Learning Active Camera for Multi-Object Navigation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Getting robots to navigate to multiple objects autonomously is essential yet difficult in robot applications. One of the key challenges is how to explore environments efficiently with camera sensors only. Existing navigation methods mainly focus on fixed cameras and few attempts have been made to navigate with active cameras. As a result, the agent may take a very long time to perceive the environment due to limited camera scope. In contrast, humans typically gain a larger field of view by looking around for a better perception of the environment. How to make robots perceive the environment as efficiently as humans is a fundamental problem in robotics. In this paper, we consider navigating to multiple objects more efficiently with active cameras. Specifically, we cast moving camera to a Markov Decision Process and reformulate the active camera problem as a reinforcement learning problem. However, we have to address two new challenges: 1) how to learn a good camera policy in complex environments and 2) how to coordinate it with the navigation policy. To address these, we carefully design a reward function to encourage the agent to explore more areas by moving camera actively. Moreover, we exploit human experience to infer a rule-based camera action to guide the learning process. Last, to better coordinate two kinds of policies, the camera policy takes navigation actions into account when making camera moving decisions. Experimental results show our camera policy consistently improves the performance of multi-object navigation over four baselines on two datasets.

1 Introduction

In the multi-object navigation task, an intelligent embodied agent needs to navigate to multiple goal objects in a 3D environment. Typically, no pre-computed map is available and the agent needs to use a stream of egocentric observations to perceive the environment. This navigation ability is the basis for indoor robots and embodied AI. Significant recent progress on this problem can be attributed to the availability of large-scale visually rich 3D datasets [65, 7, 60, 63], developments in high-quality 3D simulators [65, 55, 40, 2], and research on deep memory-based architectures that combine geometry and semantics for learning representations of the 3D environment [8, 62, 52].

Despite these advances, how to efficiently perceive the environment and locate goal objects is still an unsolved problem. Agents in current research [9, 12, 43] perceive the environment via an RGB-D camera. However, the camera is set to look forward and the range of its view is often limited. As a result, these agents can only perceive the environment in front of themselves within or slightly beyond the view range [52]. As shown at the bottom-right in Figure 1, the agent keeps looking forward and perceives redundant information which is nearly identical to the previous observations. In contrast, we human beings may be attracted by surrounding things so that we turn head actively to receive more information when keeping walking straight. In this way, we can not only reach the position in front of ourselves efficiently but also get familiar with room arrangement and object location.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

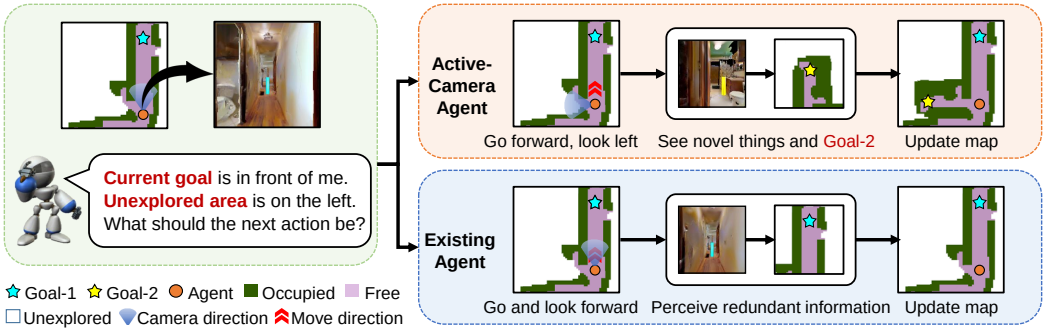

Figure 1: Illustration of an active-camera agent. At the top-right of the figure, the active-camera agent turns its camera to the left actively to look for novel things, *i.e., Goal-2*, when it is walking forward to *Goal-1*.

Motivated by the above observations, we design an agent which seeks to coordinate both camera and navigation actions. The agent, called **active-camera agent**, employs two kinds of policies, namely the navigation policy which determines where to go and a new camera policy which determines where to look. As shown at the top-right in Figure 1, the agent has found an object, *i.e.,* blue *Goal-1*, located at the end of a corridor. Two kinds of policies cooperate and actuate the agent to move forward and turn its camera to the left. As a result, the agent walks closer to the *Goal-1* and locates another goal, *i.e.,* yellow *Goal-2*, for the following navigation process. However, it is nontrivial to learn a good camera policy because of sophisticated RGB-D observations and complex indoor layouts. How to extract useful information from observations to judge which direction is worth being explored is difficult. Besides, the agent and the camera are moving simultaneously. How to coordinate these two actions for both the navigation and exploration processes is still unknown.

In this paper, we propose an **EXPloration-Oriented (EXPO) camera policy** to determine camera actions. Specifically, to better understand the sophisticated RGB-D observations, we transform them to a top-down occupancy map. Each pixel indicates whether it is an unexplored, free or occupied area. Such a map helps to simplify unstructured RGB-D observations and provides the necessary information, *e.g.,* explored areas and room layouts, for determining camera actions. Besides, to reduce the learning difficulty, we propose a heuristic module to infer an expected camera direction according to heuristic rules. We consider the heuristic module an expert and exploit it to guide the learning process. We then feed three types of information, *i.e.,* the progressively built map, the expected camera direction, and the upcoming navigation action, into a neural network to predict a camera action. The neural network considers both the heuristic rules and the navigation intention. We use a reinforcement learning algorithm to train the neural network by awarding camera actions that maximize the exploration area. We incorporate the EXPO camera policy with existing navigation policies [52, 62] and train them in an end-to-end manner for the multi-object navigation task. Extensive experiments on two benchmark demonstrate the effectiveness of the proposed methods.

Our main contributions are in threefold:

- Unlike existing agents that are set to look forward, we propose a navigation paradigm that an agent coordinates camera and navigation actions for efficiently perceiving environments to solve the multi-object navigation task.
- We propose to learn an EXPO camera policy to determine camera actions. The camera policy leverages heuristic rules to reduce the learning difficulty and takes into account the navigation intention to coordinate with the navigation policy. Such a camera policy can be incorporated with most existing navigation methods.
- Extensive experiments demonstrate consistent improvement over four navigation methods on MatterPort3D [7] and Gibson [65] datasets. More critically, the camera policy also exhibits promising transferability to unseen scenes.

