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.

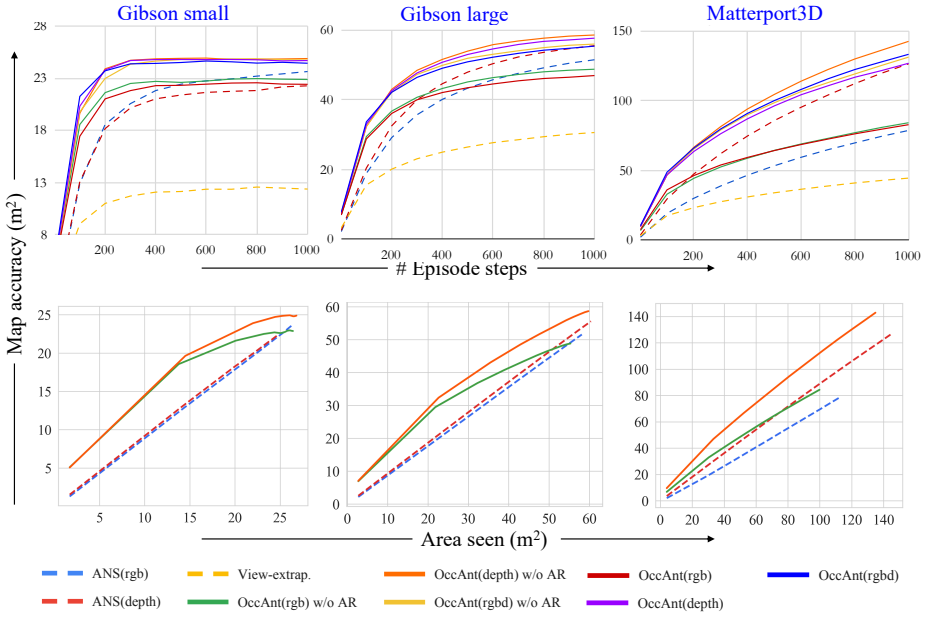

Fig. S1: **Noise-free exploration results:** Map accuracy (m^2) as a function of episode duration (top row) and area seen (bottom row) for Gibson (small and large splits) and Matterport3D under noise-free conditions. **Top:** Our OccAnt approach (solid lines) rapidly attains higher map accuracy than 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 [7,2]). 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 [7]) and the predicted maps in the center (from [2]) are indicated below each plot.

