phase. We find that this simple switch performs better than training a specific deep net to achieve the same (variations attempted in [1, 3, 4]). For *exploit→explore*, if the optimization for $\mathbf{R}, \mathbf{t}$ (see Eq. (2)) fails or if the predicted distance is greater than d_{th} (tuned to 4m), the agent returns to the goal discovery phase.

4 Experiments

We report results for image-goal navigation both in photorealistic simulation and real-world scenes.

4.1 Data and Evaluation

We evaluate image-goal navigation policies on the benchmark introduced by Hahn *et al.* [1] and follow their evaluation protocol and folds. The benchmark consists of numerous folds: {Gibson [67], MP3D [68]}×{straight, curved}×{easy, medium, hard}. For a direct comparison to prior work [3, 1, 5, 6] that reports primarily on 'Gibson-curved' fold, we follow the same in the main paper. Consistent performance trends are seen in 'Gibson-straight' and in the MP3D folds as well. These results are deferred to Appendix C and Appendix F. Performance on image-goal navigation is chiefly evaluated via two metrics – percentage of successful episodes (*success*) and success weighted by inverse path length (*SPL*) [15]. For top-performing baselines, we also include the average distance to the goal at the end of the episode in Appendix G. The objective of the image-goal navigation task is to execute `stop` within $1m$ of the goal location. The agent is allowed 500 steps.

Table 1: **Results for 'Gibson-curved' episodes** Note the significant gains by adding SLING to prior works. Consistent trends are seen in 'Gibson-straight' (Appendix C) and MP3D-curved episodes (Appendix F).

		Overall		Easy		Medium		Hard	
	Method	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑
1	BC w/ Spatial Memory [69]	1.3	1.1	3.1	2.5	0.8	0.7	0.2	0.1
2	BC w/ GRU [69, 70]	1.7	1.3	3.6	2.8	1.1	0.9	0.5	0.3
3	DDPPO [11] (from [1])	15.7	12.9	22.2	16.5	20.7	18.5	4.2	3.7
4	NRNS [1]	21.7	8.1	31.4	10.7	22.0	8.2	11.9	5.4
5	ZER [5]	33.0	23.6	48.0	34.2	36.0	25.9	15.1	10.8
6	OVRL [6]	45.6	28.0	53.6	31.7	47.6	30.2	35.6	21.9
7	DDPPO-LMN + OVRL-GD	44.3	30.1	52.4	36.6	48.6	32.6	31.9	21.2
8	SLING + Straight-GD	31.0	12.8	39.2	14.3	33.0	14.3	21.0	9.9
9	SLING + DDPPO-GD	37.9	22.8	52.2	32.7	42.2	25.2	19.4	10.5
10	SLING + NRNS-GD	43.5	15.1	58.7	17.4	47.0	17.4	25.0	10.5
11	SLING + OVRL-GD	**54.8**	**37.3**	**65.4**	**45.7**	**59.5**	**40.6**	**39.6**	**25.5**

4.2 Methods

We compare our last-mile navigation with several standardized baselines [69, 11, 1]. Note that field-of-view, rotation amplitude, *etc*. vary across baselines and we adopt the respective settings for fair comparison (implementation details of SLING are in Appendix A). Prior methods use a mix of sensors including RGB, depth, and agent pose, but no dense displacement vector to the goal. While we did include the most relevant baselines in Sec. 3.2, we also compare SLING to several other image-goal solvers. This includes imitation learning baselines such as Behavior Cloning (BC) w/ Spatial Memory [69] and BC w/ Gated Recurrent Unit [69, 70]. We also compare to established reinforcement learning baselines – DDPPO [11] and Offline Visual Representation Learning (OVRL) [6]. OVRL also makes use of pretraining using a self-supervised objective. Finally, we compare to related modular baselines include NRNS [1] and Zero Experience Replay (ZER) [5]. We defer a detailed discussion of these baselines to Appendix B.

SLING & Ablations. For a comprehensive empirical study, we combine SLING with Straight-GD, NRNS-GD, DDPPO-GD, OVRL-GD, and Oracle-GD (see Sec. 3.2 for details). We also introduce a neural baseline, DDPPO-LMN, a DDPPO model trained to perform last-mile navigation.

Further, we include clear ablations to show the efficacy of the components of our method and robustness to realistic pose and depth sensor noise:
• *w/ MLP switch:* instead of SLING's explore→exploit switch (that utilizes geometric structure), if a MLP[1] detects similarity between the agent and goal images (as in [1]).
• *w/o Recovery:* if the *exploit→explore* switch is removed *i.e*. one-sided flow of control.
• *w/o Neural Features:* if the neural features [65] are replaced with traditional features [71].
• *w/ Pose Noise:* add noise to pose that emulates real-world sensors [66, 72] (same as [3, 1]).
• *w/ Depth Noise:* imperfect depth by adopting the Redwood Noisy Depth model [73] in AI Habitat.
• *w/ Oracle-GD:* privileged baseline where NRNS-GD can access ground-truth distances to move the agent closer to the goal during exploration (see Sec. 3.2 and Appendix B).
• *w/ Oracle-LM-Pose:* privileged last-mile system with perfect displacement from agent to goal
• *w/ Oracle-LP:* privileged baseline where local policy can teleport agent to the goal prediction

4.3 Quantitative Results

In the following, we include takeaways based on the results in Tab. 1 and Tab. 2.

State-of-the-art performance. As Tab. 1 details, SLING + OVRL-GD outperforms a suite of IL, RL, and neural modular baselines. The Gibson-curved fold is widely adopted in prior works and hence the focus of the main paper. With a 54.8% overall success and 37.3 SPL we are the best-performing method on the benchmark, improving success rate by 21.8% *vs*. ZER and 9.2% *vs*. OVRL ('overall success' column of rows 5, 6, & 11). In Appendix I, we also demonstrate state-of-the-art performance when panoramic images are used.

[1] trained over an offline dataset of expert demonstrations, where adjacent nodes in a topological graph (that they maintain) are considered positives

Table 2: **Ablations on 'Gibson-curved' episodes.** Both switches are key to SLING's performance. SLING is resilient to sensor noise. Similar trends can be observed over ablations performed with OVRL-GD in Appendix F. The privileged last-mile navigation system establishes an upper bound for last-mile navigation. Even with Oracle-GD, performance improves if SLING is added.

