In Proceedings of the European Conference on Computer Vision (ECCV), 2020

Occupancy Anticipation for Efficient Exploration and Navigation

Santhosh K. Ramakrishnan[1,2], Ziad Al-Halah[1], and Kristen Grauman[1,2]

[1] The University of Texas at Austin, Austin TX 78712, USA
[2] Facebook AI Research, Austin TX 78701, USA
srama@cs.utexas.edu, ziadlhlh@gmail.com, grauman@cs.utexas.edu

Abstract. State-of-the-art navigation methods leverage a spatial memory to generalize to new environments, but their occupancy maps are limited to capturing the geometric structures directly observed by the agent. We propose *occupancy anticipation*, where the agent uses its egocentric RGB-D observations to infer the occupancy state beyond the visible regions. In doing so, the agent builds its spatial awareness more rapidly, which facilitates efficient exploration and navigation in 3D environments. By exploiting context in both the egocentric views and top-down maps our model successfully anticipates a broader map of the environment, with performance significantly better than strong baselines. Furthermore, when deployed for the sequential decision-making tasks of exploration and navigation, our model outperforms state-of-the-art methods on the Gibson and Matterport3D datasets. Our approach is the winning entry in the 2020 Habitat PointNav Challenge. *Project page: `http://vision.cs.utexas.edu/projects/occupancy_anticipation/`*

1 Introduction

In visual navigation, an agent must move intelligently through a 3D environment in order to reach a goal. Visual navigation has seen substantial progress in the past few years, fueled by large-scale datasets and photo-realistic 3D environments [4,9,74,69], simulators [74,34,3,38], and public benchmarks [13,3,38]. Whereas traditionally navigation was attempted using purely geometric representations (i.e., SLAM), recent work shows the power of *learned* approaches to navigation that integrate both geometry and semantics [79,20,56,41,77,11]. Learned approaches operating directly on pixels and/or depth as input can be robust to noise [11,10] and can generalize well on unseen environments [20,38,77,10] —even outperforming pure SLAM given sufficient experience [38].

One of the key factors for success in navigation has been the movement towards complex map-based architectures [20,46,11,10] that capture both geometry [20,11,10] and semantics [20,46,19,24], thereby facilitating efficient policy learning and planning. These learned maps allow an agent to exploit prior knowledge from training scenes when navigating in novel test environments.

Despite such progress, state-of-the-art approaches to navigation are limited to encoding *what the agent actually sees in front of it.* In particular, they build maps

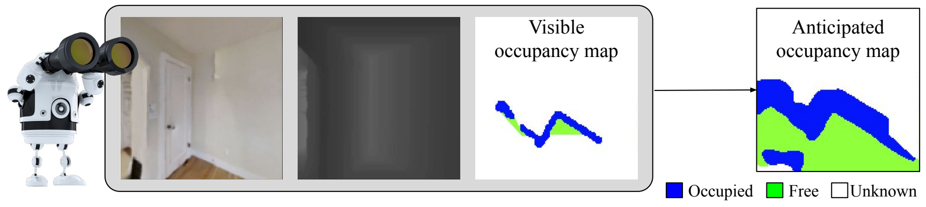

Fig. 1: **Occupancy anticipation:** A robot's perception of the 3D world is limited by its field-of-view and obstacles (the visible map). We propose to anticipate occupancy for unseen regions (anticipated map) by exploiting the context from egocentric views. We then train a deep reinforcement learning agent to move intelligently in a 3D environment, rewarding movements that improve the anticipated map.

of the environment using only the *observed* regions, whether via geometry [11,24] or learning [20,46,19,10]. Thus, while promising, today's models suffer from an important inefficiency: to map a space in the 3D environment as free or occupied, the agent must directly see evidence thereof in its egocentric camera.

Our key idea is to *anticipate occupancy*. Rather than wait to directly observe a more distant or occluded region of the 3D environment to declare its occupancy status, the proposed agent infers occupancy for unseen regions based on the visual context in its egocentric views. For example, in Fig. 1, with only the partial observation of the scene, the agent could infer that it is quite likely that the wall extends to its right, a corridor is present on its left, and the region immediately in front of it is free space. Such intelligent extrapolation beyond the observed space would lead to more efficient exploration and navigation. To achieve this advantage, we introduce a model that anticipates occupancy maps from normal field-of-view RGB(D) observations, while aggregating its predictions over time in tight connection with learning a navigation policy. Furthermore, we incorporate the anticipation objective directly into the agent's exploration policy, encouraging movements in the 3D space that will efficiently yield broader and more accurate inferred occupancy maps.

We validate our approach on Gibson [74] and Matterport3D [9], two 3D environment datasets spanning over 170 real-world spaces with a variety of obstacles and floor plans. Using only RGB(D) inputs to anticipate occupancy, the proposed agent learns to explore intelligently, achieving faster and more accurate maps compared to a state-of-the-art approach for neural SLAM [10], and navigating more efficiently than strong baselines. Furthermore, for navigation under noisy actuation and sensing, our agent improves the state of the art, winning the 2020 Habitat PointNav Challenge [1] by a margin of 6.3 SPL points.

Our main contributions are: (1) a novel occupancy anticipation framework that leverages visual context from egocentric RGB(D) views; (2) a novel exploration approach that incorporates intelligent anticipation for efficient environment mapping, providing better maps in less time; and (3) successful navigation results that improve the state of the art.

2 Related work

Navigation Classical approaches to visual navigation perform passive or active SLAM to reconstruct geometric point-clouds [71,23] or semantic maps [5,55], facilitated by loop closures or learned odometry [7,39,8]. More recent work uses deep learning to learn navigation [79,20,56,41,77,75,59,64] or exploration [48,6,57,28,51] policies in an end-to-end fashion. Explicit *map-based* navigation models [21,46,19,11] usually outperform their implicit counterparts by being more sample-efficient, generalizing well to unseen environments, and even transferring from simulation to real robots [20,10]. However, existing approaches only encode *visible* regions for mapping (i.e., the ground plane projection of the observed or inferred depth). In contrast, our model goes beyond the visible cues and anticipates maps for unseen regions to accelerate navigation.

Layout estimation Recent work predicts 3D Manhattan layouts of indoor scenes given 360 panoramas [80,76,70,73,15]. These methods predict structured outputs such as layout boundaries [80,70], corners [80], and floor/ceiling probability maps [76]. However, they do not extrapolate to unseen regions. FloorNet [36] and Floor-SP [29] use walkthroughs of previously scanned buildings to reconstruct detailed floorplans that may include predictions for the room type, doors, objects, etc. However, they assume that the layouts are polygonal, the scene is fully explored, and that detailed human annotations are available. Our occupancy map representation can be seen as a new way for the agent to infer the layout of its surroundings. Unlike any of the above approaches, our model does not make strict assumptions on the scene structure, nor does it require detailed semantic annotations. Furthermore, the proposed anticipation model is learned jointly with the exploration policy and without human guidance. Finally, unlike prior work, our goal is to accelerate navigation and map creation.

Scene completion Past work in scene completion focuses on pixelwise reconstruction of 360 panoramas with limited glimpses [28,50,51,61], inpainting [49,26,35], and inferring unseen 3D structure and semantics [68,78]. While some methods allow pixelwise extrapolation outside the current field of view (FoV) [51,68,78,27], they do not permit inferences about occluded regions in the scene. Our results show that this limitation is detrimental to successful occupancy estimation (cf. our view extrapolation baseline). SSCNet [67] performs voxelwise geometric and semantic predictions for unseen 3D structures; however, it is computationally expensive, requires voxelwise semantic labels, limits predictions to the agent's FoV, and needs carefully curated viewpoints for training. In contrast, our approach predicts 2D occupancy from egocentric RGB(D) views, and it learns to do so in an active perception setting. Since the agent controls its own camera, the viewpoints tend to be more challenging than those in curated datasets of human-taken photos used in the scene completion literature [67,68,28,50,78].

Occupancy maps In robotics, methods for occupancy focus on building continuous representations of the world [45,53,62], mapping for autonomous driving [25,40,66,37,42], and indoor robot navigation [31,16,65]. Prior extrapolation methods assume wide FoV LIDAR inputs, only exploit geometric cues from top-down views, and demonstrate results in relatively simple 2D floorplans devoid of non-wall obstacles [32,31,16,65]. In contrast, our approach does not require expensive LIDAR sensors. It operates with standard RGB(D) camera inputs, and it exploits both semantic and geometric context from those egocentric views to perform accurate occupancy anticipation. Furthermore, we demonstrate efficient navigation in visually rich 3D environments with challenging obstacles other than walls. Finally, unlike prior work, our anticipation models are learned jointly with a navigation policy that rewards accurate anticipatory mapping.

3 Approach

We propose an occupancy anticipation approach for efficient exploration and navigation. Our model anticipates areas not directly visible to the agent because of occlusion (e.g., behind a table, around a corner) or due to being outside its FoV. The agent's first-person view is provided in the form of RGB-D images (see Fig. 2 left). The goal is to anticipate the occupancy for a fixed region in front of the agent, and integrate those predictions over time as the agent moves about.

Next, we define the task setup and notation, followed by our approach for occupancy anticipation (Sec. 3.1) and a new formulation for exploration that rewards correctly anticipated regions (Sec. 3.2). Then, we explain how our occupancy anticipation model can be integrated into a state-of-the-art approach [10] for autonomous exploration and navigation in 3D environments (Sec. 3.3).

3.1 Occupancy anticipation model

We formulate occupancy anticipation as a pixelwise classification task. The egocentric occupancy is represented as a two-channel top-down map $p \in [0,1]^{2\times V \times V}$ which comprises a local area of $V \times V$ cells in front of the camera. Each cell in the map represents a 25mm $\times$ 25mm region. The two channels contain the probabilities (confidence values) of the cell being occupied and explored, respectively. A cell is considered to be occupied if there is an obstacle, and it is explored if we know whether it is occupied or free. For training, we use the 3D meshes of indoor environments (Sec. 4.1) to obtain the ground-truth local occupancy of a $V \times V$ region in front of the camera, which includes parts that may be occluded or outside the field of view (Fig. 2, bottom right).

Our occupancy anticipation model consists of three main components (Fig. 2):
(1) Feature extraction: Given egocentric RGB-D inputs, we compute:
RGB CNN features: We encode the RGB images using blocks 1 and 2 of a ResNet-18 that is pre-trained on ImageNet, followed by three additional convolution layers that prepare these features to be passed forward with the visible occupancy map. This step extracts a mixture of textural and semantic features.
Depth projection: We estimate a map of occupied, free, and unknown space by setting height thresholds on the point cloud obtained from depth and camera intrinsics [11]. Consistent with past work [11,10], we restrict the projection-based

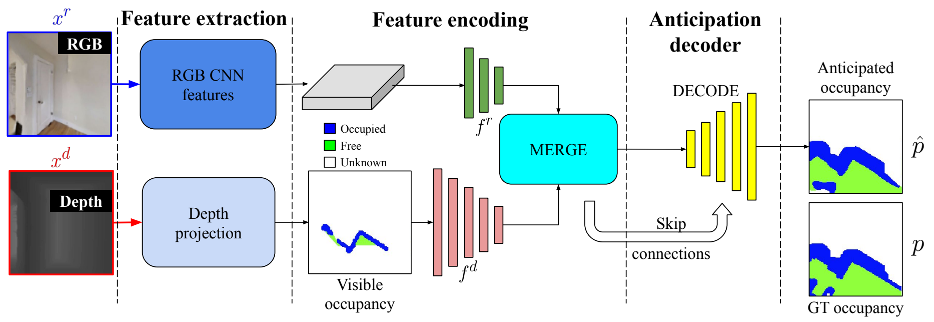

Fig. 2: Our occupancy anticipation model uses RGB(D) inputs to extract features, and processes them using a UNet to anticipate the occupancy. The depth map is projected to the ground plane to obtain the preliminary visible occupancy map. See text.

estimates to points within $\sim$ 3m, the range at which modern depth sensors would provide reliable results. This yields the initial visible occupancy map.

(2) Feature encoding: Given the RGB-D features, we independently encode them using UNet [54] encoders and project them to a common feature space. We encode the depth projection features using a stack of five convolutional blocks which results in features $\boldsymbol{f}^d = f^d_{1:5}$. Since the RGB features are already at a lower resolution, we use only three convolutional blocks to encode them, which results in features $\boldsymbol{f}^r = f^r_{3:5}$. We then combine these features using the MERGE module which contains layer-specific convolution blocks to merge each $[\boldsymbol{f}^r_i, \boldsymbol{f}^d_i]$:

$$\boldsymbol{f} = \textsc{merge}(\boldsymbol{f}^d, \boldsymbol{f}^r). \tag{1}$$

For experiments with only the depth modality, we skip the RGB feature extractor and MERGE layer and directly use the occupancy features obtained from the depth image. For experiments with only the RGB modality, we learn a model to infer the visible occupancy features from RGB (to be defined at the end of Sec. 4.1) and use that instead of the features computed from the depth image.

