Learning Exploration Policies with View-based Intrinsic Rewards

Yijie Guo[*†] **Yao Fu**[*†] **Run Peng**[†] **Honglak Lee**[†‡]
[†]University of Michigan {guoyijie, violetfy, roihn, honglak}@umich.edu
[‡]LG AI Research honglak@lgresearch.ai

Abstract

Efficient exploration in sparse-reward tasks is one of the biggest challenges in deep reinforcement learning. Common approaches introduce intrinsic rewards to motivate exploration. For example, visitation count and prediction-based curiosity utilize some measures of novelty to drive the agent to visit novel states in the environment. However, in partially-observable environments, these methods can easily be misled by relatively "novel" or noisy observations and get stuck around them. Motivated by humans' exploration behavior of seeing around the environment to get information and avoid unnecessary actions, we consider enlarging the agent's view area for efficient knowledge acquisition of the environment. In this work, we propose a novel intrinsic reward combining two components: the view-based bonus for ample view coverage and the classical count-based bonus for novel observation discovery. The resulting method, ViewX, achieves state-of-the-art performance on the 12 most challenging procedurally-generated tasks on MiniGrid. Additionally, ViewX efficiently learns an exploration policy in the task-agnostic setting, which generalizes well to unseen environments. When exploring new environments on MiniGrid and Habitat, our learned policy significantly outperforms the baselines in terms of scene coverage and extrinsic reward.

1 Introduction

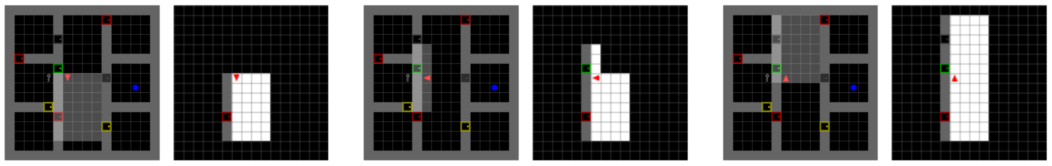
Efficient exploration is an important long-standing problem in reinforcement learning (RL). In real-world problems with unknown, vague, or multiple conflicting objectives, defining a proper reward function can be highly non-trivial. In challenging scenarios without manually-designed dense rewards, model-free RL algorithms often require numerous samples to learn a reasonable policy from the sparse environment rewards [Mnih et al., 2013, 2016, Schulman et al., 2015, 2017].

A common traditional way of solving hard-exploration tasks is to introduce intrinsic rewards that motivate the agents to explore [Schmidhuber, 1991, 2010, Oudeyer and Kaplan, 2009]. For example, count-based approaches [Strehl and Littman, 2008, Kolter and Ng, 2009, Tang et al., 2017] record all the states that the agent has visited and give high intrinsic rewards for the novel or rarely-visited states, encouraging the agent to find new states in the environment. Curiosity-driven methods [Stadie et al., 2015, Pathak et al., 2017, Burda et al., 2018b] learn dynamics models and use prediction errors in dynamics to measure the novelty of states.

While the approaches in these works encourage exploration with bonuses for novel states, they may easily fall short in many scenarios. For instance, the randomly-generated noisy observations in noisy-TV problem [Burda et al., 2018a] may trap these agents seeking novel observations. The issue may be worse in procedurally-generated environments [Cobbe et al., 2019, Chevalier-Boisvert et al., 2018], where novel states are always generated in each new episode.

Our work is in part inspired by the research about humans' visual space perception for knowledge acquisition and action guidance [Rieser et al., 1990, Loomis et al., 1992]. Unlike prior exploration

[*]Equal contribution.



(a) Step 0 (b) Step 1 (c) Step 2

Figure 1: In MiniGrid, top-down view observations o_t (left) and our constructed maps m_t (right) in consecutive steps $t = 0, 1, 2$. As is highlighted in o_t, the agent has a partially-observable view of 7x7 units but cannot see through walls or closed doors. Other grid cells are invisible to the agent.

methods, we are motivated to maximize the agent's knowledge coverage via visual perception, instead of the agent's visitation coverage of the environment. We focus more on where the agents "see" rather than where the agents "go" in the environment, which is more natural based on humans' exploration behavior. When humans explore unknown environments, they tend to broaden their knowledge of the region, i.e. "see" the most of the area. Only with the most information could people avoid unnecessary actions. Imagine that a goal object is randomly placed in one of several empty rooms, the most efficient way to find the goal is to glance around all rooms but not go into any of them until spotting the goal in one room. In order to maximize view coverage during exploration, some object interactions can be impactful because they instantly or later on help the agent reach new regions, such as opening doors and picking up keys. We hope to intrinsically motivate the agent to prioritize such object interactions and maximize its accumulated view of the space without wasting time with unimportant objects. Therefore, we propose View-based eXploration - ViewX, which assigns a higher intrinsic reward to newly discovered space than known space in the environment.