2 Related Work

Visual indoor navigation. According to the types of goal, visual navigation task can be categorized into different classes such as PointGoal [27, 9] task where the goal is a given coordinate, ObjectGoal [27, 62] task where the agent needs to navigate to an object given by language and ImageGoal [12, 68] task where the object is specified by an image. The location of the goal is not explicitly known in the last two categories of above tasks and the agent is expected to explore the environment to find the goals. Classical approaches [26, 61, 28, 57, 39] solve the navigation problem via path planning [11, 28] on a constructed map, which usually requires handcraft design. Recent works [68, 47, 67, 59, 62, 15, 58, 44, 50, 32] aim to use learning-based methods in an end-to-end manner to learn a policy for navigation. Other works [9, 12, 52, 37, 41, 5, 42, 18, 46, 17, 25, 54, 22] combine the classic and learning-based methods, which use a learned SLAM module with a spatial map or topological map. We propose a camera policy, which seeks to move the camera actively and can be incorporated with existing navigation methods, to improve the navigation performance.

Exploration for navigation. Common methods explore the environment based on heuristics like the frontier-based exploration algorithm [66, 29, 23], which chooses frontier point between explored and unexplored areas as the exploration goal. Recent works tackle the exploration problem via learning [9, 20, 6, 21, 51, 30, 38, 48], which allows the agent to incorporate knowledge from previously seen environments and generalize to novel environments. Specially, Chen *et al.* [20] and Jayaraman *et al.* [34] use end-to-end reinforcement learning policy to maximize an external reward (*i.e.,* exploration area). Burda *et al.* [6] and Dean *et al.* [21] consider intrinsic rewards such as curiosity for efficient exploration, which performs better when external rewards are sparse. Chaplot *et al.* [9], Ramakrishnan [52] and Chen *et al.* [16] infer an exploration goal by a learned policy and navigate to it using path planner, which avoids sample complexity problem in end-to-end training. Elhafsi *et al.* [24] predict the map out of view range for better path planning. Unlike existing methods, we try to actively control the camera direction for efficient exploration.

Active perception. Active perception [3] aims to change the state parameters of sensors according to intelligent control strategies and gain more information about the environment. Common methods guide the view selection using information-theoretic approaches, such as mutual information [36, 13, 14, 31]. Recent work applies this strategy on different tasks such as object recognition [35, 1, 33], object localization [10, 68, 45, 49] and scene completion [34, 53]. We refer readers to [19] and [4] for a detailed review. In this paper, we propose a camera control strategy for active camera moving to help multi-object navigation.

3 Multi-object Navigation with Active Camera

3.1 Problem Formulation

Considering an agent equipped with one RGB-D camera in a novel environment, multi-object navigation task asks the agent to explore the environment to find multiple goal objects and then navigate to them. During these processes, existing methods [62] design a navigation policy to process the observations o from the environment and infer a navigation action a^n (*i.e.,* FORWARD, TURN-LEFT, TURN-RIGHT, and FOUND). The navigation action is responsible for moving toward goal objects and indicates whether agents have found goals. However, these methods do not consider moving camera direction actively during the navigation process. Thus, agents can only perceive the field along their navigation direction, which causes low efficiency of finding goal objects [52].

To resolve the above problem, we reformulate both the navigation and camera moving process to a Markov Decision Process (MDP). Formally, the MDP can be defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition distribution, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ is the reward function. Moreover, a policy $\pi(\cdot)$ determines an action given the current state. In this paper, we propose a new navigation paradigm that consists of a navigation policy $\pi^n(\cdot)$ and a **camera policy** $\pi^c(\cdot)$. In our context, we denote the state for both two MDPs by the observation of current environment. At each time step, for a given state, these two policies predict a navigation action $a^n \sim \pi^n(\cdot)$ together with a camera action $a^c \sim \pi^c(\cdot)$ for active camera moving. The action space $\mathcal{A}$ for a^n is similar to navigation action space. The possible camera action includes {TURN-CAMERA-LEFT, TURN-CAMERA-RIGHT, and KEEP}, where KEEP indicates the camera direction remains unchanged. After performing the action, the policies will

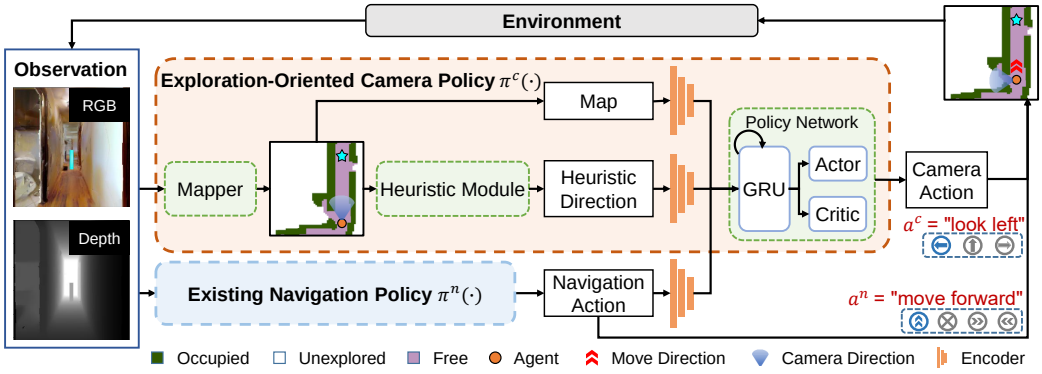

Figure 2: General scheme of an active-camera agent. The agent consists of two policies, namely the exploration-oriented camera policy determining where to look and existing navigation policy [52, 62] determining where to go.

receive a reward r whose details can be found in Section 3.3. We call the agent using this paradigm **active-camera agent** and the general scheme is shown in Figure 2.

