PEANUT: Predicting and Navigating to Unseen Targets

Albert J. Zhai, Shenlong Wang
University of Illinois at Urbana-Champaign
{azhai2, shenlong}@illinois.edu

Abstract

Efficient ObjectGoal navigation (ObjectNav) in novel environments requires an understanding of the spatial and semantic regularities in environment layouts. In this work, we present a straightforward method for learning these regularities by predicting the locations of unobserved objects from incomplete semantic maps. Our method differs from previous prediction-based navigation methods, such as frontier potential prediction or egocentric map completion, by directly predicting unseen targets while leveraging the global context from all previously explored areas. Our prediction model is lightweight and can be trained in a supervised manner using a relatively small amount of passively collected data. Once trained, the model can be incorporated into a modular pipeline for ObjectNav without the need for any reinforcement learning. We validate the effectiveness of our method on the HM3D and MP3D ObjectNav datasets. We find that it achieves the state-of-the-art on both datasets, despite not using any additional data for training. Code is available at https://ajzhai.github.io/PEANUT.

1. Introduction

Embodied visual navigation refers to a range of tasks that involve an agent navigating within the physical world based on visual sensory input [13]. One such task that has recently gained popularity is ObjectGoal navigation, also known as ObjectNav [2]. Here, the agent is placed in an unknown environment and is tasked with navigating to a specific object category (e.g. "toilet"). ObjectNav is a fundamentally important task for embodied AI because it tests both scene understanding and long-term memory. It has immediate applications for helping people with disabilities find objects in their homes. It is also a necessary precursor for many other semantic tasks, such as instruction following [1] and question answering [12].

In ObjectNav, the agent is placed in a random location in an unknown environment and is given access to RGB-D observations, its own pose, and a target category. An episode is considered successful if the agent stops near an object of the target category within a given time limit. In order to perform this task efficiently, the agent must leverage priors about environment layouts while searching for the target. For example, imagine that you observe the image in Figure 1 and your target category is "bed". The room on the right does not have any visible furniture but has a good chance to be a bedroom. On the other hand, the room on the left has what appears to be a mirror above a countertop, suggesting that it is a bathroom that probably does not contain a bed. The room directly in front could be a bedroom, but it is farther away than the other two. Thus, it would be a good idea to look inside the room on the right first and the room directly in front second. An ideal ObjectNav agent should be able to make such predictions about unexplored areas and reason about the uncertainty in its predictions to plan efficient search routes.

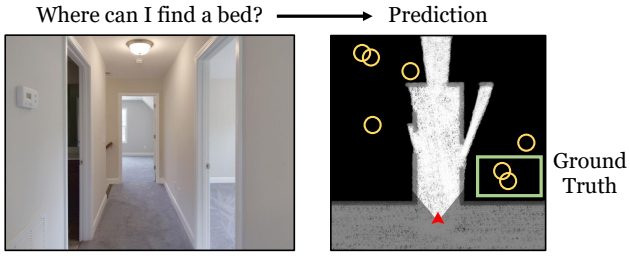

Figure 1. **Explicit Prediction for ObjectNav.** To find a semantic target in a novel environment, an agent may make explicit guesses about where the target might be. Humans excel at this type of reasoning and can generate a diverse set of reasonable guesses instantaneously. On the right is a top-down map representing the parts of the environment that have been observed. Yellow circles represent multiple human guesses as to where a bed might be.

Existing methods for ObjectNav can be divided into two categories: end-to-end learning methods and modular methods. End-to-end methods aim to learn a policy that directly maps sensor observations to actions. The policy is usually modeled as a recurrent neural network and learned either using reinforcement learning [40, 45, 30, 26, 14] or imitation learning [38]. Such methods are flexible in the policies that they can produce, but they require large amounts of data, incur high computational costs (especially for RL), and lack

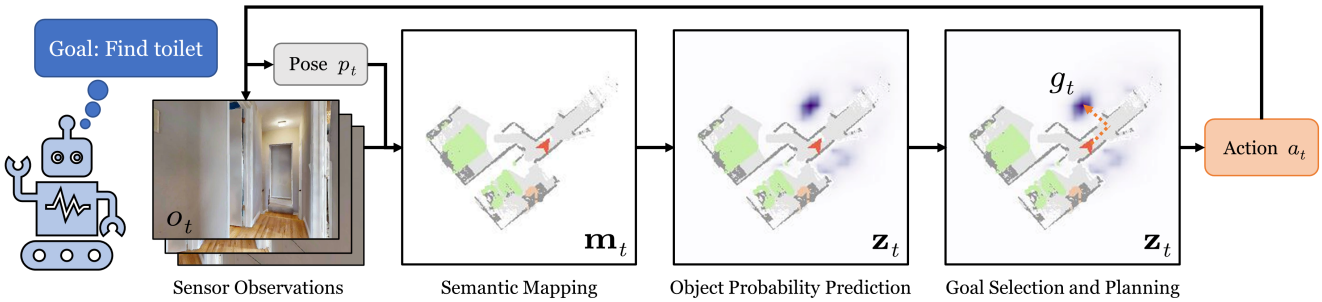

Figure 2. **Overview of PEANUT.** At each step, the agent's RGB-D observation o_t and pose observations p_t are used to update the incomplete global semantic map $\mathbf{m}_t$. This map is then used to predict a target object probability map $\mathbf{h}_t$, which is used to select long-term goals g_t. Finally, an analytical local planner is employed to calculate the low-level actions necessary to reach g_t.

interpretability. Modular methods combat these issues by decomposing the task into subproblems. Current modular methods for ObjectNav start by building a semantic map of the environment [6, 10], and then employ one module for high-level goal selection ("where to look") and another module for low-level action planning ("how to get there").