Our main contributions are summarized as follows: (1) We introduce a novel intrinsic reward for sparse-reward exploration tasks. In procedurally-generated Gym Minigrid environments, it dramatically improves the training sample efficiency in comparison with prior exploration methods. (2) We investigate our intrinsic reward in the task-agnostic exploration setting and demonstrate its exploration generalization performance. Our approach shows significant improvement over the baselines on both 2D MiniGrid and realistic 3D simulated Habitat environments. (3) We analyze the design of our intrinsic reward and compare it against several variations to verify its efficiency.

2 View-based Exploration

We consider reinforcement learning problems with partially observable Markov Decision Process (POMDP) $M = (\mathcal{S}, \mathcal{A}, \Omega, T, O, R)$, where $\mathcal{S}$, $\mathcal{A}$, Ω denotes the state space, action space and observation space. $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state transition function. $O : \mathcal{S} \times \mathcal{A} \times \Omega \rightarrow [0, 1]$ is the observation function. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function. At each step t in an episode, the agent observes $o_t \in \Omega$ in underlying state $s_t \in \mathcal{S}$, takes action $a_t \in \mathcal{A}$, and transits to the next underlying state $s_{t+1} \in \mathcal{S}$. It receives a new observation $o_{t+1} \in \Omega$ and an extrinsic reward r_t^{ext}.

2.1 View-based Intrinsic Reward

We assume a top-down view observation o'_t of the environment is included or can be learned/derived from the observation o_t. For example, in MiniGrid environments, the observation o_t is exactly o'_t, a part of the ground-truth top-down view map of the environment visible in the agent's front view. In 3D environments, the top-down view observation o'_t can be learned from the first-person view observation o_t [Gupta et al., 2017, Narasimhan et al., 2020, cha].

As is visualized in Figure 1, we gradually accumulate the top-down view observations o'_t in the past steps to construct a bird's view map m_t of the environment. The agent's relative position and orientation in any two consecutive steps are supposed to be available. Then we can append the transformation of o'_t to m_{t-1} based on the agent's relative position and orientation to construct the map m_t. The map m_t is reset as empty at the beginning of each episode, so m_t only covers regions seen by the agent in the current episode. x_t denotes the area of map m_t, which indicates how much space the agent has seen up to the episode step t. Because the map m_t is accumulated step by step, we have $x_t \geq x_{t-1}, \forall t = 1, 2, \ldots, T$.

We aim to encourage the agent to expand the frontier of its view and discover novel states for exploration. Thus, we propose a novel intrinsic reward, combining the view-based bonus and the count-based reward. Formally, we define the intrinsic reward as:

$$r_t^{int} = \frac{x_{t+1} - x_t + \lambda}{\sqrt{N(o_{t+1})}} \tag{1}$$

where N denotes the (pseudo)count of o_{t+1} throughout the training. The view-based bonus $(x_{t+1} - x_t)/\sqrt{N(o_{t+1})}$ encourages the agent to find unexplored regions. The count-based reward $1/\sqrt{N(o_{t+1})}$ drives the agent to visit globally less frequently visited states. The hyper-parameter $\lambda > 0$ controls the relative importance of count-based bonus given the view-based bonus.

2.2 Conceptual Advantages of ViewX

COUNT An illustrative example is shown in Figure 2. We have an environment with the same structure for every episode except for a different novel state at the 6^{th} position. The agent is placed in the middle of the environment and can see up to 3 grids ahead of it. Its task is to reach the goal hidden behind a closed door, while the closed door locks its visibility. It can choose to stay, go left, or go right. In episode n, COUNT assigns a high intrinsic reward for the novel state, making the agent go to the novel state and stay there for several steps until the COUNT reward goes down. However, an agent trained with ViewX may turn right but not go forward to the novel state because it already knows the three units in the right end. Instead, the ViewX agent prefers turning back to the door to discover more space.

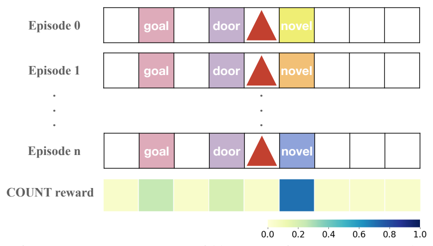


Figure 2: An illustrative example of procedurally-generated environment. The agent starts in the middle of a 1D environment and can either go left or right. The goal is hidden behind a closed door. The "novel" state emits different observations in each episode. An agent trained with COUNT prioritizes going right and gets stuck at the "novel" cell for some steps because observations there yield high COUNT rewards.