		Overall		Easy		Medium		Hard	
	Method	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑
1	NRNS [1]	21.7	8.1	31.4	10.7	22.0	8.2	11.9	5.4
2	SLING + NRNS-GD	43.5	15.1	58.7	17.4	47.0	17.4	25.0	10.5
3	w/ MLP Switch	42.5	14.8	55.4	16.7	47.3	17.3	24.9	10.5
4	w/ MLP Switch w/o Recovery	31.5	11.5	45.6	14.3	32.8	12.9	16.1	7.3
5	w/o Neural Features	33.7	11.3	47.5	13.5	35.9	13.0	17.7	7.5
6	w/ Pose Noise	43.7	14.3	58.6	16.1	47.6	16.8	24.9	10.1
7	w/ Pose & Depth Noise	43.5	14.0	56.9	15.9	47.2	15.9	26.6	10.3
	Privileged Last-Mile Navigation								
8	w/ Oracle-LP	45.1	17.8	60.8	21.2	48.7	20.3	25.8	12.1
9	w/ Oracle-LM-Pose	53.3	19.3	72.3	23.4	57.1	21.6	30.5	13.1
10	w/ Oracle-LM-Pose & Oracle-LP	53.7	22.4	72.6	27.7	57.6	24.7	31.0	14.9
	Privileged Goal Discovery								
11	NRNS + Oracle-GD *(upper bound)*	67.7	60.2	68.5	58.4	71.2	63.7	63.5	58.7
12	SLING + Oracle-GD *(upper bound)*	86.2	74.8	85.9	72.2	88.6	77.7	84.3	74.6

SLING works across methods. Using switches, we add our last-mile navigation system to DDPPO [11], NRNS [1], and OVRL [6], and observe gains across the board. As shown in Tab. 1, SLING improves the success rate of DDPPO by 22.2%, NRNS by 21.8%, and OVRL by 9.2% (rows 3 & 9, 4 & 10, 6 & 11, respectively). Quite surprisingly, SLING even with simple straight exploration, can outperform deep IL, RL, and modular baselines. (rows 1, 2, 3, 4, & 8).

SLING outperforms neural policies for last-mile navigation. SLING surpasses DDPPO trained over 400M steps for last-mile navigation by 10.5% on success rate (rows 7 & 11).

SLING succeeds across scene datasets. Similar improvements are also seen in MP3D scenes – adding SLING to OVRL improves success by 5.1%. Further details and results can be found in Appendix F.

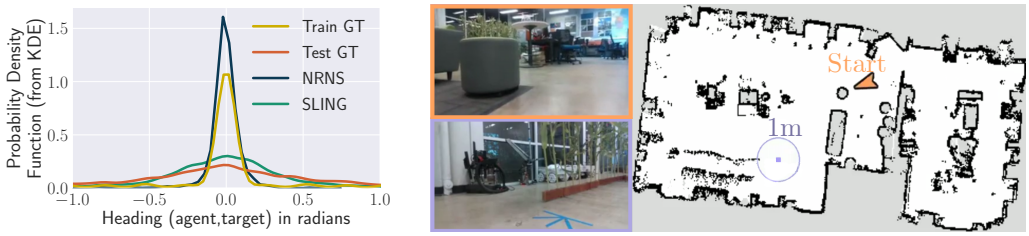
SLING is resilient to sensor noise. As shown in rows 6 & 7 of Tab. 2, minor drops in performance are observed despite challenging noise in pose and depth sensors – SPL successively reduces 15.1→14.3→14.0% (rows 2→6→7).

Geometric switches are better. Performance reduces if we swap out SLING's explore→exploit switch with the MLP switch of NRNS [1]. The effect is exasperated when SLING's exploit→explore switch is also removed, leading to a drop of 12% (Tab. 2, rows 2 & 4). The neural features utilized in SLING are useful, as seen by comparing rows 2 and 5. Further, over a set of 6500 image pairs, we evaluate the accuracy of switches. SLING's explore→exploit switch is 92.0% accurate and MLP switch [1] is only at 82.1%. Also, SLING exploit→explore switch is 84.1% accurate while NRNS doesn't have such a recovery switch (details of this study in Appendix D).

Large potential for last-mile navigation. When Oracle-LM-Pose and Oracle-LP are used there is a 10.2% overall improvement in success from 43.5 to 53.7% (Tab. 2, rows 2 & 10). Notably, in easy episodes, oracle performance is an ambitious upper bound with an increase in success of 13.9% (58.7→72.6%). For the hard (*i.e.* longer) episodes, the oracle components have a relatively lower impact. This is quite intuitive as goal discovery errors are a more prominent error mode in long-horizon episodes instead of last-mile navigation.

Improvements with Oracle-GD. Even if we assume a perfect variant of goal discovery system from [1], we observe that performance saturates at 67.7% success (row 11, Tab. 2). Comparing rows 11 and 12, SLING can boost this asymptotic success rate by 18.5% (67.7→86.2%).

Analysis: Why is SLING more robust? In Fig. 3a, we visualize the frequency distribution of heading (from the agent to the target) in expert demonstrations [1] ('train GT') and that observed at inference ('test GT'). With no geometric structure, NRNS picks up the bias in training data, particularly, towards the heading of 0 (optimal trajectories entail mostly moving forward). Concretely, 72.2% of the training data is within $[-15°, 15°]$. This drops to 39.4% at test time when the last-mile



(a) Bias in heading distribution

(b) Robot experiments (more visuals in Figures 5, 6, 7)

Figure 3: (a) Significant distribution shift between training and test heading from agent to goal (Sec. 4.3), (b) Navigation policies deployed on a robot in cluttered real-world scenes (Sec. 4.4).

navigation phase is reached (using the best-performing Oracle-GD). Quantified with (first) Wasserstein distance, $W(\text{Test GT}, \text{NRNS}) = 0.0134$ *vs.* $W(\text{Test GT}, \text{SLING}) = 0.0034$, demonstrating SLING can better match the distribution at inference.

4.4 Physical Robot Experiments

We test the navigation policies on a TerraSentia [74] wheeled robot, equipped with an Intel® RealSense™ D435i depth camera (further hardware details in Appendix E). The robot is initialized in an unseen indoor environment and provided an RGB image-goal. We ran a total of 120 trajectories, requiring 30 human hours of effort, across three scenes and two levels of difficulty. Following the previously collected simulation dataset [1], easy goals are 1.5-3m from the starting location and hard goals are 5-10m from the starting location. Particularly, we test within an office and the common areas in two department buildings, over easy and hard episodes (following definitions from [1]). The physical setup (office) is shown in Fig. 3b. As in simulation, the agent is successful if it executes `stop` action within 1m of the goal. Examples of the image-goal utilized in physical robot experiments and precautions taken are included in Appendix E.