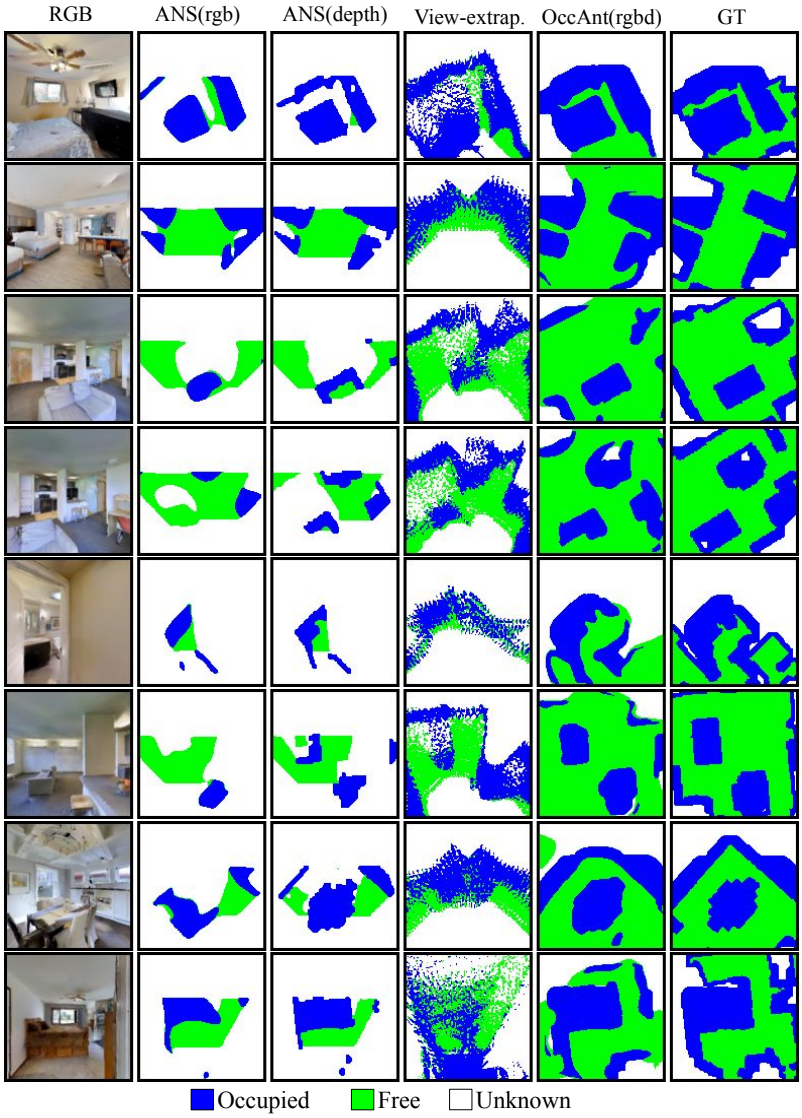

Fig. S2: **Occupancy anticipation successful cases:** ANS(rgb) is trained to predict the visible occupancy (2nd column) and ANS(depth) (3rd column) directly uses the visible occupancy (within a 3m range). Both these methods are unable to account for regions that are not visible or outside the sensing 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).

