Learning Active Camera for Multi-Object Navigation

Peihao Chen[1,6] **Dongyu Ji**[1] **Kunyang Lin**[1,2] **Weiwen Hu**[1] **Wenbing Huang**[5]
Thomas H. Li[6] **Mingkui Tan**[1,7]* **Chuang Gan**[3,4]
[1]South China University of Technology,
[2]Pazhou Laboratory, [3]MIT-IBM Watson AI Lab, [4]UMass Amherst,
[5]Gaoling School of Artificial Intelligence, Renmin University of China,
[6]Information Technology R&D Innovation Center of Peking University,
[7]Key Laboratory of Big Data and Intelligent Robot, Ministry of Education,
phchencs@gmail.com, mingkuitan@scut.edu.cn

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 [72, 7, 67, 70, 32, 29], developments in high-quality 3D simulators [72, 62, 47, 2, 30], and research on deep memory-based architectures that combine geometry and semantics for learning representations of the 3D environment [8, 69, 59, 19, 31].

Despite these advances, how to efficiently perceive the environment and locate goal objects is still an unsolved problem. Agents in current research [9, 12, 50] 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

*Corresponding author.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

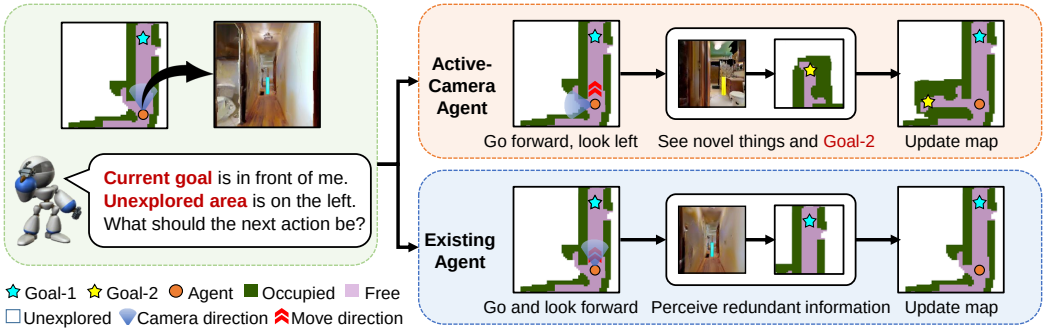

Figure 1: Illustration of an active-camera agent. The active-camera agent turns its camera to the left actively to look for novel things, *i.e., Goal-2*, when walking towards *Goal-1*.

result, these agents can only perceive the environment in front of themselves within or slightly beyond the view range [59]. 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 our heads 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.

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 [59, 69] and train them in an end-to-end manner for the multi-object navigation task. Extensive experiments on two benchmarks demonstrate the effectiveness of the proposed methods.

Our main contributions are threefold: 1) 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. 2) 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. 3) Extensive experiments demonstrate consistent improvement over four navigation methods on MatterPort3D [7] and Gibson [72] 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 tasks can be categorized into different classes such as PointGoal [33, 9] task where the goal is a given coordinate, ObjectGoal [33, 69] task where the agent needs to navigate to an object given by language and ImageGoal [12, 75] task where the object is specified by an image. The location of the goal is not explicitly known in the last two categories of the above tasks and the agent is expected to explore the environment to find the goals. Classical approaches [28, 68, 34, 64, 46] solve the navigation problem via path planning [11, 34] on a constructed map, which usually requires handcraft design. Recent works [75, 54, 74, 66, 69, 15, 65, 51, 57, 39] aim to use learning-based methods in an end-to-end manner to learn a policy for navigation. Other works [9, 12, 59, 44, 48, 5, 49, 18, 53, 17, 27, 61, 23] 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 [73, 35, 24], which chooses a frontier point between explored and unexplored areas as the exploration goal. Recent works tackle the exploration problem via learning [9, 21, 6, 25, 22, 58, 36, 45, 55], which allows the agent to incorporate knowledge from previously seen environments and generalize to novel environments. Specially, Chen *et al.* [21] and Jayaraman *et al.* [41] use end-to-end reinforcement learning policy to maximize an external reward (*i.e.,* exploration area). Burda *et al.* [6] and Dean *et al.* [22] consider intrinsic rewards such as curiosity for efficient exploration, which performs better when external rewards are sparse. Chaplot *et al.* [9], Ramakrishnan [59] 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.* [26] 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 [43, 13, 14, 38]. Recent work applies this strategy on different tasks such as object recognition [42, 1, 40], object localization [10, 75, 52, 56] and scene completion [41, 60]. We refer readers to [20] 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, the 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 [69] 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 the camera direction actively during the navigation process. Thus, agents can only perceive the field along their navigation direction, which causes low efficiency in finding goal objects [59].

To resolve the above problem, we propose a new navigation paradigm that consists of a navigation policy $\pi^n(\cdot)$ and a **camera policy** $\pi^c(\cdot)$. We reformulate both the navigation and camera moving process to a Partially Observable Markov Decision Process, where the observation o received by the agent does not fully specify the state of the environment. At each time step, for a given state estimated from o, 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 the 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 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.