The "how to get there" subproblem has been studied independently under the name of PointNav [13]. The key question for ObjectNav is how to solve the "where to look" subproblem. Recently, several works have proposed to tackle the "where to look" subproblem as an explicit prediction task by training a network to make predictions about what lies in the unexplored areas of the environment using supervised learning [28, 18, 36]. This explicit-prediction paradigm is especially attractive because it evades the sample inefficiency and high computational cost of RL [18, 36].

Previous explicit-prediction methods for ObjectNav have tried to predict unseen targets from single-view observations [18, 28] or estimate potential functions at the frontiers of the global semantic map [36]. We believe using the context provided by the global semantic map is crucial for making informative predictions. However, we choose to directly predict unseen targets instead of frontier potential functions, which do not model uncertainty and may be difficult to predict accurately.

In this paper, we introduce PrEdicting And Navigating to Unseen Targets (PEANUT), a novel explicit-prediction method for ObjectNav that predicts target object probabilities in the unexplored areas of the environment based on the agent's semantic map. Unlike previous map prediction methods [28, 18] that rely on single-view egocentric context, PEANUT performs prediction in a global, allocentric context. This approach allows the agent to leverage information from previous timesteps and is consistent with cognitive models of human and animal navigation [3, 17].

In order to select long-term goals, we take the target probability prediction and apply a simple distance-weighting scheme that encourages the agent to search closer locations before moving on to faraway ones. This produces a value map, which we select long-term goals from by simply taking the argmax. Combining this goal selection module with an analytical low-level planner yields a modular pipeline for ObjectNav that does not require any reinforcement learning and can be trained in less than one GPU day.

We evaluate PEANUT on the Habitat-Matterport3D (HM3D) dataset [37] and the Matterport3D (MP3D) dataset, which are the settings for the 2022 and 2021 Habitat ObjectNav Challenges respectively. On HM3D, PEANUT outperforms all previous published methods, including methods that utilize additional datasets for training. On MP3D, PEANUT also outperforms all previous methods, including recent modular methods. In our ablation studies, we perform experiments to demonstrate the benefit of using global context for prediction.

2. Related Work

ObjectGoal Navigation. ObjectNav remains to be a challenging task and is an active area of research in embodied AI. Existing methods for ObjectNav can be classified as either end-to-end learning methods or modular methods. End-to-end learning methods seek to learn a mapping from sensor observations to low-level actions using a single neural network, usually consisting of a visual encoder followed by a recurrent cell that aggregates information over time. Most end-to-end learning methods train the network using reinforcement learning (RL), although recently, Habitat-Web [38] collected a dataset of 80K human demonstrations and achieved strong performance using imitation learning. Significant progress has been made in end-to-end RL by introducing auxiliary tasks [45], different reward functions [45, 30], and different visual encoders such as CLIP-pretrained backbones [26], DINO-pretrained backbones [42], and graph convolutional networks on object graphs [44, 15] . Many recent works focus on finding ways to leverage additional data for training [14, 42, 8, 30]. Currently, the state-of-the-art method for ObjectNav is ProcTHOR [14], which generated thousands of synthetic scenes and applied end-to-end RL with a CLIP backbone.

Modular methods seek to build a hierarchical policy in which one module handles high-level goal selection ("where to look") and another module handles low-level action planning ("how to get there") The advantages of this decomposition are that 1) the individual problems may be easier to solve, 2) their solutions can be reused across different tasks, and 3) the final pipeline is more interpretable than an end-to-end policy.

In particular, the "how to get there" subproblem is equivalent to the task known as PointGoal Navigation (PointNav), in which the agent is tasked with reaching a given (x, y) location. In 2019, it was shown that near-perfect PointNav performance could be achieved in a noiseless setting using end-to-end RL [40]. Subsequent works focused on accounting for noisy actuation and noisy sensor readings, which are important factors to consider for real-world deployment [25]. By 2022, multiple works had shown that accurate localization could be achieved in noisy settings using either learned visual odometry models [47, 34, 9] or LiDAR-based SLAM [19], leading to the discontinuation of the Habitat PointNav Challenge [13].

However, the "where to look" subproblem remains active and is the primary focus of research on modular ObjectNav. One approach is to treat this purely as an undirected exploration problem and try to maximize the area explored by the agent [9, 29, 43]. This lends itself to simple heuristics that can perform well without any learning, but is ultimately limited in its efficiency. To leverage semantic priors for goal selection, SemExp [10] built a top-down semantic map and used it to train an RL-based policy that minimizes distance to target objects, winning the 2020 Habitat ObjectNav Challenge. More recently, a few works [36, 18] have proposed to select goals directly from supervised predictions about unexplored areas of the environment, avoiding RL entirely. This is the paradigm that we use in this paper. We discuss explicit prediction methods further in the next subsection.

