LEARNING EXPLORATION POLICIES FOR NAVIGATION

Tao Chen[1] **Saurabh Gupta**[2] **Abhinav Gupta**[1,2]
[1]Carnegie Mellon University
[2]Facebook AI Research

ABSTRACT

Numerous past works have tackled the problem of task-driven navigation. But, how to effectively explore a new environment to enable a variety of down-stream tasks has received much less attention. In this work, we study how agents can autonomously explore realistic and complex 3D environments without the context of task-rewards. We propose a learning-based approach and investigate different policy architectures, reward functions, and training paradigms. We find that use of policies with spatial memory that are bootstrapped with imitation learning and finally finetuned with coverage rewards derived purely from on-board sensors can be effective at exploring novel environments. We show that our learned exploration policies can explore better than classical approaches based on geometry alone and generic learning-based exploration techniques. Finally, we also show how such task-agnostic exploration can be used for down-stream tasks. Videos are available at: https://sites.google.com/view/exploration-for-nav/.

1 INTRODUCTION

Imagine your first day at a new workplace. If you are like most people, the first task you set for yourself is to become familiar with the office so that the next day when you have to attend meetings and perform tasks, you can navigate efficiently and seamlessly. To achieve that goal, you explore your office without the task context of target locations you have to reach and build a generic understanding of space. This step of task-independent exploration is quite critical yet often ignored in current approaches for navigation.

When it comes to navigation, currently there are two paradigms: (a) geometric reconstruction and path-planning based approaches (Hartley & Zisserman, 2003; Thrun et al., 2005; LaValle, 2006), and (b) learning-based approaches (Mirowski et al., 2017; Gupta et al., 2017; Savinov et al., 2018; Zhu et al., 2017). SLAM-based approaches, first build a map and then use localization and path planning for navigation. In doing this, one interesting question that is often overlooked is: How does one build a map? How should we explore the environment to build this map? Current approaches either use a human operator to control the robot for building the map (*e.g.* Thrun et al. (1999)), or use heuristics such as frontier-based exploration (Yamauchi, 1997). On the other hand for approaches that use learning, most only learn policies for specific tasks (Zhu et al., 2017; Mirowski et al., 2017; Gupta et al., 2017), or assume environments have already been explored (Savinov et al., 2018). Moreover, in context of such learning based approaches, the question of exploration is not only important at test time, but also at train time. And once again, this question is largely ignored. Current approaches either use sample inefficient random exploration or make impractical assumptions about full map availability for generating supervision from optimal trajectories.

Thus, a big bottleneck for both these navigation paradigms is an exploration policy: a task-agnostic policy that explores the environment to either build a map or sample trajectories for learning a navigation policy in a sample-efficient manner. But how do we learn this task-independent policy? What should be the reward for such a policy? First possible way is to not use learning and use heuristic based approaches (Yamauchi, 1997). However, there are four issues with non-learning based approaches: (a) these approaches are brittle and fail when there is noise in ego-estimation or localization; (b) they make strong assumptions about free-space/collisions and fail to generalize when navigation requires interactions such as opening doors *etc.*; (c) they fail to capture semantic priors that can reduce search-space significantly; and (d) they heavily rely on specialized sensors such as range scanners. Another possibility is to learn exploration policies on training environments. One

way is to use reinforcement learning (RL) with intrinsic rewards. Examples of intrinsic rewards can be "curiosity" where prediction error is used as reward signal or "diversity" which discourages the agent from revisiting the same states. While this seems like an effective reward, such approaches are still sample inefficient due to blackbox reward functions that can't be differentiated to compute gradients effectively. So, what would be an effective way to learn exploration policies for navigation?

In this paper, we propose an approach for learning policies for exploration for navigation. We explore this problem from multiple perspectives: (a) architectural design, (b) reward function design, and (c) reward optimization. Specifically, from perspective of reward function and optimization, we take the alternative paradigm and use supervision from human explorations in conjunction with intrinsic rewards. We notice that bootstrapping of learning from small amount of human supervision aids learning of semantics (*e.g.* doors are pathways). It also provides a good initialization for learning using intrinsic rewards. From the perspective of architecture design, we explore how to use 3D information and use it efficiently while doing exploration. We study proxy rewards that characterize coverage and demonstrate how this reward outperforms other rewards such as curiosity. Finally, we show how experience gathered from our learned exploration policies improves performance at down-stream navigation tasks.

2 Related Work

Our work on learning exploration policies for navigation in real world scenes is related to active SLAM in classical robotics, and intrinsic rewards based exploration in reinforcement learning. As we study the problem of navigation, we also draw upon recent efforts that use learning for this problem. We survey related efforts in these three directions.

Navigation in Classical Robotics. Classical approaches to navigation operate by building a map of the environment, localizing the agent in this map, and then planning paths to convey the agent to desired target locations. Consequently, the problems of mapping, localization and path-planning have been very thoroughly studied (Hartley & Zisserman, 2003; Thrun et al., 2005; LaValle, 2006). However, most of this research starts from a human-operated traversal of the environment, and falls under the purview of passive SLAM. Active SLAM, or how to automatically traverse a new environment for building spatial representations is much less studied. Cadena et al. (2016) present an excellent review of active SLAM literature, we summarize some key efforts here. Past works have formulated active SLAM as Partially Observable Markov Decision Processes (POMDPs) (Martinez-Cantin et al., 2009), or as choosing actions that reduce uncertainty in the estimated map (Carrillo et al., 2012). While these formulations enable theoretical analysis, they crucially rely on sensors to build maps and localize. Thus, such approaches are highly susceptible to measurement noise. Additionally, such methods treat exploration purely as a geometry problem, and entirely ignore semantic cues for exploration such as doors.