As shown in Tab. 3, for sim-to-real experiments, we base the goal discovery system on the NRNS model. We choose NRNS as the authors published an instantiation trained exclusively on real-world trajectories, particularly, RealEstate10K [14] (house tours videos from YouTube). In preliminary experiments, we verified that this NRNS instantiation outperformed its simulation counterpart. For a direct comparison, in SLING + NRNS-GD, we utilize the same goal discovery system but add our switching and last-mile navigation system (SLING) around it. With SLING, we improve performance from 40.0% success to 56.6%. The gains become more prominent as the task horizon increases, leading to an improvement in success rate from 3.3% to 20.0%. The large gains in hard episodes (which are exploration heavy) are accounted to SLING's better explore→exploit switch and SLING's last-mile navigation system that is not biased to zero heading (particularly important for curved and long episodes).

	Easy		Hard	
Method	Succ↑	SPL↑	Succ↑	SPL↑
NRNS [1]	40.0	37.7	3.3	3.3
+ SLING	**56.6**	**53.7**	**20.0**	**19.3**

Table 3: Results in real-world scenes.

5 Conclusion

In this work, we identify and leverage the geometric structure of last-mile navigation for the challenging image-goal navigation task [1]. With analysis of data distributions, we demonstrate that learning from expert demonstrations may lead to developing a bias. Being entirely complementary to prior work, we demonstrate that adding SLING leads to improvements across data splits, episode complexity, and goal discovery policies, establishing the new state-of-the-art for image-goal navigation [1]. We also transfer policies trained in simulation to real-world scenes and demonstrate significant gains in performance. Further improvements in the switching mechanism, neural keypoint features, visual representations from view augmentations, *etc*. complement our proposed approach to help improve performance in future work.

Like any method, SLING has several aspects where follow-up works can improve on. We list them explicitly: (1) Our method is limited by mistakes in matching correspondences. (2) We add additional parameters that need to be tuned. (3) We make a single prediction for last-mile navigation. (4) We assume access to depth and pose information. More details of these aspect as well as a discussion on pose errors, depth noise, and the nuanced image-goal navigation definition in Appendix J.

Acknowledgments

We thank the reviewers for suggesting additional experiments to make the work stronger. JW and GC are supported by ONR MURI N00014-19-1-2373. We are grateful to Akihiro Higuti and Mateus Valverde for physical robot help, Dhruv Batra for helping broaden the scope, Jae Yong Lee for help with geometric vision formulation, Meera Hahn for assistance in reproducing NRNS results, and Shubham Tulsiani for helping ground the work better to 3D vision. A big thanks to our friends who gave feedback to improve the submission draft – Homanga Bharadhwaj, Raunaq Bhirangi, Xiaoming Zhao, and Zhenggang Tang,

References

[1] M. Hahn, D. S. Chaplot, S. Tulsiani, M. Mukadam, J. M. Rehg, and A. Gupta. No rl, no simulation: Learning to navigate without navigating. *NeurIPS*, 2021.

[2] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *ICRA*, 2017.

[3] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta. Neural topological slam for visual navigation. *CVPR*, 2020.

[4] X. Meng, N. Ratliff, Y. Xiang, and D. Fox. Scaling local control to large-scale topological navigation. *ICRA*, 2020.

[5] Z. Al-Halah, S. K. Ramakrishnan, and K. Grauman. Zero experience required: Plug & play modular transfer learning for semantic visual navigation. *CVPR*, 2022.

[6] K. Yadav, R. Ramrakhya, A. Majumdar, V.-P. Berges, S. Kuhar, D. Batra, A. Baevski, and O. Maksymets. Offline visual representation learning for embodied navigation. *arXiv preprint arXiv:2204.13226*, 2022.

[7] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *NeurIPS*, 2020.

[8] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

[9] H. Team. Habitat CVPR challenge, 2020. URL https://aihabitat.org/challenge/2020/.

[10] J. Ye, D. Batra, A. Das, and E. Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. *ICCV*, 2021.

[11] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *ICLR*, 2019.

[12] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *AISTATS*, 2011.

[13] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[14] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: Learning view synthesis using multiplane images. *SIGGRAPH*, 2018.

[15] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[16] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. *ICCV*, 2019.

[17] H. Team. Habitat CVPR challenge, 2019. URL https://aihabitat.org/challenge/2019/.

[18] J. Ye, D. Batra, E. Wijmans, and A. Das. Auxiliary tasks speed up learning pointgoal navigation. *CoRL*, 2020.

[19] L. Weihs, U. Jain, I.-J. Liu, J. Salvador, S. Lazebnik, A. Kembhavi, and A. Schwing. Bridging the imitation gap by adaptive insubordination. In *NeurIPS*, 2021. the first two authors contributed equally.

[20] H. Team. Habitat CVPR challenge, 2021. URL https://aihabitat.org/challenge/2021/.

[21] S. Datta, O. Maksymets, J. Hoffman, S. Lee, D. Batra, and D. Parikh. Integrating egocentric localization for more realistic point-goal navigation agents. *CoRL*, 2020.

[22] L. Weihs, J. Salvador, K. Kotar, U. Jain, K.-H. Zeng, R. Mottaghi, and A. Kembhavi. Allenact: A framework for embodied ai research. *arXiv preprint arXiv:2008.12760*, 2020.

[23] C. Chen, U. Jain, C. Schissler, S. V. A. Gari, Z. Al-Halah, V. K. Ithapu, P. Robinson, and K. Grauman. Soundspaces: Audio-visual navigation in 3d environments. *ECCV*, 2020.

[24] C. Gan, J. Schwartz, S. Alter, D. Mrowca, M. Schrimpf, J. Traer, J. De Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, et al. Threedworld: A platform for interactive multi-modal physical simulation. *NeurIPS Datasets and Benchmarks Track*, 2021.

[25] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018.

[26] A. Suhr, C. Yan, J. Schluger, S. Yu, H. Khader, M. Mouallem, I. Zhang, and Y. Artzi. Executing instructions in situated collaborative interactions. In *EMNLP*, 2019.

[27] M. Hahn, J. Krantz, D. Batra, D. Parikh, J. M. Rehg, S. Lee, and P. Anderson. Where are you? localization from embodied dialog. In *EMNLP*, 2020.

[28] S. Raychaudhuri, S. Wani, S. Patel, U. Jain, and A. Chang. Language-aligned waypoint (law) supervision for vision-and-language navigation in continuous environments. In *EMNLP*, 2021.

[29] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, K. Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *CoRL*, 2022.

[30] A. Padmakumar, J. Thomason, A. Shrivastava, P. Lange, A. Narayan-Chen, S. Gella, R. Piramithu, G. Tur, and D. Hakkani-Tur. TEACh: Task-driven embodied agents that chat. *arXiv*, 2021.

[31] X. Puig, T. Shu, S. Li, Z. Wang, Y.-H. Liao, J. B. Tenenbaum, S. Fidler, and A. Torralba. Watch-and-help: A challenge for social perception and human-{ai} collaboration. In *ICLR*, 2021.

[32] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 2020.