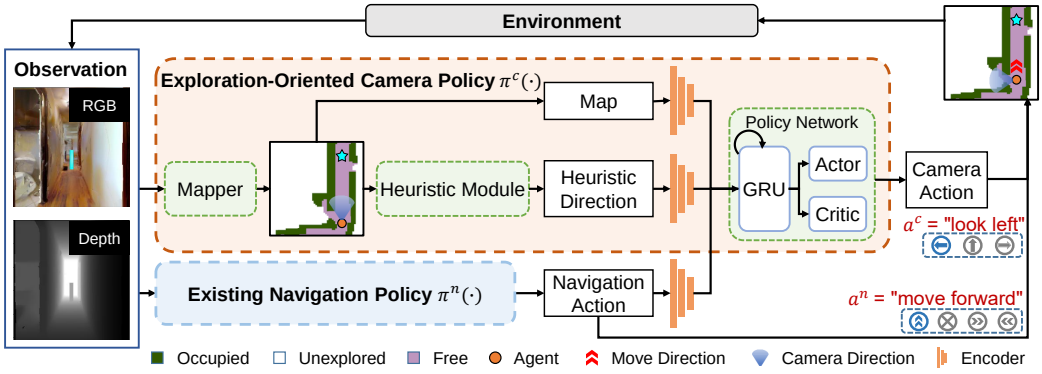

Figure 2: General scheme of an active-camera agent. The agent consists of two policies, namely the exploration-oriented camera policy determining where to look and existing navigation policy [59, 69] determining where to go.

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 camera policies must coordinate with navigation policies. In this paper, we design an EXPO camera policy that 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 the room layout and location of unexplored areas straightforwardly. The heuristic module infers a heuristic direction that is 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 [59] 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 $D(\cdot)$ [34]. 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 the 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 an 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 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 the value of the agent's current state.

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

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 the heuristic direction. This heuristic reward is defined as the reduction of s in successive time steps, *i.e.,* $r_{\text{heuristic}} = s_{t-1} - s_t$. The $s \in [0^\circ, 180^\circ]$ represents the angle between the camera direction and the heuristic direction. 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 follows:

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

where α and β are two scaling hyper-parameters. We train the camera policy to maximize such exploration reward using PPO [63]. 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 to 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 real-world coordinates. 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 modular SLAM-based [59, 73] and end-to-end learning-based [69, 71]. For the first type of navigation method, we use the existing navigation policy 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. In this case, it is worth mentioning that the camera policy coordinates with the navigation policy by conditioning the output of the navigation policy. The navigation policy decides navigation action by exploiting a progressively built map, which is built from historical camera policy output. For the end-to-end learning-based navigation method, the existing navigation policy typically contains the same recurrent network as our camera policy and uses it to predict navigation action. 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 in this case 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. The paradigm is shown in Figure A in Supplementary Materials.

4 Experiments

4.1 Experimental Setup

Task details. We follow MultiON [69] 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 Supplementary Materials. We perform experiments on two photorealistic 3D indoor environments, *i.e.,* Matterport3D [7] and Gibson [72].

We follow the existing work [69] 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% of 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 [59], 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 the same set of random seeds and report the mean results and standard deviation. More details are shown in Supplementary Materials.

4.2 Baselines

Mapping + FBE [73]: This SLAM-based navigation method breaks the problem into mapping and path planning. We use depth projection [21] for mapping and frontier-boundary-exploration (FBE) method [73] 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 [34].

OccAnt [59]: This baseline is the same as the previous one except that we replace the depth projection with an occupancy anticipation neural network [59]. 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.

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

DD-PPO [71]: 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 action.

MultiON [69]: 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 [69].

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 with 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 surprising 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, 72]. 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.

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**$_{\pm 0.3}$

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 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 a 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 an obstacle). 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 in 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 the 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 the 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 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 a noisy map. To this end, we use a learned neural network [59] 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.,*

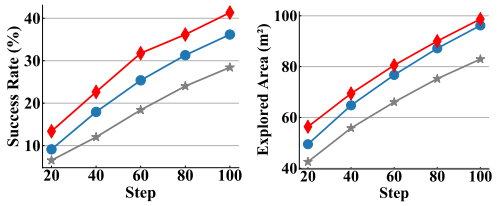

Figure 3: Exploration results. Left: success rate of finding three objects. Right: explored areas within limited time step. Standard deviation are negligibly small. -●-: w/o camera action. -⋆-: Random camera action. -◆-: Our camera policy.