Learning for Navigation. In order to leverage such semantic cues for navigation, recent works have formulated navigation as a learning problem (Zhu et al., 2017; Gupta et al., 2017; Mirowski et al., 2017; Savinov et al., 2018). A number of design choices have been investigated. For example, these works have investigated different policy architectures for representing space: Zhu et al. (2017) use feed-forward networks, Mirowski et al. (2017) use vanilla neural network memory, Gupta et al. (2017) use spatial memory and planning modules, and Savinov et al. (2018) use semi-parametric topological memory. Different training paradigms have also been explored: Gupta et al. (2017) learn to imitate behavior of an optimal expert, Mirowski et al. (2017) and Zhu et al. (2017) use extrinsic reward based reinforcement learning, Pathak et al. (2018) learn an inverse dynamics model on the demonstrated trajectory way-points from the expert, while Savinov et al. (2018) use self-supervision. In our work here, we build on insights from these past works. Our policy architecture and reward definition use spatial memory to achieve long-horizon exploration, and we imitate human exploration demonstrations to boot-strap policy learning. However, in crucial distinction, instead of studying goal-directed navigation (either in the form of going to a particular goal location, or object of interest), we study the problem of autonomous exploration of novel environments in a task-agnostic manner. In doing so, unlike past works, we do not assume access to human demonstrations in the given novel test environment like Savinov et al. (2018), nor do we assume availability of millions of samples of experience or reward signals in the novel test environment like Zhu et al. (2017) or Mirowski et al. (2017). Moreover, we do not depend on extrinsically defined reward

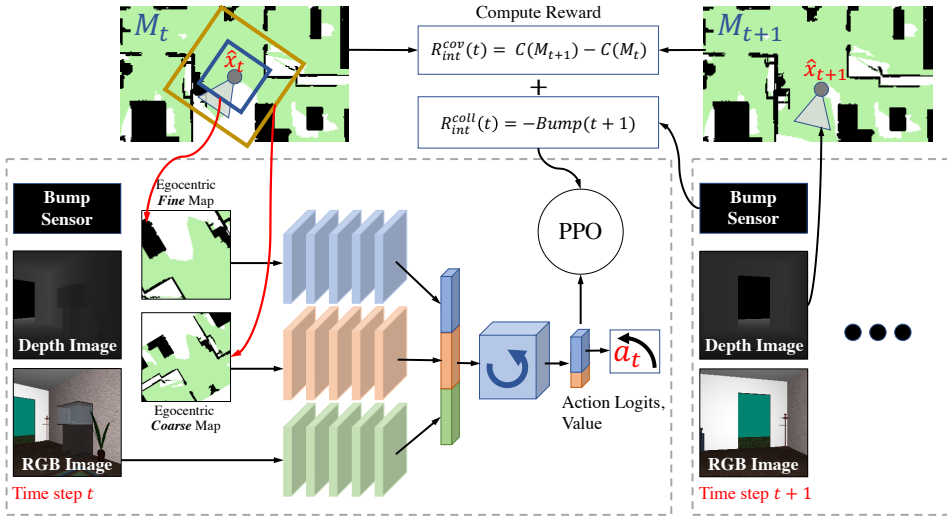


Figure 1: Policy and Training Architecture: Our scheme for learning exploration policies for navigation. We assume a mobile robot with an RGB-D camera and a bump sensor. We maintain a running estimate of its position $\hat{x}_t$. These position estimates are used to stitch together an approximate map of the environment using the depth images. The exploration policy π_e is a recurrent network takes egocentric crops of this map at two scales, and the current RGB image as input. The policy is trained using an intrinsic coverage reward that is computed from the approximate map and a collision penalty obtained from the bump sensor.

signals for training. We derive them using on-board sensors. This makes our proposed approach amenable to real world deployment. Learning based approaches (Gandhi et al., 2017; Sadeghi & Levine, 2017) have also been used to learn low-level collision avoidance policies. While they do not use task-context, they learn to move towards open space, without the intent of exploring the whole environment. Zhang et al. (2017) uses a differentiable map structure to mimic the SLAM techniques. Such works are orthogonal to our effort on exploration as our exploration policy can benefit from their learned maps instead of only using reconstructed occupancy map.

Exploration for Navigation. A notable few past works share a similar line of thought, and investigate exploration in context of reinforcement learning (Schmidhuber, 1991; Stadie et al., 2015; Pathak et al., 2017; Fu et al., 2017; Lopes et al., 2012; Chentanez et al., 2005). These works design intrinsic reward functions to capture novelty of states or state-action transitions. Exploration policies are then learned by optimizing these reward functions using reinforcement learning. Our work is most similar to these works. However, we focus on this problem in the specific context of navigation in complex and realistic 3D environments, and propose specialized policy architectures and intrinsic reward functions. We experimentally demonstrate that these specializations improve performance, and how learning based exploration techniques may not be too far from real world deployment. Jayaraman & Grauman (2018) use a related reward function (pixel reconstruction) to learn policies to look around (and subsequently solve tasks). However, they do it in context of 360° images, and their precise reward can't be estimated intrinsically. Xu et al. (2017) generate smooth movement path for high-quality camera scan by using time-varying tensor fields. Bai et al. (2016) propose an information-theoretic exploration method using Gaussian Process regression and show experiments on simplistic map environments. Kollar & Roy (2008) assume access to the ground-truth map and learn an optimized trajectory that maximizes the accuracy of the SLAM-derived map. In contrast, our learning policy directly tells the action that the agent should take next and estimates the map on the fly.

System Identification. Finally, the general idea of exploration prior to goal-driven behavior, is also related to the classical idea of system identification (Ljung, 1987). Recent interest in this idea with end-to-end learning (Yu et al., 2017; Duan et al., 2016; Mishra et al., 2018) has been shown to successfully adapt to novel mazes. In comparison, we tackle navigation in complex and realistic 3D environments, with very long spatio-temporal dependencies.