(3) Anticipation decoding: Given the encoded features $\boldsymbol{f}$, we use a UNet decoder that outputs a $2 \times V \times V$ tensor of probabilities:

$$\hat{p} = \sigma(\textsc{Decode}(\boldsymbol{f})), \tag{2}$$

where $\hat{p} \in [0,1]^{2 \times V \times V}$ is the estimated egocentric occupancy and σ is the sigmoid activation function. For training the occupancy anticipation model, we use binary cross entropy loss per pixel and per channel:

$$L = \sum_{i=1}^{V^2} \sum_{j=1}^{2} -\Big[p_{ij} \log \hat{p}_{ij} + (1 - p_{ij}) \log(1 - \hat{p}_{ij})\Big], \tag{3}$$

where p is the ground-truth (GT) occupancy map that is derived from the 3D mesh of training environments (see Sec. S5 in Supp. for details).

So far, we have presented our occupancy anticipation approach supposing a single RGB-D observation as input. However, our model is ultimately used in the context of an embodied agent that moves in the environment and actively collects a sequence of RGB-D views to build a complete map of the environment. Next, we introduce a new reward function that utilizes the agent's anticipation performance to guide its exploration during training.

3.2 Anticipation reward for exploration policy learning

In *visual exploration*, an agent must quickly map a new environment without having a specified target. Prior work on exploration [11,17,10,52] often uses area-coverage—the area seen in the environment during navigation—as a reward function to guide exploration. However, the traditional area-coverage approach is limited to rewarding the agent only for *directly seeing* areas. Arguably, an ideal exploration agent would obtain an accurate and complete map of the environment *without* necessarily directly observing all areas.

Thus, we propose to encourage exploratory behaviors that yield a correctly *anticipated* map. In this case, the occupancy entries in the map need not be obtained via direct agent observations to register a reward; it is sufficient to correctly infer them. In particular, we reward agent actions that yield accurate occupancy predictions for the global environment map, i.e., the number of grid cells where the predicted occupancy matches the layout of the environment.

More concretely, let $\hat{m}_t \in [0,1]^{2\times G\times G}$ be the global environment map obtained by anticipating occupancy for the RGB-D observations $\{x^r_{1:t}, x^d_{1:t}\}$ from time 1 to t, and then geometrically registering the predictions to a single global map based on the agent's pose estimates at each time step (see Fig. 3). Note $G > V$. Let m be the ground-truth layout of the environment. Then, the unnormalized accuracy of a map prediction $\hat{m}$ is measured as follows:

$$\text{Accuracy}(\hat{m}, m) = \sum_{i=1}^{G^2}\sum_{j=1}^{2} \mathbb{1}[\hat{m}_{ij} = m_{ij}], \tag{4}$$

where $\mathbb{1}[\hat{m}_{ij} = m_{ij}]$ is an indicator function that returns one if $\hat{m}_{ij} = m_{ij}$ and zero otherwise. We reward the increase in map accuracy from time $t-1$ to t:

$$R_t^{anticp} = \text{Accuracy}(\hat{m}_t, m) - \text{Accuracy}(\hat{m}_{t-1}, m). \tag{5}$$

This function rewards actions leading to correct global map predictions, irrespective of whether the agent actually *observed* those locations. For example, if the agent correctly anticipates free space behind a table and is rewarded for that, it then learns to avoid spending additional time around tables in the future to observe that space directly. Resources can be instead allocated to visiting more interesting regions that are harder to anticipate. Additionally, this reward

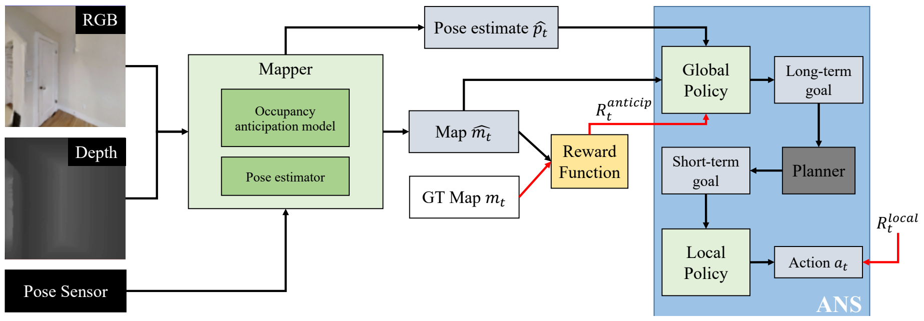

Fig. 3: **Exploration with occupancy anticipation:** We introduce two key upgrades to the original Active Neural SLAM (ANS) model [10] (see text): (1) We replace the projection unit in the mapper with our occupancy anticipation model (see Fig. 2). (2) We replace the area-coverage reward function with the proposed reward (Eqn. 5), which encourages the agent to efficiently explore and build accurate maps through occupancy anticipation. Note that the reward signals (in red) are provided only during training.

provides a better learning signal while training under noisy conditions by accounting for mapping errors arising from noisy pose and map predictions. Thus, our approach encourages more intelligent exploration behavior by injecting our anticipated occupancy idea directly into the agent's sequential decision-making.

3.3 Exploration and navigation with occupancy anticipation

Having defined the core occupancy anticipation components, we now demonstrate how our model can be used to benefit embodied navigation in 3D environments. We consider both exploration (discussed above) and *PointGoal navigation* [58,2], a.k.a PointNav, where the agent must navigate efficiently to a target specified by a displacement vector from the agent's starting position.

For both tasks, we adapt the state-of-the-art Active Neural SLAM (ANS) architecture [10] that previously achieved the best exploration results in the literature and was the winner of the 2019 Habitat PointNav challenge. However, our anticipation model is generic and can be easily integrated with most map-based embodied navigation models [20,11,18].

The ANS model is a hierarchical, modular policy for exploration that consists of a mapper, a planner, a local policy, and a global policy (shown in Fig. 3). Given RGB images, the mapper estimates the egocentric occupancy and agent pose, and then temporally aggregates the maps into a global top-down map using the pose estimates. At regular time intervals Δ, the global policy picks a location on the global map to explore. A shortest-path planner decides what trajectory to take from the current position to the target and picks an intermediate goal (within 1.25m) to navigate to. The local policy then selects actions that lead to the intermediate goal; it gets another intermediate goal upon reaching the current goal. See [10] for details. Critically, and like other prior work, the model of [10] is supervised to generate occupancy estimates based solely on the *visible* occupancy obtained from the egocentric views.

We adapt ANS by modifying the mapper and the reward function. For the mapper, we replace the projection unit from ANS with our anticipation model (see Fig. 3). Additionally, we account for incorrect occupancy estimates in two ways: (1) we filter out high entropy predictions and (2) we maintain a moving average estimate of occupancy at each location in the global map (see Sec. S7 in Supp.). For the reward function, we use the anticipation-based reward presented in Sec. 3.2.

We train the exploration policy with our anticipation model end-to-end, as this allows adapting to the changing distribution of the agent's inputs. Both the local and the global reinforcement learning policies are trained with Proximal Policy Optimization (PPO) [60]. In our model, the reward of the global policy is our anticipation-based reward defined in Eqn. 5. This replaces the traditional area-coverage reward used in ANS and other current models [11,10,52], which rewards the increment in the actual area seen, not the correctly registered area in the map. The reward for the local policy is simply based on the reduction in the distance to the local goal: $R_t^{local} = d_{t-1} - d_t$, where d is the Euclidean distance between the current position and the local goal.

4 Experiments

In the following experiments we demonstrate that 1) our occupancy anticipation module can successfully infer unseen parts of the map (Sec. 4.2) and 2) trained together with an exploration and navigation policy, it accelerates active mapping and navigation in new environments (Sec. 4.3 and Sec. 4.4).

4.1 Experimental setup

We use the Habitat [38] simulator along with Gibson [74] and Matterport3D [9] environments. Each dataset contains around 90 challenging large-scale photo-realistic 3D indoor environments such as houses and office buildings. On average, the Matterport3D environments are larger. Our observation space consists of 128×128 RGB-D observations and odometry sensor readings that denote the change in the agent's pose x, y, θ. Our action space consists of three actions: MOVE-FORWARD by 25cm, TURN-LEFT by $10°$, TURN-RIGHT by $10°$. For navigation, we add a STOP action, which the agent emits when it believes it has reached the goal. We simulate noisy actuation and odometer readings for realistic evaluation (see Sec. S6 in Supp.).

We train our exploration models on Gibson, and then transfer them to PointGoal navigation on Gibson and exploration on Matterport3D. We use the default train/val/test splits provided for both datasets [38] with disjoint environments across the splits. For evaluation on Gibson, we divide the validation environments into small (area less than 36m^2) and large (area greater than 36m^2) to observe the influence of environment size on results. For policy learning, we use

Method	IoU %			F1 score %		
	free	occ.	mean	free	occ.	mean
all-free	30.1	0	15.1	43.6	0	21.8
all-occupied	0	25.1	12.6	0	39.2	19.6
ANS(rgb)	12.1	14.9	13.5	19.6	24.9	22.5
ANS(depth)	14.5	24.1	19.3	23.1	37.6	30.4
View-extrap.	15.5	26.4	21.0	25.0	40.4	32.7
OccAnt(rgb)	44.4	47.9	46.1	58.2	62.9	60.6
OccAnt(depth)	50.4	**61.9**	56.1	63.8	**75.0**	69.4
OccAnt(rgbd)	**51.5**	61.5	**56.5**	**64.9**	74.8	**69.8**

Table 1: **Occupancy anticipation results** on the Gibson validation set. Our models, OccAnt($\cdot$), substantially improve the map quality and extent, showing the advantage of learning to anticipate 3D structures beyond those directly observed.

the Adam optimizer and train on episodes of length 1000 for $1.5 - 2$ million frames of experience. Please see Sec. S8 in Supp. for more details.

Baselines: We define baselines based on prior work:

– **ANS(rgb)** [10]: This is the state-of-the-art Active Neural SLAM approach for exploration and navigation. We use the original mapper architecture [10], which infers the visible occupancy from RGB.[3]
– **ANS(depth)**: We use depth projection to infer the visible occupancy (similar to [11]) instead of predicting it from RGB.
– **View-extrap.**: We extrapolate an 180° FoV depth map from 90° FoV RGB-D and project it to the top-down view. This is representative of scene completion approaches [68,78]. See Sec. S11 in Supp. for network details.
– **OccAnt(GT)**: This is an upper bound that cheats by using the ground-truth anticipation maps for exploration and navigation.

We implement all baselines on top of the ANS framework. Our goal is to show the impact of our occupancy model, while fixing the backbone navigation architecture and policy learning approach across methods for a fair comparison. We consider three versions of our models based on the input modality:

– **OccAnt(depth)**: anticipate occupancy given the visible occupancy map.
– **OccAnt(rgb)**: anticipate occupancy given only the RGB image. We replace the depth projections in Fig. 2 with the pre-trained ANS(rgb) estimates (kept frozen throughout training).
– **OccAnt(rgbd)**: anticipate occupancy given the full RGB-D inputs.

By default, our methods use the proposed anticipation reward from Sec. 3.2. We denote ablations without this reward as “w/o AR”.

[3] We use our own implementation of ANS since authors' code was unavailable at the time of our experiments. See Sec. S7 in Supp. for details.

4.2 Occupancy anticipation results

First we evaluate the per-frame prediction accuracy of the mapping models trained during exploration. We evaluate on a separate dataset of images sampled from validation environments in Gibson at uniform viewpoints from discrete locations on a 1m grid, a total of $1,034$ (input, output) samples. This allows standardized evaluation of the mapper, independent of the exploration policy.

To quantify the local occupancy maps' accuracy, we compare the predicted maps to the ground truth. We report the Intersection over Union (IoU) and F1 scores for the "free" and "occupied" classes independently. In addition to the baselines from Sec. 4.1, we add two naive baselines that classify all locations as free (all-free), or occupied (all-occupied).

Table 1 shows the results. Our anticipation models OccAnt are substantially better than all the baselines. Comparing different modalities, OccAnt(depth) is much better than OccAnt(rgb) under all the metrics. This makes sense, as visible occupancy is directly computable from the depth input, but must be inferred for RGB (see Fig. 4). Interestingly, the rgbd models are not better than the depth-only models, likely because (1) geometric cues are more easily learned from depth than RGB, and (2) the RGB encoder contains significantly more parameters and could lead to overfitting. See Table S5 in Supp. for network sizes. Overall, Table 1 demonstrates our occupancy anticipation models successfully broaden the coverage of the map beyond the visible regions.

4.3 Exploration results

Next we deploy our models for visual exploration. The agent is given a limited time budget (T=1000) to intelligently explore and build a 2D top-down occupancy map of a previously unseen environment.