Coverage Maximization Some recent works [Chaplot et al., 2020a, Chen et al., 2019] directly use coverage increase $x_{t+1} - x_t$ as an intrinsic reward for policy learning or as a measurement of performance, where x_t is the absolute or relative area in the space known to be traversable or non-traversable for the agent. This view coverage maximization method seeks information gain for exploration in spatial environments [Mirchev et al., 2018, Charrow et al., 2015] (In Appendix E, we derive the relation in detail.) With such an intrinsic reward function, the agent aims to see the most of the environment. However, using this reward alone has some potential disadvantages. First, the view coverage maximization reward is sparse because it is non-zero only when the agent discovers some new space for the first time[Ramakrishnan et al., 2020]. Imagine a large room without any obstacles. An agent with a large field of view can simply rotate around to see the whole space and get high intrinsic rewards. However, it gets no more reward for any actions in the room after the rotation. This will lead to random exploration in this large room, which is inefficient for the agent to discover the door and see more space. Second, unlike the COUNT reward, view coverage maximization reward does not asymptotically approach 0 during training, because the map m_t is reset as empty in each episode to deal with the procedurally-generated environments. Therefore, it is more likely to introduce bias to the original RL objective. We experimentally show this in Section 4.4.

ViewX ViewX is designed to take advantage of both COUNT and view coverage maximization and mitigate their issues. We use view coverage change as a weight for COUNT. Therefore the agent is intrinsically rewarded the most when it finds some new regions for the first time. Besides, the intrinsic reward theoretically diminishes at the later stage of the training to trace back to the original objective of maximizing extrinsic rewards. Similar to the global score in RAPID[Zha et al., 2021], we also add a small constant $\lambda > 0$ to stimulate the visitation of globally novel states and prevent the intrinsic reward from being too sparse.

2.3 Exploration with Intrinsic Rewards

Many exploration methods in RL focus on the task-driven exploration setting, where an extrinsic reward function is well-defined and the agent's goal is to maximize accumulated extrinsic rewards. Intrinsic reward works as an bonus to facilitate reaching states with high extrinsic rewards. In this setup, the total reward received by the agent at step t is $r_t = r_t^{ext} + \alpha r_t^{int}$. In Section 4.2, we study the effectiveness of ViewX as an intrinsic bonus on sparse-reward tasks in procedurally-generated environments, where the agent only gets extrinsic rewards when it successfully reaches the goal.

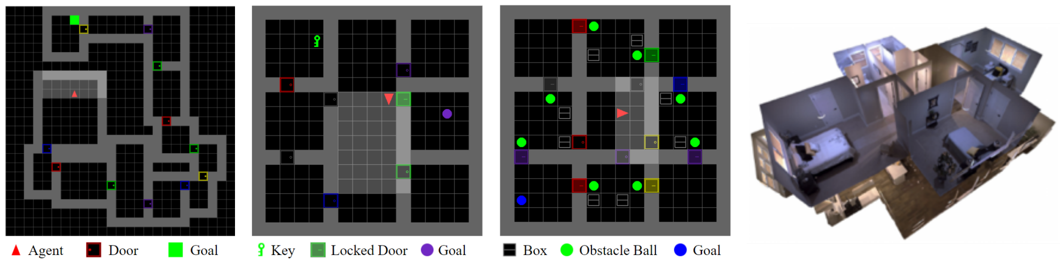
On the other hand, task-agnostic exploration considers policy learning without any external rewards. The goal is to learn an exploration policy useful for downstream tasks. A good exploration policy may be able to collect diverse and informative transition data for multiple downstream tasks [Zhang et al., 2020b], or be able to explore new environments efficiently and achieve high space coverage [Parisi et al., 2021, Chen et al., 2019]. In Section 4.3, we learn the exploration policy with only intrinsic reward r_t^{int} in a task-agnostic manner and further test the learned policy in unseen environments.

3 Related Work

Intrinsic Motivation Intrinsic motivation with bonus rewards is a natural way to encourage exploration. Prior works [Tang et al., 2017, Bellemare et al., 2016, Ostrovski et al., 2017, Burda et al., 2018b, Pathak et al., 2017, Burda et al., 2018a] use pseudo count or prediction error as intrinsic reward signals to incentivize visiting novel states. While these prior methods encourage the agent to observes novel observations, ours additionally drives the agent to seek larger view coverage of the environment for efficient exploration. On the other hand, some prior works aim to maximize information gain for exploration [Storck et al., 1995, Little and Sommer, 2013, Mobin et al., 2014, Shyam et al., 2019]. Information gain corresponds to knowledge increase acquired upon each step to update the agent's internal environment model. The view coverage maximization method in navigation tasks [Chen et al., 2019, cha, Chaplot et al., 2020b] fall into this category (See Appendix E for more explanation). Apart from calculating coverage, these works use accurate top-down view maps as inputs of the policy or planner. However, ViewX only needs the area of view coverage rather than the accurate map inference to guide exploration. Furthermore, coverage maximization reward does not asymptotically vanish to 0 in procedurally-generated environments. So the final policy does not necessarily maximize the environment reward. ViewX combines the advantages of count-based bonus and view coverage maximization bonus and alleviates the issues of both components.