3.2 Learning to Determine Camera Action

Learning a camera policy for determining camera actions is challenging because 1) it is difficult to understand complex room structure from RGB-D observations; 2) the position of agents is changing all the time so that camera policies must coordinate with navigation policies. In this paper, we design an EXPO camera policy which consists of three components, *i.e.,* a mapper, a heuristic module, and a policy network. The mapper transforms the RGB-D image to an occupancy top-down map, which reveals room layout and location of unexplored areas straightforwardly. The heuristic module infers a heuristic direction being worth exploring. The inferred direction can be served as a reference for camera policy and reduce the learning difficulty. Then, a policy network predicts camera actions considering the encoded features of the map, heuristic direction, and upcoming navigation action. The upcoming navigation action informs the policy network of the next location of agents. The overview of the camera policy is shown in Figure 2. Next, we introduce each component in detail.

Mapper. We follow existing work [52] to build an occupancy map by a mapper. Each map pixel has three possible values, *i.e.,* $0, 1, 2$, indicating an unexplored, free or occupied area, respectively. Specifically, the mapper takes RGB-D as input and predicts a local map $\mathbf{M}^l$. This local map represents a small area in front of the camera. Then we integrate the local map into a global map $\mathbf{M}^g$ via a registration function $R(\cdot)$, *i.e.,* $\mathbf{M}^g = R(\mathbf{M}^l, \mathbf{M}^g, q)$, where q is the camera's pose representing its position and orientation. The global map covers the whole explored environment.

Heuristic module. We seek to find a heuristic direction indicating which area is worth being explored by using heuristic rules. The principle is to find a direction where the camera sight is not blocked by occupancy (*e.g.,* wall). Following this principle, we draw K points $\mathcal{P} = \{p_i\}_{i=1}^K$ uniformly around the agent. Each point is located at angle θ_i from the current navigation direction and r meters away from the agent. We calculate geodesic distance $\mathcal{D} = \{D(p_i)\}_{i=1}^K$ from agent to these points on the progressively built global map $\mathbf{M}^g$ via A* path planning algorithm $D(\cdot)$ [28]. During path planning, we consider both the free and unexplored areas navigable. If the geodesic distance d_i is smaller than a threshold γ ($\gamma > r$) and the point p_i locates at unexplored areas in the global map, we denote these points explorable because there is no obstacle locating between the agent and point p_i. Moving the camera to these explorable directions allows the agent to see unexplored parts of environment. If there exist such explorable points, we select one point closest to the camera direction and take its angle θ_i as the heuristic direction θ^*. Otherwise, we set $\theta^* = 0$, *i.e.,* facing the navigation direction.

Policy network. With the progressively built map and inferred heuristic direction, we use a recurrent network to aggregate this information at each time step. The network takes as input map features, heuristic direction features and navigation action features. Specifically, a convolutional neural network is exploited to extract map features from a egocentric map which is cropped from the global map. The other two types of features are encoded from the heuristic direction θ^* and navigation action

Algorithm 1 Training method for active-camera agent.

Require: The parameters of the navigation policy $\pi^n(\cdot)$ and camera policy $\pi^c(\cdot)$, a mapper $m(\cdot)$, the number of points K, the radius r, the distance threshold γ, map registration function $R(\cdot)$, A* algorithm $D(\cdot)$, angle selection function $S(\cdot)$.

Initialize the parameters of $\pi^n(\cdot)$ and $\pi^c(\cdot)$ randomly.
Initialize an occupancy global map $\mathbf{M}^g = \mathbf{0}$.
while not convergent **do**
  Collect observation o from environment.
  // Update the global map
  Let local map $\mathbf{M}^l = m(o)$ and $\mathbf{M}^g = R(\mathbf{M}^l, \mathbf{M}^g, q)$, where q is agent's pose.
  Crop an egocentric map $\mathbf{M}^e$ from $\mathbf{M}^g$.
  // Infer heuristic direction
  Obtain K points $\mathcal{P}=\{p_i\}_{i=1}^K$ being r meters away from agent's position at angles $\Theta=\{\theta_i|\theta_i=\frac{2\pi i}{K}\}_{i=1}^K$.
  Calculate geodesic distance from the agent to these points $\mathcal{D} = \{d_i|d_i=D(p_i)\}_{i=1}^K$ on $\mathbf{M}^g$.
  Select a index set of explorable point $\mathcal{I} = \{i \mid \mathbf{M}^g[p_i]=0, d_i < \gamma\}$.
  Obtain heuristic direction $\theta^*=S(\{\theta_i, i \in \mathcal{I}\})$, where S returns the closest direction to current camera direction.
  // Sample action and update policies
  Sample navigation action $a^n \sim \pi^n(\cdot|o)$ and camera action $a^c \sim \pi^c(\cdot|\mathbf{M}^e, \theta^*, a^n)$.
  Compute reward via Eq. (1).
  Update the navigation and camera policies via PPO.
end while

a^n, respectively by using a learned embedding layer. The output state features from the recurrent network are fed into an actor-critic network. The actor predicts a camera action and the critic predicts a value of the agent's current state.

3.3 Reward Function for Camera Policy

We expect the camera policy to help agents explore environments more efficiently. We achieve this objective in two ways. On the one hand, we encourage the camera policy to follow the heuristic rules by awarding actions that move cameras toward heuristic direction. Such heuristic reward is defined as the reduction of angle $s \in [0^\circ, 180^\circ]$ between camera direction and heuristic direction *i.e.,* $r_{\text{heuristic}} = s_{t-1} - s_t$. On the other hand, because it is impractical to design a heuristic rule to cover all situations, we explicitly encourage the camera policy to explore more areas by an area reward $r_{\text{area}} = c_t - c_{t-1}$, which indicates the increase of explored area c in successive time steps. The camera agent makes its own decision according to recommended direction and occupancy map information. In addition, to avoid constant camera moving and reduce energy consumption, we expect agents to execute camera motor for camera moving as less frequently as possible by introducing a turn-penalty reward $r_{\text{turn-penalty}} = \mathbb{1}[\text{turn-camera}]$, where $\mathbb{1}[\text{turn-camera}]$ equals to 1 when the agent actuates camera motor to move a camera and otherwise equals to 0. The detail about camera motor execution will be described in Section 3.4. To sum up, the reward for camera policy is as follow:

$$r = \alpha \cdot r_{\text{heuristic}} + \beta \cdot r_{\text{area}} - r_{\text{turn-penalty}}, \tag{1}$$

where α and β are two scaling hyper-parameters. We train the camera policy to maximize such exploration reward using PPO [56]. In this way, the EXPO camera policy is encouraged to take into account heuristic rule-based decisions and to explore more areas by executing the camera motor as less frequently as possible.