To quantify exploration, we measure both map quality and speed (number of agent actions): (1) **Map accuracy (m^2):** the area in the global map built during exploration (both free and occupied) that matches with the ground-truth layout of the environment. The map is built using predicted occupancy maps which are registered using estimated pose (may be noisy). Note that this is an unnormalized accuracy measure (see Eqn. 4). (2) **IoU:** the intersection over union between that same global map and the ground-truth layout of the environment. (3) **Area seen (m^2):** the amount of free and occupied regions *directly seen* during exploration. The map for this metric is built using ground-truth pose and depth-projections (similar to [11,10]). (4) **Episode steps:** the number of actions taken by the agent. While the first two metrics measure the quality of the created map, the latter two are a function of how (and how long) the agent moved to get that map. Higher accuracy in fewer steps or lower area-seen is better.

All agents are trained on 72 scenes from Gibson under noisy odometry and actuation (see Sec. 4.1), and evaluated on Gibson and Matterport3D under both noisy and noise-free conditions.

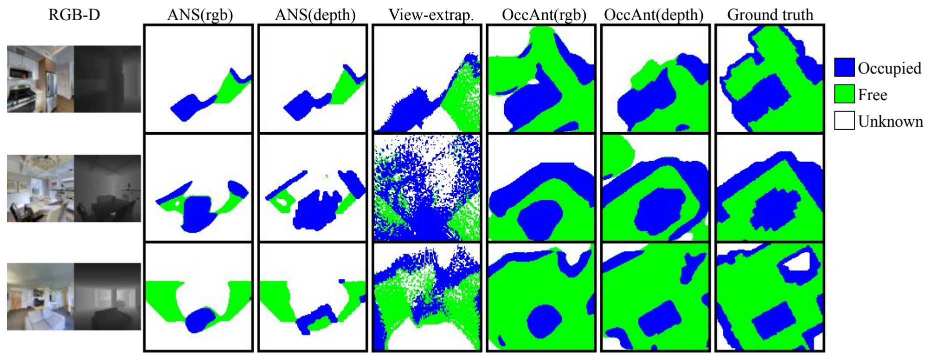

Fig. 4: **Per-frame local occupancy predictions:** First and last columns show the RGB-D input and anticipation ground-truth, respectively. ANS(*) are restricted to only predicting occupancy for visible regions. View-extrap. extrapolates, but is unable to predict occupancy for occluded regions (first row) and struggles to make correct predictions in cluttered scenes (second row). Our model successfully anticipates with either RGB or depth. For example, in the first row, we successfully predict the presence of a corridor and another room on the left. In the second row, we successfully predict the presence of navigable space behind the table. In the third row, we are able to correctly anticipate the free space behind the chair and the corridor to the right.

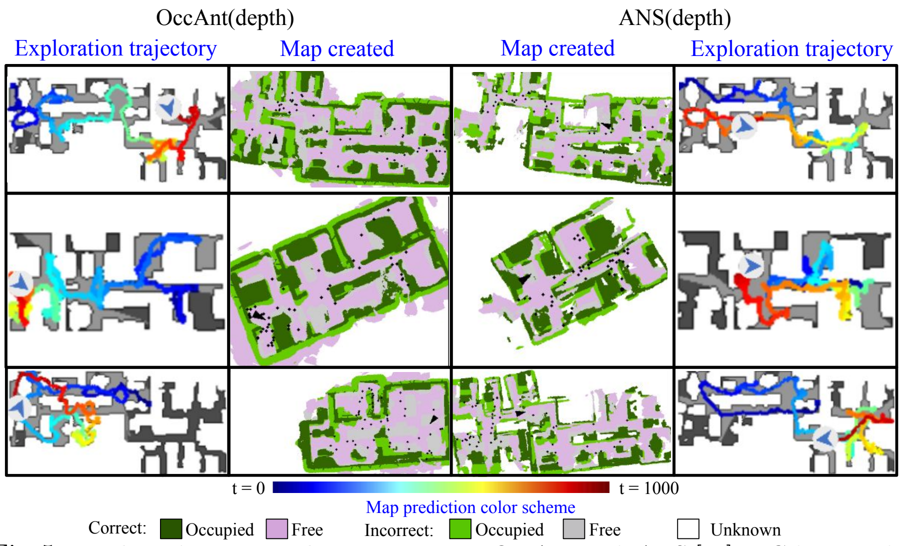

Fig. 5: **Exploration examples:** We compare OccAnt with ANS [10] in Gibson under noisy actuation and odometry. The exploration trajectories and the corresponding maps are shown at the extremes and center, respectively. **Row 1:** Both methods cover similar area, but our method better anticipates the unseen parts with fewer registration errors. **Row 2:** Our method achieves better area coverage and mapping quality whereas the baseline gets stuck in a small room for extended periods of time. **Row 3:** A failure case for our method, where it gets stuck in one part of the house after anticipating that a narrow corridor leading to a different room was occupied.

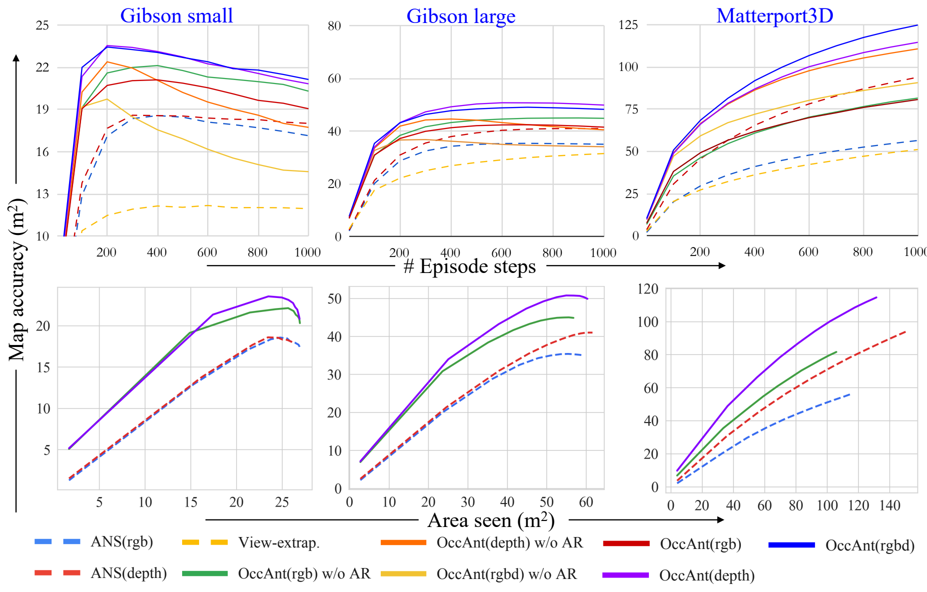

Fig. 6: **Exploration results:** Map accuracy (m^2) as a function of episode duration (top row) and area seen (bottom row) for Gibson (small and large splits) and Matterport3D under noisy conditions (see Sec. S1 in Supp. for noise-free). Higher and steeper curves are better. **Top:** Our OccAnt approach rapidly attains higher map accuracy than the baselines (dotted lines). **Bottom:** OccAnt achieves higher map accuracy for the same area seen (we show the best variants here to avoid clutter). These results show the agent *actively moves better* to explore the environment with occupancy anticipation.

	Noisy test conditions						Noise-free test conditions					
	Gibson small		Gibson large		Matterport3D		Gibson small		Gibson large		Matterport3D	
Method	Map acc.	IoU	Map acc.	IoU	Map acc.	IoU	Map acc.	IoU	Map acc.	IoU	Map acc.	IoU
ANS(rgb) [10]	18.5	55	35.0	47	44.7	18	22.4	76	43.4	64	53.4	23
ANS(depth)	18.5	56	39.4	53	72.5	26	21.4	74	48.0	72	85.9	34
View-extrap.	12.0	26	28.1	27	39.4	14	12.1	27	26.5	27	33.9	13
OccAnt(rgb) w/o AR	21.8	66	44.2	57	65.8	23	22.6	71	45.2	60	64.4	24
OccAnt(depth) w/o AR	20.2	58	44.2	54	92.7	29	**24.9**	**84**	**54.1**	**75**	. **104.7**	**38**.
OccAnt(rgbd) w/o AR	16.9	45	35.6	40	76.3	23	24.8	**84**	52.0	71	98.7	34
OccAnt(rgb)	20.9	62	42.1	54	66.2	22	22.3	70	43.5	58	64.4	22
OccAnt(depth)	**22.7**	**71**	**50.3**	**67**	94.1	**33**	24.8	83	53.1	74	96.5	35
OccAnt(rgbd)	**22.7**	**71**	48.4	62	**99.9**	32	24.5	82	51.0	69	100.3	34
OccAnt(GT)	21.7	67	51.9	63	-	-	26.1	93	65.4	91	-	-

Table 2: **Timed exploration results:** Map quality at T=500 for all models and datasets. See text for details.

Fig. 6 shows the exploration results. Our approach generally outperforms the baselines, improving the map quality more rapidly, whether in terms of time (top row) or area seen (bottom row). When compared on a same-modality basis, we see that OccAnt(rgb) converges much faster than ANS(rgb). Similarly, OccAnt(depth) is able to rapidly improve the map quality and outperforms ANS(depth) on all cases. This apples-to-apples comparison shows that anticipating occupancy leads to much more efficient mapping in unseen environments. Again, using depth generally provides more reliable mapping than pure RGB.

Furthermore, the proposed anticipation reward generally provides significant benefits to map accuracy in the noisy setting (compare our full model to the "w/o AR" models in Fig. 6). While map accuracy generally increases over time for noise-free conditions (see Sec. S1 in Supp.), it sometimes saturates early or even declines slightly over time in the noisy setting as noisy pose estimates accumulate and hurt map registration accuracy. This is most visible in Gibson small (top left plot). However, our anticipatory reward alleviates this decline.

Table 2 summarizes the map accuracy and IoU for all methods at T=500. Our method obtains significant improvements, supporting our claim that occupancy anticipation accelerates exploration and mapping. Additionally, perfect anticipation with the OccAnt(GT) model gives comparably good noisy exploration, and good gains in noise-free exploration (+10-20% IoU). This shows that there is indeed a lot of mileage in anticipating occupancy; our model moves the state-of-the-art towards this ceiling. Fig. 5 shows example exploration trajectories and the final global map predictions on Gibson.

4.4 Navigation results

Next we evaluate the utility of occupancy anticipation for quickly reaching a target. In PointNav [58,2], the agent is given a 2D coordinate (relative to its position) and needs to reach that target as quickly as possible. Following [10], we use noise-free evaluation and directly transfer the mapper, planner, and local policy learned during exploration to this task. In this way, instead of navigating to a point specified by the global policy, the agent has to navigate to a fixed goal location. To evaluate navigation, we use the standard metrics—success rate, success rate normalized by inverse path length (SPL) [2], and time taken. The agent succeeds if it stops within 0.2m of the target under a time budget of $T = 1000$.

Table 3 shows the navigation results on the Gibson validation set. Our approach outperforms the baselines. Thus, not only does occupancy anticipation successfully map the environment, but it also allows the agent to move to a specified goal more quickly by modeling the navigable spaces. This apples-to-apples comparison shows that our idea improves the state of the art for PointNav. As with exploration, using ground truth (GT) anticipation leads to good gains in the navigation performance, and our methods bridge the gap between the prior state of the art and perfect anticipation.

In concurrent work, the DD-PPO approach [72] obtains 0.96 SPL for PointNav, but it requires 2.5 billion frames of experience to do so (and it fails for noisy conditions; see below). To achieve the performance of our method (0.8 SPL in 2M frames), DD-PPO requires more than 50× the experience. Our sample efficiency can be attributed to explicit mapping along with occupancy anticipation.

Method	SPL %	Success %	Time taken
ANS(rgb) [10]	66.8	87.9	254.109
ANS(depth)	76.8	86.6	226.161
View-extrap.	10.4	33.3	835.556
OccAnt(rgb)	71.2	88.2	223.411
OccAnt(depth)	77.8	91.3	194.751
OccAnt(rgbd)	**80.0**	**93.0**	**171.874**
OccAnt(GT)	89.5	96.0	125.018

Table 3: **PointNav results:** Our approach provides more efficient navigation.

	Test standard			Test challenge		
Rank	Team	SPL %	Success %	Team	SPL %	Success %
1	**OccupancyAnticipation**	19.2	24.8	**OccupancyAnticipation**	20.9	27.5
2	ego-localization [14]	10.4	13.6	ego-localization [14]	14.6	19.2
3	Information Bottleneck	5.0	7.5	DAN [30]	13.2	25.3
4	cogmodel_team	0.8	1.3	Information Bottleneck	6.0	8.8
5	UCULab	0.5	0.8	cogmodel_team	0.7	1.2
6	Habitat Team (DD-PPO) [72]	0.0	0.2	UCULab	0.1	0.2

Table 4: **Habitat Challenge 2020 results:** Our approach is the winning entry.