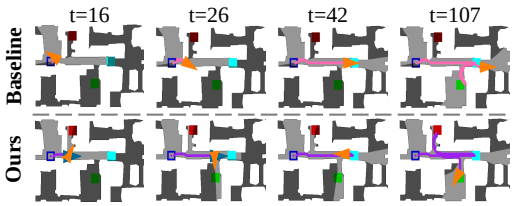

Figure 4: Visualization of multi-object navigation. The start position: □. Goal order: 1 ■, 2 ■, 3 ■. Camera direction: ▸. Navigation direction: ▸. When only ▸ is displayed, the camera and navigation directions are the same.

predicting a wall as free space). These noisy points may mislead the camera policy to make a wrong camera action. Experimental results on the 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 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 for 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 the 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 [73] to explore the environment. Given a limited time step budget, we evaluate the success rate of finding all three goal objects and explored areas 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 our 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 the 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 Supplementary Materials.

5 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 cameras 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.

Acknowledgments

This work was partially supported by National Key R&D Program of China (No. 2020AAA0106900), National Natural Science Foundation of China (No. 62072190 and No. 62006137), Program for Guangdong Introducing Innovative and Enterpreneurial Teams (No. 2017ZT07X183).

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. 3

[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. 1

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

[4] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018. 3

[5] E. Beeching, J. Dibangoye, O. Simonin, and C. Wolf. Learning to plan with uncertain topological maps. In *ECCV*, pages 473–490, 2020. 3

[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. 3

[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. 1, 2, 6, 7, 18

[8] D. S. Chaplot, D. Gandhi, A. Gupta, and R. R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *NeurIPS*, 2020. 1, 6

[9] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural SLAM. In *ICLR*, 2020. 1, 3, 18, 20

[10] D. S. Chaplot, H. Jiang, S. Gupta, and A. Gupta. Semantic curiosity for active visual learning. In *ECCV*, pages 309–326, 2020. 3

[11] D. S. Chaplot, D. Pathak, and J. Malik. Differentiable spatial planning using transformers. In *ICML*, pages 1484–1495, 2021. 3

[12] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta. Neural topological SLAM for visual navigation. In *CVPR*, pages 12872–12881, 2020. 1, 3

[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. 3

[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. 3

[15] C. Chen, Z. Al-Halah, and K. Grauman. Semantic audio-visual navigation. In *CVPR*, pages 15516–15525, 2021. 3

[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. 3

[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. 3

[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. 3

[19] P. Chen, D. Ji, K. Lin, R. Zeng, T. H. Li, M. Tan1, and C. Gan. Weakly-supervised multi-granularity map learning for vision-and-language navigation. In *NeurIPS*, 2022. 1

[20] 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. 3

[21] T. Chen, S. Gupta, and A. Gupta. Learning exploration policies for navigation. In *ICLR*, 2019. 3, 6

[22] V. Dean, S. Tulsiani, and A. Gupta. See, hear, explore: Curiosity via audio-visual association. In *NeurIPS*, 2020. 3

[23] Z. Deng, K. Narasimhan, and O. Russakovsky. Evolving graphical planner: Contextual global planning for vision-and-language navigation. *NeurIPS*, pages 20660–20672, 2020. 3

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

[25] Y. Du, C. Gan, and P. Isola. Curious representation learning for embodied intelligence. In *ICCV*, pages 10408–10417, 2021. 3

[26] A. Elhafsi, B. Ivanovic, L. Janson, and M. Pavone. Map-predictive motion planning in unknown environments. In *ICRA*, pages 8552–8558, 2020. 3

[27] 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. 3

[28] 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. 3

[29] C. Gan, Y. Gu, S. Zhou, J. Schwartz, S. Alter, J. Traer, D. Gutfreund, J. B. Tenenbaum, J. H. McDermott, and A. Torralba. Finding fallen objects via asynchronous audio-visual integration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10523–10533, 2022. 1

[30] C. Gan, J. Schwartz, S. Alter, M. Schrimpf, J. Traer, J. De Freitas, J. Kubilius, A. Bhandwaldar, N. Haber, M. Sano, et al. Threedworld: A platform for interactive multi-modal physical simulation. *NeurIPS*, 2021. 1

[31] C. Gan, Y. Zhang, J. Wu, B. Gong, and J. B. Tenenbaum. Look, listen, and act: Towards audio-visual embodied navigation. In *ICRA*, pages 9701–9707, 2020. 1

[32] C. Gan, S. Zhou, J. Schwartz, S. Alter, A. Bhandwaldar, D. Gutfreund, D. L. Yamins, J. J. DiCarlo, J. McDermott, A. Torralba, et al. The threedworld transport challenge: A visually guided task-and-motion planning benchmark towards physically realistic embodied ai. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8847–8854, 2022. 1

[33] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, pages 7272–7281, 2017. 3

[34] 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. 3, 4, 6

[35] D. Holz, N. Basilico, F. Amigoni, and S. Behnke. Evaluating the efficiency of frontier-based exploration strategies. In *ISR*, pages 1–8, 2010. 3

[36] 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. 3

[37] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, pages 2017–2025, 2015. 17

[38] M. G. Jadidi, J. V. Miró, and G. Dissanayake. Mutual information-based exploration on continuous occupancy maps. In *IROS*, pages 6086–6092, 2015. 3

[39] U. Jain, I. Liu, S. Lazebnik, A. Kembhavi, L. Weihs, and A. G. Schwing. Gridtopix: Training embodied agents with minimal supervision. In *ICCV*, 2021. 3