3.4 Combination of Navigation and Camera Policies

Navigation and camera actions execution. We assume the camera is attached on a mobile robot platform using a revolute joint. The robot can move to a location using a wheel actuator and turn the camera using a camera motor simultaneously. The action space in this paper is based on the real-world coordinate. The rotation angles for navigation and camera actions are set to the same. Thus, if both navigation and camera actions tend to move to the same direction, *e.g.,* turning the platform and camera to the left, we only need to actuate the robot wheel and do not need to actuate the camera motor.

Incorporating camera policy into existing navigation methods. Existing navigation methods can be mainly categorized into two types, namely module SLAM-based and end-to-end learning-based. For the first type of navigation method, we use the existing navigation policy [52] to infer a navigation action, which is then fed to the proposed camera policy to infer a camera action. The paradigm is shown in Figure 2. For the end-to-end learning-based navigation method, the existing navigation policy [62] typically contains a same recurrent network as our camera policy and uses it to predict navigation actions. To better coordinate two kinds of policies, we use one recurrent network to predict both navigation action and camera action. Specifically, we feed this recurrent network both camera policy input (*i.e.,* map and heuristic direction) and navigation policy input (typically RGB-D images). Unlike the paradigm in Figure 2, the recurrent network does not need the features of navigation action anymore because it predicts both types of actions at the same time. We use the summation of exploration reward in Equation 1 and navigation reward in existing methods (typically the distance reduced to goals) to evaluate the goodness of predicted navigation-camera actions.

4 Experiments

4.1 Experimental Setup

Task details. We follow MultiON [62] to place multiple goal objects in the environment randomly. The objects are embodied with cylinders in different colors. In this way, We are free to decide the location and the number of goal objects to adjust the task difficulty. The agent succeeds in an episode if it calls FOUND action within a 1.5 meters radius of all objects in order. We call FOUND automatically when the agent is near the current goal object because we focus on evaluating the effectiveness of the agent finding objects and navigating to them. By default, we place three goal objects and denote it 3-ON task. We also show results of 2-ON and 1-ON in the supplementary material. We perform experiments on two photorealistic 3D indoor environments, *i.e.,* Matterport3D [7] and Gibson [65].

We follow the existing work [62] to evaluate the navigation success rate using **Success** and **Progress** metrics. Success indicates the percentage of successfully finished episodes, while Progress indicates the percentage of objects being successfully found. We also evaluate the navigation efficiency using **SPL** and **PPL**, which are short for Success weighted by Path Length and Progress weighted by Path Length, respectively. The weight is proportional to the navigation efficiency and is defined as $d/\bar{d}$, where d is the length of the ground-truth path and $\bar{d}$ is the path length traveled by an agent.

Implementation details. For navigation and camera actions, a FORWARD action moves the agent forward by 0.25 meters and a TURN action turns by 30°. The maximum episode time step is 500 because an agent with a global ground-truth map (*i.e.,* oracle agent) finishes more than 97% episodes within 500 steps. Our camera policy tries to narrow the performance gap between such an oracle agent and the agent with a progressively built map. We set $K = 8$, $r = 2.4$, $\gamma = r \times 1.2 = 2.88$ in heuristic module empirically. We use a mapper that outputs the ground-truth of occupancy anticipation [52], because how to train a good mapper is orthogonal to our work. Reward scaling factors α and β are set to 10 and 1 respectively such that three reward terms are in the same order of magnitude at initialization. We evaluate models for five runs using a same set of random seeds and report the mean results and standard deviation. More details are shown in the supplementary material.

4.2 Baselines

Mapping + FBE [66]: This modular SLAM-based navigation method breaks the problem into mapping and path planning. We use depth projection [20] for mapping and use the frontier-boundary-exploration (FBE) method [66] to select an exploration point. Once the built map covers goal objects, the agent navigates to them by taking deterministic actions [8] along the path planned by A* algorithm [28].

OccAnt [52]: This baseline is the same as the previous one except that we replace the depth projection with an occupancy anticipation neural network [52]. Such a network can infer the occupancy state beyond the visible regions. We assume the neural network is well-trained so that we use the ground-truth occupancy state within the field of view for experiments.

DD-PPO [64]: This end-to-end learning-based baseline performs navigation using RL algorithm. It consists of a recurrent policy network, which takes as input RGB-D images, goal object categories and previous actions for predicting navigation actions.

Table 1: Multi-object navigation results (%) for incorporating camera policy into different baselines on Matterport3D and Gibson datasets.