Finally, we validate our approach on the 2020 Habitat PointNav Challenge [1], which requires the agent to adapt to noisy RGB-D sensors and noisy actuators, and to operate without an odometer. This presents a much more difficult evaluation setup than past work which assumes perfect odometry as well as noise-free sensing and actuation [38,10,72]. See Sec. S13 in Supp. for more details. Table 4 shows the results. Our method won the challenge, outperforming the competing approaches by large margins. While our approach generalizes well to this setting, DD-PPO [72] fails (0 SPL) due to its reliance on perfect odometry.

5 Conclusion

We introduced the idea of occupancy anticipation from egocentric views in 3D environments. By learning to anticipate the navigable areas beyond the agent's actual field of view, we obtain more accurate maps more efficiently in novel environments. We demonstrate our idea both for individual local maps, as well as integrated within sequential models for exploration and navigation, where the agent continually refines its (anticipated) map of the world. Our results clearly demonstrate the advantages on multiple datasets, including improvements to the state-of-the-art embodied AI model for exploration and navigation.

Acknowledgements

UT Austin is supported in part by DARPA Lifelong Learning Machines and the GCP Research Credits Program. We thank Devendra Singh Chaplot for clarifying the implementation details for ANS.

References

1. The Habitat Challenge 2020. `https://aihabitat.org/challenge/2020/`
2. Anderson, P., Chang, A., Chaplot, D.S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., et al.: On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06757 (2018)
3. Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., van den Hengel, A.: Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
4. Armeni, I., Sax, A., Zamir, A.R., Savarese, S.: Joint 2D-3D-Semantic Data for Indoor Scene Understanding. ArXiv e-prints (Feb 2017)
5. Bao, S.Y., Bagra, M., Chao, Y.W., Savarese, S.: Semantic structure from motion with points, regions, and objects. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2703–2710. IEEE (2012)
6. Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., Efros, A.A.: Large-scale study of curiosity-driven learning. In: arXiv:1808.04355 (2018)
7. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.J.: Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on robotics **32**(6), 1309–1332 (2016)
8. Carrillo, H., Reid, I., Castellanos, J.A.: On the comparison of uncertainty criteria for active slam. In: 2012 IEEE International Conference on Robotics and Automation. pp. 2080–2087. IEEE (2012)
9. Chang, A., Dai, A., Funkhouser, T., Nießner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3d: Learning from rgb-d data in indoor environments. In: Proceedings of the International Conference on 3D Vision (3DV) (2017), matterPort3D dataset license available at: `http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf`.
10. Chaplot, D.S., Gupta, S., Gandhi, D., Gupta, A., Salakhutdinov, R.: Learning to explore using active neural mapping. 8th International Conference on Learning Representations, ICLR 2020 (2020)
11. Chen, T., Gupta, S., Gupta, A.: Learning exploration policies for navigation. In: 7th International Conference on Learning Representations, ICLR 2019 (2019)
12. Choi, S., Zhou, Q.Y., Koltun, V.: Robust reconstruction of indoor scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
13. Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., Batra, D.: Embodied question answering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 2054–2063 (2018)
14. Datta, S., Maksymets, O., Hoffman, J., Lee, S., Batra, D., Parikh, D.: Integrating egocentric localization for more realistic pointgoal navigation agents. CVPR 2020 Embodied AI Workshop (2020)
15. Dhamo, H., Navab, N., Tombari, F.: Object-driven multi-layer scene decomposition from a single image. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
16. Elhafsi, A., Ivanovic, B., Janson, L., Pavone, M.: Map-predictive motion planning in unknown environments. arXiv preprint arXiv:1910.08184 (2019)
17. Fang, K., Toshev, A., Fei-Fei, L., Savarese, S.: Scene memory transformer for embodied agents in long-horizon tasks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 538–547 (2019)

18. Gan, C., Zhang, Y., Wu, J., Gong, B., Tenenbaum, J.B.: Look, listen, and act: Towards audio-visual embodied navigation. arXiv preprint arXiv:1912.11684 (2019)
19. Gordon, D., Kembhavi, A., Rastegari, M., Redmon, J., Fox, D., Farhadi, A.: Iqa: Visual question answering in interactive environments. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4089–4098 (2018)
20. Gupta, S., Davidson, J., Levine, S., Sukthankar, R., Malik, J.: Cognitive mapping and planning for visual navigation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2616–2625 (2017)
21. Gupta, S., Fouhey, D., Levine, S., Malik, J.: Unifying map and landmark based representations for visual navigation. arXiv preprint arXiv:1712.08125 (2017)
22. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968). https://doi.org/10.1109/tssc.1968.300136, `https://doi.org/10.1109/tssc.1968.300136`
23. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge university press (2003)
24. Henriques, J.F., Vedaldi, A.: Mapnet: An allocentric spatial memory for mapping environments. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8476–8484 (2018)
25. Hoermann, S., Bach, M., Dietmayer, K.: Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 2056–2063. IEEE (2018)
26. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (ToG) **36**(4), 1–14 (2017)
27. Jayaraman, D., Gao, R., Grauman, K.: Shapecodes: self-supervised feature learning by lifting views to viewgrids. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 120–136 (2018)
28. Jayaraman, D., Grauman, K.: Learning to look around: Intelligently exploring unseen environments for unknown tasks. In: Computer Vision and Pattern Recognition, 2018 IEEE Conference on (2018)
29. Jiacheng Chen, Chen Liu, J.W., Furukawa, Y.: Floor-sp: Inverse cad for floorplans by sequential room-wise shortest path. In: The IEEE International Conference on Computer Vision (ICCV) (2019)
30. Karkus, P., Ma, X., Hsu, D., Kaelbling, L.P., Lee, W.S., Lozano-Pérez, T.: Differentiable algorithm networks for composable robot learning. arXiv preprint arXiv:1905.11602 (2019)
31. Katyal, K., Popek, K., Paxton, C., Burlina, P., Hager, G.D.: Uncertainty-aware occupancy map prediction using generative networks for robot navigation. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 5453–5459. IEEE (2019)
32. Katyal, K., Popek, K., Paxton, C., Moore, J., Wolfe, K., Burlina, P., Hager, G.D.: Occupancy map prediction using generative and fully convolutional networks for vehicle navigation. arXiv preprint arXiv:1803.02007 (2018)
33. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
34. Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., Farhadi, A.: AI2-THOR: An Interactive 3D Environment for Visual AI. arXiv (2017)

35. Li, Y., Liu, S., Yang, J., Yang, M.H.: Generative face completion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3911–3919 (2017)
36. Liu, C., Wu, J., Furukawa, Y.: Floornet: A unified framework for floorplan reconstruction from 3d scans. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 201–217 (2018)
37. Lu, C., Dubbelman, G.: Hallucinating beyond observation: Learning to complete with partial observation and unpaired prior knowledge (2019)
38. Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., Batra, D.: Habitat: A Platform for Embodied AI Research. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
39. Martinez-Cantin, R., De Freitas, N., Brochu, E., Castellanos, J., Doucet, A.: A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. Autonomous Robots **27**(2), 93–103 (2009)
40. Mohajerin, N., Rohani, M.: Multi-step prediction of occupancy grid maps with recurrent neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 10600–10608 (2019)
41. Mousavian, A., Toshev, A., Fišer, M., Košecká, J., Wahid, A., Davidson, J.: Visual representations for semantic target driven navigation. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 8846–8852. IEEE (2019)
42. Müller, M., Dosovitskiy, A., Ghanem, B., Koltun, V.: Driving policy transfer via modularity and abstraction. arXiv preprint arXiv:1804.09364 (2018)
43. Murali, A., Chen, T., Alwala, K.V., Gandhi, D., Pinto, L., Gupta, S., Gupta, A.: Pyrobot: An open-source robotics framework for research and benchmarking. arXiv preprint arXiv:1906.08236 (2019)
44. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill **1**(10), e3 (2016)
45. O'Callaghan, S.T., Ramos, F.T.: Gaussian process occupancy maps. The International Journal of Robotics Research **31**(1), 42–62 (2012)
46. Parisotto, E., Salakhutdinov, R.: Neural map: Structured memory for deep reinforcement learning. arXiv preprint arXiv:1702.08360 (2017)
47. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
48. Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: International Conference on Machine Learning (2017)
49. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
50. Ramakrishnan, S.K., Grauman, K.: Sidekick policy learning for active visual exploration. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 413–430 (2018)

51. Ramakrishnan, S.K., Jayaraman, D., Grauman, K.: Emergence of exploratory look-around behaviors through active observation completion. Science Robotics **4**(30) (2019). https://doi.org/10.1126/scirobotics.aaw6326, `https://robotics.sciencemag.org/content/4/30/eaaw6326`
52. Ramakrishnan, S.K., Jayaraman, D., Grauman, K.: An exploration of embodied visual exploration. arXiv preprint arXiv:2001.02192 (2020)
53. Ramos, F., Ott, L.: Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent. The International Journal of Robotics Research **35**(14), 1717–1730 (2016)
54. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
55. Salas-Moreno, R.F., Newcombe, R.A., Strasdat, H., Kelly, P.H., Davison, A.J.: Slam++: Simultaneous localisation and mapping at the level of objects. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1352–1359 (2013)
56. Savinov, N., Dosovitskiy, A., Koltun, V.: Semi-parametric topological memory for navigation. arXiv preprint arXiv:1803.00653 (2018)
57. Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., Gelly, S.: Episodic curiosity through reachability. arXiv preprint arXiv:1810.02274 (2018)
58. Savva, M., Chang, A.X., Dosovitskiy, A., Funkhouser, T., Koltun, V.: Minos: Multimodal indoor simulator for navigation in complex environments. arXiv preprint arXiv:1712.03931 (2017)
59. Sax, A., Emi, B., Zamir, A.R., Guibas, L., Savarese, S., Malik, J.: Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. arXiv preprint arXiv:1812.11971 (2018)
60. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
61. Seifi, S., Tuytelaars, T.: Where to look next: Unsupervised active visual exploration on 360 {\deg} input. arXiv preprint arXiv:1909.10304 (2019)
62. Senanayake, R., Ganegedara, T., Ramos, F.: Deep occupancy maps: a continuous mapping technique for dynamic environments (2017)
63. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proceedings of the National Academy of Sciences **93**(4), 1591–1595 (1996)
64. Shen, W.B., Xu, D., Zhu, Y., Guibas, L.J., Fei-Fei, L., Savarese, S.: Situational fusion of visual representation for visual navigation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2881–2890 (2019)
65. Shrestha, R., Tian, F.P., Feng, W., Tan, P., Vaughan, R.: Learned map prediction for enhanced mobile robot exploration. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 1197–1204. IEEE (2019)
66. Sless, L., Cohen, G., Shlomo, B.E., Oron, S.: Self supervised occupancy grid learning from sparse radar for autonomous driving. arXiv preprint arXiv:1904.00415 (2019)
67. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic scene completion from a single depth image. Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition (2017)
68. Song, S., Zeng, A., Chang, A.X., Savva, M., Savarese, S., Funkhouser, T.: Im2pano3d: Extrapolating 360 structure and semantics beyond the field of view. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3847–3856 (2018)

69. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H.M., Nardi, R.D., Goesele, M., Lovegrove, S., Newcombe, R.: The Replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019)
70. Sun, C., Hsiao, C.W., Sun, M., Chen, H.T.: Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
71. Thrun, S.: Probabilistic robotics. Communications of the ACM **45**(3), 52–57 (2002)
72. Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., Batra, D.: Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames (2020)
73. Wu, W., Fu, X.M., Tang, R., Wang, Y., Qi, Y.H., Liu, L.: Data-driven interior plan generation for residential buildings. ACM Trans. Graph. **38**(6) (Nov 2019). https://doi.org/10.1145/3355089.3356556, `https://doi.org/10.1145/3355089.3356556`
74. Xia, F., Zamir, A.R., He, Z., Sax, A., Malik, J., Savarese, S.: Gibson env: Real-world perception for embodied agents. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9068–9079 (2018), gibson dataset license agreement available at `https://storage.googleapis.com/gibson_material/Agreement%20GDS%2006-04-18.pdf`
75. Yang, J., Ren, Z., Xu, M., Chen, X., Crandall, D., Parikh, D., Batra, D.: Embodied amodal recognition: Learning to move to perceive objects. In: ICCV (2019)
76. Yang, S.T., Wang, F.E., Peng, C.H., Wonka, P., Sun, M., Chu, H.K.: Dula-net: A dual-projection network for estimating room layouts from a single rgb panorama. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3363–3372 (2019)
77. Yang, W., Wang, X., Farhadi, A., Gupta, A., Mottaghi, R.: Visual semantic navigation using scene priors. arXiv preprint arXiv:1810.06543 (2018)
78. Yang, Z., Pan, J.Z., Luo, L., Zhou, X., Grauman, K., Huang, Q.: Extreme relative pose estimation for rgb-d scans via scene completion. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
79. Zhu, Y., Gordon, D., Kolve, E., Fox, D., Fei-Fei, L., Gupta, A., Mottaghi, R., Farhadi, A.: Visual Semantic Planning using Deep Successor Representations. In: Computer Vision, 2017 IEEE International Conference on (2017)
80. Zou, C., Colburn, A., Shan, Q., Hoiem, D.: Layoutnet: Reconstructing the 3d room layout from a single rgb image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2051–2059 (2018)

Occupancy Anticipation for Efficient Exploration and Navigation Supplementary Materials

Santhosh K. Ramakrishnan[1,2], Ziad Al-Halah[1], and Kristen Grauman[1,2]

[1] The University of Texas at Austin, Austin TX 78712, USA
[2] Facebook AI Research, Austin TX 78701, USA
`srama@cs.utexas.edu, ziadlhlh@gmail.com, grauman@cs.utexas.edu`

This document provides additional information about the experimental settings as well as qualitative and quantitative results to support the experiments from the main paper. Below is a summary of the sections in the supplementary file:

S1 Noise-free exploration results

As noted in the main paper, we evaluate on both noisy and noise-free conditions. We showed the change in map accuracy as a function of episode steps and area seen under noisy conditions in Fig. 6 in the main paper. We show the same results on noise-free conditions here in Fig. S1. Similar to the noisy case, OccAnt approach (solid lines) rapidly leads to higher map accuracy when compared to the baselines (dotted lines). However, we can see that adding the anticipation reward (AR) in this noise-free setting does not lead to improvements in performance in contrast to what was observed for the more realistic noisy setup (Fig. 5 in main).