3 Approach

Let us consider a mobile agent that is equipped with an RGB-D camera and that can execute basic movement macro-actions. This agent has been dropped into a novel environment that it has never been in before. We want to learn an exploration policy π_e that enables this agent to efficiently

explore this new environment. A successful exploration of the new environment should enable the agent to accomplish down-stream tasks in this new environment efficiently.

We formulate estimation of π_e as a learning problem. We design a reward function that is estimated entirely using on-board sensors, and learn π_e using RL. Crucially, π_e is learned on a set of environments $\mathcal{E}_{train}$, and tested on a held-out set of environments $\mathcal{E}_{test}$, *i.e.*, $\mathcal{E}_{train} \bigcap \mathcal{E}_{test} = \phi$.

The key novelties in our work are: a) the design of the policy π_e, b) the design of the reward function R, and c) use of human demonstrations to speed up training of π_e. The design of the policy and the reward function depend on a rudimentary map of the environment that we maintain over time. We first describe the map construction procedure, and then detail the aforementioned aspects.

3.1 Map Construction

As the agent moves around, it uses its depth camera to build and update a map of the world around it. This is done by maintaining an estimate of the agent's location over time and projecting 3D points observed in the depth image into an allocentric map of the environment.

More specifically, let us assume that the agent's estimate of its current location at time step t is $\hat{x}_t$, and that the agent starts from origin (*i.e.* $\hat{x}_0 = \mathbf{0}$). When the agent executes an action a_t at time step t, it updates its position estimate through a known transition function f, *i.e.*, $\hat{x}_{t+1} = f(\hat{x}_t, a_t)$.

The depth image observation at time step $t+1$, D_{t+1} is back-projected into 3D points using known camera intrinsic parameters. These 3D points are transformed using the estimate $\hat{x}_{t+1}$, and projected down to generate a 2D map of the environment (that tracks what parts of space are known to be traversable or known to be non-traversable or unknown). Note that, because of slippage in wheels *etc.*, $\hat{x}_{t+1}$ may not be the same as the true relative location of the agent from start of episode x_{t+1}. This leads to aliasing in the generated map. We do not use expensive non-linear optimization (bundle adjustment) to improve our estimate $\hat{x}_{t+1}$, but instead rely on learning to provide robustness against this mis-alignment. We use the new back-projected 3D points to update the current map M_t and to obtain an updated map M_{t+1}.

3.2 Policy Architecture

Before we describe our policy architecture, let's define what are the features that a good exploration policy needs: (a) good exploration of a new environment requires an agent to meaningfully move around by detecting and avoiding obstacles; (b) good policy also requires the agent to identify semantic cues such as doors that may facilitate exploration; (c) finally, it requires the agent to keep track of what parts of the environment have or have not been explored, and to estimate how to get to parts of the environment that may not have been explored.

This motivates our policy architecture. We fuse information from RGB image observations and occupancy grid-based maps. Information from the RGB image allows recognition of useful semantic cues. While information from the occupancy maps allows the agent to keep track of parts of the environment that have or have not been explored and to plan paths to unexplored regions without bumping into obstacles.

The policy architecture is shown in Fig. 1. We describe it in detail here:

1. **Information from RGB images**: RGB images are processed through a CNN. We use a ResNet-18 CNN that has been pre-trained on the ImageNet classification task, and can identify semantic concepts in images.
2. **Information from Occupancy Maps**: We derive an occupancy map from past observations (as described in Sec. 3.1), and use it with a ResNet-18 CNN to extract features. To simplify learning, we do not use the allocentric map, but transform it into an egocentric map using the estimated position $\hat{x}_t$ (such that the agent is always at the center of the map, facing upwards). This map canonicalization aids learning. It allows the CNN to not only detect unexplored space but to also locate it with respect to its current location. Additionally, we use two such egocentric maps, a coarse map that captures information about a $40m \times 40m$ area around the agent, and a detailed map that describes a $4m \times 4m$ neighborhood. Some of these design choices were inspired by

recent work from Gupta et al. (2017), Parisotto & Salakhutdinov (2018) and Henriques & Vedaldi (2018), though we make some simplifications.

3. **Recurrent Policy**: Information from the RGB image and maps is fused and passed into an RNN. Recurrence can allow the agent to exhibit coherent behavior.

3.3 Coverage Reward

We now describe the intrinsic reward function that we use to train our exploration policy. We derive this intrinsic rewards from the map M_t, by computing the coverage. Coverage $C(M_t)$ is defined as the total area in the map that is known to be traversable or known to be non-traversable. Reward $R^{cov}_{int}(t)$ at time step t is obtained via gain in coverage in the map: $C(M_{t+1}) - C(M_t)$. Intuitively, if the current observation adds no obstacles or free-space to the map then it adds no information and hence no reward is given. We also use a collision penalty, that is estimated using the bump sensor, $R^{coll}_{int}(t) = -Bump(t+1)$, where $Bump(t+1)$ denotes if a collision occurred while executing action a_t. R^{cov}_{int} and R^{coll}_{int} are combined to obtain the total reward.

3.4 Training Procedure

Finally, we describe how we optimize the policy. Navigating in complex realistic 3D houses, requires long-term coherent behavior over multiple time steps, such as exiting a room, going through doors, going down hallways. Such long-term behavior is hard to learn using reinforcement learning, given sparsity in reward. This leads to excessively large sample complexity for learning. To overcome this large sample complexity, we pre-train our policy to imitate human demonstrations of how to explore a new environment. We do this using trajectories collected from AMT workers as they were answering questions in House3D (Das et al., 2018). We ignore the question that was posed to the human, and treat these trajectories as a proxy for how a human will explore a previosuly unseen environment. After this phase of imitation learning, π_e is further trained via policy gradients (Williams, 1992) using proximal policy optimization (PPO) from Schulman et al. (2017).