[40] 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. 3

[41] D. Jayaraman and K. Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *CVPR*, pages 1238–1247, 2018. 3

[42] 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. 3

[43] 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. 3

[44] P. Karkus, S. Cai, and D. Hsu. Differentiable slam-net: Learning particle SLAM for visual navigation. In *CVPR*, pages 2815–2825, 2021. 3

[45] 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. 3

[46] 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. 3

[47] 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. 1

[48] E. S. Lee, J. Kim, and Y. M. Kim. Self-supervised domain adaptation for visual navigation with global map consistency. In *WACV*, 2022. 3

[49] Y. Liang, B. Chen, and S. Song. Sscnav: Confidence-aware semantic scene completion for visual semantic navigation. In *ICRA*, pages 13194–13200, 2021. 3

[50] 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. 1

[51] 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. 3

[52] S. Mathe, A. Pirinen, and C. Sminchisescu. Reinforcement learning for visual object detection. In *CVPR*, pages 2894–2902, 2016. 3

[53] X. Meng, N. D. Ratliff, Y. Xiang, and D. Fox. Scaling local control to large-scale topological navigation. In *ICRA*, pages 672–678, 2020. 3

[54] 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. 3

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

[56] D. Nilsson, A. Pirinen, E. Gärtner, and C. Sminchisescu. Embodied visual active learning for semantic segmentation. In *AAAI*, pages 2373–2383, 2021. 3

[57] 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. 3

[58] D. Pathak, D. Gandhi, and A. Gupta. Self-supervised exploration via disagreement. In *ICML*, pages 5062–5071, 2019. 3

[59] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman. Occupancy anticipation for efficient exploration and navigation. In *ECCV*, pages 400–418, 2020. 1, 2, 3, 4, 5, 6, 8, 16

[60] S. K. Ramakrishnan and K. Grauman. Sidekick policy learning for active visual exploration. In *ECCV*, pages 424–442, 2018. 3

[61] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018. 3

[62] 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. 1, 18

[63] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. 5

[64] 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. 3

[65] 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. 3

[66] 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. 3

[67] 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. 1

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

[69] S. Wani, S. Patel, U. Jain, A. X. Chang, and M. Savva. Multion: Benchmarking semantic map memory using multi-object navigation. In *NeurIPS*, 2020. 1, 2, 3, 4, 5, 6, 7, 16, 17, 18

[70] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi. Visual room rearrangement. In *CVPR*, pages 5922–5931, 2021. 1

[71] 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. 5, 7, 16, 17

[72] 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. 1, 2, 6, 7, 18

[73] B. Yamauchi. A frontier-based approach for autonomous exploration. In *CIRA*, pages 146–151, 1997. 3, 5, 6, 9, 16

[74] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi. Visual semantic navigation using scene priors. In *ICLR*, 2019. 3

[75] 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. 3

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 J in the supplemental materials.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see Section J in the supplemental materials.
   (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 materials.
   (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 materials.
(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]

Supplementary Materials for "Learning Active Camera for Multi-Object Navigation"

In the supplementary, we provide more implementation details and more experimental results of our EXPO camera policy. We organize the supplementary as follows.

- In Section A, we provide the differences of active cameras from panoramic cameras.
- In Section B, we provide more discussion on how to coordinate two types of policies.
- In Section C, we provide more architecture details on our EXPO camera policy.
- In Section D, we provide more details on updating the top-down map.
- In Section E, we provide more experimental details, including training settings, environment details, and evaluation metrics.
- In Section F, we provide experimental results on multi-object navigation with different number of goal objects.
- In Section G, we provide experimental results to explain the choice of step budget for evaluation.
- In Section H, we provide more discussions on oracle information used in experiments.
- In Section I, we provide analysis on failure cases.
- In Section J, we discuss potential future researches and social impact.

A Differences of active cameras from panoramic cameras

In this paper, we propose an active camera to coordinate camera and navigation actions for perceiving a novel environment more efficiently. An alternative is to use panoramic cameras for capturing panoramic images. However, panoramic images are often in a very large size and contain a lot of redundant information, requiring a huge amount of resources to analyze. It may not be practical to equip panoramic cameras on mobile platforms. In this case, designing an active camera would be a more practical way to capture necessary environmental information required for navigation at a much cheaper cost.

An agent with multiple fixed pinhole cameras could make for an interesting comparison with our active-camera agent. To compare our active-camera agent with the multi-fixed-camera agent, we equip an agent with four fixed pinhole cameras facing different directions (*i.e.,* front, left, right, and back). Specifically, we concatenate the features of these four images and feed them to an end-to-end navigation baseline (*i.e.,* MultiON). We train this baseline for 30 million frames, which is the same as our active-camera agent. In Table A, the multi-fixed-camera agent performs better than the single-fixed-camera agent but still worse than our single-active-camera agent. We speculate this is because the end-to-end learned neural network is hard to map this larger amount of information (four RGBD images) to correct actions. We note that the multi-fixed-camera results in Table A are different from the results reported in Openreview during the rebuttal phase. This is because we train the agent using different numbers of parallel threads. We found that using less threads for training significantly decreases performance. The possible reason is that the model may be unstable during training if capturing training data from only a few environments. In Table A, all variants use the same number of parallel threads for training (*i.e.,* 36).