[33] C. Pérez-D'Arpino, C. Liu, P. Goebel, R. Martín-Martín, and S. Savarese. Robot navigation in constrained pedestrian environments using reinforcement learning. In *ICRA*, 2021.

[34] U. Jain, L. Weihs, E. Kolve, M. Rastegari, S. Lazebnik, A. Farhadi, A. G. Schwing, and A. Kembhavi. Two body problem: Collaborative visual task completion. In *CVPR*, 2019.

[35] U. Jain, L. Weihs, E. Kolve, A. Farhadi, S. Lazebnik, A. Kembhavi, and A. G. Schwing. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *ECCV*, 2020.

[36] J. Thomason, M. Murray, M. Cakmak, and L. Zettlemoyer. Vision-and-dialog navigation. In *CoRL*, 2020.

[37] U. Jain, I.-J. Liu, S. Lazebnik, A. Kembhavi, L. Weihs, and A. G. Schwing. Gridtopix: Training embodied agents with minimal supervision. In *ICCV*, 2021.

[38] I.-J. Liu, Z. Ren, R. A. Yeh, and A. G. Schwing. Semantic tracklets: An object-centric representation for visual multi-agent reinforcement learning. In *IROS*, 2021.

[39] I.-J. Liu, U. Jain, R. A. Yeh, and A. Schwing. Cooperative exploration for multi-agent deep reinforcement learning. In *ICML*, 2021.

[40] M. Deitke, D. Batra, Y. Bisk, T. Campari, A. X. Chang, D. S. Chaplot, C. Chen, C. P. D'Arpino, K. Ehsani, A. Farhadi, et al. Retrospectives on the embodied ai workshop. *arXiv preprint arXiv:2210.06849*, 2022.

[41] O. Kwon, N. Kim, Y. Choi, H. Yoo, J. Park, and S. Oh. Visual graph memory with unsupervised representation for visual navigation. *ICCV*, 2021.

[42] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. *CVPR*, 2021.

[43] N. Hirose, F. Xia, R. Martín-Martín, A. Sadeghian, and S. Savarese. Deep visual mpc-policy learning for navigation. *RA-L*, 2019.

[44] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine. ViNG: Learning Open-World Navigation with Visual Goals. *ICRA*, 2020.

[45] D. Shah and S. Levine. ViKiNG: Vision-Based Kilometer-Scale Navigation with Geographic Hints. *RSS*, 2022.

[46] S. Wani, S. Patel, U. Jain, A. Chang, and M. Savva. Multion: Benchmarking semantic map memory using multi-object navigation. *NeurIPS*, 2020.

[47] S. Patel, S. Wani, U. Jain, A. Schwing, S. Lazebnik, M. Savva, and A. Chang. Interpretation of emergent communication in heterogeneous collaborative embodied agents. *ICCV*, 2021.

[48] P. Chattopadhyay, J. Hoffman, R. Mottaghi, and A. Kembhavi. Robustnav: Towards benchmarking robustness in embodied navigation. *ICCV*, 2021.

[49] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[50] C.-P. Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *TPAMI*, 2000.

[51] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *IJCV*, 2009.

[52] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. *CVPR*, 2020.

[53] S. Agarwala, L. Jin, C. Rockwell, and D. F. Fouhey. Planeformers: From sparse view planes to 3d reconstruction. *ECCV*, 2022.

[54] C. Rockwell, J. Johnson, and D. F. Fouhey. The 8-point algorithm as an inductive bias for relative pose prediction by vits. *3DV*, 2022.

[55] M. El Banani, L. Gao, and J. Johnson. UnsupervisedR&R: Unsupervised Point Cloud Registration via Differentiable Rendering. *CVPR*, 2021.

[56] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 2017.

[57] M. Labbé and F. Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *JFR*, 2019.

[58] P. Anderson, A. Shrivastava, J. Truong, A. Majumdar, D. Parikh, D. Batra, and S. Lee. Sim-to-real transfer for vision-and-language navigation. *CoRL*, 2020.

[59] J. Truong, S. Chernova, and D. Batra. Bi-directional domain adaptation for sim2real transfer of embodied navigation agents. *RA-L*, 2021.

[60] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *RA-L*, 2020.

[61] T. Chen, S. Gupta, and A. Gupta. Learning exploration policies for navigation. *ICLR*, 2019.

[62] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[63] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi. Simple but effective: Clip embeddings for embodied ai. *CVPR*, 2022.

[64] A. Eftekhar, A. Sax, J. Malik, and A. Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. *ICCV*, 2021.

[65] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018.

[66] D. S. Chaplot, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural mapping. *ICLR*, 2020.

[67] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. *CVPR*, 2018.

[68] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *3DV*, 2017.

[69] M. Bain and C. Sammut. A framework for behavioural cloning. *Machine Intelligence*, 1995.

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

[71] D. Lowe. Object recognition from local scale-invariant features. *ICCV*, 1999.

[72] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019.

[73] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. *CVPR*, 2015.

[74] V. A. Higuti, A. E. Velasquez, D. V. Magalhaes, M. Becker, and G. Chowdhary. Under canopy light detection and ranging-based autonomous navigation. *JFR*, 2019.

[75] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[76] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive Mapping and Planning for Visual Navigation. *CVPR*, 2017.

[77] S. Gupta, D. Fouhey, S. Levine, and J. Malik. Unifying map and landmark based representations for visual navigation. *arXiv preprint arXiv:1712.08125*, 2017.

[78] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi. Visual Semantic Planning using Deep Successor Representations. *ICCV*, 2017.

[79] L. Mezghani, S. Sukhbaatar, T. Lavril, O. Maksymets, D. Batra, P. Bojanowski, and K. Alahari. Memory-augmented reinforcement learning for image-goal navigation. *arXiv preprint arXiv:2101.05181*, 2021.

[80] M. V. Gasparino, A. N. Sivakumar, Y. Liu, A. E. B. Velasquez, V. A. H. Higuti, J. Rogers, H. Tran, and G. Chowdhary. Wayfast: Traversability predictive navigation for field robots. *IROS*, 2022.

[81] M. Camplani and L. Salgado. Efficient spatio-temporal hole filling strategy for kinect depth maps. *Three-dimensional image processing (3DIP) and applications Ii*, 2012.

[82] J. Shen and S.-C. S. Cheung. Layer depth denoising and completion for structured-light rgb-d cameras. *CVPR*, 2013.

Appendix – Last-Mile Embodied Visual Navigation

In this Appendix, we include additional details about the following:

Code of SLING with DDPPO-GD and OVRL-GD:
https://github.com/Jbwasse2/SLING

Code of SLING with Straight-GD, NRNS-GD, and Oracle-GD:
https://github.com/Jbwasse2/SLING/tree/nrns

A Implementation Details of SLING

Here we include finer details of SLING that we deferred from the main paper. It was straightforward to connect prior baselines with SLING (~ 100 lines of additional code) as prior baselines are reused largely. We were able to test SLING on the robot without any online fine-tuning in the real world.