Exploration in Procedurally-generated Environments Many existing methods in RL study the exploration in "singleton" environments [Ecoffet et al., 2019, Burda et al., 2018a, Choi et al., 2018, Guo et al., 2020, Badia et al., 2020], where the environment and tasks keep the same in every episode. Procedurally-generated environments [Cobbe et al., 2019, 2020, Justesen et al., 2018, Chevalier-Boisvert et al., 2018, Küttler et al., 2020] pose a new challenge for exploration. The environment structure changes in every episode and the agent is unlikely to ever visit the same state twice. Therefore, it requires an exploration policy generalizing well across a very large state space. In order to solve these exploration challenges, RIDE [Raileanu and Rocktäschel, 2020] proposes to reward the agent's actions that have an impact on the environment state representation; RAPID[Zha et al., 2021] employs imitation learning to encourage episode-level good exploration behavior with high visitation coverage; NovelD [Zhang et al., 2021b] applies regulated difference of novelty measurement using random network distillation. MADE [Zhang et al., 2021a] aims to maximize the deviation of the occupancy of the next policy from the explored region. ViewX shares the intuition of encouraging exploration in episode-level and push the exploration frontier in the environment. However, we emphasize that the exploration frontier for ViewX is where the agent sees or knows about the environment, rather than where the agent actually visits.

Task-agnostic Exploration Task-agnostic exploration (or pure exploration) probes the problem of learning exploration policies in the absence of any external rewards. The objective is to effectively accumulate information about the environment for data collection [Zhang et al., 2020b], state representation learning [Yarats et al., 2021], model learning [Shyam et al., 2019], or policy learning [Chen et al., 2019, Parisi et al., 2021]. The collected trajectories, learned representation, model or policy can be exploited to assist solving diverse downstream tasks. C-BET [Parisi et al., 2021] proposes to first learn exploration without any extrinsic reward and then transfer the learned exploration policy to different environments that share some structure similarities. In this work, we follow C-BET to evaluate the learned exploration policy via offline transfer/generalization to new environments. While C-BET tends to more actively interact with objects, ViewX is better at exploration in larger environments with more complicated layouts.



(a) MultiRoom-N12-S10 (b) KeyCorridorS6R3 (c) ObstructedMaze-Full (d) Apartment 0 in Replica

Figure 3: Example of the environments Minigrid and Habitat.

4 Experiments

In this section, we aim to answer the following research questions: (1) can ViewX outperform previous state-of-the-art exploration methods in terms of training-time sample efficiency? (2) can ViewX efficiently learn an exploration policy in the task-agnostic setting and generalize well to unseen test environments? (3) how does each component of our intrinsic reward (Equation 1) contribute to the performance? To answer these questions, we investigate our method on two benchmarks: 2D Minigrid environments[Chevalier-Boisvert et al., 2018] with procedurally-generated hard-exploration tasks and simulated 3D Habitat environments[Szot et al., 2021].

4.1 Minigrid Environments and Baselines

Figure 3 shows the rendering of a few example environments we use in the experiments. MiniGrid [Chevalier-Boisvert et al., 2018] is a set of procedurally-generated environments. In this work, we test our intrinsic reward design on MultiRooms (MR), KeyCorrdor (KC), Obstructed Maze (OM), and BlockedUnlockPickup (BUP) tasks. In each task, the agent is randomly placed in a 2D grid board and is expected to achieve a goal by exploring the environment and interacting with objects. At each step, the agent has a field of vision of 7x7 cells in the partially-observable environment and can conduct one of the seven actions: turn left, turn right, move forward, pick up, drop, toggle, and done.

Specifically, in Multiroom-Ny-Sz environments (see Figure 3a as an example), y represents the number of rooms and z represents the maximum size of each room. The agent needs to navigate rooms through the closed doors to find the goal in the last room. In KeyCorridor-SyRz environments (Figure 3b), y represents the room size and z represents the number of rows. The agent has to pick up an object goal which is placed behind a locked door. ObstructedMaze (Figure 3c) and BlockedUnlockPickup are similar to KeyCorridor but with more constraints. The agent has to reach a goal behind a door, but the doors are locked, keys are hidden in some boxes, and doors may be obstructed by balls.

For the task-driven exploration, we compare with previous state-of-the-art algorithms including MADE[Zhang et al., 2020b], NovelD[Zhang et al., 2021b], RIDE[Raileanu and Rocktäschel, 2020], and RND[Burda et al., 2018b]. In all experiments, we follow the same basic RL algorithm and network architecture as in RIDE and only change the intrinsic rewards. For task-agnostic setting, we compare against C-BET[Parisi et al., 2021], COUNT, RIDE[Raileanu and Rocktäschel, 2020], and RND[Burda et al., 2018b] which are the strongest methods for exploration generalization as reported in [Parisi et al., 2021]. We employ the same codebase as C-BET for a fair comparison of different designs of intrinsic rewards. Implementation details of our method and baselines are in Appendix A.

4.2 Task-driven Exploration

Figure 4 shows the learning curves of ViewX and state-of-the-art exploration baselines MADE, NovelD, RIDE, and RND in 12 challenging Minigrid environments, including MultiRooms (MR), KeyCorridor (KC), and ObstructedMaze (OM). In all of the environments, ViewX significantly outperforms previous methods in terms of sample efficiency. For instance, on MR-N12-S10, ViewX is about two times more sample efficient than MADE and NovelD. On the hardest environment OM-Full, our method learns to solve the task within 45M timesteps, which greatly improves the previous state-of-the-arts. We remark that a single $\lambda = 0.01$ works well across all the tasks.