Table A: Comparisons of different camera types based on an end-to-end agent (*i.e.,* MultiON).

	MatterPort3D			
Camera Types	SPL	PPL	Success	Progress
Single-Fixed-Camera (MultiON)	33.0	43.8	44.1	60.5
Multi-Fixed-Camera	37.75	48.3	49.1	65.6
Single-Active-Camera (Ours)	**38.7**	**49.5**	**51.1**	**67.3**

We also have conducted multi-fixed-camera experiments on the SLAM-based method. The information captured by multiple cameras helps agents build a map efficiently. This map provides more information for path-planning. The results are shown in Table B. The multi-fixed-camera agent

outperforms our active camera agent. Because the SLAM-based multi-fixed-camera agent does not need to learn a neural network to map RBGD observation to navigation action, it will not suffer from learning difficulty problems as in e2e-based agents. It is worth noting that although using a single camera, our active camera policy helps the agent achieve comparative performance compared with an agent with four fixed cameras.

Table B: Comparisons of different camera types based on slam-based agents.

	MatterPort3D			
Camera Types	SPL	PPL	Success	Progress
Single-Fixed-Camera (OccAnt)	53.0	57.7	72.0	80.2
Multi-Fixed-Camera	**68.3**	**72.2**	**79.4**	**85.3**
Single-Active-Camera (Ours)	57.9	62.1	75.6	82.6

B More discussion on the coordination between two types of policies

When incorporating camera policy with an end-to-end navigation policy, we use one policy network (*i.e.,* Navigation-Camera Joint Policy) to jointly predict both camera and navigation actions, as described in Section 3.4 in the main paper. The paradigm is shown in Figure A. We feed this policy network both camera policy inputs (*i.e.,* map and heuristic direction) and navigation policy inputs (*i.e.,* RGB-D images, target, and previous action). In this way, the policy network better coordinates two types of actions. We have tried a variant that uses two separate policy networks to predict camera and navigation actions, respectively, which is similar to the paradigm in Figure 2 in the main paper. In Table C, this separated variant performs worse than our joint policy and even worse than the baseline without an active camera. We suspect this is because the navigation policy can not control camera direction to perceive desired observations for deciding navigation action. As a result, an agent performs poorly in navigation. These observations are consistent with the experimental results in Table 3 in the main paper. In Table 3, the agent performs worse when using rule-based camera actions because these camera actions are uncontrollable by the navigation policy either.

Table C: Comparisons between separated and joint navigation-camera policies when incorporating a camera policy with an end-to-end navigation policy.

	MatterPort3D			
Method	SPL	PPL	Success	Progress
MultiON	33.0	43.8	44.1	60.5
+ Active Camera (Separated)	29.1	40.4	37.3	54.1
+ Active Camera (Joint)	**38.7**	**49.5**	**51.1**	**67.3**
DD-PPO	16.7	29.2	22.2	40.9
+ Active Camera (Separated)	12.2	27.1	13.6	32.7
+ Active Camera (Joint)	**19.1**	**34.0**	**24.0**	**43.9**

When incorporating a camera policy with a SLAM-based navigation policy, two types of policies decide actions separately as shown in Figure 2 in the paper. Although the agent does not decide two types of actions jointly, two types of policies cooperate with each other from the following two aspects. 1) The camera policy cooperates with navigation action by taking it as input when deciding camera action. 2) The navigation policy cooperates with camera action by an off-the-shelf path-planning algorithm (*e.g.,* if the agent finds a better path toward goal objects after executing a camera action, the navigation policy will change the navigation action accordingly).

C More architecture details on camera policy

The general scheme of our active-camera agents, *i.e.,* , incorporating a camera policy with a modular SLAM-based [59, 73] navigation policy or an end-to-end learning-based [69, 71] navigation policy, are shown in Figure 2 and Figure A, respectively. We feed map features, heuristic direction features and navigation features to a policy network to predict camera action. These features are extracted from three encoders, respectively. Next, we describe the architecture details of these encoders and the policy network.