Input configurations (Extending Sec. 4.1) At every time step the agent receives an RGB image, depth image, and the pose of the agent. Policies that are based on NRNS-GD and Straight-GD, following prior work [1], the images are given as 640×480 with a FoV of $120°$. Policies that are based on OVRL-GD and DDPPO-GD, for a head-on comparison, the agent is given 128×128 images with a FoV of $90°$. The pose is given as the position and heading of the agent in the environment.

Depth noise. We use the Redwood depth noise model [73] to insert noise into the depth image. For the pose noise, we follow the convention from prior work [1, 3], for a direct and fair comparison.

Pose noise. Prior work [66] built a Gaussian mixture model to capture pose noise from a real-world LoCoBot. We take the same error model, sample from it, and add the sampled noise to the agent's pose.

Local policy (Extending Sec. 3.3) For a fair comparison, we use the same local policy as prior visual navigation works [1, 66]. The local policy takes distance and heading to create a waypoint to navigate to. Building over (near-solved) setting of point-goal navigation, and a fast marching method to build a local map, the agent can localize itself and the goal and navigate towards it.

Additional hyperparameters. Beyond the hyperparameters of goal discovery modules (Straight-GD [61], DDPPO-GD [11, 6], NRNS-GD [1], OVRL-GD [6], and Oracle-GD), our last-mile navigation module introduces only a few hyperparameters which we include in Tab. 4. Note that we use different # of matches (50 in SLING + NRNS-GD *vs*. 20 in SLING + OVRL-GD) because the different methods use different input image sizes. No automated or grid-search tuning was conducted to find these hyperparameters.

B Goal Discovery Systems and Image-Goal Navigation Solvers (Extending Sec. 4.2 and Sec. 3.2)

The goal discovery modules, that show the compatibility and efficacy of SLING, are utilized in Sec. 4. Next, we include additional details for these.

Table 4: Key hyperparameter choices for SLING.

Hyperparamter	Value
last-mile navigation module	
Min # of Matches for *explore→exploit* switch (for NRNS/Straight)	50
Min # of Matches for *for explore→exploit* switch (OVRL)	20
Max Predicted Distance	4 meters
Confidence threshold for feature matcher module [52]	0.5

Behavior Cloning with Spatial Memory. Applies imitation learning (IL) wherein $\mathbf{I}_a$ and $\mathbf{I}_g$ are represented with ResNet18 [75]. Moreover, using the depth map $\mathbf{D}_a$, the observations are represented as a spatial metric map (found to be effective across embodied AI tasks [66, 46]).

Behavior Cloning with GRU. Another IL baseline where the observations at each time step are encoded identically to the above. However, instead of the spatial metric map, a GRU [70] is employed (CNN-RNN architectures have been effective for semantic navigation [76, 77, 23, 78]).

Zero Experience Replay (ZER) [5]. A recent plug-and-play RL policy trained using rewards obtained from moving closer to the goal and looking towards it, as well as view augmentation. The authors shared metrics over the folds of the benchmark [1], particularly Gibson-curved and *cross-domain* transfer results on MP3D, which we include in Tab. 1.

DDPPO [11], NRNS [1], OVRL [6]. These methods have been described as part of goal discovery, see Sec. 3.2. For DDPPO, we report results from Hahn *et al*. [1], trained for 100M steps (10x more compute than NRNS). For NRNS, we report the reproducible metrics from their official implementation (differs slightly from the paper [1]). At the time of submission, OVRL is the best-performing method on Gibson-curved fold of the benchmark [1]. For OVRL, we requested their checkpoints and re-evaluated them to report detailed metrics across easy-medium-hard folds.

DDPPO-LMN DDPPO [11] is a widely-adopted end-to-end deep RL baseline for embodied AI tasks [18, 10, 6] in AIHabitat [16]. We train DDPPO, exclusively for last-mile navigation. This last-mile navigation DDPPO (termed DDPPO-LMN) was trained on agents initialized at most 3m from the goal. DDPPO-LMN was trained to convergence on these trajectories over 400M steps, in the Gibson scenes. For a fair comparison, we allow DDPPO-LMN to use the same explore→exploit switch as SLING.

Straight-GD. Following the strategies of robot vacuums and studies in [61], this exploration module moves straight till it collides with an obstacle and then turns right (15°). A collision is estimated if, after completing an action, the pose difference between the agent's movement and its expected displacement is less than $0.1m$.

NRNS-GD. As introduced by Hahn *et al*. [1], NRNS utilizes four graph convolution layers to extract an embedding from the topological map. The extracted graph embedding (of size 768) along with the goal embedding are fed into a linear layer which predicts the distance estimate from the unexplored nodes to the goal. The next node that the agent navigates to is the node that minimizes this distance plus the distance from the agent to the node. We remove redundant nodes from being added to the topological map, which led to improved performance.

DDPPO-GD. For DDPPO-GD, we used the trained checkpoint we obtained from the OVRL [6] authors, added SLING over it and evaluated it on different data folds. The DDPPO agent was trained for 500M (NRNS [1] train for a max of 100M) steps over RGB observations on the episode dataset from [79].

OVRL-GD. For our OVRL experiments, we use the pretrained visual encoders provided by OVRL's authors. The downstream policy is trained using DDPPO on either the MP3D episode dataset (matching [1]) or the Gibson episode dataset (following [79]) for their respective experiments. We match OVRL's [6] training setup by using a set of 32 GPUs with 10 episodes each and train the agent for 500M steps. Each worker is allowed to collect up to 64 frames of experience in the environment and then trained using 2 PPO epochs with 2 mini-batches, with a learning rate of 2.5 x 10^{-4}.

Oracle-GD. This has the same architecture as NRNS-GD from Hahn *et al*. [1], where the agent builds a topological map to represent the environment. However, this method has privileged access, particularly, to the perfect distances from each node in the map to the goal. The planner will then

deterministically navigate the agent to the node in the map that has the lowest distance to the goal. We also found additional tweaks and edits to improve performance: (1) removing the agent to node distance and (2) removal of redundant nodes in the topological map.

Importantly, even when using Oracle-GD for goal discovery, the last-mile navigation modules do not have access to the ground truth distance to the goal. Furthermore, the oracle does not use its information to switch between goal discovery and last-mile navigation. So there is still a long way to perfect navigation, despite using a Oracle-GD.