As we will qualitatively demonstrate in Sec. S4, the main benefit of using the anticipation reward is that it leads to better noise correction in the pose estimates under noisy test conditions, resulting in more effective map registration. This is due to the fact that achieving high AR (i.e., the map accuracy) inherently depends on better map registration. If the per-frame maps are not registered correctly, AR is likely to be low even if the per-frame map estimates are very good. Therefore, in addition to covering more area, the agent also has to better train the pose estimator which would then lead to higher AR over time. Since noise correction is not needed under noise-free conditions, using AR has limited impact on the final performance.

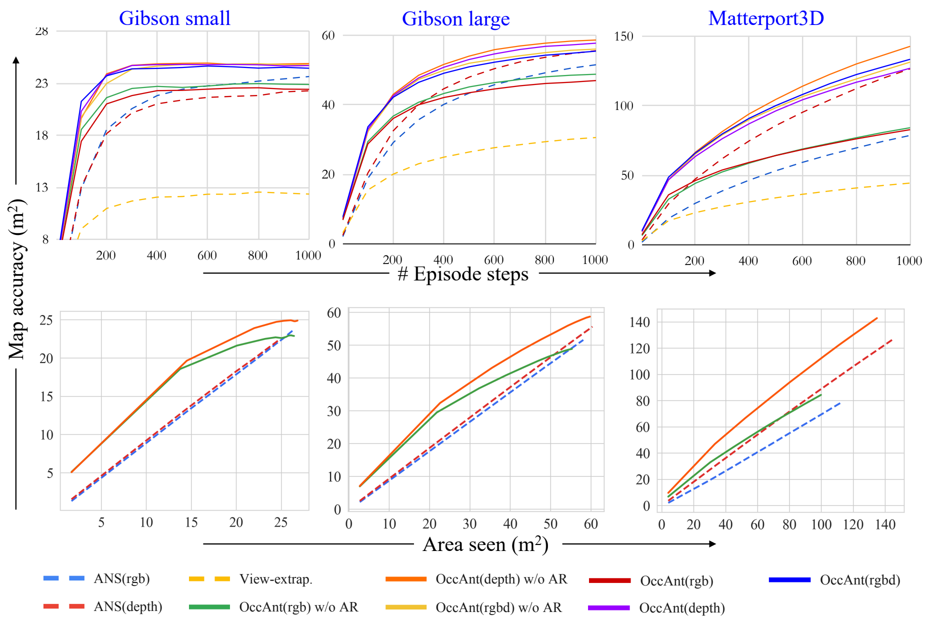

Fig. S1: **Noise-free exploration results:** Map accuracy (m^2) as a function of episode duration (top row) and area seen (bottom row) for Gibson (small and large splits) and Matterport3D under noise-free conditions. **Top:** Our OccAnt approach (solid lines) rapidly attains higher map accuracy than the baselines (dotted lines). Using anticipation reward (AR) largely retains the original performance in the noise-free conditions (but improves significantly in the noisy conditions, see Fig. 5 main paper). **Bottom:** OccAnt achieves higher map accuracy for the same area covered (we show best variants here to avoid clutter). These results show the agent *actively moves better* to explore the environment with our occupancy anticipation idea.

S2 Occupancy anticipation ablation study

As discussed in the main paper, our key contributions are a novel framework for occupancy anticipation and a novel anticipation reward which encourages the agent to build more accurate maps (as opposed to covering more area). To isolate the gains achieved by these individual contributions, we view the results from the main paper (Tables 1, 2, and 3 in main paper) in a different way. We first group the results based on the modality (rgb/depth/rgbd), and further sort the methods based on whether they use occupancy anticipation (OccAnt) or the anticipation reward (AR). We present these ablations for the per-frame map evaluation (Table S1), the exploration evaluation (Table S2), and the navigation evaluation (Table S3). By default, the ANS baselines do not use occupancy anticipation or the anticipation reward and our methods always use occupancy anticipation.

For per-frame maps, in Table S1 we see that adding occupancy anticipation to the base model significantly improves the IoU and F1 scores as expected. Adding the anticipation reward leads to comparable or better results, showing that it leads to better training of the mapper during the exploration training.

For exploration, in Table S2 we see that adding occupancy anticipation generally leads to better map quality than ANS across different modalities and testing conditions. Adding the anticipation reward (AR) leads to significant improvements in the map quality under noisy conditions for both depth and rgbd modalities (rgb slightly underperforms). This is primarily due to improved training of the mapper module which leads to better map registration (see Sec. S4). As we also noted in Sec. S1, using AR in noise-free conditions has limited impact on the performance as the pose-estimation is assumed to be perfect in these cases. It mainly benefits exploration in the more real-world testing scenarios with noisy actuation and sensing.

For navigation, in Table S3 we see that adding occupancy anticipation leads to significant improvements in all three metrics. The impact of using AR here is limited because we assume noise-free test conditions for PointNav (following [38,10]). However, the challenge results reported in the main paper remove this assumption to test PointNav with noisy odometry and actuation.

S3 Occupancy anticipation qualitative examples

See Figs. S2 and S3 for some successful cases and failure cases for our best method from Table S1 when compared with the baselines.

S4 Exploration with occupancy anticipation examples

In Table 2 and Fig. 6 from the main paper, and Table S2 in this supplementary, we see that adding occupancy anticipation on top of the ANS baseline leads to better performance, and adding anticipation reward (AR) leads to better mapping in the noisy cases.

Here, we highlight some example episodes to show that (1) using occupancy anticipation avoids local navigation difficulties and obtains higher map qualities for lower area coverage (Fig. S4), while sometimes being susceptible to inaccuracies in map predictions (Fig. S5), and (2) the anticipation reward leads to better map registration (i.e., good pose estimates) which results in higher map quality (Fig. S6). The color scheme for the trajectories (from [38]) and the predicted maps in the center (from [10]) are indicated below each plot.

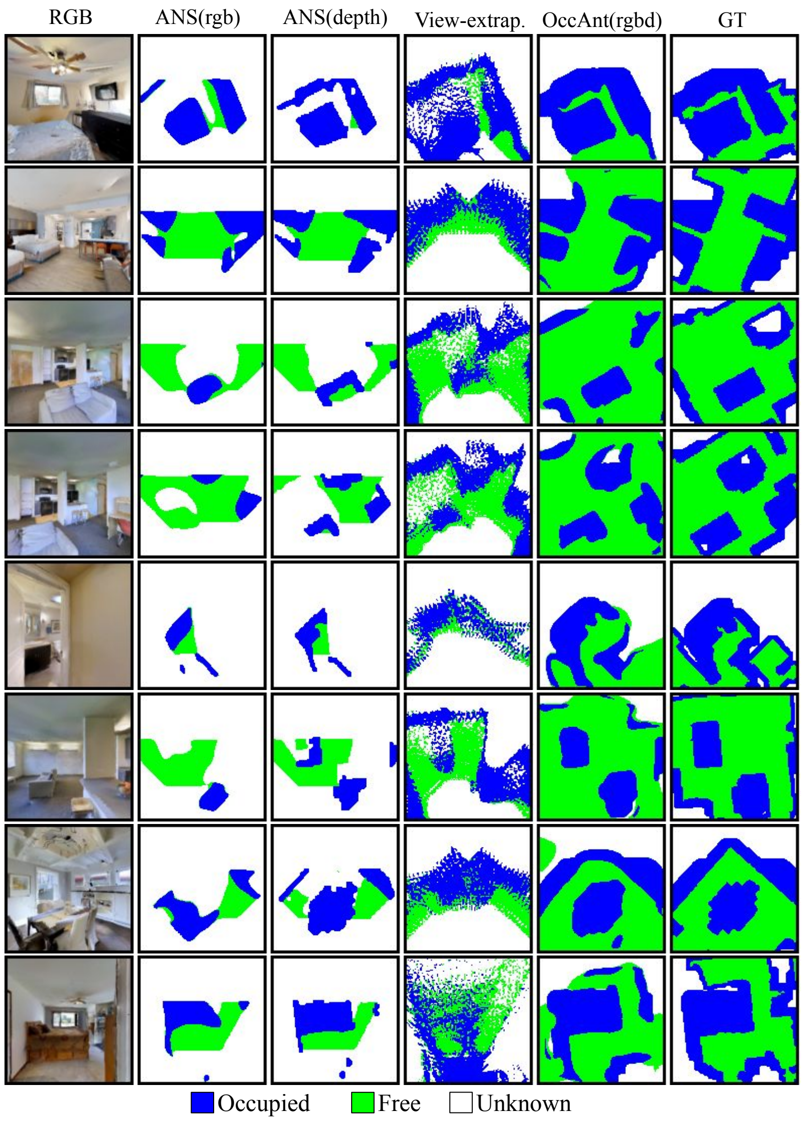

Fig. S2: **Occupancy anticipation successful cases:** ANS(rgb) is trained to predict the visible occupancy (2nd column) and ANS(depth) (3rd column) directly uses the visible occupancy (within a 3m range). Both these methods are unable to account for regions that are not visible or outside the sensing range. While View-extrap (4th column) is able to expand beyond a 90° FoV, its predictions are often noisy and do not include occluded regions. Also, the predictions are not guaranteed to be smooth in the top-down projection as smoothness in the depth-image prediction space does not necessarily lead to smoothness in the top-down maps, resulting in speckled outputs. Our method OccAnt(rgbd) (5th column) is able to successfully anticipate occupancy for regions that are occluded and outside the field-of-view with high accuracy (see ground-truth in column 6).

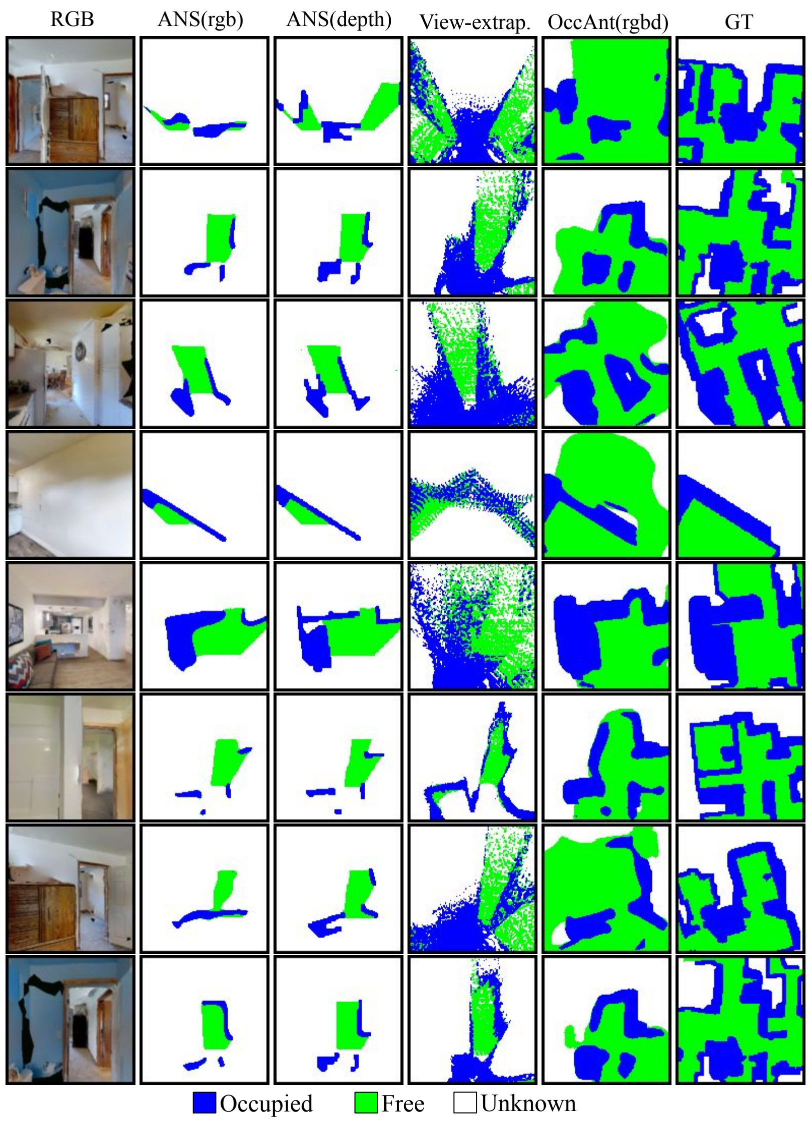

Fig. S3: **Occupancy anticipation failure cases:** Our approach OccAnt(rgbd) incorrectly predicts narrow corridors as occupied, and is unable to handle cases where multiple solutions may exist. For example, in rows 2, 5 and 8, it predicts that the corridors in the center of the map are blocked. In row 1, it predicts that the two doors correspond to the same room, even though the wall colors are different and it is unlikely that a small room would have two doors. In row 3, 4 and 6, it predicts entrances to spaces that do not exist. Such predictions are generally difficult to make given only the context of the current first-person view, and therefore our model tends to fail at these cases.