Explicit Prediction for Navigation. Efficient navigation in unknown environments requires prior models of environment layouts. End-to-end learning methods for navigation can make implicit predictions about unobserved areas of the environment, as suggested by visualizations of learned feature maps [20]. Several works in recent years have aimed to predict specific properties of the environment layout for which explicit supervision can be obtained, allowing for straightforward offline training procedures.

Such explicit-prediction methods have appeared in a variety of navigation tasks, including PointNav [35], RoomNav [33], ImageNav [11, 21], and ObjectNav [28, 36, 18]. The prediction outputs range from occupancy maps [35] to distances to the target [36, 21] to semantic maps [28, 18]. The works that predict semantic maps for ObjectNav are the most similar to our approach. SSCNav [28] predicts the unexplored semantic map in a window around the agent along with a confidence map, and uses them as input to an RL-based goal selection policy. L2M [18] uses an ensemble of map predictors and selects goals by maximizing an upper confidence bound on the probability of the target category. However, both of these methods make predictions in an egocentric window using sensor inputs from only one timestep, whereas our method makes predictions on the allocentric global map using the information gathered from all previous timesteps. Another method, PONI [36], predicts potential functions (a combination of the connected free space and the proximity to the target) at the frontiers of the map in a global manner. We demonstrate in our experiments that directly predicting target object probabilities outperforms PONI in terms of navigation performance.

3. Approach

Problem Formulation. ObjectNav is an episodic navigation task. In each episode, the agent starts at a random location in an unknown environment and is given a target category $c_{target} \in \{1, 2, \ldots, C\}$, where C is the number of possible target categories. At each timestep t, the agent observes an RGB-D image o_t and an (x, y, θ) pose reading p_t, and then must output one of four discrete actions: `MOVE_FORWARD`, `TURN_LEFT`, `TURN_RIGHT`, and `STOP`. An episode is considered successful if the agent executes `STOP` within $1.0\,\mathrm{m}$ of a target object and the object can be viewed from the agent's position. Each episode has a time limit of 500 steps.

Overview. Our proposed method, PEANUT, is a modular approach for ObjectNav (Fig. 2). Like most modular approaches, PEANUT uses a top-down semantic map as its internal representation of the environment. At the heart of our method is a novel unseen object prediction model. It addresses the question of "where to look" by making explicit predictions of the probability of the target appearing in each unexplored location of the semantic map, extracting a value map from these predictions, and then selecting the highest-value location as the agent's goal. We train our prediction model via supervised learning using a dataset of actual semantic maps collected by a passive exploration agent.

We decompose the PEANUT pipeline into four components: semantic mapping, unseen object probability prediction, prediction-based goal selection, and low-level planning. For semantic mapping, we use the same procedure as prior works [10, 36], which we review in section 3.1. In section 3.2, we describe our explicit prediction model and how we gather data for training the model. In section 3.3, we describe our simple strategy for selecting the agent's long term goal based on the prediction output. In section 3.4, we describe the analytical local policy we use for planning actions to reach the goal.

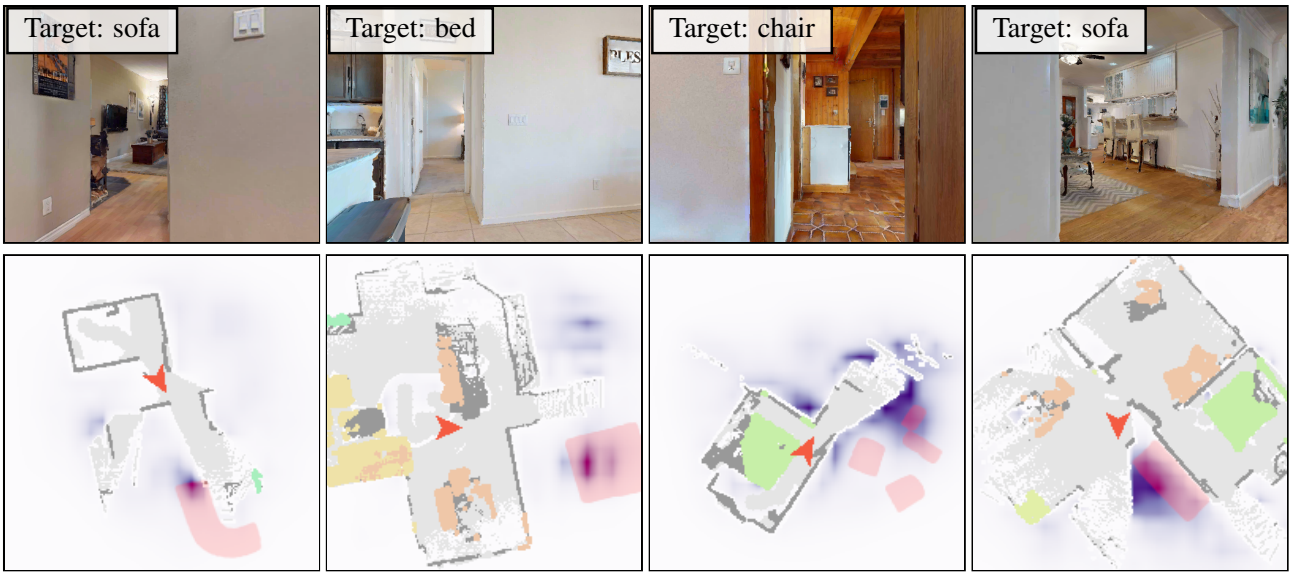

Figure 3. **Example target predictions**. We visualize some predictions made by our model in four different scenes from HM3D (val). The top row shows the agent's RGB observation, and the bottom row shows the incomplete semantic map overlaid with the target probability prediction (in purple) and ground-truth unseen objects (in light red). The model has learned to use semantic cues to improve its predictions, as demonstrated by its ability to predict the sofa facing the TV on the left. It is also able to produce multimodal predictions, which is especially apparent in the "bed" prediction. Note that the prediction network only takes the semantic map as input, not the RGB images.