C Results on Straight Data Folds (Extending Sec. 4.3)

In Tab. 5, we supplement the results of Gibson-curved and MP3D-curved (from Tab. 1 and Tab. 8), to include takeaways based on the straight counterparts. ZER [5] does not report results on the straight split, hence, could not be included in Tab. 5.

Table 5: **Results for 'Gibson-straight' and 'MP3D-straight' episodes.** Note the significant gains by adding SLING to prior works. SLING + NRNS-GD performs the best on Gibson-straight and SLING + Straight-GD performs best on MP3D-straight.

	Method	Overall		Easy		Medium		Hard	
		Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑
	Dataset = *Gibson-straight*								
1	BC w/ Spatial Memory [69]	12.5	12.1	24.8	23.9	11.5	11.2	1.3	1.2
2	BC w/ GRU State [69, 70]	19.5	18.7	34.9	33.4	17.6	17.0	6.0	5.9
3	DDPPO [11] (from [1])	29.0	26.8	43.2	38.5	36.4	34.8	7.4	7.2
4	OVRL [6]	44.9	30.0	53.6	34.7	48.6	33.3	32.5	21.9
5	NRNS [1]	47.3	39.8	70.1	62.7	50.7	41.5	21.2	15.4
6	SLING + Straight-GD	63.5	55.5	84.3	**79.0**	65.6	57.3	40.6	30.2
7	SLING + DDPPO-GD	38.6	26.0	54.9	39.5	41.0	27.2	20.0	11.3
8	SLING + OVRL-GD	58.1	42.5	71.2	54.1	60.3	44.4	43.0	29.1
9	SLING + NRNS-GD	**68.4**	**58.0**	**85.0**	76.8	**71.3**	**60.6**	**49.0**	**36.6**
	Dataset = *MP3D-straight*								
10	BC w/ Spatial Memory [69]	13.3	12.7	25.8	24.8	11.3	10.6	3.0	2.9
11	BC w/ GRU State [69, 70]	15.7	15.4	30.2	29.5	12.7	12.4	4.4	4.3
12	DDPPO [11] (from [1])	27.4	24.5	36.4	30.8	33.8	31.4	12.0	11.5
13	OVRL [6]	52.6	39.4	69.5	54.0	51.7	39.2	36.7	25.0
14	NRNS [1]	36.7	30.2	56.9	49.2	33.7	27.1	19.6	14.4
15	SLING + Straight-GD	**61.3**	**54.3**	**83.0**	**77.9**	**60.2**	**52.3**	**40.9**	**32.9**
16	SLING + DDPPO-GD [2]	31.7	21.4	49.6	36.2	31.4	20.1	14.2	7.9
17	SLING + OVRL-GD	58.3	47.1	78.8	68.5	58.7	46.3	37.4	26.5
18	SLING + NRNS-GD	60.6	49.6	82.0	76.1	59.1	46.5	40.8	26.3

State-of-the-art performance also on straight data folds. Similar to the curved results, utilizing SLING with previous goal discovery modules results in the highest performance across the Gibson-straight and MP3D-straight data folds. Over previous state-of-the-art (NRNS [1]), we improve the success rate by 21.1% (rows 5 & 9 under overall success) on Gibson and by 24.6% (rows 14 & 15 under overall success) on the MP3D dataset.

SLING significantly boosts all prior policies. On the Gibson dataset, using SLING improved the success rate on NRNS by 21.1% (rows 5 and 9), on DDPPO by 9.6% (rows 3 and 7), and OVRL by 13.2% (rows 4 and 8). Similar trends hold for the MP3D dataset. Quite surprisingly, even using the very simple Straight-GD with SLING (row 6) works really well for the straight fold. Note that this straight-exploring agent outperforms all other neural policies on MP3D.

[2]Due to limited compute, we were unable to retrain DDPPO-GD from scratch. Therefore, we use DDPPO-GD trained on Gibson, without SLING this model had an overall success and SPL of 9.0% and 4.4% respectively.

D Details of Switch Experiment (Extending Sec. 4.3)

In Sec. 4.3, under 'Geometric switches are better', we presented that our switches are more accurate. Particularly, SLING's explore→exploit switch is 92.0% accurate and MLP switch [1] is only at 82.1%. Also, SLING exploit→explore switch is 84.1% accurate while NRNS doesn't have such a recovery switch. These are summarized in Tab. 6. Next, we provide the deferred details of this study and evaluation data.

We sampled 500 image pairs per scene – 250 positives and 250 negatives for last-mile navigation. These are sampled randomly from 13 test environments (a total of 6500 image pairs). What is a positive for last-mile navigation? This is not well defined in prior works [1, 3, 4]. We say two views are positive (for last-mile navigation), if they are (1) less than $3m$ apart, (2) the angular difference is less than $22.5°$, and (3) the ratio of the geodesic distance over the euclidean distance is less than 1.2. An example of a positive and negative pair for last-mile navigation is visualized in Fig. 4a and Fig. 4b.

(a) Positive image pair

(b) Negative image pair

Figure 4: **Positive and negative pairs for last-mile navigation.** (a) The given image pair is similar (or positive) as the views are close and have significant overlap. (b) The image pair is dissimilar (or a negative) because they were taken in different rooms (their euclidean distance and geodesic to euclidean distance ratio are quite high (> 1.2).

Table 6: **Comparing accuracy of switches.** Our *explore→exploit* simple switching mechanisms are more accurate than MLP switches [1].

Switching Mechanism	*Explore→Exploit* Accuracy	*Exploit→Explore* Accuracy
MLP switch from NRNS [1]	82.1	N/A
SLING switch	**92.0**	84.1

E Robotic Experiments (Extending Sec. 4.4)

While we included all major real-robot results in the main paper (Sec. 4.4), we deferred several details, which we describe next.

Sensing details. The TerraSentia robot utilizes an Intel® RealSense™ D435i depth camera with a horizontal and vertical FoV of $69°$ and $42°$, respectively. The depth image is spatially aligned to the RGB image. When using Robot Operating System (ROS), the RGB and depth images are not necessarily published at the same time. Therefore, RGB and depth images are paired with the closest temporal message. To obtain the pose estimate, we utilize ORB-SLAM2 [56].

Safety. In order to protect the motors on the robot from getting damaged, and to make the image-goal task more realistic, we stop the robot when it crashes into an obstacle. After stopping we take the measurements needed to acquire the reported metrics.

Nonlinear model predictive control. Our real-robot system deploys Nonlinear Model Predictive Control (NMPC) [80] for the robot to execute actions. We utilize skid-steer dynamics to model the behavior of the TerraSentia. The controller optimizes the cost function consisting of penalties for errors between the robot's states and states up to and including the final estimation state, and the magnitude of the control input.