4 Experiments

The goal of this paper is to build agents that can autonomously explore novel complex 3D environments. We want to understand this in context of the different choices we made in our design, as well as how our design compares to alternate existing techniques for exploration. While coverage of the novel environment is a good task-agnostic metric, we also design experiments to additionally quantify the utility of generic task-agnostic exploration for downstream tasks of interest. We first describe our experimental setup that consists of complex realistic 3D environments and emphasizes the study of generalization to novel environments. We then describe experiments that measure task-agnostic exploration via coverage. And finally, we present experiments where we use different exploration schemes for downstream navigation tasks.

4.1 Experimental Setup

We conducted our experiments on the House3D simulation environment Wu et al. (2018). House 3D is based on realistic apartment layouts from the SUNCG dataset Song et al. (2017) and simulates first-person observations and actions of a robotic agent embodied in these apartments. We use 20 houses each for training and testing. These sets are sampled from the respective sets in House 3D, and do not overlap. That is, testing is done on a set of houses not seen during training. This allows us to study generalization, *i.e.*, how well our learned policies perform in novel, previously unseen houses. We made one customization to the House 3D environment: by default doors in House 3D are rendered but not used for collision checking. We modified House 3D to also not render doors, in addition to not using them for collision checking.

Observation Space. We assume that the agent has an RGB-D camera with a field of view of 60°, and a bump sensor. The RGB-D camera returns a regular RGB image and a depth image that records the distance (depth information is clipped at 3m) of each pixel from the camera. The bump sensor returns if a collision happened while executing the previous action.

Action Space. We followed the same action space as in EmbodiedQA (Das et al., 2018), with 6 motion primitives: *move forward* 0.25m, *move backward* 0.25m, *strafe left* 0.25m, *strafe right* 0.25m, *turn left* 9°, and *turn right* 9°.

Extrinsic Environmental Reward. We do not use any externally specified reward signals from the environment: $R_{ext}(t) = 0$.

Intrinsic Reward. As described in Sec. 3.3, the intrinsic reward of the agent is based upon the map that it constructs as it moves around (as described in Sec. 3.1), and readings from the bump sensor. Reward $R_{int}(t) = \alpha R^{\text{cov}}_{int}(t) + \beta R^{\text{coll}}_{int}(t)$. Here R^{cov}_{int} is the coverage reward, and R^{coll}_{int} is the collision reward. α, β are hyper-parameters to trade-off how aggressive the agent is.

Training. As described in Sec. 3.4, we train policies using imitation of human trajectories and RL.

1. *Imitation from Human Exploration Trajectories.* We leverage human exploration trajectories collected from AMT workers as they answered questions for the EmbodiedQA task (Das et al., 2018). We ignore the question that was posed to the AMT worker, and train our policy to simply mimic the actions that the humans took. As the humans were trying to answer these questions in previously unseen environments, we assume that these trajectories largely exhibit exploration behavior. We used a total of 693 human trajectories for this imitation.
2. *Reinforcement Learning*: After learning via imitation, we further train the policy using reinforcement learning on the training houses. We use PPO (Schulman et al., 2017) to optimize the intrinsic reward defined above. At the start of each episode, the agent is initialized at a random location inside one of the training houses. Each episode is run for 500 time-steps. We run a total of 6400 episodes which amounts to a total of 3.2M steps of experience.

Baselines. We next describe the various baseline methods that we experimented with. We implemented a classical baseline that purely reasons using geometry, and a learning baseline that uses curiosity for exploration.

1. *Frontier-based Exploration.* As a classical baseline, we experimented with frontier-based exploration (Yamauchi, 1997; Dornhege & Kleiner, 2013). This is a purely geometric method that utilizes the built map M_t. Every iteration, it samples a point in currently unexplored space, and plans a path towards it from the current location (unobserved space is assumed to be free). As the plan is executed, both the map and the plan are updated. Once the chosen point is reached, this process is repeated.
2. *Curiosity-based Exploration.* The next baseline we tried was curiosity-based exploration. In particular, we use the version proposed by Pathak et al. (2017) that uses prediction error of a forward model as reward. We use the modifications proposed by Burda et al. (2019), and only train a forward model. We prevent degeneracy in forward model by learning it in the fixed feature space of a ResNet-18 model that has been pre-trained on the ImageNet classification task.

4.2 Coverage Quality

We first measure exploration by itself, by measuring the *true* coverage of the agent. We compute the true coverage using the map as described in Sec. 3.1, except for using the true location of the agent rather than the estimated location (*i.e.* x_t instead of $\hat{x}_t$). We study the following three scenarios:

Without Estimation Noise: We first evaluate the performance of the agent that is trained and tested without observation noise, that is, $\hat{x}_t = x_t$. Note that *this setting is not very realistic as there is always observation error in an agent's estimate of its location.* It is also highly favorable to the frontier-based exploration agent, which very heavily relies on the accuracy of its maps. Fig. 2(left) presents the performance of different policies for this scenario. We first note that the curiosity based agent (**IL + RL with Curiosity**) explores better than a random exploration policy (that executes a random action at each time step). **Straight** is a baseline where the agent moves along a straight line and executes a random number of 9° turns when a collision occurs, which is a strategy used by many robot vacuums. Such strategy does not require RGB or depth information, and performs better than the curiosity based agent. However, both policies are worse than an RGB only version of our method, an RGB only policy that is trained with our coverage reward (**IL + RL with RGB**). Our full system (**IL + RL with Map + RGB**) that also uses maps as input performs even better.