3.1. Semantic Mapping

Our semantic mapping module follows the same procedure as previous works in semantic exploration [10, 36]. The module assumes access to RGB-D observations and the agent's pose. At each timestep, the RGB-D image is segmented into semantic categories using an existing segmentation model. The set of categories must include at least the C target categories, but may also include other categories. Then, the depth observation is used to lift each pixel along with its semantic label to 3D. Points that are within a certain height range (based on the agent's height) are marked as obstacles. The point cloud is then converted into a voxel occupancy grid and is summed across the height dimension to obtain an egocentric map. This egocentric map is transformed into an allocentric coordinate system using the agent's pose and is aggregated with the existing global map.

In total, the semantic map $\mathbf{m}_t$ has $(N+4) \times H \times W$ elements, where N is the number of semantic categories. Channels 1 and 2 represent the obstacle map and the explored areas. Channels 3 and 4 contain the agent's current location and all past locations. The next C channels correspond to the C target semantic categories. The other $N-C$ channels represent the remaining semantic categories.

3.2. Object Probability Prediction

The core component of our navigation pipeline is the explicit prediction model, which represents the agent's belief of what lies outside the regions of the environment that it has explored. Most of the time, there are many possibilities as to what the environment layout may be, so the model must be able to express uncertainty in its predictions. Thus, we represent the prediction as another top-down map, aligned with $\mathbf{m}_t$, in which each element contains the probability of each target category appearing in that location. We train a network to predict these probabilities using the entire global map $\mathbf{m}_t$ as input, unlike previous works that predict based on observations from a single frame [28, 18]. We choose to directly predict target locations instead of potential functions [36], which vary greatly depending on the layout of unseen obstacles. We hypothesize that by modeling uncertainty and focusing on prediction of target objects instead of obstacles, we are able to learn a useful prediction model more effectively from limited amounts of data.

Problem Formulation. Given an incomplete semantic map and a set of target categories, we wish to predict for every unexplored map location the probability of each target category appearing in that location. Formally, let $\mathbf{m}_t \in \mathbb{R}^{(N+4)\times H\times W}$ be the incomplete map as described above, and let $\mathcal{C} = \{1, 2, \ldots, C\}$ be the set of possible target categories. Let $\mathbf{e}_t$ be an exploration mask indicating which locations have been explored, and let $\mathbf{M} \in \mathbb{R}^{C\times H\times W}$ be the ground-truth full semantic map of the environment containing the C target categories. For each location (i, j) satisfying $\mathbf{e}_t[i, j] = 0$ and each target category $c \in \mathcal{C}$, we wish to learn a model $f_\theta(c, i, j|\mathbf{m}_t)$ of the probability that an object of category c exists at location (i, j) in the complete map $\mathbf{M}$, given the incomplete map $\mathbf{m}_t$.

Network Architecture and Loss Function. We formulate this as a dense pixel prediction task and train the network to predict zero in the already-explored areas. This allows us to use any image-to-image network architecture for f_θ. In this work, we use a PSPNet [46] architecture, but modify the number of input channels to $N+4$ and the number of output channels to C. A detailed description of the architecture is provided in the supplementary material.

To train the network, we calculate training outputs $\mathbf{y}_t$ from ground-truth maps $\mathbf{M}$ by setting the already-explored areas to zero:

$$\mathbf{y}_t[c,i,j] = (1 - \mathbf{e}_t[i,j])\mathbf{M}[c,i,j]. \tag{1}$$

We then train using binary cross-entropy loss, averaged over the C categories:

$$\begin{aligned} L = \frac{1}{CHW} \sum_{c,i,j} (&\mathbf{y}_t[c,i,j] \log f_\theta(c,i,j|\mathbf{m}_t) \\ &+ (1 - \mathbf{y}_t[c,i,j]) \log(1 - f_\theta(c,i,j|\mathbf{m}_t))). \end{aligned} \tag{2}$$

Training Data Generation. We generate training data for our prediction task from semantic maps collected during episodes of exploration. Using the mapping procedure described in section 3.1, we let an exploration agent wander around for 500 steps in different scenes from different starting locations and save the incomplete map every 25 steps. In our experiments, the agent follows the baseline exploration strategy from [29] due to its empirical efficiency, but any reasonable exploration strategy can be applied instead. We use the maps from steps 25 to 250 as incomplete input maps $\mathbf{m}_t$ and generate $\mathbf{M}$ from the map at step 500, where the agent has usually explored the entire scene.

Inference. During navigation, we simply apply the prediction network on the collected semantic map $\mathbf{m}_t$ and extract the probability map $\mathbf{z}_t$ corresponding to c_{target}:

$$\mathbf{z}_t[i,j] = f_\theta(c_{target}, i, j|\mathbf{m}_t), \tag{3}$$

and use it to select long-term goals for the agent. We update the prediction every 10 steps or whenever the previous goal has been reached.