	MatterPort3D				Gibson			
Method	SPL	PPL	Success	Progress	SPL	PPL	Success	Progress
OccAnt	53.0$\pm$0.0	57.7$\pm$0.1	72.0$\pm$0.1	80.2 $\pm$0.1	76.1$\pm$0.1	77.8$\pm$0.1	89.0$\pm$0.1	91.8$\pm$0.1
+ Naive Camera Action	48.7$\pm$0.5	53.1$\pm$0.4	69.1$\pm$0.6	76.8$\pm$0.2	69.9$\pm$0.6	71.9$\pm$0.4	84.8$\pm$1.1	88.3$\pm$0.8
+ Our Camera Policy	**57.9**$\pm$0.5	**62.1**$\pm$0.2	**75.6**$\pm$0.5	**82.6**$\pm$0.2	**78.9**$\pm$0.5	**80.7**$\pm$0.5	**89.9**$\pm$0.5	**92.4**$\pm$0.5
Mapping + FBE	40.1$\pm$0.0	45.4$\pm$0.1	62.3$\pm$0.0	72.5$\pm$0.2	62.1$\pm$0.3	64.9$\pm$0.2	80.9$\pm$0.4	86.2$\pm$0.4
+ Naive Camera Action	35.2$\pm$1.1	41.2$\pm$1.0	55.3$\pm$1.4	66.9$\pm$1.3	55.8$\pm$0.5	58.5$\pm$0.4	77.5$\pm$0.6	83.1$\pm$0.3
+ Our Camera Policy	**44.6**$\pm$0.3	**49.8**$\pm$0.1	**64.2**$\pm$0.9	**74.1**$\pm$0.3	**68.7**$\pm$0.1	**71.2**$\pm$0.1	**84.4**$\pm$0.1	**88.9**$\pm$0.4
MultiON	33.0$\pm$0.5	43.8$\pm$0.6	44.1$\pm$0.6	60.5$\pm$0.5	56.5$\pm$0.3	62.2$\pm$0.2	68.4$\pm$0.1	77.3$\pm$0.0
+ Naive Camera Action	32.4$\pm$0.1	45.1$\pm$0.2	41.9$\pm$0.1	60.2$\pm$0.8	54.1$\pm$0.3	61.6$\pm$0.0	64.5$\pm$0.5	75.2$\pm$0.2
+ Our Camera Policy	**38.7**$\pm$0.7	**49.5**$\pm$0.9	**51.1**$\pm$0.0	**67.3**$\pm$0.3	**59.6**$\pm$0.3	**66.8**$\pm$0.2	**69.1**$\pm$0.3	**79.0**$\pm$0.1
DD-PPO	16.7$\pm$0.2	29.2$\pm$0.2	22.2$\pm$0.1	40.9$\pm$0.3	30.1$\pm$0.4	41.2$\pm$0.4	39.4$\pm$0.7	54.8$\pm$0.5
+ Naive Camera Action	16.7$\pm$0.1	30.4$\pm$0.1	20.8$\pm$0.3	39.2$\pm$0.4	31.3$\pm$0.3	43.4$\pm$0.2	38.8$\pm$0.6	54.5$\pm$0.5
+ Our Camera Policy	**19.1**$\pm$0.4	**34.0**$\pm$0.4	**24.0**$\pm$0.2	**43.9**$\pm$0.3	**33.9**$\pm$0.3	**45.3**$\pm$0.3	**40.5**$\pm$0.3	**55.3**$\pm$0.3

MultiON [62]: This is the variant of DD-PPO, with a progressively built object map as an extra input. Each cell of the object map is a one-hot vector indicating the existence of the goal objects. We store an object on the map once it is within the field of view of agents. We encode the egocentric cropped object map and feed it to the policy network. This baseline is the same as *OracleEgoMap* variant presented by [62].

4.3 Multi-object Navigation Results

Results on Matterport3D dataset. In Table 1, the agent with our EXPO camera policy performs better on multi-object navigation task upon four baselines. Specifically, our agent increases Success and Progress metrics for a large margin, indicating incorporating our camera policy helps agents successfully navigate to more goal objects. We attribute the improvement to a better exploration ability of our agent. With such ability, the agent finds more goal objects in a limited time step and then navigates to them. Besides, the improvement on SPL and PPL indicates our agent navigates to goal objects along a shorter path. The agent does not need to walk inside all rooms. Instead, the actively moving camera allows the agent to perceive what is inside a room when it passes by. The above results show our agent navigates to goal objects more efficiently in a higher success rate. We encourage readers to see the visualization in Figure 4 and watch supplementary videos.

The camera action provides more movement freedom for the agent. We are interested in the question that whether the improvement comes from simply extending the action space. To this end, we remove our camera policy. The agent determines camera actions naively. Specifically, for SLAM-based baselines, the agent chooses a camera action randomly. For learning-based baselines, the agent learns to predict both camera and navigation actions using the original navigation input and rewards. In Table 1, using these naive camera actions brings little improvement or even negative influence. This is not surprised because it is nontrivial to coordinate the camera and navigation actions. Also, a larger action space may increase the learning difficulty. These results further suggest the importance of the proposed camera policy for determining a reasonable camera action.

Transferability of camera policy to Gibson dataset. We evaluate the transferability of the learned camera policy on Gibson dataset. In Table 1 , a similar trend is observed on all baselines, *i.e.,* using naive camera actions does not help for navigation while our EXPO camera policy performs better than baselines. It is worth noting that the EXPO camera policy is trained on Matterport3D and has not been fine-tuned on Gibson dataset. There are significant differences between these two datasets in scene style and layout distribution [7, 65]. The consistent improvement demonstrates that our active-camera agent has learned general exploration and navigation skills for the multi-object navigation task. It also shows the possibility of transferring the agent to a real-world scene.

4.4 Further Analysis

Rule-based camera actions *vs.* learned camera policy. In contrast to learning a neural network, one may use handcrafted rules to decide camera actions. We compare the learned camera policy

Table 2: Ablation study on input information and reward types of camera policy based on OccAnt baseline.

Method	SPL	PPL	Success	Progress
w/o Map Input	55.9$_{\pm 0.7}$	60.1$_{\pm 0.6}$	73.2$_{\pm 0.7}$	80.3$_{\pm 0.5}$
w/o Heuristic Input	56.5$_{\pm 0.6}$	60.4$_{\pm 0.4}$	74.0$_{\pm 0.3}$	80.7$_{\pm 0.2}$
w/o NavAction Input	56.9$_{\pm 0.4}$	60.7$_{\pm 0.3}$	74.7$_{\pm 0.5}$	81.2 $_{\pm 0.3}$
w/o Heuristic Reward	53.8$_{\pm 0.5}$	58.1$_{\pm 0.3}$	72.7$_{\pm 0.7}$	80.3$_{\pm 0.4}$
w/o Area Reward	56.9$_{\pm 0.2}$	61.0$_{\pm 0.2}$	74.3$_{\pm 0.1}$	81.4$_{\pm 0.5}$
w/o Turn Reward	57.2$_{\pm 0.8}$	61.1$_{\pm 0.5}$	74.6$_{\pm 0.5}$	81.7$_{\pm 0.4}$
Ours	**57.9**$_{\pm 0.5}$	**62.1**$_{\pm 0.2}$	**75.6**$_{\pm 0.5}$	**82.6**$_{\pm 0.2}$

Table 3: Comparison between rule-based camera actions and learned camera policy based on two types of baselines.