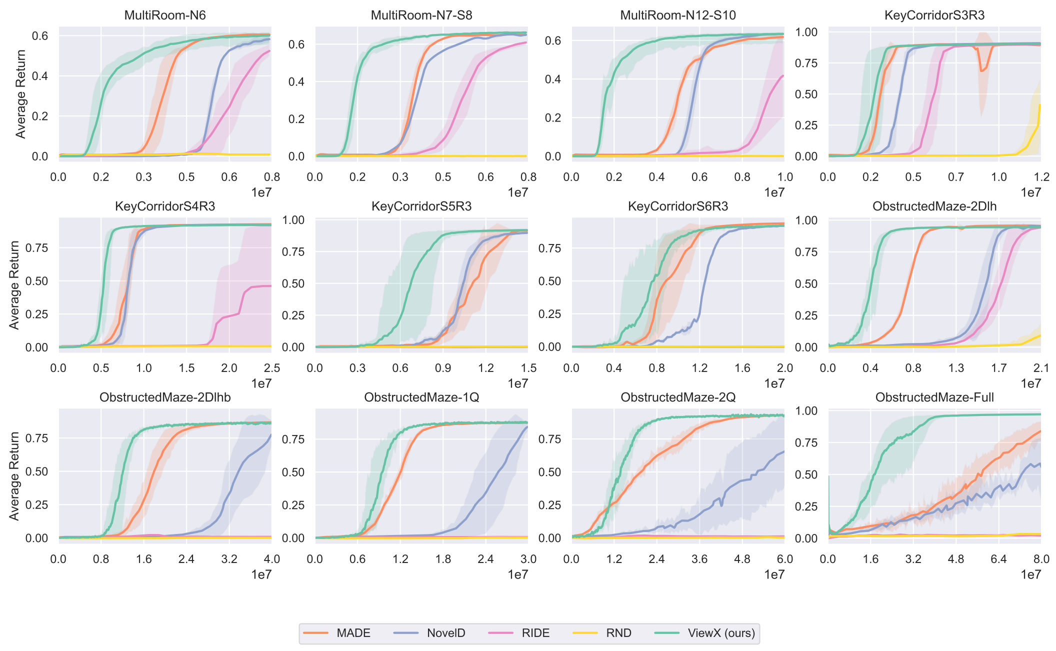


Figure 4: Task-driven training performance of ViewX and the baselines on 12 MiniGrid environments. The x-axis shows the number of training timesteps. The curves are averaged across 7 seeds.

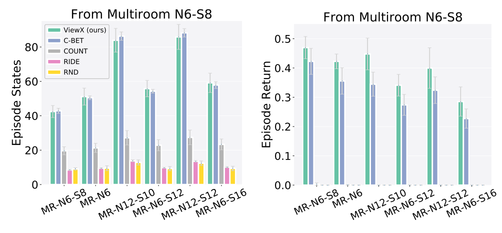
4.3 Task-agnostic Exploration

We further investigate our proposed intrinsic reward in the task-agnostic setting. To prevent the ViewX intrinsic reward from vanishing to 0, we randomly reset the observation count $N(o_t)$ with a probability 0.001 at any time step during the training (similarly to C-BET [Parisi et al., 2021]). We set $\lambda = 1$ without any hyper-parameter tuning for ViewX in the following experiments.

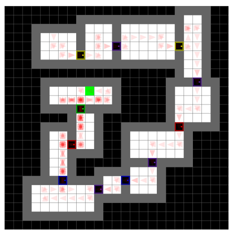
4.3.1 Minigrid Environments

The exploration policy is trained on one environment for 10M timesteps with only intrinsic rewards. Then we evaluate the learned exploration policies on the test environments that share some similarities with the training environment. We measure the quality of the exploration policy with two criteria: number of visited cells in the grid and episode return. We expect that good exploration policies can bring the agent to the goal more often by visiting more cells in the grid faster. Within the limit of training steps (10M), compared with other methods, ViewX learns a better exploration policy during training with higher sample efficiency and the learned policy better explores new environments.

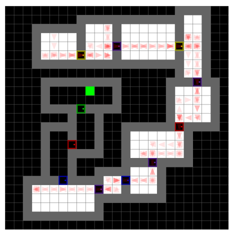
MultiRoom On Multiroom-Ny-Sz environments, increasing room number y and room size z makes the exploration more challenging. We examine the exploration policy on test environments (e.g. MR-N12-S10, MR-N6-S12, etc.) with higher difficulty levels than the training environment MR-N6-S8. In Figure 5a, ViewX visits more states and obtains higher episode return than baselines.



(a) Exporation generatlization to larger environments of MR. Average and standard deviation over 7 runs.



(b) Exploration behavior of ViewX on MR-N12-S10.