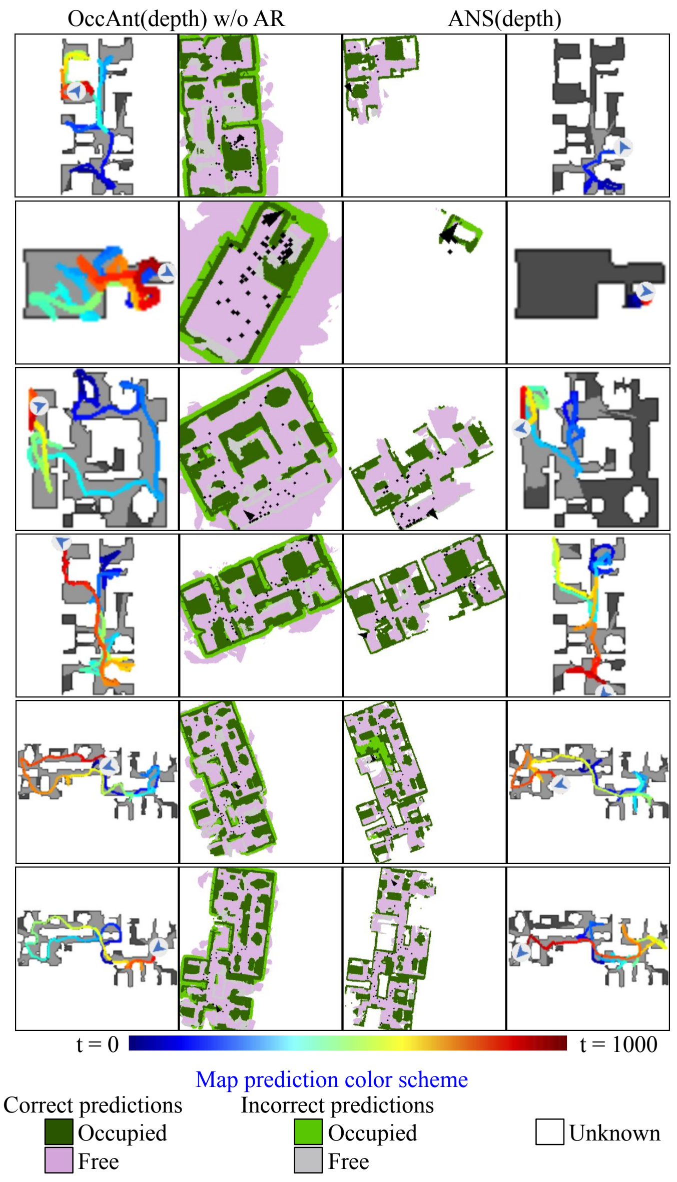

Fig. S4: We enumerate some of the key advantages of exploration using occupancy anticipation by comparing OccAnt(depth) w/o AR with ANS(depth) in Gibson under noise-free conditions. The exploration trajectories and the map created during exploration are shown at the extremes and the center, respectively. ‘ANS(depth) tends to achieve worse exploration in some cases where the visible occupancy is incorrectly estimated (top 3 rows), causing the agent to get stuck in local regions. In other cases, the map accuracy is generally higher for OccAnt(depth) w/o AR for similar amounts of area seen (bottom 3 rows) as it is better at filling up the occupancy for unvisited regions.

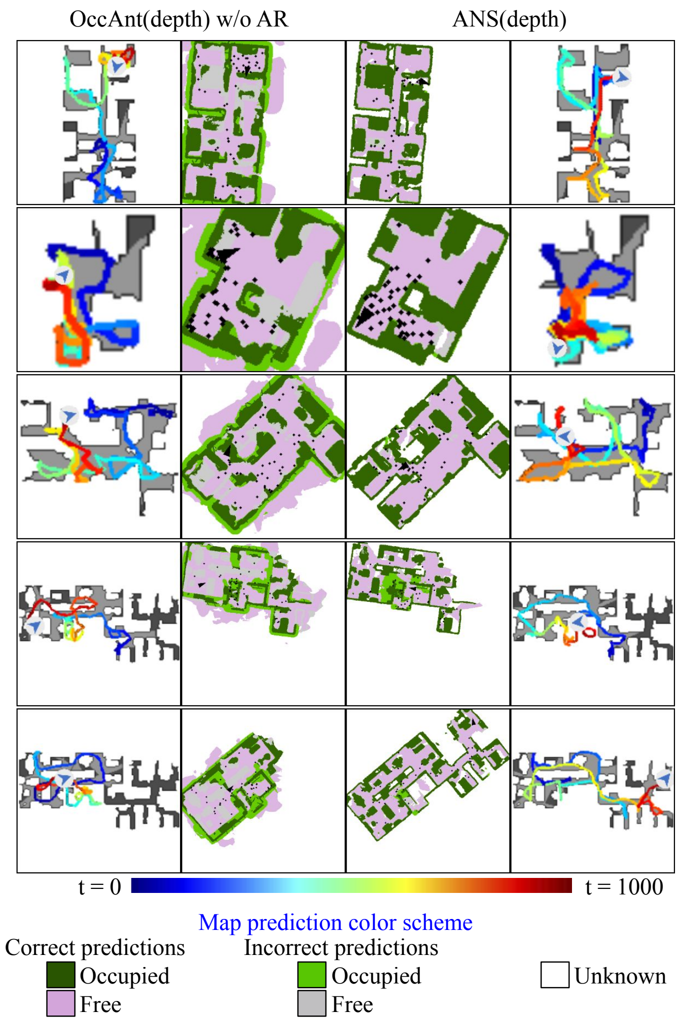

Fig. S5: We highlight one key weakness of exploration using occupancy anticipation, which is the impact of classification errors in occupancy estimates. We compare OccAnt(depth) w/o AR with ANS(depth) in Gibson. In some cases, OccAnt(depth) w/o AR tends to generate false negatives for occupied regions, classifying some of the explored obstacles as free-space (gray regions in the first 3 rows, 2nd column). While this does not impact the area seen, it does reduce the map quality. On the flip side, OccAnt(depth) w/o AR may prematurely classify some narrow corridors as blocked (similar to Fig. S3) causing the agent to stop exploring beyond that corridor (light green regions in last two rows, 2nd column).

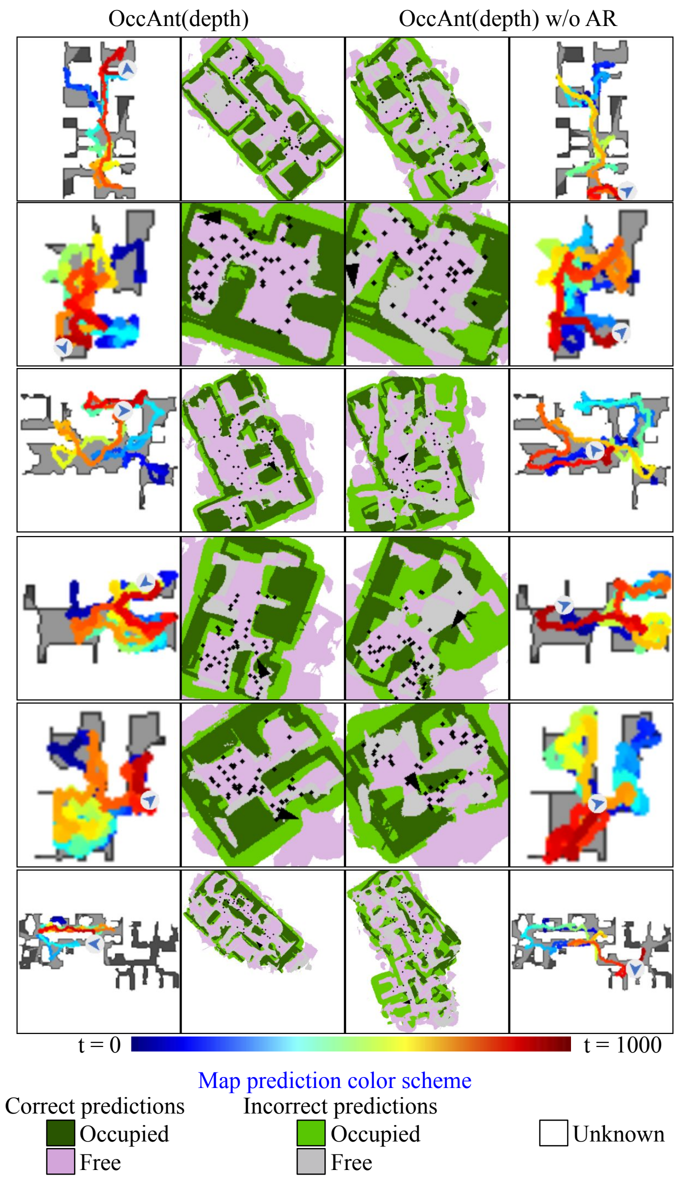

Fig. S6: **The impact of using anticipation reward:** In Table 6 from the main paper and Table S2 in Supp., we could see that models using anticipation reward generally leads to higher map qualities in noisy test conditions. Here, we show that, when the model that uses the anticipation reward (OccAnt(depth) on the left) accounts much better for the noise in map registration when compared to a vanilla anticipation model that does not use it (OccAnt(depth) w/o AR on the right).

Method	OccAnt	AR	IoU %			F1 score %		
			free	occ.	mean	free	occ.	mean
ANS(rgb)	✗	✗	12.1	14.9	13.5	19.6	24.9	22.5
OccAnt(rgb) w/o AR	✓	✗	**44.6**	**47.9**	**46.2**	**58.4**	**62.9**	**60.6**
OccAnt(rgb)	✓	✓	44.4	**47.9**	46.1	58.2	**62.9**	**60.6**
ANS(depth)	✗	✗	14.5	24.1	19.3	23.1	37.6	30.4
OccAnt(depth) w/o AR	✓	✗	50.3	61.7	56.0	**63.8**	74.9	69.3
OccAnt(depth)	✓	✓	**50.4**	**61.9**	**56.1**	**63.8**	**75.0**	**69.4**
OccAnt(rgbd) w/o AR	✓	✗	50.1	60.5	55.3	63.6	74.1	68.8
OccAnt(rgbd)	✓	✓	**51.5**	**61.5**	**56.5**	**64.9**	**74.8**	**69.8**

Table S1: Per-frame occupancy anticipation ablation study

Noisy test conditions								
			Gibson small		Gibson large		Matterport3D	
Method	OccAnt	AR	Map acc.	IoU %	Map acc.	IoU %	Map acc.	IoU %
ANS(rgb) [10]	✗	✗	18.46	55	34.95	47	44.70	18
OccAnt(rgb) w/o AR	✓	✗	**21.77**	**66**	**44.15**	**57**	65.76	**23**
OccAnt(rgb)	✓	✓	20.87	62	42.08	54	**66.15**	22
ANS(depth)	✗	✗	18.54	56	39.35	53	72.48	26
OccAnt(depth) w/o AR	✓	✗	20.22	58	44.18	54	92.70	29
OccAnt(depth)	✓	✓	**22.74**	**71**	**50.30**	**67**	**94.12**	**33**
OccAnt(rgbd) w/o AR	✓	✗	16.92	45	35.60	40	76.32	23
OccAnt(rgbd)	✓	✓	**22.70**	**71**	**48.42**	**62**	**99.92**	**32**
Noise-free test conditions								
			Gibson small		Gibson large		Matterport3D	
Method	OccAnt	AR	Map acc.	IoU %	Map acc.	IoU %	Map acc.	IoU %
ANS(rgb) [10]	✗	✗	22.43	**76**	43.41	**64**	53.40	23
OccAnt(rgb) w/o AR	✓	✗	**22.60**	71	**45.19**	60	**64.44**	**24**
OccAnt(rgb)	✓	✓	22.32	70	43.52	58	64.35	22
ANS(depth)	✗	✗	21.39	74	48.01	72	85.91	34
OccAnt(depth) w/o AR	✓	✗	**24.91**	**84**	**54.05**	**75**	**104.68**	**38**
OccAnt(depth)	✓	✓	24.80	83	53.08	74	96.45	35
OccAnt(rgbd) w/o AR	✓	✗	**24.80**	**84**	**51.99**	**71**	98.70	34
OccAnt(rgbd)	✓	✓	24.51	82	50.97	69	**100.25**	34

Table S2: **Timed exploration ablation:** Map quality at T=500 for all models and datasets.