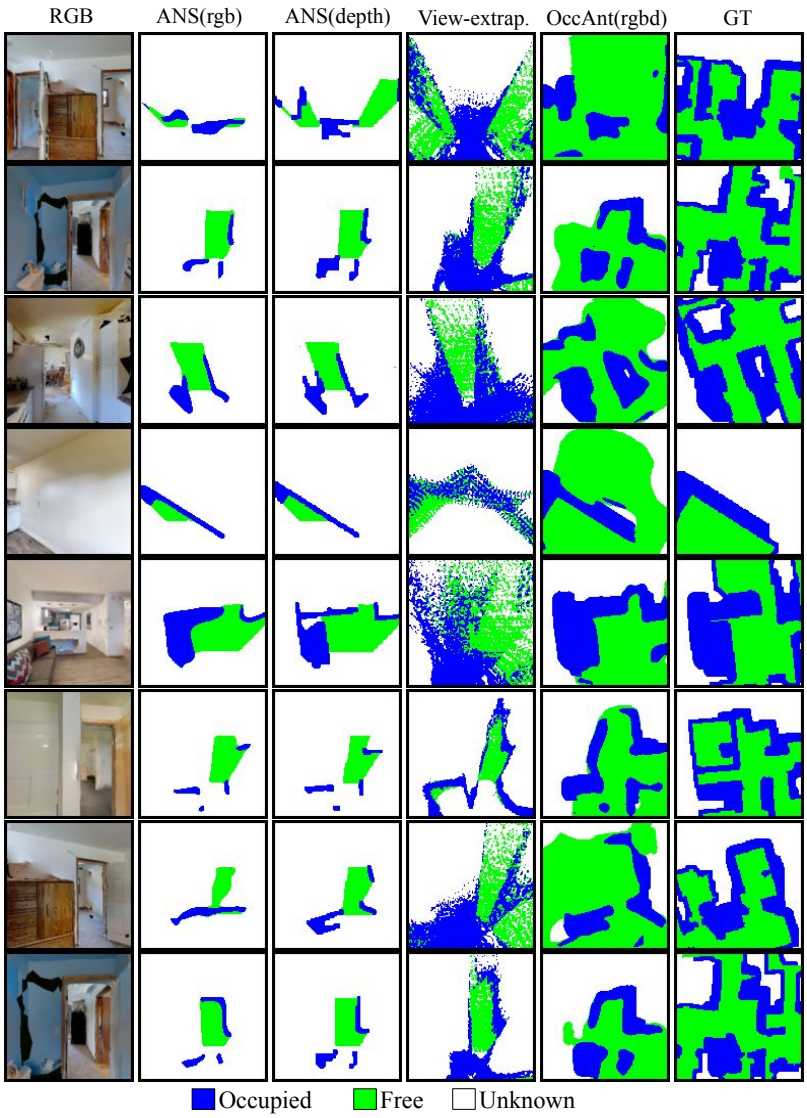

Fig. S3: **Occupancy anticipation failure cases:** Our approach OccAnt(rgbd) incorrectly predicts narrow corridors as occupied, and is unable to handle cases where multiple solutions may exist. For example, in rows 2, 5 and 8, it predicts that the corridors in the center of the map are blocked. In row 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.

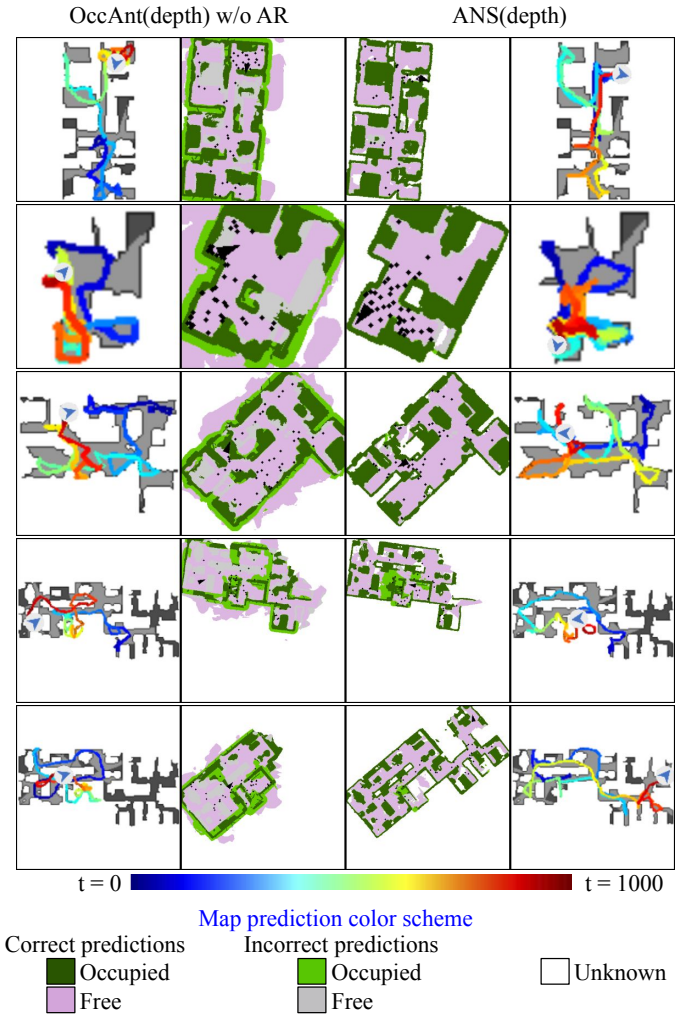

Fig. S5: We highlight one key weakness of exploration using occupancy anticipation, which is the impact of classification errors in occupancy estimates. We compare OccAnt(depth) w/o AR with ANS(depth) in Gibson. In some cases, OccAnt(depth) w/o AR tends to generate false negatives for occupied regions, 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).