Method	SPL	PPL	Success	Progress
OccAnt	52.9$_{\pm 0.0}$	57.7$_{\pm 0.1}$	72.0$_{\pm 0.1}$	80.2$_{\pm 0.1}$
+ Random Action	48.7$_{\pm 0.5}$	53.1$_{\pm 0.4}$	69.1$_{\pm 0.6}$	76.8$_{\pm 0.2}$
+ Swing Action	55.6$_{\pm 0.1}$	60.1$_{\pm 0.0}$	73.1$_{\pm 0.1}$	81.4$_{\pm 0.1}$
+ Heuristic Action	56.3$_{\pm 0.1}$	60.3$_{\pm 0.1}$	74.0$_{\pm 0.1}$	80.9$_{\pm 0.1}$
+ Learned Policy	**57.9**$_{\pm 0.5}$	**62.1**$_{\pm 0.2}$	**75.6**$_{\pm 0.5}$	**82.6**$_{\pm 0.2}$
MultiON	33.0$_{\pm 0.5}$	43.8$_{\pm 0.6}$	44.1$_{\pm 0.6}$	60.5$_{\pm 0.5}$
+ Random Action	3.1$_{\pm 0.6}$	10.2$_{\pm 0.5}$	4.8$_{\pm 0.9}$	19.8$_{\pm 0.1}$
+ Swing Action	13.0$_{\pm 0.8}$	25.4$_{\pm 0.7}$	19.7$_{\pm 1.3}$	41.1$_{\pm 1.4}$
+ Heuristic Action	19.9$_{\pm 0.7}$	32.2$_{\pm 0.4}$	27.6$_{\pm 0.8}$	46.3$_{\pm 0.4}$
+ Learned Policy	**38.7**$_{\pm 0.7}$	**49.5**$_{\pm 0.9}$	**51.1**$_{\pm 0.0}$	**67.3**$_{\pm 0.3}$

with three types of rule-based camera actions, *i.e.,* 1) selecting a random camera action; 2) forcing the agent to look forward and swing within 90° around the navigation direction; 3) following the heuristic direction inferred from the heuristic module.

The results upon a SLAM-based baseline, *i.e.,* OccAnt, are shown in Table 3. Exploiting random camera actions drops the performance because the agent often looks backward and captures redundant useless information. The other two types of rule-based camera actions improve the performance slightly. These camera actions help the agent build a map covering more areas. Consequently, the SLAM-based navigation policy can plan a better path for navigation using path planning algorithm. However, it is hard for us to design a robust rule covering all situations. For example, the swing camera actions may miss some explorable areas because the agent has passed by these areas before the camera swing to the direction pointing to them. Also, there exist false positive areas in the occupancy map (*e.g.,* free space behind a table is predicted as obstacles). These areas may mislead the heuristic module to consider an unexplored area as an explored one. Compared with these camera actions, our learned camera policy brings a more significant improvement. We attribute the improvement to the exploration reward in Equation 1. With such a reward, the agent is encouraged to take into account not only the handcraft rules but also the noisy occupancy map to predict a better camera action.

As for learning-based baseline, *i.e.,* MultiON, using these three types of rule-based camera actions significantly drops the performance. In these experiments, it is worth noting that we have fed previous camera action to inform the policy which direction the observation is taken from. We suspect the poor performance is because the camera movement decided by rules is uncontrollable by the navigation policy. As a result, the navigation policy can not get desired observations for predicting next navigation action. In contrast, the learned camera policy, which is trained together with the navigation policy, allows the agent to determine how to move its camera by itself. The above results further demonstrate the importance of the learned EXPO camera policy.

Ablation study on camera policy inputs and rewards. We conduct this ablation study by removing one of inputs and rewards of a camera policy upon OccAnt baseline on MatterPort3D dataset. In Table 2, removing any input or reward will drop the performance. We note that human knowledge (*i.e.,* heuristic input and reward) is important for the camera policy. Awarding the policy to follow this knowledge can be considered as a form of regularization and guidance for learning. The egocentric occupancy map input and area reward are also critical. They encourage the agent to explore more areas. With a better exploration ability, the agent can find goal objects and navigate to them more efficiently. The turn-penalty reward has little influence on the navigation performance. However, it helps to reduce the frequency of actuating camera motor described in Section 3.4. Experimental results show that without this reward, agents actuate the camera motor for 12.41% of total time steps. Adding turn-penalty reward decreases the number to 5.70%.

Does camera policy work with imperfect mapper? Our EXPO camera policy obtains indoor layout information mainly from the progressively built map. In this subsection, we would like to evaluate whether the proposed camera policy works with noisy map. To this end, we use a learned neural network [52] to predict the occupancy map from RGB-D images. The predicted map contains many false positive points (*e.g.,* predicting the free space as occupancy) and false negative points (*e.g.,* predicting a wall as free space). These noisy points may mislead the camera policy to make a wrong camera action. Experimental results on OccAnt baseline show that incorporating camera policy brings consistent improvement. Specifically, SPL and PPL increase from 13.6 to 18.1 and from 20.5 to 25.4, respectively. Success and Progress increase from 18.39 to 24.0 and from 28.4 to

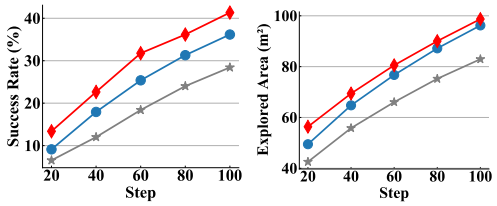

Figure 3: Exploration results. Left: success rate of finding three objects. Right: explored area within limited time step. Standard deviation are negligibly small. -●-: w/o camera action. -★-: Random camera action. -◆-: Our camera policy.