Method	OccAnt	AR	SPL %	Success Rate %	Time taken
ANS(rgb) [10]	✗	✗	66.8	87.9	254.109
OccAnt(rgb) w/o AR	✓	✗	**71.2**	**88.2**	**223.411**
OccAnt(rgb)	✓	✓	66.1	81.3	293.321
ANS(depth)	✗	✗	76.8	86.6	226.161
OccAnt(depth) w/o AR	✓	✗	**78.6**	**92.2**	**187.358**
OccAnt(depth)	✓	✓	77.8	91.3	194.751
OccAnt(rgbd) w/o AR	✓	✗	77.9	92.9	174.105
OccAnt(rgbd)	✓	✓	**80.0**	**93.0**	**171.874**

Table S3: **PointGoal navigation ablation:** Time taken refers to the average number of agent actions required; the maximum time budget is T=1000.

S5 Generating ground-truth for occupancy anticipation

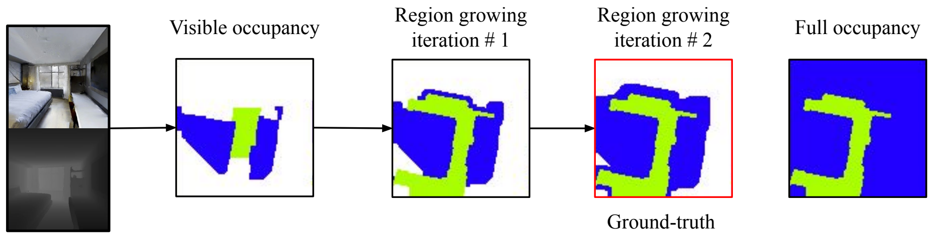

Fig. S7: Pipeline for generating anticipation ground-truth.

Using 3D meshes of indoor environments from Gibson and Matterport3D, we obtain the ground-truth local occupancy of a $V \times V$ region in front of the camera which includes parts that may be occluded or outside the field-of-view (see Fig.2 from main paper). However, this may include regions in the environment that are outside the bounds of the environment's mesh. To alleviate this problem, we devise a simple heuristic that generates the ground truth by masking out regions in the occupancy map that are outside the bounds of the environment (highlighted in Fig. S7).

We first obtain the visible occupancy via a geometric projection of the depth inputs (2nd column). We then selectively sample the ground-truth layout (last column) around the visible regions by growing a mask around the visible occupancy by sequential hole-filling and morphological dilation operations. We perform two iterations of this region growing to obtain the final ground-truth used to train our model (3rd & 4th columns). This heuristic captures the occupied regions that are closer to navigable space in the environment (likely to be objects, walls, etc), while ignoring regions outside the bounds of the environment. This is necessary since the occupancy map from the simulator does not distinguish between obstacles and regions outside the bounds of the environment mesh. Note that these steps apply only in training; during inference the occupancy anticipation proceeds solely in the end-to-end model.

S6 Noise models for actuation and odometry

Following [10], we simulate realistic actuation and odometry to train and evaluate our exploration agents. For this purpose, we use the PyRobot actuation model provided by Habitat which consists of truncated Gaussians for both the rotation and translation motions.[3] Specifically, we use the default LoCoBot noise-model with the ILQR controller. For simulating noise in the odometry, we similarly use

[3] `https://github.com/facebookresearch/habitat-sim/habitat_sim/agent/controls/pyrobot_noisy_controls.py`

truncated Gaussians for both rotation and translation measurements. For the translation measurement, we use a mean of 0.025m and a standard deviation of 0.001 For the rotation measurement, we use a mean of 0.9° and standard deviation of 0.057°. The distributions are truncated at 2 standard deviations. These are based on approximate values provided by the authors of ANS.

S7 Differences in ANS implementation

We implemented the ANS approach using the published details in [10] and instructions obtained directly from the authors via private communication as code was not publicly available at the time of our research. Our implementation has a few differences from that in [10], which we discuss in the following. For shortest path planning, we use an A* planner instead of fast-marching [63] used in [10] since we were able to find a fast A* implementation that was publicly available.[4]

For aggregating the local occupancy maps $\hat{p}_t$ from each observation[5] into the global map $\hat{m}_{t-1}$ from the previous time-step , the authors in [10] use channelwise max-pooling of the local and global maps to obtain the updated global map $\hat{m}_t$.

$$\hat{m}_t = \texttt{ChannelwiseMax}(\hat{m}_{t-1}, \hat{p}_t) \tag{1}$$

Instead, we opt to perform a moving-average over time to allow the agent to account for errors in the map prediction by averaging predictions from multiple views over time.

$$\hat{m}_t = \alpha_e \hat{m}_{t-1} + (1 - \alpha_e)\hat{p}_t \tag{2}$$

We found that this provided robustness to false positives in the map predictions and registration errors due to odometry noise.

Additionally, since our proposed model anticipates occupancy beyond the visible regions, we found that it is helpful to filter out low-confidence predictions of occupancy on a per-frame basis using the `EntropyFilter()` operation. Given prediction $\hat{p}_t$, `EntropyFilter()` masks out the predictions for locations i, j in $\hat{p}_t$ where the binary-entropy of the probabilities across the map channels are larger than a threshold τ_{ent} before performing the moving-average aggregation. These low-confidence predictions generally correspond to regions that are hard to anticipate or may have multiple solutions. Hence, our global map update formula is:

$$\hat{m}_t = \alpha_e \hat{m}_{t-1} + (1 - \alpha_e)\texttt{EntropyFilter}(\hat{p}_t). \tag{3}$$

[4] A* implementation: `https://github.com/hjweide/a-star`

[5] $\hat{p}_t$ is the local map at t registered to the global coordinates using the agent's pose estimate.

S8 Implementation details

The key hyperparameters for learning the policy and mapper are specified in Table S4.

Policy learning	
Optimizer	Adam [33]
# processes	24
Learning rate	0.00025
Value loss coef	0.5
Entropy coef	0.001
Discount factor γ	0.99
GAE τ	0.95
Episode length	1000
# training frames	1.5-2 million
PPO clipping	0.2
PPO epochs	4
# PPO minibatches	16
Global policy Δ	25
Global policy update interval	20
Global policy reward scaling	0.0001
Local policy reward scaling	1.0
Local policy update interval	25
Mapper learning	
Optimizer	Adam [33]
Learning rate	0.0001
Replay buffer size	25000
Mapper update interval	5
Mapper batch size	32
Mapper update batches	20
Map scale	0.05m
Local map size (V)	101
Global map size (G)	961
Aggregation factor (α_e)	0.9

Table S4: Policy and mapper hyperparameters used to train our models

S9 Occupancy anticipation architecture

The architecture diagrams for the individual components of our occupancy anticipation model (Fig. 2 in main paper) are shown in Figs. S8, S9, S10 and S11 with a brief description of the role of each module. We follow the PyTorch [47] conventions to describe individual layers, with the tensor shapes represented in (C, H, W) notations. The descriptions for individual layers are:

- **ConvBR:** a combination of `nn.Conv2d`, `nn.BatchNorm2d` and `nn.ReLU` layers with the arguments representing the input channels, output channels, kernel size, stride and padding.
- **MaxPool:** an instantiation of the `nn.MaxPool2d` layer with the arguments representing the kernel size, stride and padding.
- **Conv:** a `nn.Conv2d` layer with the arguments representing the input channels, output channels, kernel size, stride and padding.
- **AvgPool:** an instantiation of the `nn.AvgPool2d` layer with the arguments representing the kernel size, stride and padding.
- **2x Upsample:** an instantiation of the `nn.Upsample` layer with a scaling factor of 2.

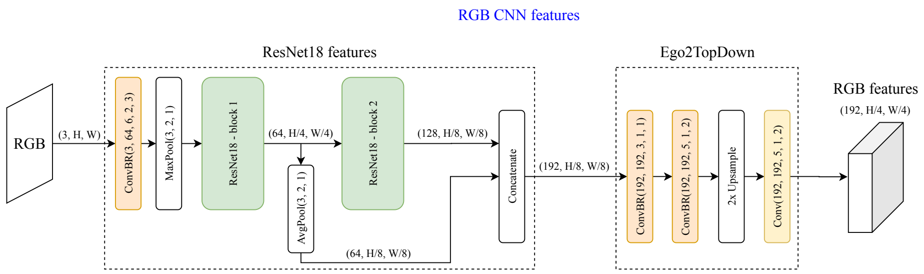

Fig. S8: **RGB CNN features:** extracts features from RGB images using ResNet18 blocks, and further processes the features to obtain compatible RGB features in a top-down view.

S10 ANS projection unit architecture

The projection unit architecture for the ANS(rgb) baseline is shown in Fig. S12. This is based on the architecture in [10] with some minor differences. It uses `nn.BatchNorm` + `nn.ReLU` blocks instead of `nn.Dropout` in the fully connected layers, it has a larger convolutional decoder to account for our larger map outputs, and it consists of `nn.Conv2d` + `nn.Upsample` layers instead of than `nn.ConvTranspose2D` layers as this has been shown to reduce checkerboard artifacts [44].

S11 View extrapolation baseline

We now provide more details on the task-defintion and architecture for the View-extrap. baseline introduced in Sec. 4.1 in the main paper. The goal is to extrapolate 180° FoV depth from 90° FoV RGB-D inputs in order to evaluate the performance of scene completion approaches [68,78]. Since Habitat [38] does

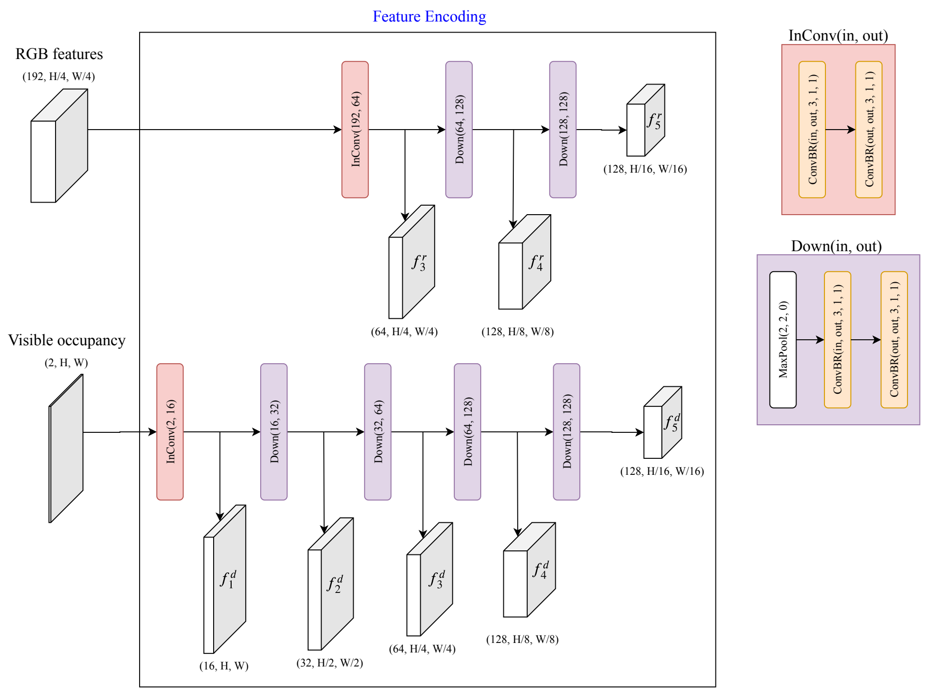

Fig. S9: **Feature encoding:** The RGB features and visible occupancy are encoded using independent UNet encoding layers. The expanded view of the "InConv" and "Down" blocks are shown on the right. The encoded RGB and visible occupancy features at different levels are $f_{3:5}^r$ and $f_{1:5}^d$, respectively.