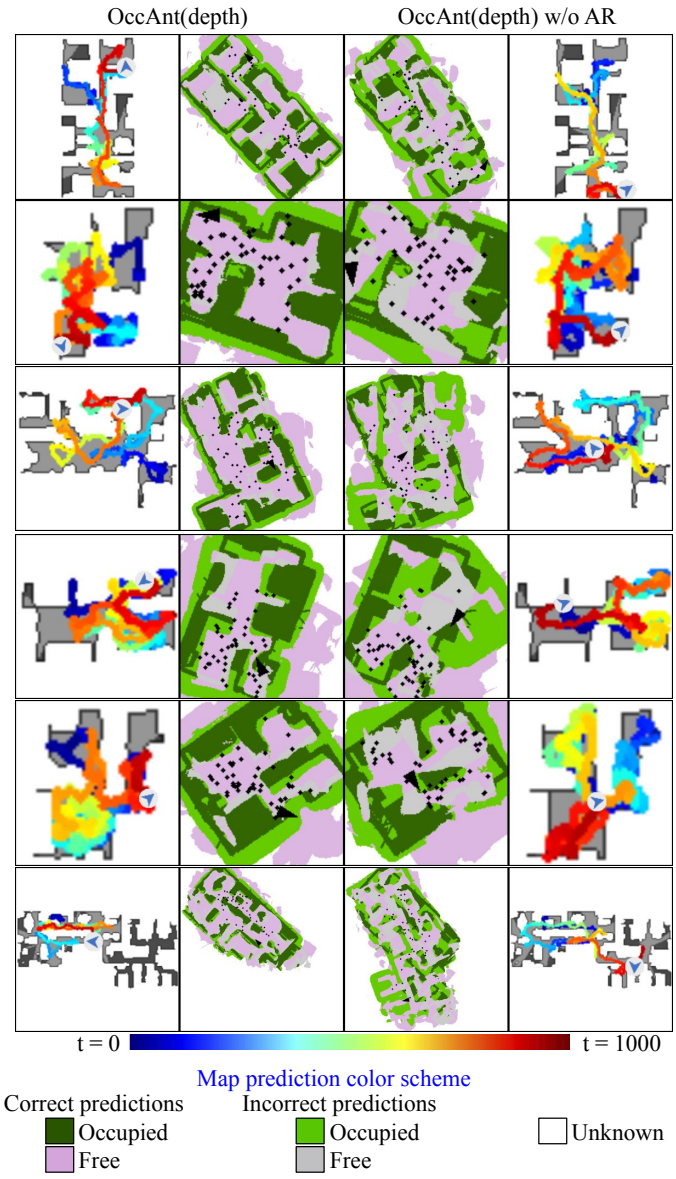

Fig. S6: **The impact of using anticipation reward:** In Table 6 from the main paper and Table S2 in Supp., we could see that models using anticipation reward generally leads to higher map qualities in noisy test conditions. Here, we show that, when the model that uses the anticipation reward (OccAnt(depth) 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) [2]	✗	✗	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) [2]	✗	✗	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) [2]	✗	✗	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

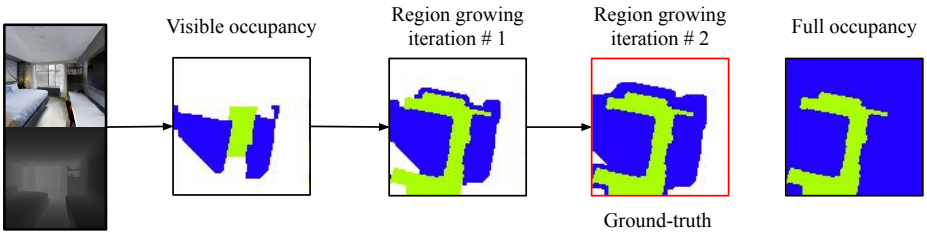

Fig. S7: Pipeline for generating anticipation ground-truth.

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 [2], 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 [2] 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 [2], which we discuss in the following. For shortest path planning, we use an A* planner instead of fast-marching [11] used in [2] 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 [2] 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 [6]
# 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 [6]
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 [10] 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.

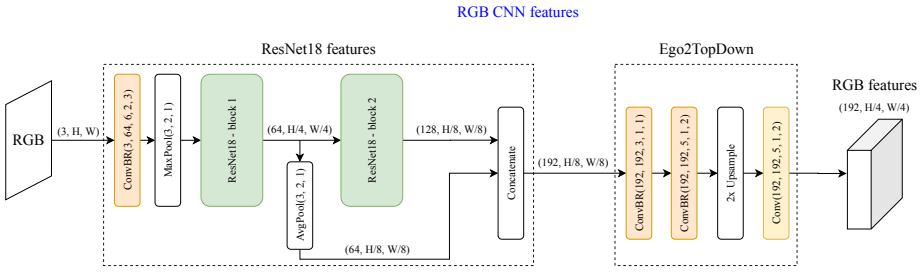

Fig. S8: **RGB CNN features:** extracts features from RGB images using ResNet18 blocks, and further processes the features to obtain compatible RGB features in a top-down view.