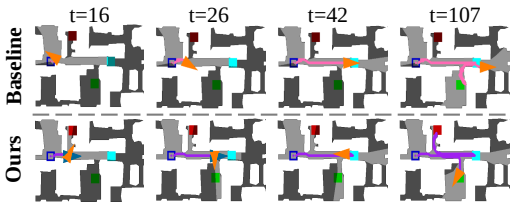

Figure 4: Visualization of multi-object navigation. The start position: □. Goal order: 1 ■, 2 ■, 3 ■. Camera direction: ▸. Navigation direction: ▸. When only ▸ is displayed, the camera and navigation directions are the same.

34.2, respectively. These results demonstrate that the proposed EXPO camera policy is robust to the map noise and can be deployed in a robot with no need of a ground-truth map.

Does improvement come from extra information? Compared with baselines, our active-camera agent leverages extra information (*i.e.,* occupancy map, heuristic direction and exploration reward) for determining camera actions. For a fair comparison, we add this extra information into MultiON to build an enhanced baseline. Results on MatterPort3D dataset demonstrate that the enhanced baseline performs slightly better than MultiON baseline but worse than our active-camera agent. Specifically, SPL, PPL, Success and Progress are 35.1, 45.9, 48.0, and 63.2, respectively. We suspect the improvement of the enhanced baseline comes from the fact that heuristic direction provides location information about unexplored areas. Also, the extra rewards encourage agents to explore these areas. However, due to the limited action space, the agent in enhanced baseline can not coordinate their camera and navigation actions well, which limits the performance.

Exploration performance. One of critical abilities for the multi-object navigation task is exploring the environment efficiently to locate all goal objects. To evaluate the exploration ability, we place the agent in a novel environment with three goal objects located in different places. We follow FBE method [66] to explore the environment. Given a limited time step budget, we evaluate success rate of finding all three goal objects and explored area in Figures 3. Compared with the baselines that the agent is always looking forward or moving its camera randomly, the agent with our EXPO camera policy finds more goal objects and explores more areas. These results have the same trend of navigation performance in Table 1, suggesting that the learned EXPO camera policy helps to explore the environment more efficiently and consequently boosts the multi-object navigation performance.

Visualization. In Figure 4, both baseline (*i.e.,* OccAnt) and our agent are navigating to *Goal-1* at the beginning. During this process, our agent moves its camera actively at time step $t = 16$ and $t = 26$, finding *Goal-2* and *Goal-3* respectively. Knowing the location of goal objects, our agent plans a shortest path for navigation. In contrast, the baseline agent goes straight to *Goal-1*, with the camera looking forward constantly. Consequently, after it navigates to *Goal-1*, it cannot find other goal objects and has to waste time exploring the environment again. Failure case analysis can be found in the supplementary material.

5 Discussion

Limitations and future work. Although our active camera agent shows promising performance in photo-realistic simulated data, it has not been thoroughly evaluated in real world, which is an interesting future work. Before through evaluation, directing adapting it to real environments may cause accidents such as breaking somethings and hitting pedestrian.

Conclusion. In order to solve the uncoordinated camera-navigation actions problem of existing agents, we propose a navigation paradigm in which agents can dynamically move their camera for perceiving environments more efficiently. Such exploration ability is important for multi-object navigation. To determine the camera actions, we learn a camera policy via reinforcement learning by awarding it to explore more areas. Also, we use heuristic rules to guide the learning process and reduce the learning difficulty. The proposed camera policy can be incorporated into most existing navigation methods. Experimental results show that our camera policy consistently improves the multi-object navigation performance of multiple existing methods on two benchmark datasets.

References

[1] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg. A dataset for developing and benchmarking active vision. In *ICRA*, pages 1378–1385, 2017.

[2] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. D. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, pages 3674–3683, 2018.

[3] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[4] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.

[5] E. Beeching, J. Dibangoye, O. Simonin, and C. Wolf. Learning to plan with uncertain topological maps. In *ECCV*, pages 473–490, 2020.

[6] Y. Burda, H. Edwards, D. Pathak, A. J. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

[7] A. X. Chang, A. Dai, T. A. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from RGB-D data in indoor environments. In *3DV*, pages 667–676, 2017.

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

[9] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural SLAM. In *ICLR*, 2020.

[10] D. S. Chaplot, H. Jiang, S. Gupta, and A. Gupta. Semantic curiosity for active visual learning. In *ECCV*, pages 309–326, 2020.

[11] D. S. Chaplot, D. Pathak, and J. Malik. Differentiable spatial planning using transformers. In *ICML*, pages 1484–1495, 2021.

[12] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta. Neural topological SLAM for visual navigation. In *CVPR*, pages 12872–12881, 2020.

[13] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar. Information-theoretic planning with trajectory optimization for dense 3d mapping. In *RSS*, 2015.

[14] B. Charrow, S. Liu, V. Kumar, and N. Michael. Information-theoretic mapping using cauchy-schwarz quadratic mutual information. In *ICRA*, pages 4791–4798, 2015.

[15] C. Chen, Z. Al-Halah, and K. Grauman. Semantic audio-visual navigation. In *CVPR*, pages 15516–15525, 2021.

[16] C. Chen, S. Majumder, Z. Al-Halah, R. Gao, S. K. Ramakrishnan, and K. Grauman. Audio-visual waypoints for navigation. *CoRR*, abs/2008.09622, 2020.

[17] K. Chen, J. K. Chen, J. Chuang, M. Vázquez, and S. Savarese. Topological planning with transformers for vision-and-language navigation. In *CVPR*, pages 11276–11286, 2021.

[18] K. Chen, J. P. de Vicente, G. Sepulveda, F. Xia, A. Soto, M. Vázquez, and S. Savarese. A behavioral approach to visual navigation with graph localization networks. In *RSS*, 2019.

[19] S. Chen, Y. Li, and N. M. Kwok. Active vision in robotic systems: A survey of recent developments. *International Journal of Robotics Research*, 30(11):1343–1377, 2011.

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

[21] V. Dean, S. Tulsiani, and A. Gupta. See, hear, explore: Curiosity via audio-visual association. In *NeurIPS*, 2020.