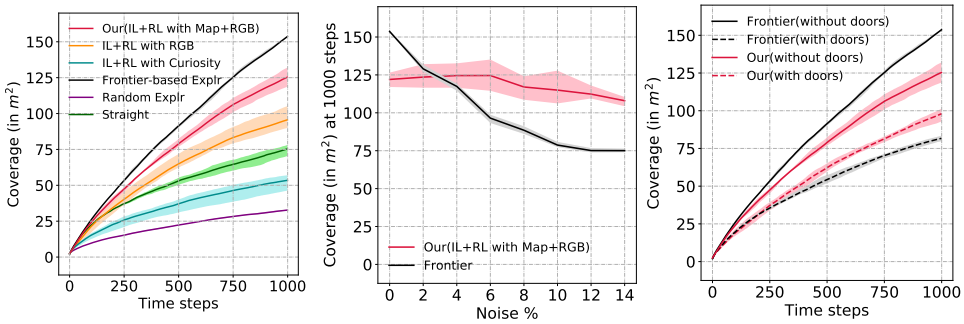


Figure 2: Coverage Performance. Policies are tested on 100 exploration runs (5 random start locations each on 20 testing houses). **Left figure** plots average coverage as a function of number of time steps in the episode. **Center figure** studies impact of noise in state estimation. We plot average coverage at end of episode as a function of amount of noise. **Right figure** studies the case when there is a mis-match between geometry and affordance, and plots average coverage as a function of time steps in episode. All plots report mean performance over 3 runs, and the shaded area represents the minimum and maximum performance.

Frontier-based exploration (**Frontier-based Explr**) has the best performance in this scenario. As noted, this is to expect as this method gets access to perfect, fully-registered maps, and employs optimal path planning algorithm to move from one place to another. It is also worth noting that, it is hard to use such classical techniques in situations where we only have a RGB images. In contrast, learning allows us to easily arrive at policies that can explore using only RGB images at test time.

With Estimation Noise: We next describe a more realistic scenario that has estimation noise, *i.e.* $\hat{x}_{t+1}$ is estimated using a noisy dynamics function. In particular, we add truncated Gaussian noise to the transition function f at each time step. The details of the noise generation is elaborated in Appendix C.4. The noise compounds over time. Even though such a noise model leads to compounding errors over time (as in the case of a real robot), we acknowledge that this simple noise model may not perfectly match noise in the real world. Fig. 2(center) presents the coverage area at the end of episode (1000 time steps) as a function of the amount of noise introduced[1]. When the system suffers from sensor noise, the performance of the frontier-based exploration method drops rapidly. In contrast, our learning-based agent that wasn't even trained with any noise continues to performs well. Even at relatively modest noise of 4% our learning based method already does better than the frontier-based agent. We additionally note that our agent can even be trained when the intrinsic reward estimation itself suffers from state estimation noise: for instance performance with 10% estimation noise (for intrinsic reward computation and map construction) is $98.9m^2$, only a minor degradation from $117.4m^2$ (10% estimation noise for map construction at test time only).

Geometry and Affordance Mismatch: Next, to emphasize the utility of learning for this task, we experiment with a scenario where we explicitly create a mismatch between geometry and affordance of the environment. We do this by rendering doors, but not using them for collision checking (*i.e.* the default House 3D environment). This setting helps us investigate if learning based techniques go beyond simple geometric reasoning in any way. Fig. 2(right) presents performance curves. We see that there is a large drop in performance for the frontier-based agent. This is because it is not able to distinguish doors from other obstacles, leading to path planning failures. However, there is a relatively minor drop in performance of our learning-based agent. This is because it can learn about doors (how to identify them in RGB images and the fact that they can be traversed) from human demonstrations and experience during reinforcement learning.

4.2.1 Ablation Study

We also conducted ablations of our method to identify what parts of our proposed technique contribute to the performance. We do these ablations in the setting without any estimation noise, and use coverage as the metric.

Imitation Learning: We check if pre-training with imitation learning is useful for this task. We test this by comparing to the models that were trained only with RL using the coverage reward. The left two plots in Fig. 3 shows performance of agents with following combinations: *RL*: policy trained

[1]A noise of η means we sample perturbations from a truncated Gaussian distribution with zero-mean, η standard deviation, and a total width of 2η. This sampled perturbation is scaled by the step length (25cm for x, y and 9° for azimuth θ) and added to the state at each time step.

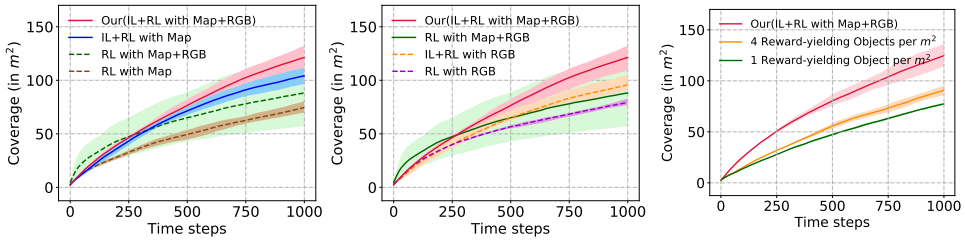


Figure 3: Ablation Study: We report average coverage as a function of time step in episode. As before we plot mean over 3 runs and minimum and maximum performance. **Left figure** shows that using the RGB image helps improve performance, **center figure** shows that using the map helps improve performance. We can also see that imitation learning improves coverage and reduces the variance in performance. **Right figure** shows the comparison between different reward design. We can see that our intrinsic reward enables the agent to explore more efficiently in the testing time.

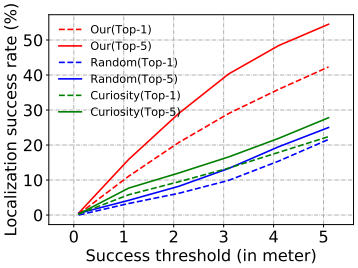
with PPO only, *IL*: policy trained with imitation learning, *Map*: policy uses constructed maps as input, *RGB*: policy uses RGB images as input. In all settings, pre-training with imitation learning helps improve performance, though training with RL improves coverage further. Also, policies trained using RL have a fairly large variance. Imitation learning also helps reduce the variance.

RGB Observations and Map: Fig. 3 (left) and Fig. 3 (center) respectively show that both RGB images and map inputs improve performance.