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 [2] 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 [9].

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 [12,14]. Since Habitat [7] does

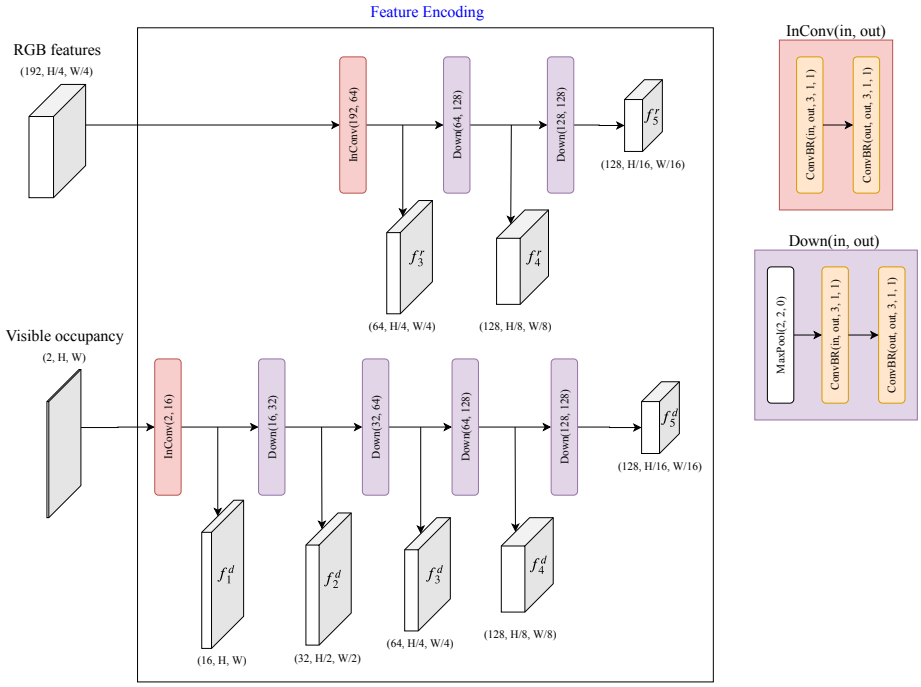

Fig. S9: **Feature encoding:** The RGB features and visible occupancy are encoded using independent UNet encoding layers. The expanded view of the "InConv" and "Down" blocks are shown on the right. The encoded RGB and visible occupancy features at different levels are $f^r_{3:5}$ and $f^d_{1:5}$, respectively.

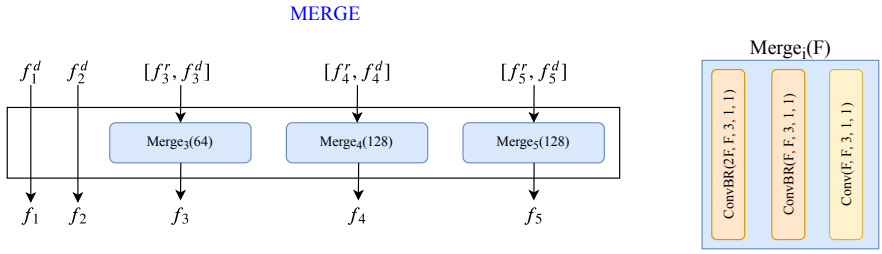

Fig. S10: **MERGE:** combines the RGB ($f^r_{3:5}$) and visible occupancy ($f1:5^d$) features obtained from Feature encoding layers in a layerwise fashion to obtain a set of merged features $f = f_{1:5}$. Since the RGB features are not available at levels 1 and 2, it simply uses the visible occupancy 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 ±45° heading offsets and aim to regress those from the egocentric view (see Fig. S13). Since each camera has a 90° FoV,