3.3. Prediction-Based Goal Selection

We provide a simple method for selecting long-term goals using the target probability predictions $\mathbf{z}_t$. A simple way to plan using $\mathbf{z}_t$ is to set the goal g_t to be the location with the highest probability $\operatorname{argmax}_{(i,j)} \mathbf{z}_t[i,j]$. However, it is usually more efficient for the agent to search in locations closer to its current location before moving on to locations farther away. Thus, for each location (i,j), we use the Fast Marching Method [39] to calculate the geodesic distance $d_t(i,j)$ from the agent's current location using the current obstacle map (channel 1 of $\mathbf{m}_t$). The geodesic distance to (i,j) is the length of the shortest path to (i,j), which is a better estimate of the time needed to reach (i,j) compared to the Euclidean distance. We then use d_t as an exponential weight factor for $\mathbf{z}_t$ and set g_t to be the argmax of the weighted result. Formally,

$$g_t = \operatorname*{argmax}_{(i,j)} \exp\left(-d_t(i,j)/\lambda\right)\mathbf{z}_t[i,j], \tag{4}$$

where λ is a tunable parameter that allows for a tradeoff between higher-probability locations and closer locations. Note that the calculated geodesic distance is not aware of any obstacles outside what the agent has currently seen – it is fully optimistic about traversability in unexplored areas.

3.4. Analytical Local Policy

Our local policy converts the long-term goal g_t into low-level actions using the same technique as previous works [9, 10, 36]. It computes the shortest path in terms of (i,j) locations using the Fast Marching Method and then extracts a waypoint from this path using the agent's step distance. Replanning is done at every timestep. We also deploy the recovery behavior described in [29] when the agent gets stuck between obstacles.

4. Experiments

We evaluate PEANUT on standard ObjectNav datasets [37, 7] and show that it outperforms previous methods, including state-of-the-art methods that rely on additional data for training. We also perform several ablation experiments to study the effect of various design choices in our pipeline.

4.1. Experimental Setup

Datasets. We conduct experiments on both the Habitat-Matterport3D (HM3D) dataset [37] and the Matterport3D (MP3D) dataset [7] using the Habitat simulator [31]. Both datasets contain 3D reconstructions of real-world indoor environments from which realistic RGB-D observations can be rendered. HM3D is more recent and is of slightly higher quality, but there are not as many publicly available results for it compared to MP3D.

For HM3D, our setup is consistent with the 2022 Habitat ObjectNav Challenge [41], which has 6 goal categories and 80 train / 20 val / 20 test scenes. For MP3D, our setup is consistent with the 2021 Habitat ObjectNav Challenge [2], which has 21 goal categories and 56 train / 11 val / 18 test scenes. In both settings, the agent's RGB-D observation has a resolution of 640×480 with a horizontal field of view of 79 degrees. The step distance for `MOVE_FORWARD` is $0.25\,\mathrm{m}$ and the turn angle is 30 degrees.

Figure 4. **Example navigation episode using PEANUT.** We visualize a full episode of navigation in a scene from HM3D (val). The top row shows the agent's RGB observation, and the bottom row shows the incomplete semantic map overlaid with the target probability prediction. The agent's selected long-term goal is marked by an orange cross. At $t = 1$, without having much information, the agent's prediction is somewhat random. At $t = 10$, it updates its prediction to assign higher probabilities around the hallway to the right, and begins to move over there. At $t = 20$, after seeing more of the hallway, it updates its prediction to sharply favor the open room on the right (which is indeed a bathroom). At $t = 49$, the agent detects the toilet, and at $t = 61$, it ends the episode successfully.

Metrics. We evaluate navigation performance using two standard metrics. **Success** is the proportion of episodes in which the agent successfully stopped at a goal object. **SPL** is the success weighted by the length of the agent's path relative to the oracle shortest path length [2].

Baselines. For HM3D, we compare PEANUT with four public methods. To the best of our knowledge, these are the only methods for which papers are available.

- **DD-PPO** [40] applies end-to-end RL with large-scale distributed training.
- **Habitat-Web** [38] applies end-to-end imitation learning using human teleoperated demonstrations.
- **OVRL** [42] pretrains a visual encoder using DINO [5] on images from the Omnidata dataset [16] before applying end-to-end RL.
- **ProcTHOR** [14] applies end-to-end RL on a large procedurally generated dataset of synthetic scenes. Unlike the other methods, it does not use depth and pose observations as inputs. It currently holds the state-of-the-art SPL on HM3D.

For MP3D, we compare PEANUT with eight other methods. Of these eight methods, five use end-to-end learning:

- **DD-PPO** [40], **Habitat-Web** [38], and **OVRL** [42] are described above.
- **Red Rabbit** [45] incorporates 6 auxiliary tasks for end-to-end RL. It is the winner of the 2021 Habitat ObjectNav Challenge.
- **TreasureHunt** [30] artificially inserts objects from the YCB dataset [4] into MP3D scenes to provide additional data for end-to-end RL.

The other three methods are modular methods:

- **ANS** [9] is a modular method that trains an RL-based goal selection policy to maximize area explored.
- **PONI** [36] is a modular method that predicts potential functions on the frontiers of a top-down map, and then selects goals by picking the highest-potential frontier. The potential function is a sum of the connected free space and the inverse distance to the target.
- **Stubborn** [29] is a modular method that simply sets the goal to be one of four corners of a local crop of the map, rotating when reaching a dead-end. It also heuristically fuses target detections across frames. It is currently the best-performing modular method on MP3D.