Intrinsic Reward: We also compare our intrinsic reward design with extrinsic reward design, as shown in Fig. 3 (right). The extrinsic reward is setup as follows: we randomly generated a number of locations evenly distributed across the traversable area of the houses, where the agent will get a positive reward if the agent is close to any of these locations. Once the agent gets a reward from a location, this location will be no longer taken into account for future reward calculation. We tried two settings where we place 1 or 4 reward-yielding objects per m^2. We can see that our coverage map reward provides a better reward signal and in fact can be estimated intrinsically without needing to instrument the environment.

4.3 Using Exploration for Downstream Task

Now that we have established how to explore well, we next ask if task-agnostic exploration is even useful for downstream tasks. We show this in context of goal-driven navigation. We execute the learned exploration policy π_e (for 1500 time steps) to explore a new house, and collect experience (set of images $\mathcal{I}$ along with their pose $\mathbf{p}$). Once this exploration has finished, the agent is given different navigation tasks. These navigation tasks put the agent at an arbitrary location in the environment, and give it a *goal image* that it needs to reach. More specifically, we reset the agent to 50 random poses (positions and orientations) in each testing house (20 testing houses in total) and get the RGB camera view. The agent then needs to use the experience of the environment acquired during exploration to efficiently navigate to the desired target location. The efficiency of navigation on these test queries measures the utility of exploration.



SPL (No exploration)	SPL (Our exploration)
0.69	**0.87**

Figure 4: Exploration for down-stream tasks. We evaluate utility of exploration for the down-stream navigation tasks. **Top plots** shows effectiveness of the exploration trajectory for localization of goal images in a new environment. **Bottom table** compares efficiency of reaching goals without and with maps built during exploration.

This evaluation requires a navigation policy π_n, that uses the exploration experience and the goal image to output actions that can convey the agent to the desired target location. We opt for a simple policy π_n. π_n first localizes the target image using nearest neighbor matching to the set of collected images $\mathcal{I}$ (in ImageNet pre-trained ResNet-18 feature space). It then plans a path to the this estimated target location using a occupancy map computed from $\mathcal{I}$. We do this experiment in the setting without any state estimation noise.

We independently measure the effectiveness of exploration data for a) localization of the given target images, and b) path planning efficiency in reaching desired locations. **Localization performance:** We measure the distance between the agent's estimate of the goal image location, and the true goal image location. Fig. 4 (top) plots the success at localization as a function of the success threshold

(distance at which we consider a localization as correct). We report top-1 and top-5 success rates. We compare to the random exploration baseline and curiosity-driven baseline which serve to measure the hardness of the task. We see our exploration scheme performs well. **Path planning efficiency:** Next, we measure how efficiently desired goal locations can be reached. We measure performance using the SPL metric as described by Anderson et al. (2018) (described in the appendix, higher is better). We compare against a baseline that does not have any prior experience in this environment and derives all map information on the fly from goal driven behavior (going to the desired test location). Both agents take actions based on the shortest-path motion planning algorithm. Once again these serve as a measure of the hardness of the topology of the underlying environment. As shown in Fig. 4 (bottom), using exploration data from our policy improves efficiency of paths to reach target locations.

5 Discussion

In this paper, we motivated the need for learning explorations policies for navigation in novel 3D environments. We showed how to design and train such policies, and how experience gathered from such policies enables better performance for downstream tasks. We think we have just scratched the surface of this problem, and hope this inspires future research towards semantic exploration using more expressive policy architectures, reward functions and training techniques.

References

Robert G Abbott. Behavioral cloning for simulator validation. In *Robot Soccer World Cup*, pp. 329–336. Springer, 2007. 14

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Zamir. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 9, 12

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 14

Shi Bai, Jinkun Wang, Fanfei Chen, and Brendan Englot. Information-theoretic exploration with bayesian optimization. In *IROS*, 2016. 3

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019. 6

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 2016. 2

Henry Carrillo, Ian Reid, and José A Castellanos. On the comparison of uncertainty criteria for active slam. In *ICRA*, 2012. 2

Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *NIPS*, 2005. 3

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. In *CVPR*, 2018. 5, 6, 14

Christian Dornhege and Alexander Kleiner. A frontier-void-based approach for autonomous exploration in 3D. *Advanced Robotics*, 2013. 6

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016. 3

Justin Fu, John Co-Reyes, and Sergey Levine. EX^2: Exploration with exemplar models for deep reinforcement learning. In *NIPS*, 2017. 3

Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *IROS*, 2017. 3

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017. 1, 2, 5

Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1, 2, 13

Joao F Henriques and Andrea Vedaldi. MapNet: An allocentric spatial memory for mapping environments. In *CVPR*, 2018. 5

Dinesh Jayaraman and Kristen Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *CVPR*, 2018. 3

Thomas Kollar and Nicholas Roy. Trajectory optimization using reinforcement learning for map exploration. *IJRR*, 2008. 3

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 1, 2

Lennart Ljung. *System identification: theory for the user*. Prentice-hall, 1987. 3

Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *NIPS*, 2012. 3

Ruben Martinez-Cantin, Nando de Freitas, Eric Brochu, José Castellanos, and Arnaud Doucet. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 2009. 2

Ryszard S Michalski, Ivan Bratko, and Avan Bratko. *Machine learning and data mining; methods and applications*. John Wiley & Sons, Inc., 1998. 14

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. In *ICLR*, 2017. 1, 2

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018. 3

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *ICLR*, 2018. 5

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 3, 6

Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. In *ICLR*, 2018. 2

Fereshteh Sadeghi and Sergey Levine. CAD^{2}RL: Real single-image flight without a single real image. In *RSS*, 2017. 3

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018. 1, 2

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, 1991. 3

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5, 6

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017. 5

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015. 3

Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B Cremers, Frank Dellaert, Dieter Fox, Dirk Hahnel, Charles Rosenberg, Nicholas Roy, Jamieson Schulte, et al. MINERVA: A second-generation museum tour-guide robot. In *ICRA*, 1999. 1