[22] Z. Deng, K. Narasimhan, and O. Russakovsky. Evolving graphical planner: Contextual global planning for vision-and-language navigation. *NeurIPS*, pages 20660–20672, 2020.

[23] C. Dornhege and A. Kleiner. A frontier-void-based approach for autonomous exploration in 3d. *Advanced Robotics*, 27(6):459–468, 2013.

[24] A. Elhafsi, B. Ivanovic, L. Janson, and M. Pavone. Map-predictive motion planning in unknown environments. In *ICRA*, pages 8552–8558, 2020.

[25] K. Fang, A. Toshev, L. Fei-Fei, and S. Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *CVPR*, pages 538–547, 2019.

[26] J. Fuentes-Pacheco, J. R. Ascencio, and J. M. Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81, 2015.

[27] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, pages 7272–7281, 2017.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[29] D. Holz, N. Basilico, F. Amigoni, and S. Behnke. Evaluating the efficiency of frontier-based exploration strategies. In *ISR*, pages 1–8, 2010.

[30] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel. VIME: variational information maximizing exploration. In *NeurIPS*, pages 1109–1117, 2016.

[31] M. G. Jadidi, J. V. Miró, and G. Dissanayake. Mutual information-based exploration on continuous occupancy maps. In *IROS*, pages 6086–6092, 2015.

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

[33] D. Jayaraman and K. Grauman. Look-ahead before you leap: End-to-end active recognition by forecasting the effect of motion. In *ECCV*, pages 489–505, 2016.

[34] D. Jayaraman and K. Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *CVPR*, pages 1238–1247, 2018.

[35] D. Jayaraman and K. Grauman. End-to-end policy learning for active visual categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1601–1614, 2019.

[36] B. J. Julian, S. Karaman, and D. Rus. On mutual information-based control of range sensing robots for mapping applications. *International Journal of Robotics Research*, 33(10):1375–1392, 2014.

[37] P. Karkus, S. Cai, and D. Hsu. Differentiable slam-net: Learning particle SLAM for visual navigation. In *CVPR*, pages 2815–2825, 2021.

[38] K. D. Katyal, A. Polevoy, J. L. Moore, C. Knuth, and K. M. Popek. High-speed robot navigation using predicted occupancy maps. In *ICRA*, pages 5476–5482, 2021.

[39] J. Kim, E. S. Lee, M. Lee, D. Zhang, and Y. M. Kim. Sgolam: Simultaneous goal localization and mapping for multi-object goal navigation. *CoRR*, abs/2110.07171, 2021.

[40] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, abs/1712.05474, 2017.

[41] E. S. Lee, J. Kim, and Y. M. Kim. Self-supervised domain adaptation for visual navigation with global map consistency. In *WACV*, 2022.

[42] Y. Liang, B. Chen, and S. Song. Sscnav: Confidence-aware semantic scene completion for visual semantic navigation. In *ICRA*, pages 13194–13200, 2021.

[43] Q. Luo, M. Sorokin, and S. Ha. A few shot adaptation of visual navigation skills to new observations using meta-learning. In *ICRA*, pages 13231–13237, 2021.

[44] P. Marza, L. Matignon, O. Simonin, and C. Wolf. Teaching agents how to map: Spatial reasoning for multi-object navigation. *CoRR*, abs/2107.06011, 2021.

[45] S. Mathe, A. Pirinen, and C. Sminchisescu. Reinforcement learning for visual object detection. In *CVPR*, pages 2894–2902, 2016.

[46] X. Meng, N. D. Ratliff, Y. Xiang, and D. Fox. Scaling local control to large-scale topological navigation. In *ICRA*, pages 672–678, 2020.

[47] A. Mousavian, A. Toshev, M. Fiser, J. Kosecká, A. Wahid, and J. Davidson. Visual representations for semantic target driven navigation. In *ICRA*, pages 8846–8852, 2019.

[48] T. Nagarajan and K. Grauman. Learning affordance landscapes for interaction exploration in 3d environments. In *NeurIPS*, 2020.

[49] D. Nilsson, A. Pirinen, E. Gärtner, and C. Sminchisescu. Embodied visual active learning for semantic segmentation. In *AAAI*, pages 2373–2383, 2021.

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

[51] D. Pathak, D. Gandhi, and A. Gupta. Self-supervised exploration via disagreement. In *ICML*, pages 5062–5071, 2019.

[52] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman. Occupancy anticipation for efficient exploration and navigation. In *ECCV*, pages 400–418, 2020.

[53] S. K. Ramakrishnan and K. Grauman. Sidekick policy learning for active visual exploration. In *ECCV*, pages 424–442, 2018.

[54] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018.

[55] M. Savva, J. Malik, D. Parikh, D. Batra, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, and V. Koltun. Habitat: A platform for embodied AI research. In *ICCV*, pages 9338–9346, 2019.

[56] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[57] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):1591–5, 1996.

[58] Z. Seymour, K. Thopalli, N. C. Mithun, H. Chiu, S. Samarasekera, and R. Kumar. Maast: Map attention with semantic transformers for efficient visual navigation. In *ICRA*, pages 13223–13230, 2021.

[59] W. B. Shen, D. Xu, Y. Zhu, F. Li, L. J. Guibas, and S. Savarese. Situational fusion of visual representation for visual navigation. In *ICCV*, pages 2881–2890, 2019.

[60] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. D. Nardi, M. Goesele, S. Lovegrove, and R. A. Newcombe. The replica dataset: A digital replica of indoor spaces. *CoRR*, abs/1906.05797, 2019.

[61] S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

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

[63] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi. Visual room rearrangement. In *CVPR*, pages 5922–5931, 2021.

[64] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. In *ICLR*, 2020.

[65] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, pages 9068–9079, 2018.

[66] B. Yamauchi. A frontier-based approach for autonomous exploration. In *CIRA*, pages 146–151, 1997.

[67] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi. Visual semantic navigation using scene priors. In *ICLR*, 2019.

[68] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will publish our code upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We have included the training details in the supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We have run evaluation using 5 seeds and reported error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We have included the details of the computational resources in the supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] We have added the license in the supplemental material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]