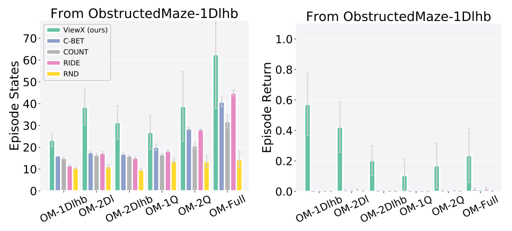


(c) Exploration behavior of C-BET on MR-N12-S10.

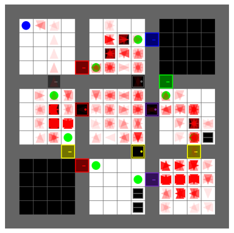
Figure 5: Exploration performance on MR. In (b)(c), small red triangles annotate the agent's positions and orientations and the darker red color means more visitations. White regions are seen by the agent.

Figure 5b, 5c visualize the agent's exploration behavior on test environment MR-N12-S10. While ViewX agent successfully opens all the doors along the path to the goal (green square), C-BET fails to get to the last three rooms within the limited time steps. C-BET agent learns to move around doors for many times because opening a door leads to environment panorama change. With the intrinsic reward $1/(N(change) + N(o))$, C-BET agent receives a high intrinsic reward when interacting with doors. The reward is still attractive (though gradually decreasing) when it interacts with the door for more times. On the contrary, ViewX agent opens each door just once, because only opening a door for the first time increases its view area. More interactions with an opened door means no view-based reward. Therefore, ViewX agent is more efficient navigating more rooms within limited time steps.

ObstructedMaze We train exploration policies in the obstructed maze with only two rooms and test their generalization ability on obstructed mazes with more rooms. Figure 6a shows that ViewX learns a reasonable policy on the training environment OM-1Dlhb with satisfactory episode return. In contrast, other methods fail to explore the training environment widely enough to receive any reward of discovering the goal. Note that here episode return only works as an evaluation metric, and the agent does not receive any extrinsic reward during training.



(a) Exploration generalization to larger environments of OM. Average and standard deviation over 7 runs.


(b) Exploration behavior of ViewX on OM-Full.

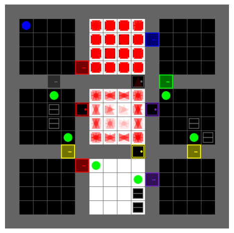
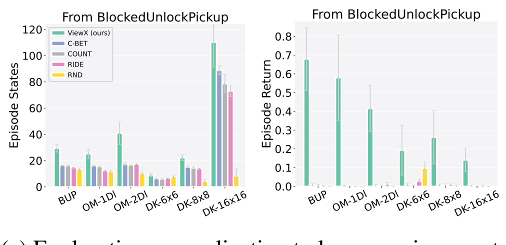

(c) Exploration behavior of C-BET on OM-Full.

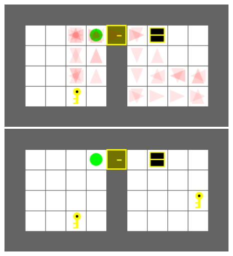
Figure 6: Performance of the exploration policies on training and test ObstructedMaze environments.

On the test environments with larger and more complicated maps, ViewX exploration policy demonstrates better generalization performance (Figure 6a). ViewX agent focuses on view coverage and is highly rewarded when it first opens a door to see a new space. C-BET agent is interested in the interactions with objects but fails to quickly complete the complicated process of getting hidden keys and removing obstacles to open a door. As illustrated in Figure 6b, 6c, ViewX agent visits more locations than C-BET in the test environment OM-Full. This better state coverage increases its possibility of finding the goal state.

BlockedUnlockPickup The training environment BlockedUnlockPickup (BUP) requires the agent to pick up a box which is placed in another room. We evaluate the generalization performance on other environments with larger room size and room number. In Figure 7a, ViewX outperforms the baselines in terms of state visitation and goal discovery.



(a) Exploration generalization to larger environments of ObstructedMaze (OM) and DoorKey (DK). Average and standard deviation over 5 runs.


(b) Exploration behavior of ViewX on BUP.

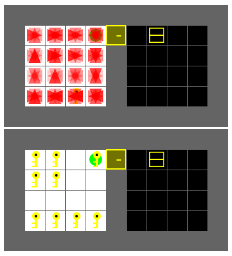

(c) Exploration behavior of C-BET on BUP.

Figure 7: Performance of the exploration policies on training and test environments.