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005. 1, 2

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992. 5

Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018. 5, 13

Kai Xu, Lintao Zheng, Zihao Yan, Guohang Yan, Eugene Zhang, Matthias Niessner, Oliver Deussen, Daniel Cohen-Or, and Hui Huang. Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields. In *SIGGRAPH Asia*, 2017. 3

Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation (CIRA)*, 1997. 1, 6

Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *RSS*, 2017. 3

Jingwei Zhang, Lei Tai, Joschka Boedecker, Wolfram Burgard, and Ming Liu. Neural slam: Learning to explore with external memory. *arXiv preprint*, 2017. URL http://arxiv.org/abs/1706.09520. 3

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 1, 2

APPENDIX A SPL METRIC

As defined by Anderson et al. (2018), Success weighted by (normalized inverse) Path Length or SPL is computed as follows. Here, ℓ_i is the shortest-path distance between the starting and goal position, p_i is the length of the path actually executed by the agent, and S_i is a binary variable that indicates if the agent succeeded. SPL is computed over N trials as follows:

$$SPL = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{\ell_i}{\max(p_i, \ell_i)}. \tag{1}$$

APPENDIX B EXAMPLES OF POLICY INPUTS AND MAPS

We show an example of the policy inputs (RGB image and maps in two different scales) at time t and $t + 10$ in Fig. B.1. Green area means the known(seen) traversable area, blue area means known non-traversable area, and the white area means unknown area. Fig. B.2 shows how the map evolves and the agent moves as it explores a new house.

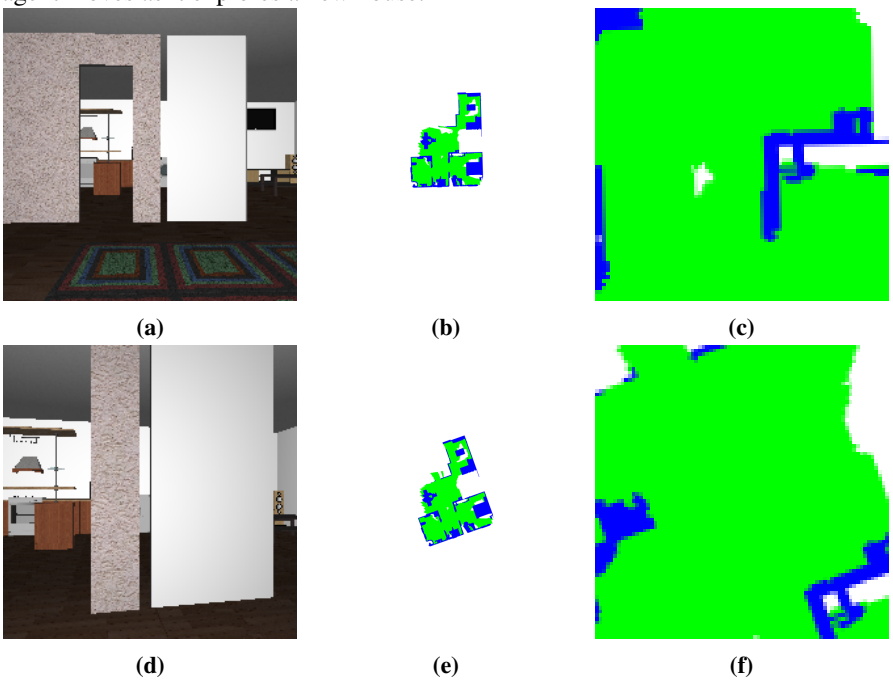


Figure B.1: Examples of policy inputs at time step t and $t + 10$. (a) and (d) are the RGB images, (b) and (e) are the coarse maps, and (c) and (f) are the fine maps.

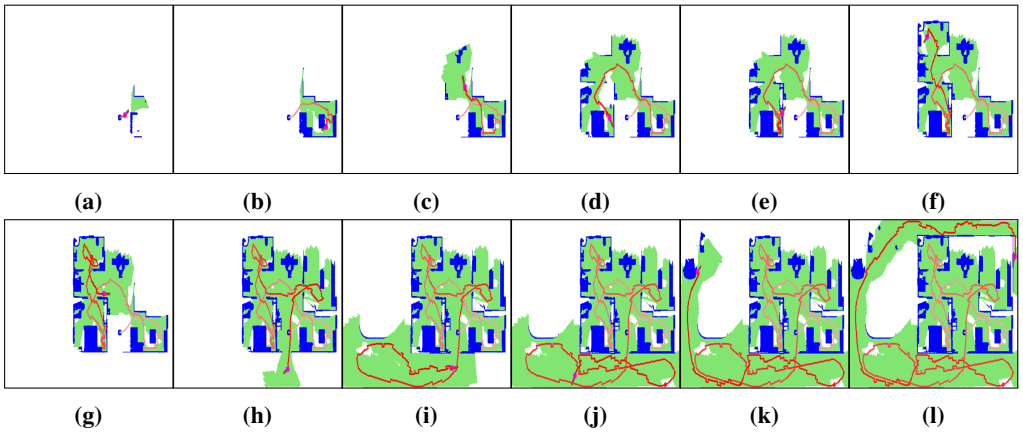


Figure B.2: Snapshots of the built map as the agent explores the house

APPENDIX C EXPERIMENTAL DETAILS

C.1 ENVIRONMENT DETAILS

In this paper, we simulated the agent in House3D environment Wu et al. (2018) and used 20 houses for training and 20 new houses for testing. Fig. C.1 shows the distribution of the total traversable area of these houses. Fig. C.2 shows some examples of the top-down views for training and testing houses (white is traversable, black is occupied).