Environments and examples. We choose a diverse set of three scenes for our real-robot study including a total of 120 demonstrations. These environments are challenging, containing diverse

layouts and furniture, several obstacles, varying lighting conditions, long hallways, and visually-confounding common spaces (due to repeated patterns.) We show qualitative examples for each scene, particularly, a reconstruction from the robot, third-person views to show the scene, and trajectory examples. These trajectories and reconstructions were not used for real-world robotics experiments. They are strictly added for visualization purposes. The `office`, `department1`, and `department2` environments are visualized in Fig. 5, Fig. 6, and Fig. 7, respectively.

F Additional Results for Gibson-curved and MP3D-curved (Extending Sec. 4)

We supplement the results on the curved dataset. Particularly, we include ablations of SLING + OVRL-GD (Tab. 7), MP3D curved episodes (Tab. 8), and multi-seed runs (Tab. 9).

Ablation results on OVRL (Extending Tab. 2). We performed several ablations (see Sec. 4.2) on the SLING + NRNS-GD method in Tab. 2. We test the same on the SLING + OVRL-GD in Tab. 7. Results follow a trend similar to SLING + NRNS-GD, with performance drops when key components of SLING are taken away. The biggest drop (overall success drop from 55.4→37.9) is observed without our switches *i.e.* replacing our explore→exploit switch with MLP switch [1] and removing the exploit→explore switch (useful for error recovery). Notably, our method shows resistance to noise with an SPL change of +0.2% (rows 2 and 8) when depth noise is added. Unlike the NRNS-GD counterpart, OVRL-GD goal discovery module is quite less resilient to pose noise.

Table 7: **SLING + OVRL-GD ablations on 'Gibson-curved' episodes.** Ablations demonstrate the need for using two switches as well as utilizing learned features. Further testing demonstrates SLING + OVRL-GD is resilient to sensor noise.

		Overall		Easy		Medium		Hard	
	Method	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑
1	OVRL [6]	45.6	28.0	53.6	31.7	47.6	30.2	35.6	21.9
2	SLING + OVRL-GD	54.8	37.3	65.4	45.7	59.5	40.6	39.6	25.5
3	w/ MLP Switch	43.5	18.0	50.9	19.5	46.0	21.7	31.9	16.3
4	w/ MLP Switch w/o Recovery	37.9	16.7	47.7	18.1	43.0	21.3	28.2	15.2
5	w/o Neural Features	53.7	34.6	64.0	41.9	56.9	36.8	40.2	25.2
6	w/ Pose Noise	46.6	29.2	55.1	33.0	50.4	33.0	34.4	21.5
7	w/ Pose & Depth Noise	46.0	28.4	54.7	33.1	49.5	31.2	33.8	20.9
8	w/ Depth Noise	55.8	37.6	67.6	45.9	58.1	40.3	41.8	26.7

Additional in-domain MP3D results (extending Tab. 1). Consistent with previous results, SLING improves performance across several methods (compare rows 3 *vs*. 7, 4 *vs*. 8, and 5 *vs*. 9 of Tab. 8). Results for ZER are not included as they do not present results for this split.

Table 8: **Results for 'MP3D-curved' episodes.** Extending Tab. 1, adding SLING to prior works improves navigation results.

		Overall		Easy		Medium		Hard	
	Method	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑
	Dataset = *MP3D-curved*								
1	BC w/ Spatial Memory [69]	2.2	1.9	4.9	4.2	1.4	1.2	0.4	0.3
2	BC w/ GRU [69, 70]	1.3	1.1	3.1	2.6	0.8	0.7	0.1	0.0
3	DDPPO [11] (from [1])	12.9	10.0	17.9	13.2	15.0	12.1	5.9	4.8
4	NRNS [1]	15.5	7.4	23.1	10.8	15.1	7.3	8.4	4.1
5	OVRL [6]	41.6	24.4	52.4	35.2	42.6	26.3	**29.7**	16.9
6	SLING + Straight-GD	25.3	10.8	31.1	11.9	27.2	12.1	17.7	8.5
7	SLING + DDPPO-GD [3]	27.1	15.8	41.1	25.3	27.7	15.5	12.6	6.5
8	SLING + NRNS-GD	32.6	14.9	43.2	19.7	32.5	15.1	22.1	9.9
9	SLING + OVRL-GD	**46.7**	**30.1**	**62.6**	**41.1**	**48.4**	**31.5**	29.2	**17.7**

Table 9: **Multi-seed runs.** Mean and standard deviation of three runs with random seeds.

Method	Overall		Easy		Medium		Hard	
	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑
Dataset = *Gibson-curved*								
SLING + DDPPO-GD	39.4 ± 2.5	24.8 ± 2.3	52.8 ± 2.4	35.2 ± 2.8	43.4 ± 2.6	27.1 ± 2.9	22.0 ± 2.9	12.0 ± 1.4
SLING + OVRL-GD	56.3 ± 2.4	37.3 ± 0.7	66.6 ± 2.0	45.0 ± 0.7	61.2 ± 3.8	41.5 ± 1.7	41.1 ± 1.6	25.4 ± 0.8

Table 10: **Distance to goal decreases with SLING.** Adding SLING to previous goal discovery methods decreases their average distance from the final agent location to the goal.

Method	Overall Final Dist. ↓	Easy Final Dist. ↓	Med Final Dist. ↓	Hard Final Dist. ↓
DDPPO	3.05	2.28	2.75	4.11
SLING + DDPPO-GD	2.61	1.68	2.18	3.96
NRNS	2.96	1.99	2.74	4.15
SLING + NRNS-GD	2.42	1.41	2.10	3.75
OVRL	2.43	1.58	2.12	3.59
SLING + OVRL-GD	**2.17**	**1.28**	**1.70**	**3.52**

Multi-seed runs for OVRL-GD and DDPPO-GD (Extending Tab. 1). In this experiment we ran SLING + OVRL-GD and SLING + DDPPO-GD on 2 more seeds and present the results in Tab. 9. We see a slight improvement in the performance of both of the methods compared to the results presented in Tab. 1. Note that Yadav *et al*. [6] report overall success metrics and conduct a similar robustness study with 3 random seeds. They report a similar standard deviation for OVRL of 2.7% in overall success rate (SLING + OVRL-GD is 2.4%) and 1.7% in SPL (SLING + OVRL-GD is 0.7%).

G Final Distance to Goal for Top-Performing Baselines (Extending Sec. 4.3)

Recall, the final distance to goal metric reports the distance, from agent to goal, at the end of an episode. This metric is averaged across test episodes and reported in Tab. 10.