To be concrete, the ViewX agent picks up the key, moves the obstacle ball and opens the door in BUP successfully, as shown by the agent's path in Figure 7b (top). Figure 7b (bottom) demonstrates that the ViewX agent never unnecessarily drops the key in the left room. However, C-BET agent pays

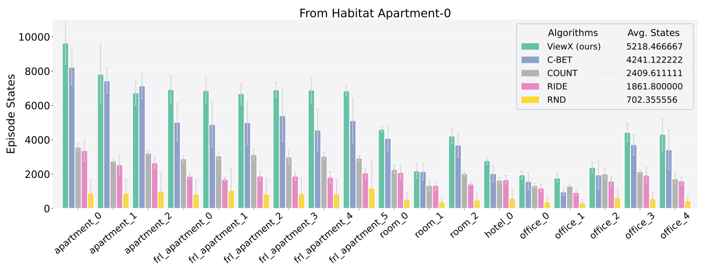


Figure 8: Exploration performance on scenes in Replica datasets.

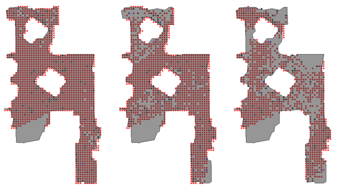


Figure 9: State visitation during exploration on FRL apartment 0, with ViewX (left), C-BET (middle), and COUNT (right) agents.

attention to interactions with objects, such as dropping and picking up the key in different locations of the environment. Figure 7c (bottom) shows where C-BET agent drops the key. These interactions cause novel panorama change to delight the agent with C-BET intrinsic reward, but actually distract the agent in the goal-reaching task.

4.3.2 Habitat Environments

Habitat simulator [Szot et al., 2021] provides a platform to run embodied AI tasks. Following the prior work [Parisi et al., 2021], we set up experiments with the Replica dataset [Straub et al., 2019], a set of indoor scenes with realistic visual observations. The agent partially observes the environment when navigating around, and its field of view is limited from first-person view point. Its visibility may be obstructed by walls, closed door, or corners of obstacles. We first train the exploration policy in one scene Apartment 0 with only intrinsic rewards for 2M steps. The start position is randomly sampled in the environment in each episode. We test the learned exploration policy on new scenes with layouts different from the training scene. On each scene, we report the total number of visited states in 100 test episodes. Figure 8 shows that ViewX agent visits more locations in comparison with the baselines. We also visualize the state coverage for each exploration policy in the test scene FRL apartment 0. In Figure 9, we annotate the visited states on ground truth environment map with red dots and qualitatively show the efficacy of our learned exploration policy.

4.4 Ablation Study

In this section, we analyze the importance of each component of our intrinsic reward by doing ablation tests. Specifically, in the task-driven exploration (Section 4.2) and task-agnostic exploration (Section 4.3) settings, we run experiments using the following intrinsic reward functions.

- ViewX: $(x_{t+1} - x_t + \lambda)/\sqrt{N(o_{t+1})}$
- R1: view coverage maximization reward $x_{t+1} - x_t$
- R2: without the constant term $(x_{t+1} - x_t)/\sqrt{N(o_{t+1})}$
- R3: indicator of whether view expands $(\mathbb{1}\{x_{t+1} - x_t > 0\} + \lambda)/\sqrt{N(o_{t+1})}$

The results of task-driven training performance are summarized in Appendix B.1. We make the following observations: (1) R1 results in slightly sub-optimal policies, especially on harder environments like OM. (2) R2 contributes the most to the success of ViewX. Comparing ViewX and R2, the constant term λ in ViewX slightly helps its performance on all the tasks. (3) R3 still works on easy environment like MR, but totally fails on harder ones.

The reason behinds observation (1) is that the view coverage maximization reward $r_t^{int} = x_{t+1} - x_t$ lies in a range between 0 and the agent's largest view area. This reward never goes to 0 for all new states. Given the reward function $r_t = r_t^{ext} + \alpha r_t^{int}$ in task-driven exploration setting, $r_t^{int} = x_{t+1} - x_t$ is problematic because non-zero r_t^{int} changes the reward function r_t^{ext} of the original POMDP. The optimal policy for the new POMDP with intrinsic rewards differs from the optimal policy for the original POMDP. Thus, the final policy of R1 does not maximize the cumulative extrinsic reward. The count-based reward $1/\sqrt{N(o_{t+1})}$ asymptotically vanishes to 0 during training. Therefore, a multiplication of R1 view coverage maximization reward $x_{t+1} - x_t$ with the count-based reward $1/\sqrt{N(o_{t+1})}$ can mitigate the misleading effect of R1.

An illustrative example is provided in Figure 10. The example shows a KC-S3R3 environment, where the agent is expected to find the key to unlock the locked blue door and pick up the yellow ball behind the door. Figure 10a shows that the agent has reached the key. Figure 10b 10c show how ViewX and

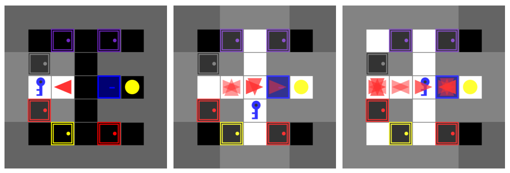


Figure 10: Comparison between ViewX and R1 on KC-S3R3 in task-driven setting. The agent (red triangle) should pick up the yellow ball behind the locked blue door. It has a view of 7x7 units (highlighted). (a) Visualization of agent reaching the blue key. (b) ViewX agent's behavior after picking up the key. (c) R1 agent's behavior after picking up the key.