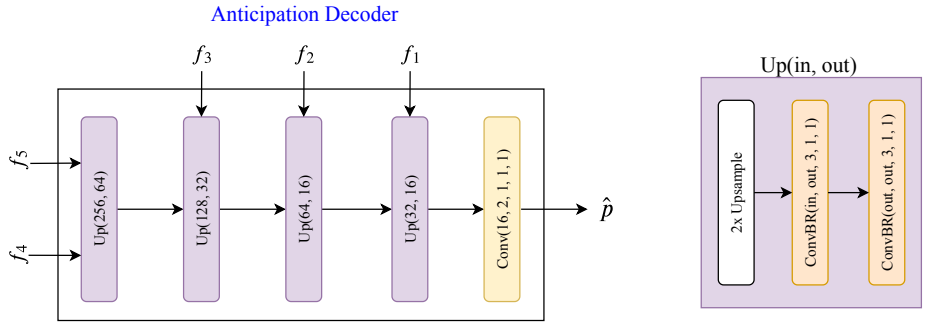

Fig. S11: **Anticipation Decoder:** a typical UNet decoder that takes features provided by MERGE ($f = f_{1:5}$) and decodes them using residual connections to obtain the anticipated occupancy map $\hat{p}$. The expanded view of the "Up" block is shown on the right.

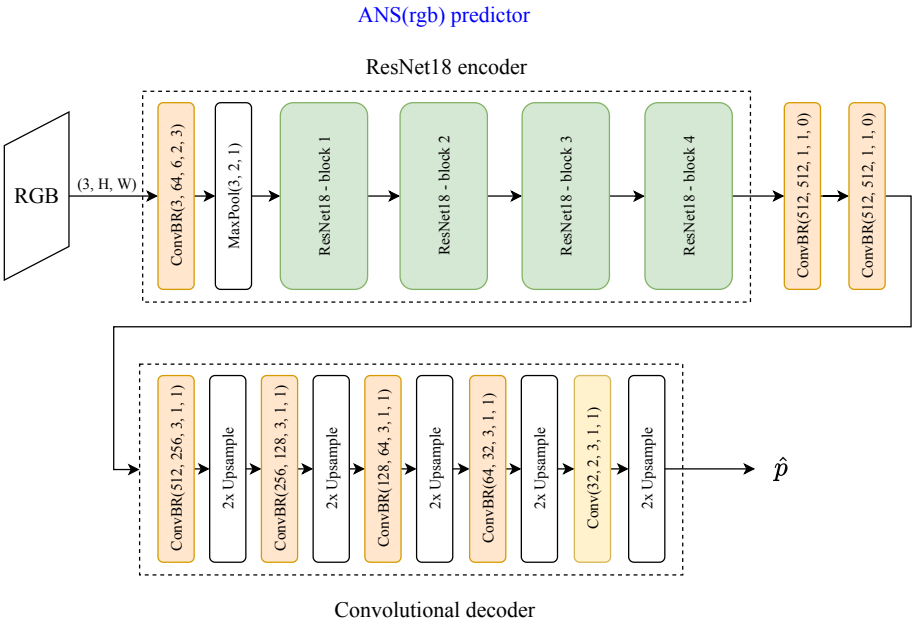

Fig. S12: **ANS projection unit:** ResNet-18 features are extracted, followed by two fully connected layers (represented by 1×1 convolutions) and a convolutional decoder that uses "Upsample" blocks to increase the output resolution and predict the occupancy estimates $\hat{p}$. Note that this is supervised 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 [14] 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.