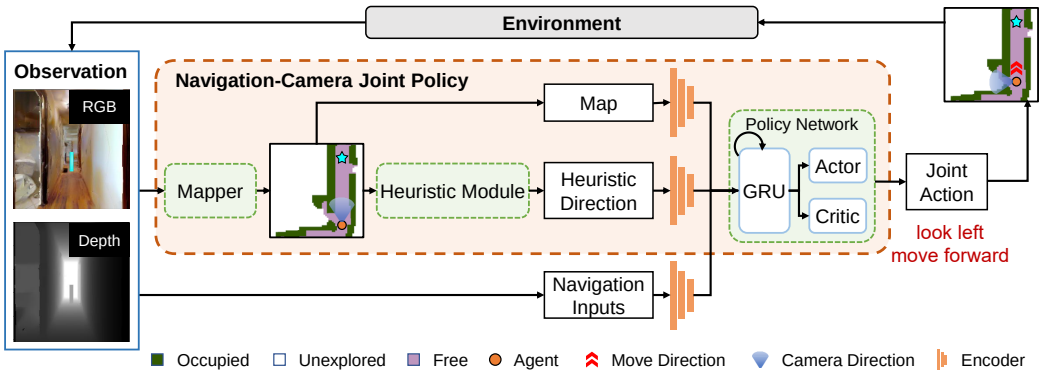

Figure A: General scheme of an active-camera agent when incorporating camera policy with an end-to-end navigation policy.

The map encoder consists of three convolution layers followed by a fully connected layer. We use ReLU as an activation function after each layer. The setting of three convolution layers (kernel size, stride) are $(4 \times 4, 2)$, $(3 \times 3, 2)$ and $(2 \times 2, 1)$, respectively. The fully connected layer outputs 512-dimension map features $\mathbf{v}_m$. The heuristic direction encoder is a learned embedding layer, transforming the input heuristic action into 32-dimension features $\mathbf{v}_h$. As for navigation encoder, its architecture depends on what kind of navigation policy we incorporate the camera policy with. If we use a SLAM-based navigation policy that output an navigation action, the navigation encoder is a learned embedding layer which transforms the action into 32-dimension features $\mathbf{v}_n$. If we use an end-to-end navigation policy, the encoder of this navigation policy [69, 71] would become our navigation encoder, transforming navigation input (typically RGB-D images) to latent features. With the features from these three encoders, we concatenate them to build the input $\mathbf{v} = [\mathbf{v}_m, \mathbf{v}_h, \mathbf{v}_n]$ for the policy network. The policy network consist of a one-layer gate recurrent unit (GRU), followed by two individual fully connected layers corresponding for predicting camera action and values of the agent's current state, respectively.

D Updating top-down map by a registration function $R(\cdot)$

As described in Section 3.2 in the paper, we use a registration function $R(\cdot)$ to update the global top-down occupancy map $\mathbf{M}^g$, *i.e.,* $\mathbf{M}^g = R(\mathbf{M}^l, \mathbf{M}^g, q)$, where $\mathbf{M}^l$ is a local map and q is the relative pose change of the agent from the first time step. After that, we crop an egocentric map from the global as camera policy input. We describe the details of this registration function below.

All these three types of maps are with the same resolution. Each pixel in a map represents $0.08 \times 0.08m^2$ area in the real world. The size of local map, global map and egocentric map are 61×61, 3000×3000, and 125×125, respectively. These maps represent $4.88 \times 4.88m^2$ area in front of the agent, $240 \times 240m^2$ area in the real-world, and $10 \times 10m^2$ area around the agent, respectively. The global map is with a larger size and covers the whole environment. In contrast, the local map represents only a small area, which is projected from the visual observation within a 79° field-of-view in front of an agent. To merge these two maps, we initialize a blank map with the same size as the global map. We put the local map in the center. Then, we use affine transformation [37] on this map based on the pose q. This affine transformation transforms the local to its location on the global map. After that, we aggregate the transformed map with the global map by a max-pooling operation.

E More experimental details

More implementation details. We use PyTorch to implement our active-camera agent. We train it using 36 parallel threads on 3 Nvidia Titan X GPUs. Each thread contains one of the scenes from the training set. The agent is trained using PPO with 4 mini-batches and 2 epochs in each PPO update. We use Adam optimizer with a learning rate of 2.5e-4, a discount factor of $\gamma = 0.99$, an entropy coefficient of 0.001, a value loss coefficient of 0.5. We train the proposed camera policy together with the SLAM-based and learning-based methods for 2 and 30 million frames, respectively.

More details on environments. We use Habitat simulator[2] [62] along with two large-scale photorealistic 3D indoor environments (*i.e.,* Matterport3D [7] and Gibson [72]). Both environments consist various of scenes such as house, church, and factory. For Matterport3D, we use the dataset proposed in MultiON [69] for the multi-object navigation task. In this dataset, 79 scenes are divided into two disjoint parts, 61 scenes for training and 18 scenes for testing. For Gibson, we evaluate the transferability of our camera policy. As the MultiON paper does not provide a corresponding multi-object dataset, we use the test scene split provided by Active Neural SLAM[9] and follow the episode generation rule in MultiON (*i.e.,* the geodesic distance between two successive goals is in the range of 2m and 20m) to generate 861 multi-object episodes within 14 different scenes for testing. For each episode, we ensure that all objects and start positions are on the same floor.

More details on metrics. We follow MultiON [69] to evaluate multi-object navigation in terms of success rate and navigation efficiency. A good agent should successfully navigate to all goal objects through the shortest path. The details are described below.