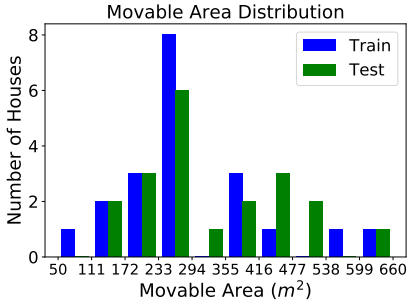


Figure C.1: Area distribution of training and testing houses (20 houses in each case). The average movable area of the training houses is 289.7 m^2. The average movable area of the testing houses is 327.9 m^2.

Figure C.2: Examples of house layouts (top-down views). **The left** three figures show 3 examples of the training houses. **The right** three figures show 3 examples of the testing houses. White regions indicate traversable area, black regions represent occupied area.

C.2 MAP RECONSTRUCTION

Given a sequence of camera poses (extrinsics) $([\boldsymbol{R}_1, \boldsymbol{t}_1], [\boldsymbol{R}_2, \boldsymbol{t}_2], ..., [\boldsymbol{R}_N, \boldsymbol{t}_N])$, and the corresponding depth images $(\boldsymbol{D}_1, \boldsymbol{D}_2, ..., \boldsymbol{D}_N)$ at each time step as well as the camera intrinsics $\boldsymbol{K} \in \mathbb{R}^{3\times3}$, we can reconstruct the point cloud of the scene based on the principle of mulitple-view geometry (Hartley & Zisserman, 2003). We can convert the points from pixel space to the camera space and then to the world space. More specifically, the formulas to achieve this can be summarized as follows:

$$\lambda_{ij}\boldsymbol{x}_{ij} = \boldsymbol{K}\left[\boldsymbol{R}_i, \boldsymbol{t}_i\right]\boldsymbol{w}_{ij} \quad \forall j \in \{1, 2, ..., S\}, i \in \{1, ..., N\} \tag{2}$$

$$\boldsymbol{W} = \bigcup_{i=1}^{N}\bigcup_{j=1}^{S}\boldsymbol{w}_{ij} \tag{3}$$

where S is the total number of pixels in each depth image, $\boldsymbol{x}_{ij} \in \mathbb{R}^3$ is the homogeneous coordinates for j_{th} pixel on i_{th} depth image D_i, $\lambda_{ij} \in \mathbb{R}$ is the depth value of the j_{th} pixel on i_{th} depth image D_i, and $\boldsymbol{w}_{ij} \in \mathbb{R}^4$ is the homogeneous coordinates for the corresponding point in the world coordinate system. We can get $\boldsymbol{w}_{ij}$ from $\boldsymbol{x}_{ij}$ based on Equation (2). And we can merge points via Equation (3).

C.3 TRAINING DETAILS

The occupancy map generated by the agent itself uses a resolution of 0.05m. Our policy uses a coarse map, and a detailed map. The coarse map captures information about a 40m×40m area around the agent, at a resolution of of $0.5m$. The occupancy map is down-sampled from 800×800 to 80×80 to generate the coarse map that is fed into the policy network. The detailed map captures information about a 4m×4m area around the agent at a $0.05m$ resolution. RGB images are rendered at 224×224 resolution in House3D and re-sized into 80×80 before they are fed into the policy network.

The size of the last fully-connected layer in ResNet18 architecture is modified to 128. Outputs of the three ResNet18 networks are concatenated into a 384-dimensional vector. This is transformed into a 128-dimensional vector via a fully-connected layer. Next, it's fed into a single-layer RNN(GRU) layer with 128 hidden layer size. The output of RNN layer is followed by two heads. One head (the policy head) has two fully-connected layers $(128 - 32 - 6)$. The other head (the value head) has two fully-connected layers $(128 - 32 - 1)$. We use ELU as the nonlinear activation function for the fully-connected layers after ResNet18.

Each episode consists of 500 steps in training. RNN time sequence length is 20. Coverage area $C(M_t)$ is measured by the number of covered grids in the occupancy map. Coefficient α for the coverage reward $R_{int}^{\text{cov}}(t)$ is 0.0005, coefficient β for $R_{int}^{\text{coll}}(t)$ is 0.006. PPO entropy loss coefficient is 0.01. Network is optimized via Adam optimizer with a learning rate of 0.00001.

C.4 Noise Model

Details of noise generation for experiments with estimation noise in Section 4.2:

1. Without loss of generality, we initialize the agent at the origin, that is $\hat{x}_0 = x_0 = \mathbf{0}$.
2. The agent takes an action a_t. We add truncated Gaussian noise to the action primitive(e.g., move forward 0.25m) to get the estimated pose $\hat{x}_{t+1}$, i.e., $\hat{x}_{t+1} = \hat{x}_t + \tilde{a}_t$ where $\hat{x}_t$ is the estimated pose in time step t and $\tilde{a}_t$ is the action primitive a_t with added noise.
3. Iterate the second step until the maximum number of steps is reached.

Thus, in this noise model, the agent estimates its new pose based on the estimated pose from the last time step and the executed action. Thus, we dont use oracle odometry in the noise experiments. This noise model leads to compounding errors over time (as in the case of a real robot), though we acknowledge that this simple noise model may not perfectly match noise in the real world.

C.5 Imitation Learning Details

We use behavioral cloning technique (Argall et al., 2009; Michalski et al., 1998; Abbott, 2007) to imitate the human behaviors in exploring the environments. We got the human demonstration trajectories from (Das et al., 2018). We ignored the question-answering part of the data and only used the exploration trajectories. We cleaned up the data by removing the trajectories that have less than 100 time steps. The trajectories are then converted into short sequences of trajectory segments ($(s_i, a_i, s_{i+1}, a_{i+1}, ..., s_{i+T}, a_{i+T})$, where T is based on the RNN sequence length). The policy is pretrained with behavioral cloning by imitating actions $(a_i, a_{i+1}, ..., a_{i+T})$ from the human demonstrations given the states $(s_i, s_{i+1}, ..., s_{i+T})$.