Implementation Details. Our semantic mapping module requires a 2D semantic segmentation model. For HM3D, we finetune a COCO-pretrained Mask-RCNN [22] (on images from HM3D) to predict the 6 goal categories along with 3 other categories (*fireplace*, *mirror*, and *bathtub*). For MP3D, we use the publicly available RedNet [24] model from [45], which predicts 21 object categories.

To train PEANUT, we collect maps as described in section 3.2 via Habitat. For fair comparisons with other works, we only train on scenes from MP3D when evaluating ObjectNav performance on MP3D, and likewise for HM3D,

Table 1. ObjectNav results on HM3D (test-standard). The "Ext. Data" column indicates whether the method uses non-HM3D data for anything beyond pretraining a 2D perception model.

Method	SPL (↑)	Success (↑)	Ext. Data
DD-PPO [40]	0.12	0.26	no
Habitat-Web [38]	0.22	0.55	yes
OVRL [42]	0.27	0.60	no
ProcTHOR [14]	0.32	0.54	yes
PEANUT (Ours)	**0.33**	**0.64**	no

although better performance may be achieved by combining the two or adding other datasets. For both MP3D and HM3D, we collect sequences of maps from 50 starting locations for each scene. This gives 4000 train / 1000 val sequences for HM3D and 2800 train / 550 val sequences for MP3D. During training, we apply random rotation, flip, and padded-crop operations for data augmentation.

Our prediction network uses a PSPNet [46] architecture with a ResNet50 [23] backbone and an auxiliary loss weight of 0.4. For both datasets, we use an Adam optimizer [27] with $\alpha = 0.0005$, $(\beta_1, \beta_2) = (0.9, 0.999)$, and a batch size of 8, with α decaying according to the "poly" learning rate policy described in [46]. For HM3D, we stop training at 28 000 iterations. For MP3D, we stop training at 22 000 iterations. For goal selection, we set $\lambda = 5\,\mathrm{m}$ based on validation performance.

4.2. ObjectNav on HM3D

Quantitative Results. We report results from the test-standard split of the 2022 Habitat ObjectNav Challenge leaderboard in Tab. 1. PEANUT achieves an SPL of 0.33 and a success rate of 0.64, outperforming all other published methods in both metrics. Note that two of the other methods rely on additional data for training their navigation policy, in the form of procedurally generated scenes for ProcTHOR [14] and human demonstrations for Habitat-Web [38]. PEANUT does not use any data outside of HM3D other than through the COCO-pretrained Mask-RCNN [22], and is trained on a relatively small amount of interaction within HM3D. By making globally-informed predictions about where the target objects may be, PEANUT can select goals that help the agent search the scene efficiently.

Qualitative Results. We show example target predictions made by our prediction model in Fig. 3. We can see that the model has learned strong priors on the spatial regularities of indoor layouts, enabling it to make rather sharp predictions. In addition, it is able to exploit semantic regularities such as the co-occurrence of TVs and sofas. We can also confirm that the model is able to produce multimodal predictions, reflecting the inherent uncertainty in the prediction task.

We visualize a full episode of navigation using PEANUT in Fig. 4. We observe that the prediction-based goal selection policy allows the agent to search for the target in an intelligent manner. We also notice that the prediction can change dramatically over a short timeframe, suggesting that the model takes advantage of new information quickly.

Table 2. ObjectNav results on MP3D (val). The "Ext. Data" column indicates whether the method uses non-MP3D data for anything beyond pretraining a 2D perception model.

Method	SPL (↑)	Success (↑)	Ext. Data
DD-PPO [40]	0.018	0.080	no
Habitat-Web [38]	0.102	0.354	yes
OVRL [42]	0.074	0.286	no
Red-Rabbit [45]	0.079	0.346	no
TreasureHunt [30]	0.110	0.284	yes
ANS [9]	0.092	0.273	no
PONI [36]	0.121	0.227	no
Stubborn [29]	0.149	**0.407**	no
PEANUT (Ours)	**0.158**	0.405	no

4.3. ObjectNav on MP3D

Quantitative Results. We report results on the MP3D val split (2195 episodes) in Tab. 2. For DD-PPO [40] and ANS [9], we report results from [36]. For Stubborn [29], we obtain results by running the official code. For all other methods, we report results from their respective papers. As with HM3D, PEANUT outperforms all previous methods on MP3D in terms of SPL. Its success rate of 0.405 is matched only by Stubborn [29], which fuses detection scores over time for more robust target detection. PEANUT significantly outperforms PONI [36], another explicit-prediction method. A possible reason for this is that frontier potentials vary widely depending on the layout of unseen obstacles and are thus difficult to estimate accurately.

Qualitative Results. We show example target predictions from PEANUT and compare them with potential functions estimated by PONI [36] in Fig. 5. The PONI predictions were obtained using their publicly available pretrained model. We find that PEANUT's predictions appear to be more informative and useful than PONI's, as PONI often predicts high potentials for frontiers that are not worth exploring (such as in the corners of already-explored rooms).