- **Success**: ratio of successfully navigating to all goal objects and calling FOUND in a correct order within an allowed time step budget.
- **Progress**: the fraction of goal objects being successfully found.
- **SPL**: Success weighted by Path Length. Concretely, $\mathrm{SPL} = s \times d/max(d, \bar{d})$, where s indicates the value of Success metric, d is the shortest geodesic distance from the starting point to all objects in order, $\bar{d}$ indicates the geodesic distance traveled by the agent.
- **PPL**: Progress weighted by Path Length. This is an extended version of SPL based on progress. Concretely, $\mathrm{PPL} = p \times d'/max(d', \bar{d}')$, where p indicates the value of Progress metric, d' and $\bar{d}'$ is similar to d and $\bar{d}$ in SPL metric but only account for the path through all founded objects.

F More results on 1-ON, 2-ON and 3-ON episodes

In order to evaluate the efficiency of our camera policy for the navigation tasks with different difficulties, except for 3-ON episodes (*i.e.,* navigating to 3 different objects in an environment), we follow MultiON [69] to evaluate on 2-ON and 1-ON episodes on both Matterport3D and Gibson datasets.

Performance on Matterport3D dataset. In Table D our camera policy consistently improves the navigation performance based on different baselines. These results suggest that actively moving camera following our camera policy helps agents explore the environment and locate objects more efficiently. These abilities benefit both the single object navigation task and the multi-object navigation task.

Table D: Object navigation results (%) on 1-ON, 2-ON and 3-ON episodes on Matterport3D dataset.

Method	SPL			PPL			Success			Progress		
	1-ON	2-ON	3-ON	1-ON	2-ON	3-ON	1-ON	2-ON	3-ON	1-ON	2-ON	3-ON
OccAnt	60.9	55.2	53.0	60.9	58.3	57.7	89.2	79.4	72.0	89.2	84.3	80.2
+Naive Camera Policy	58.6	52.2	48.7	58.6	55.4	53.1	85.7	75.5	69.1	85.7	80.6	76.8
+Our Camera Policy	**66.9**	**61.4**	**57.9**	**66.9**	**64.1**	**62.1**	**90.3**	**81.8**	**75.6**	**90.3**	**86.0**	**82.6**
Mapping+FBE	49.1	44.0	40.1	49.1	46.9	45.4	82.6	**72.6**	62.3	82.6	77.6	72.5
+Naive Camera Policy	45.5	38.7	35.2	45.5	42.5	41.2	79.3	65.4	55.3	79.3	72.3	66.9
+Our Camera Policy	**55.0**	**47.2**	**44.6**	**55.0**	**51.1**	**49.8**	**84.7**	71.8	**64.2**	**84.7**	**78.2**	**74.1**
MultiON	55.9	43.6	33.0	55.9	49.3	43.8	77.9	59.9	44.1	77.9	68.9	60.5
+Naive Camera Policy	59.6	44.6	32.4	59.6	52.0	45.1	80.2	59.2	41.9	80.2	69.9	60.2
+Our Camera Policy	**62.1**	**50.8**	**38.7**	**62.1**	**56.4**	**49.5**	**84.2**	**67.5**	**51.1**	**84.2**	**75.9**	**67.3**
DD-PPO	43.5	27.5	16.7	43.5	35.5	29.2	62.3	38.0	22.2	62.3	50.2	40.9
+Naive Camera Policy	47.5	27.7	16.7	47.5	37.7	30.4	61.1	35.2	20.8	61.1	48.4	39.2
+Ours Camera Policy	**51.2**	**31.8**	**19.1**	**51.2**	**41.2**	**34.0**	**66.3**	**41.5**	**24.0**	**66.3**	**53.8**	**43.9**

[2]https://aihabitat.org/

Transferability results to Gibson dataset. In Table E, we show the Transferability results to the Gibson dataset. Most of the results share the same trend as the results on the Matterport3D dataset, *i.e.,* our EXPO camera policy improves the object navigation performance in terms of Success, Progress, SPL, and PPL based on different baselines. These results demonstrate that our method can generalize well to different domains. This allows us to train the agent on the collected photorealistic datasets and then easily deploy it in the real-world environment.

Table E: Transferability results (%) to Gibson dataset on 1-ON, 2-ON and 3-ON episodes.