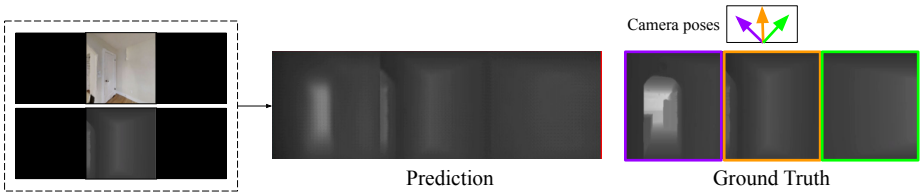

Fig. S13: **View extrapolation task:** Given the agent's egocentric RGB-D input, we predict the depth-map for two additional depth-sensors placed at 45° angles to the left (purple) and right (green) of the central input (orange). These are geometrically projected to the top-down view to obtain the occupancy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

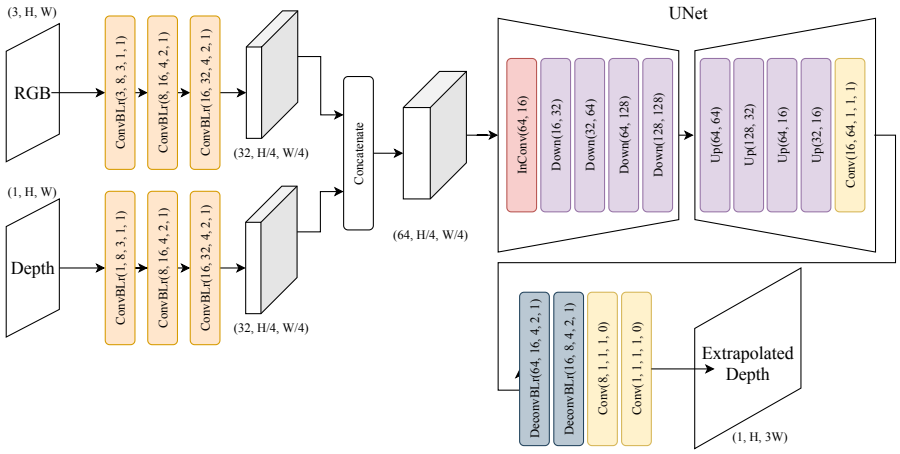

Fig. S14: **View extrapolation architecture [14]:** The 90° FoV RGB and depth inputs are independently encoded using Convolutional layers, concatenated and processed using a UNet model. The decoded features from UNet are used to extrapolate the final depth predictions. "DeconvBLr" uses `nn.ConvTranspose2D` 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

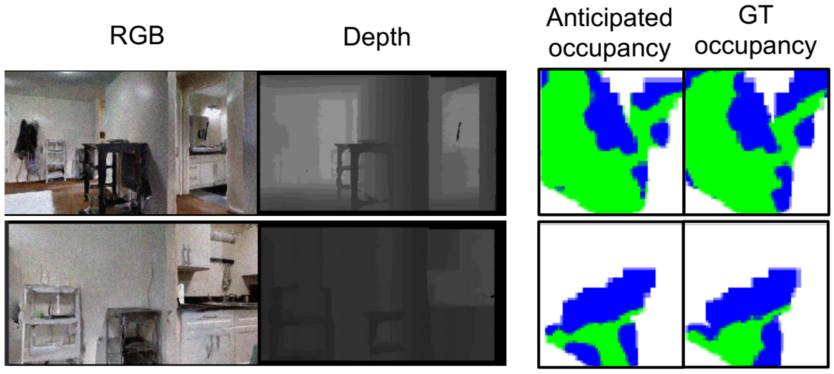

Fig. S15: Qualitative results from the 2020 Habitat Challenge: On the left, we show the noisy RGB and depth inputs. On the right, we show the corresponding anticipated and ground-truth occupancy. Our model learns to anticipate accurately in the presence of noise.

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 [4] and learned [13] 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 [3] for the depth sensor. Additionally, actuation noise in the robot motions is simulated by using a noise model obtained from the LoCoBot [8].

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 [5].[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.

References

1. The Habitat Challenge 2020. https://aihabitat.org/challenge/2020/
2. 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)
3. Choi, S., Zhou, Q.Y., Koltun, V.: Robust reconstruction of indoor scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
4. 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)
5. 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
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. 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)
8. 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)
9. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill **1**(10), e3 (2016)
10. 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
11. 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)
12. 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)

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

13. 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)
14. 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)