4.4. Discussions

Global Context for Prediction. We study the effect of leveraging full global context for prediction as opposed to only using single-frame observations for prediction, as done by SSCNav [28] and L2M [18]. We emulate egocentric single-frame maps by taking $6\,\mathrm{m} \times 6\,\mathrm{m}$ crops of $\mathbf{m}_t$ from random locations along the agent's trajectory, and masking

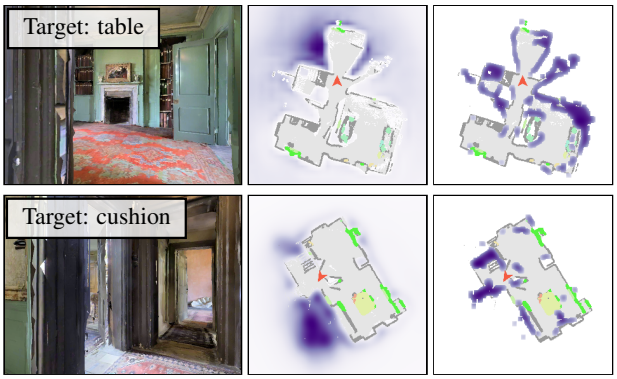

Figure 5. **Qualitative comparison with potential functions (PONI) [36].** We visualize predictions on MP3D (val) made by PEANUT (left) and compare them with potential functions estimated by PONI (right). We observe that PEANUT tends to make cleaner, sharper predictions than PONI, which often assigns high potentials to frontiers that clearly cannot lead to any target object.

out everything outside of the agent's viewing frustum projected onto the map. Note that this actually gives more information than an actual single-frame observation, since it does not account for occlusion from the agent's view. We apply random rotation and flipping and train a network to predict object probabilities in the unexplored areas of the crop in the exact same manner as the global context version.

We compare prediction quality using two metrics: **DTO** looks at the highest-probability location from the prediction and measures its Euclidean distance to the nearest ground-truth object (if one exists). **NLL** is the average negative log-likelihood over the ground-truth object locations after dividing the prediction probabilities by their sum across all locations. We evaluate the metrics over 4000 crops from our HM3D val map dataset and report the results in Tab. 3. We observe that the predictions made using global context are significantly more accurate than those using single-frame inputs. Since more accurate predictions enable more efficient downstream navigation, we conclude that leveraging global context is beneficial for prediction-based navigation.

Distance Weighting for Goal Selection. We study the effect of weighting the predicted probabilities by their distance to encourage selection of closer goals. We also evaluate an agent that does not use predictions at all and always navigates to the nearest unexplored location, which is equivalent to frontier-based exploration [43]. For this agent, we clip the minimum distance to $3\,\mathrm{m}$ to prevent excessive back-and-forth movement. Navigation performance measured over 500 episodes from the HM3D val split is reported in Tab. 4. We find that, without distance weighting, the agent performs slightly worse because it is may ignore closer target objects in favor of farther ones. Without predictions, the agent performs significantly worse because it wastes time searching in useless locations.

Table 3. Effect of using global context instead of single-frame egocentric context for prediction on maps from HM3D (val).

Input	DTO (↓)	NLL (↓)
Egocentric Crop	1.974	9.460
Global Map	**1.518**	**8.873**

Table 4. Ablation studies on HM3D (val). We study the effect of 1) not using distance-weighting (DW) during goal selection, 2) not using the target predictions (Pred), and 3) using ground-truth target segmentation (GT).

Pred	DW	GT Seg	SPL (↑)	Success (↑)
✓	✓		**0.320**	**0.638**
✓			0.310	0.598
	✓		0.235	0.478
✓	✓	✓	0.400	0.756

Noisy Actuation and Imperfect Localization. We also measure PEANUT's performance on the same HM3D val split under noisy actuation and without using the Habitat-provided pose observations. We use the PyRobot LoCoBot actuation noise model [25, 32] with the noise multiplier set to 0.1, and perform localization using a simple depth-based odometry algorithm, which at each step tracks the relative transformations by using a grid filter to minimize point-to-point distance with a uniform motion prior. We find that PEANUT achieves an SPL of 0.305 and a success rate of 0.610 in this setting, indicating that it is reasonably robust to some degree of noisy actuation and localization.

Limitations. A significant amount of PEANUT's failures are caused by errors in the initial semantic segmentation, as demonstrated by the large increase in performance when using ground-truth target segmentation instead of the Mask-RCNN (Tab. 4). Another source of failures is scenes that contain multiple floors. Since PEANUT relies on a single top-down map, it is unable to traverse staircases and is prone to getting stuck within them. We also note that the above experiments are performed with noiseless depth inputs, and performance may degrade with noisy depth.

5. Conclusion

We presented PEANUT, a modular method for Object-Goal navigation that predicts unseen target objects from a top-down semantic map and navigates to them. The prediction model can be trained in a supervised manner using passively collected maps. Unlike previous map prediction methods, our method uses the global context from all explored areas to produce more accurate predictions, which in turn allows for more efficient navigation. Experiments showed that PEANUT outperforms the state-of-the-art on standard ObjectNav benchmarks. In the future, PEANUT

may be improved by leveraging RGB and 3D shape information to build even more powerful prediction models.