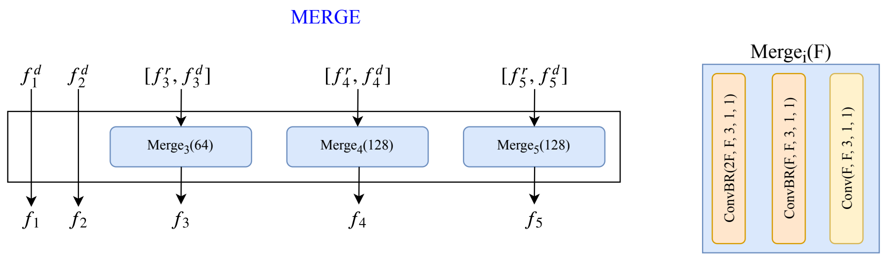

Fig. S10: **MERGE:** combines the RGB ($f_{3:5}^r$) and visible occupancy ($f1 : 5^d$) features obtained from Feature encoding layers in a layerwise fashion to obtain a set of merged features $f = f_{1:5}$. Since the RGB features are not available at levels 1 and 2, it simply uses the visible occupancy features for those levels. The expanded view of the "Merge$_i$(F)" block is shown on the right.

not natively support panoramic rendering, we use a simpler solution to account for this. We place two cameras with $\pm 45°$ heading offsets and aim to regress those from the egocentric view (see Fig. S13). Since each camera has a 90° FoV,

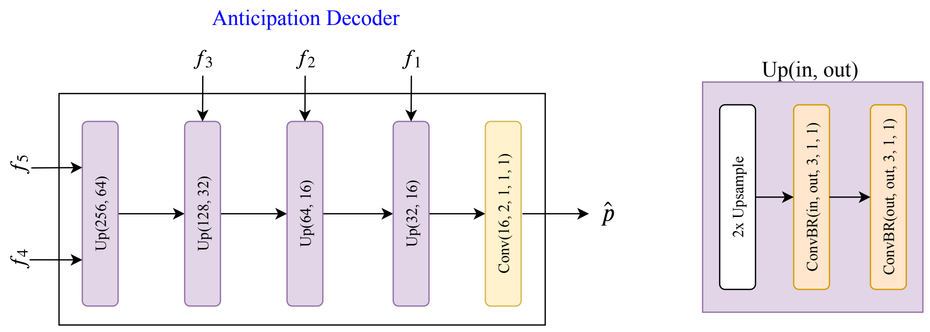

Fig. S11: **Anticipation Decoder:** a typical UNet decoder that takes features provided by MERGE ($f = f_{1:5}$) and decodes them using residual connections to obtain the anticipated occupancy map $\hat{p}$. The expanded view of the "Up" block is shown on the right.

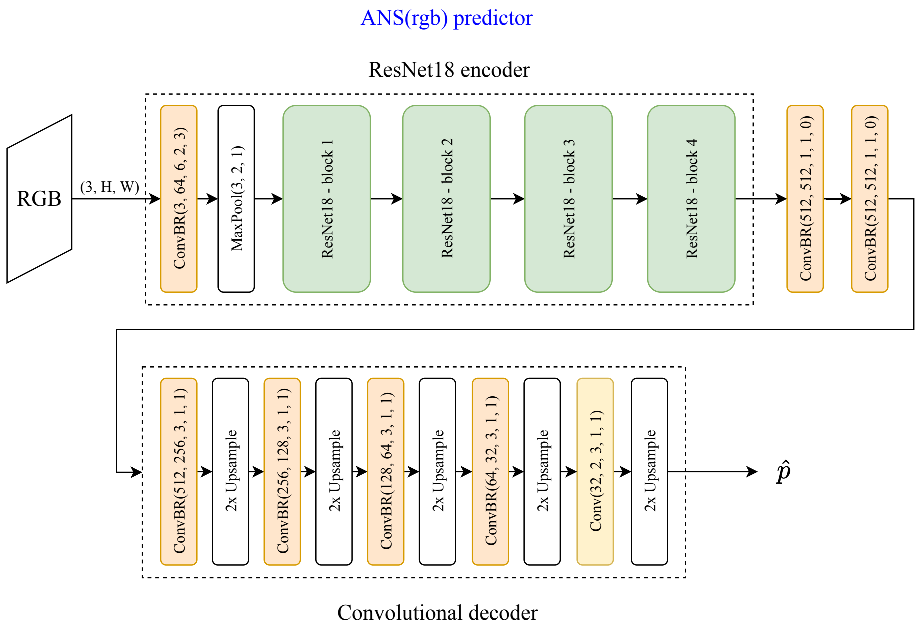

Fig. S12: **ANS projection unit:** ResNet-18 features are extracted, followed by two fully connected layers (represented by 1×1 convolutions) and a convolutional decoder that uses "Upsample" blocks to increase the output resolution and predict the occupancy estimates $\hat{p}$. Note that this is supervised to predict the visible occupancy map, not the anticipated occupancy map (see Fig. 1 in main paper).

this leads to an effective coverage of 180° once the agent anticipates the unobserved portions. We base our architecture for view extrapolation on the model from [78] with a capacity similar to our model to permit online training during policy learning (see Fig. S14). It takes as input the 90° FoV RGB-D images and regresses the left and right cameras. It is trained to minimize the pixelwise ℓ_1 loss between the prediction and the ground-truth.

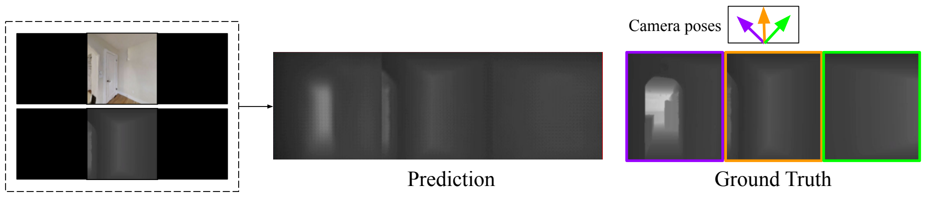

Fig. S13: **View extrapolation task:** Given the agent's egocentric RGB-D input, we predict the depth-map for two additional depth-sensors placed at 45° angles to the left (purple) and right (green) of the central input (orange). These are geometrically projected to the top-down view to obtain the occupancy estimates.

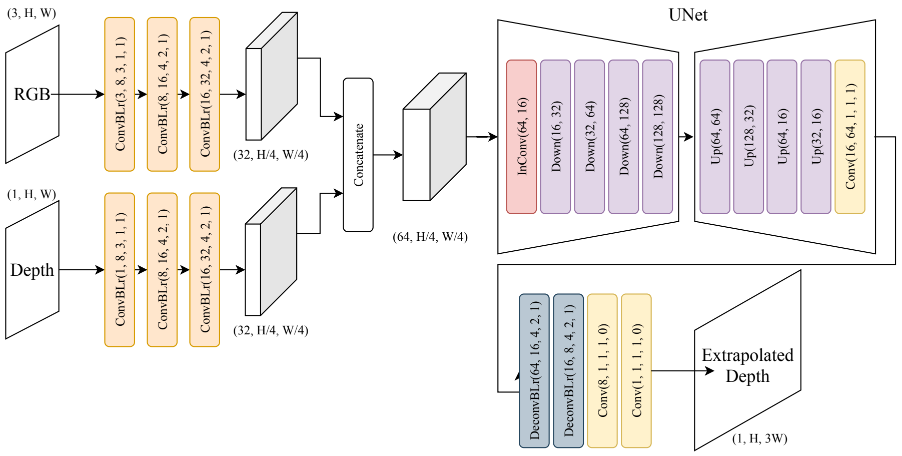

Fig. S14: **View extrapolation architecture [78]:** The 90° FoV RGB and depth inputs are independently encoded using Convolutional layers, concatenated and processed using a UNet model. The decoded features from UNet are used to extrapolate the final depth predictions. "DeconvBLr" uses `nn.ConvTranspose2D` to perform the upsampling. Note that "ConvBLr" and "DeconvBLr" use `nn.LeakyReLU(0.1)` instead of "nn.ReLU()".

S12 Comparing the model capacities of different methods

We compare the overall model capacity of our approaches with the baselines in Table S5. The depth-only models (bottom 2 rows) tend to have fewer parameters than the rgb-only models as they rely on geometric projection for processing depth (no ResNet backbone). Our depth model has comparable number of parameters with the depth-only baselines. Our rgb model has slightly more parameters than the rgb baseline. However, this is due to the fact that OccAnt(rgb) takes the output of ANS(rgb) as an additional input. However, since ANS(rgb) is kept frozen throughout the training of OccAnt(rgb), this effectively gives us 5.7M trainable parameters.

Method	Parameters (in millions)
ANS(rgb)	14.16
OccAnt(rgb)	19.86
ANS(depth)	0.87
OccAnt(depth)	1.72

Table S5: Comparing model capacity of different approaches

S13 Habitat Challenge 2020

We detail the key issues we had to address for the PointNav track of Habitat Challenge 2020 [1] and the changes to our system required to achieve our state-of-the-art results. Compared to the 2019 Habitat Challenge, there were two key changes that increased the task difficulty:

Lack of GPS+Compass sensor: The presence of the GPS+Compass sensor used in earlier challenges continually provides the agent with a perfect estimate of the position and heading angle of the goal relative to its current position. Such perfect localization has been exploited in the past by purely geometric [20] and learned [72] approaches to achieve high-quality PointNav performance. However, such high precision localization is hard to achieve in practice. The 2020 challenge instead requires navigation in the absence of the GPS+Compass sensor. Instead, the goal location is only specified initially at the start of the episode, requiring the agent to accurately keep track of its position in the environment to successfully reach the goal.

Noisy actuation and sensing: In the 2020 challenge, RGB-D sensing noise is simulated artificially by using a Gaussian noise-model for the RGB sensor and the Redwood noise model [12] for the depth sensor. Additionally, actuation noise in the robot motions is simulated by using a noise model obtained from the LoCoBot [43].

We adapted our model in several ways to address these challenges. To address the lack of GPS+Compass sensor, we used an online pose estimator that uses RGB-D inputs x_t and x_{t+1} to estimate the relative change in the pose Δp_{t+1}. These pose changes are summed up over time to track the agent's pose p_{t+1}. When compared to the original ANS model, we found that using RGB-D inputs gave slightly better estimates and was more computationally efficient than using top-down maps. The pose estimator consists of a 6 convolutional layers followed by 3 fully-connected layers to predict the pose for each modality (RGB, depth) independently. The predictions are combined by using input-conditioned weighting factors that are estimated using a learned MLP with 4 fully-connected layers.

To handle noisy sensing, we train our occupancy anticipation model end-to-end on the noisy inputs, which gave accurate predictions (see Fig. S15). We found that OccAnt(depth) gave the best performance, and that adding RGB information to occupancy anticipation did not lead to significant changes in performance.

To deal with noisy actuation, we found that the learned pose estimator gave robust estimates of the agent position. Despite having this pose estimator, we experienced large drifts in the estimate over time due to high variance in the actuation noise. To partially mitigate this issue, we focused on efficient navigation with safe planning that maintains sufficient distance from obstacles while planning shortest paths. In practice, we found that reducing the number of collisions leads to faster navigation and lower drift in the pose estimates. We achieve this by using a weighted variant of the classic A-star search algorithm [22].[6]

Additionally, we incorporated some simple heuristics from the original Active Neural SLAM implementation to update the occupancy maps based on collisions, and used an analytical local policy for navigation instead of a learned policy.

[6] Weighted A-star implementation: `https://github.com/srama2512/astar_pycpp`

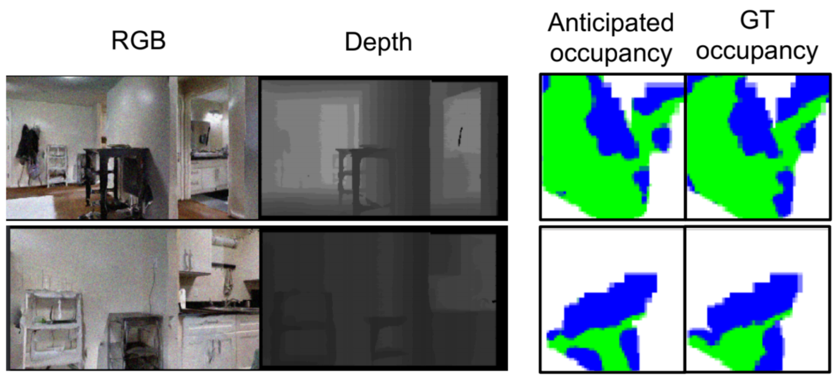

Fig. S15: Qualitative results from the 2020 Habitat Challenge: On the left, we show the noisy RGB and depth inputs. On the right, we show the corresponding anticipated and ground-truth occupancy. Our model learns to anticipate accurately in the presence of noise.