Method	SPL			PPL			Success			Progress		
	1-ON	2-ON	3-ON	1-ON	2-ON	3-ON	1-ON	2-ON	3-ON	1-ON	2-ON	3-ON
OccAnt	79.6	77.8	76.1	79.6	78.7	77.8	94.5	92.0	89.0	94.5	93.2	91.8
+Naive Camera Policy	73.9	72.0	69.9	73.9	72.9	71.9	91.7	88.4	84.8	91.7	90.0	88.3
+Ours Camera Policy	**82.4**	**80.8**	**78.9**	**82.4**	**81.6**	**80.7**	**94.7**	**92.7**	**89.9**	**94.7**	**93.7**	**92.4**
Mapping+FBE	67.7	65.0	62.1	67.7	66.3	64.9	91.5	86.3	80.9	91.5	88.9	86.2
+Naive Camera Policy	61.2	58.5	55.8	61.2	59.9	58.5	88.7	83.0	77.5	88.7	85.9	83.1
+Ours Camera Policy	**73.8**	**71.2**	**68.7**	**73.8**	**72.5**	**71.2**	**93.6**	**88.7**	**84.4**	**93.6**	**91.1**	**88.9**
MultiON	68.1	62.1	56.5	68.1	65.1	62.2	86.2	77.2	68.4	86.2	81.7	77.3
+Naive Camera Policy	69.2	61.2	54.1	69.2	65.2	61.6	86.0	74.5	64.5	86.0	80.0	75.2
+Ours Camera Policy	**73.1**	**66.7**	**59.6**	**73.1**	**69.9**	**66.8**	**88.9**	**79.3**	**69.1**	**88.9**	**84.1**	**79.0**
DD-PPO	53.6	39.8	30.1	53.6	46.8	41.2	**72.1**	52.6	39.4	**72.1**	62.6	54.8
+Naive Camera Policy	56.9	42.3	31.3	56.9	49.6	43.4	71.7	52.4	38.8	71.7	62.1	54.5
+Ours Camera Policy	**57.7**	**44.9**	**33.9**	**57.7**	**51.3**	**45.3**	71.9	**54.6**	**40.5**	71.9	**63.3**	**55.3**

G Selection of time step budget for evaluation

Exploring the environment and locating objects efficiently are important abilities for multi-object navigation. We propose EXPO camera policy to help agents coordinate their camera and navigation actions for exploring the environment more efficiently. To better evaluate the efficiency, we evaluate the navigation success rate given a limited time step budget. If given an infinite time step budget for an agent, it is trivial for the agent to transverse the entire environment and then go to goal objects.

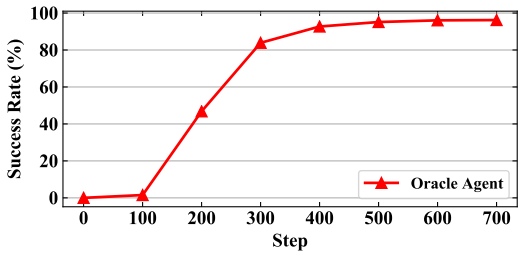

Figure B: Success rate of multi-object navigation task for an oracle agent. The success rate reaches 97% at 500 time steps and then remains the same.

To determine the value of time step budget, we consider an oracle agent with a ground-truth occupancy and object map. In other words, the oracle agent knows the position of objects and thus it does not need to explore the environment and find them. We evaluate how many time steps the oracle agent need for finishing the multi-object navigation task. In Figure B, the oracle agent successfully finishes more than 97% episodes using 500 time steps and the success rate remains the same when the time step is greater than 500. According to these results, we select 500 as the time step budget for evaluation because 500 time steps are sufficient for an oracle agent to navigate to all goal objects. We hope our EXPO camera policy helps the agents without map knowledge narrow the performance gap to the oracle agent.

H More discussions on oracle information used in experiments

In our experiments, to indicate whether an agent has reached goal objects, we use an oracle FOUND action (*i.e.,* automatically execute FOUND action once the agent reaches goal objects). We argue that this simplification is reasonable because the agent can easily judge whether a goal object appears near it with the help of a well-trained semantic segmentation model and a depth image. After simplifying this problem by using the oracle FOUND action, we are allowed to focus more on evaluating the exploration ability of active-camera agents.

I Failure cases analysis

We analyze failure cases where our active-camera agent takes more time for reaching goal objects compared with baselines. We observe that most of these cases are because our agent selects the wrong way for exploration. This is inevitable for all agents because goal objects are placed randomly in our experiments. We believe this problem can be solved if we have more information about the object's location (*e.g.,* finding a specific object like food, which is likely located in the kitchen). We can involve some constraints to guide our agent to explore relevant areas actively. We leave this in future works. Even so, as seen in Figure 3 in the paper, given the same time step, our agent explores more areas and finds all objects more quickly, which demonstrates its superior exploration ability. We strongly recommend readers watch the supplementary video for more examples.

J Potential future researches and social impact

In this paper, we present an active-camera agent. This agent can explore the environment and finish multi-object navigation task more efficiently by dynamically moving RGB-D camera sensors. In the real world, we may equip robots with different sensors, such as microphones and cameras. How to coordinate the movement of these sensors is worth exploring. Besides, we have assumed perfect camera pose localization in our experiments. However, there exist actuation and sensor noise in the real world. Leveraging a neural network [9] to solve this problem is an interesting research direction. Besides, even though we have evaluated the transferability of camera policy among different datasets (transferring from MatterPort3D to Gibson) in our paper, evaluating the robustness of the camera policy among different robot types and different embodied tasks is also a valuable research direction.

In the future, these agents may be able to help people deliver parcels and even take care of patients. On the other hand, an imperfect robotic agent may cause accidents such as breaking some things and hitting pedestrians. A possible way to avoid these accidents is to first develop such a robotic agent in a simulator environment and then deploy it into a controlled real environment for evaluation.