Acknowledgments. This project is partially funded by the Illinois Smart Transportation Initiative STII-21-07, an Amazon Research Award, an Intel Research Gift, and an IBM IIDAI grant. We also thank NVIDIA for their Academic Hardware Grant.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 1

[2] 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. 1, 5, 6

[3] Daniel Bush, Caswell Barry, Daniel Manson, and Neil Burgess. Using grid cells for navigation. *Neuron*, 87(3):507–520, 2015. 2

[4] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha S Srinivasa, Pieter Abbeel, and Aaron M Dollar. Ycb benchmarking project: Object set, data set and their applications. *Journal of The Society of Instrument and Control Engineers*, 56(10):792–797, 2017. 6

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *CVPR*, 2021. 6

[6] Vincent Cartillier, Zhile Ren, Neha Jain, Stefan Lee, Irfan Essa, and Dhruv Batra. Semantic mapnet: Building allocentric semantic maps and representations from egocentric views. In *AAAI*, 2021. 2

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *3DV*, 2017. 5

[8] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Semantic visual navigation by watching youtube videos. In *NeurIPS*, 2020. 2

[9] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2019. 3, 5, 6, 7

[10] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *NeurIPS*, 2020. 2, 3, 4, 5

[11] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, 2020. 3

[12] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *CVPR*, 2018. 1

[13] Matt Deitke, Dhruv Batra, Yonatan Bisk, Tommaso Campari, Angel X Chang, Devendra Singh Chaplot, Changan Chen, Claudia Pérez D'Arpino, Kiana Ehsani, Ali Farhadi, et al. Retrospectives on the embodied ai workshop. *arXiv preprint arXiv:2210.06849*, 2022. 1, 2, 3

[14] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, et al. Procthor: Large-scale embodied ai using procedural generation. *arXiv preprint arXiv:2206.06994*, 2022. 1, 2, 6, 7

[15] Heming Du, Xin Yu, and Liang Zheng. Learning object relation graph and tentative policy for visual navigation. In *ECCV*. Springer, 2020. 2

[16] 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. 6

[17] Uğur M Erdem and Michael Hasselmo. A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *European Journal of Neuroscience*, 35(6):916–931, 2012. 2

[18] Georgios Georgakis, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, and Kostas Daniilidis. Learning to map for active semantic goal navigation. In *ICLR*, 2021. 2, 3, 4, 7

[19] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Devendra Singh Chaplot. Navigating to objects in the real world. *arXiv*, 2022. 3

[20] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017. 3

[21] Meera Hahn, Devendra Singh Chaplot, Shubham Tulsiani, Mustafa Mukadam, James M Rehg, and Abhinav Gupta. No rl, no simulation: Learning to navigate without navigating. In *NeurIPS*, 2021. 3

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 6, 7

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7

[24] 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. 6

[25] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *RAL*, 2020. 3, 8

[26] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. In *CVPR*, 2022. 1, 2

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[28] Yiqing Liang, Boyuan Chen, and Shuran Song. Sscnav: Confidence-aware semantic scene completion for visual semantic navigation. In *ICRA*, 2021. 2, 3, 4, 7

[29] Haokuan Luo, Albert Yue, Zhang-Wei Hong, and Pulkit Agrawal. Stubborn: A strong baseline for indoor object navigation. *arXiv preprint arXiv:2203.07359*, 2022. 3, 5, 6, 7

[30] Oleksandr Maksymets, Vincent Cartillier, Aaron Gokaslan, Erik Wijmans, Wojciech Galuba, Stefan Lee, and Dhruv Batra. Thda: Treasure hunt data augmentation for semantic navigation. In *ICCV*, 2021. 1, 2, 6, 7

[31] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *ICCV*, 2019. 5

[32] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019. 8

[33] Medhini Narasimhan, Erik Wijmans, Xinlei Chen, Trevor Darrell, Dhruv Batra, Devi Parikh, and Amanpreet Singh. Seeing the un-scene: Learning amodal semantic maps for room navigation. In *ECCV*. Springer, 2020. 3

[34] Ruslan Partsey, Erik Wijmans, Naoki Yokoyama, Oles Dobosevych, Dhruv Batra, and Oleksandr Maksymets. Is mapping necessary for realistic pointgoal navigation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17232–17241, 2022. 3

[35] Santhosh K Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation. In *ECCV*. Springer, 2020. 3

[36] 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. 2, 3, 4, 5, 6, 7, 8

[37] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 2, 5

[38] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *CVPR*, 2022. 1, 2, 6, 7

[39] James A. Sethian. Fast marching methods. *SIAM Rev.*, 41(2):199–235, 1999. 5

[40] 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. *arXiv preprint arXiv:1911.00357*, 2019. 1, 3, 6, 7

[41] Karmesh Yadav, Santhosh Kumar Ramakrishnan, John Turner, Aaron Gokaslan, Oleksandr Maksymets, Rishabh Jain, Ram Ramrakhya, Angel X Chang, Alexander Clegg, Manolis Savva, Eric Undersander, Devendra Singh Chaplot, and Dhruv Batra. Habitat challenge 2022. https://aihabitat.org/challenge/2022/, 2022. 5

[42] 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. 2, 6, 7

[43] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *CIRA*, 1997. 3, 8

[44] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018. 2

[45] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *ICCV*, 2021. 1, 2, 6, 7

[46] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 5, 7

[47] Xiaoming Zhao, Harsh Agrawal, Dhruv Batra, and Alexander G Schwing. The surprising effectiveness of visual odometry techniques for embodied pointgoal navigation. In *CVPR*, pages 16127–16136, 2021. 3