Across DDPPO, NRNS, and OVRL, consistent trends hold. First, SLING significantly reduces the final distance to goal. Next, the final distance to goal is much lower than the initial distance to the goal. As stated in averages, base OVRL starts ∼2.25m from the goal in easy episodes (1.5-3m) and reaches 1.58m from it, starts ∼4m from the goal in medium episodes (3-5m) and reaches 2.12m from it, and starts ∼7.5m from the goal in hard episodes (5-10m) and reached 3.59m from it. The final trend we find is that the final distance to goal is within range of last-mile navigation. Showing that last-mile navigation is a challenge for many previous methods.

H Potential of Last-Mile Navigation – 'Stop' Budget Study (Extending Sec. 2 and Sec. 3.4)

Recall that the image-goal navigation task can be completed either by calling the 'stop' action, or having the agent reach the maximum number of steps in an episode. In this study, we evaluate if last-mile navigation is a prominent error mode for image-goal navigation like it has been shown for other datasets and tasks [48, 46, 10]. Following the corresponding study for multi-object navigation [46], we study the performance of NRNS [1] as we increase the budget of the 'stop' action errors. This stop budget allows the agent to continue last-mile navigation beyond a hard failure, until this 'stop' budget is exhausted. As shown in Tab. 11, with just a budget of one, success increased from 28% to 51%. This shows that improving the last-mile of navigation and recovering from mistakes has immense potential that SLING taps into.

[3]Due to limited compute, we were unable to retrain DDPPO-GD from scratch. Therefore, we use DDPPO-GD trained on Gibson, without SLING this model had an overall success and SPL of 7.5% and 3.2% respectively.

Table 11: **Results testing tolerance towards.** Adding an increasing 'stop' budget causes the agent to perform better. This shows that being able to recover from mistakes has great potential to improve navigation success. †denotes that we edited the NRNS implementation to prevent redundant nodes from being added to the topological map. This leads to clear gains at no cost.

'Stop' action budget	Overall Success ↑	Overall SPL ↑
0 from [1]	21.7	8.1
0 (reproduced†)	27.8	10.7
1	50.8	17.0
2	68.4	21.4
3	81.2	25.1
4	90.9	28.3
5	96.2	30.0
6	98.8	31.0
7	99.8	31.2
8	100.0	31.3

I SLING with Panoramic Images (Extending Sec. 4.3)

Image-Goal navigation performance is highly correlated to the field-of-view (FoV) of the agent. This is intuitive as an agent that sees more about the environment and associated context in one observation will do better. Methods like NTS [3] and VGM [41] operate on panoramic observations and enjoy this advantage. However, other methods benchmarked in most prior works [6, 11, 5, 1] and ours operated on non-panoramic images. In order to have a fair comparison to methods that use panoramic images, we retrain OVRL with panoramic images and then apply SLING to this new model. To get SLING to work with panoramic images, we take the front-facing subsection of the panoramic goal and agent image and give it to SLING. For this experiment, we once again utilize the start and goal images on the Gibson-curved split, but with panoramic images. The results of this experiment are shown in Tab. 12 where we demonstrate that utilizing SLING with OVRL-GD yields the overall state-of-the-art on image-goal navigation while utilizing panoramic images. Notably, SLING improves the overall success and SPL of OVRL by 1.9% and 1.0% respectively (rows 2 and 3).

Table 12: **Results on the Gibson-Curved Panoramic dataset.** Adding SLING to OVRL allows us to improve their model to yield the new state-of-the-art when panoramic images are used for the image-goal navigation task.

		Overall		Easy		Medium		Hard	
	Method	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑	Succ↑	SPL↑
1	VGM [41]	74	51	81	46	79	60	62	**47**
2	OVRL [6]	76.7	59.4	88.8	71.8	78.6	62.9	62.9	43.6
3	SLING + OVRL-GD	**78.6**	**60.4**	**90.1**	**72.9**	**82.1**	**65.0**	**63.7**	43.4

J Limitations (Extending Sec. 5)

First, we rely on correspondences *i.e.* mistakes in keypoint feature extraction or matching failures directly lead to errors in predicted actions (neural features [65] reduce this effect). *Second*, as we add structure to the last-mile navigation problem, we also add design parameters like distance threshold d_{th} and correspondence threshold n_{th}. The latter we tuned depending on the size of the image (640×480 in NRNS and 128×128 in OVRL). *Third*, currently in SLING, we utilize only the agent's current observation for estimating distance and heading. Using temporal smoothening could make our prediction more robust.

Following the baselines proposed in the benchmark [1], we also assume access to depth and pose sensors. Studies in [5] also show improvements when using depth and pose sensors. For physical experiments, the robot comes equipped with an inexpensive depth camera and uses SLAM [56] for

pose estimations. While we demonstrate robustness to sensor noise (Tab. 2), in future work, we could try relaxing this assumption with a depth prediction module.

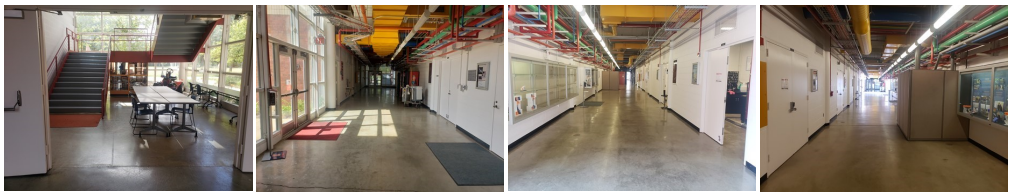
Pertinent to physical experiments, other limitations are:
(1) Errors in pose prediction when the keypoints are located in a small area of the image. However, this can be fixed heuristically with the exploit→explore switch.
(2) Large depth noises; this could be managed with various denoising techniques [81, 82].
(3) SLING must directly observe the goal image in order to have enough overlap to navigate to it. Because the current image-goal navigation success criterion only requires the agent to be within 1 meter from the goal, we can not take full advantage of the task definition. However, we assert that looking at the image-goal would be more aligned with how a human would attempt image-goal navigation.

(a) Scene snapshots

(b) SLING example trajectories.

(c) NRNS example trajectories.

Figure 5: `Office scene`. The map size is approximately $30m$ by $11m$. (a) office equipment serves as obstacles in this scene (b,c) Topdown map reconstruction with RTAB-Map of image-goal navigation task.



(a) Scene snapshots

(b) SLING examples

Figure 6: `Department1` **scene.** The Map size is approximately $95m$ by $32m$ (large width due to the LIDAR going through a door in the upper left side of image). (a) long corridors make images agent's views quite similar and make navigation challenging (b) Topdown map reconstruction with RTAB-Map of image-goal navigation task.

(a) Scene snapshots

(b) SLING examples

(c) NRNS examples.

Figure 7: `Department2` **scene.** The map size is approximately $42m$ by $22m$ for the shown floor. (a) several furniture items and specular floors are challenging for navigation, (b,c) Topdown map reconstruction with RTAB-Map of image-goal navigation task.