Setting	ViewX	R1	R2	R3
BUP	**0.6788** (±0.1669)	0.0131 (±0.0129)	0.0123 (±0.0041)	0.0049 (±0.0039)
OM-1Dl	**0.5798** (±0.2270)	0.0257 (±0.0243)	0.0296 (±0.0169)	0.0078 (±0.0051)
OM-2Dl	**0.4154** (±0.1222)	0.0444 (±0.0212)	0.0629 (±0.0151)	0.0125 (±0.0034)
DK-6x6	**0.1920** (±0.1318)	0.1077 (±0.0254)	0.1473 (±0.0466)	0.0629 (±0.0137)
DK-8x8	**0.2617** (±0.1420)	0.0901 (±0.0370)	0.1349 (±0.0584)	0.0408 (±0.0182)
DK-16x16	**0.1405** (±0.0598)	0.0328 (±0.0174)	0.0555 (±0.0307)	0.0127 (±0.0043)

Table 1: Mean episodic rewards and standard deviations after training with a specific timesteps in task-agnostic setting.

R1 agents behave after picking up the key. ViewX agent directly turns back and opens the locked door. However, R1 agent opens the gray and red doors to gain coverage reward before going back to the locked door. R1 agent achieves larger view coverage than ViewX, but it costs more steps to reach the goal, which is a sub-optimal solution with lower extrinsic reward in this task.

The results of ablations on task-agnostic exploration can be found in Table 1. To sum up, when trained on a difficult task like BlockedUnlockPickup (BUP), ViewX agent explores environments better than the variations R1, R2, and R3. R1 and R2 agents fail to solve the BUP task because the view coverage maximization reward is mostly 0 before opening the door. Simply doing random exploration without any rewards makes it challenging to open the door through a tedious process of picking up the key and moving around the obstacle. The agent has a slight chance to correctly interact with objects or find passages to leave the current room via random exploration. R3 agent takes the indicator of coverage increase for bonus reward, but this bonus is less informative compared to ViewX. At the same time, the use of indicator may make R3 agent prefer to only see a bit more in each step, and thus, it explores much worse than ViewX in complex environments.

5 Limitations and Discussion

The scope of this work is for navigation-like tasks, which is the same as many prior works [Raileanu and Rocktäschel, 2020, Zha et al., 2021, Zhang et al., 2020a, 2021a, Parisi et al., 2021]. The underlying assumption to support ViewX is that seeing more space in the environment expedites the discovery of higher extrinsic rewards. However, obtaining a large view space coverage does not always mean advancing to the task goal. For example, the agent may be expected to conduct careful and complicated operations only in a small portion of a large environment. Therefore, we propose to deal with this issue by further extending our key idea to maximize knowledge coverage. In navigation tasks, the view coverage in ViewX is an easily-accessible proxy for the knowledge coverage. In general RL problems, our motivation is to consider how much area in the state space the agent has known to be reachable or non-reachable in an episode. We do not necessarily require the agent to really visit a new state for a bonus reward, but knowing whether a new state is reachable or not yields an intrinsic reward. We believe this high-level idea (e.g. maximizing knowledge coverage rather than visitation coverage of the state space) is more general and could be applicable to broader RL problems with some extensions. We leave this as future work. Besides, our approach assumes the agent's relative position and orientation are available information to construct the map. Combination with localization techniques to release this assumption is another interesting future direction.

6 Conclusion

Exploration in sparse-reward tasks, especially in procedurally-generated environments with partial observability, is challenging due to misleading novel observations. We propose a simple intrinsic reward to simultaneously seek large view space and novel observations to take advantage of both intrinsic motivations. We develop a novel algorithm, ViewX, for task-driven and task-agnostic exploration and empirically show its efficiency in various environments, including the 2D minigrid domain and the 3D visually-realistic environment. We verify that ViewX offers substantial gains in sample efficiency when solving tasks in procedurally-generated environments, and it outperforms previous state-of-the-art exploration methods with intrinsic rewards.

Acknowledgements

This work was supported in part by grants from NSF F062238-FW-HTF-R and LG AI Research.

References

A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018a.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018b.

D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to explore using active neural SLAM. *CoRR*, abs/2004.05155, 2020a. URL `https://arxiv.org/abs/2004.05155`.

D. S. Chaplot, H. Jiang, S. Gupta, and A. Gupta. Semantic curiosity for active visual learning. In *European Conference on Computer Vision*, pages 309–326. Springer, 2020b.

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 *Robotics: Science and Systems*, volume 11, pages 3–12, 2015.

T. Chen, S. Gupta, and A. Gupta. Learning exploration policies for navigation. *arXiv preprint arXiv:1903.01959*, 2019.

M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

J. Choi, Y. Guo, M. Moczulski, J. Oh, N. Wu, M. Norouzi, and H. Lee. Contingency-aware exploration in reinforcement learning. *arXiv preprint arXiv:1811.01483*, 2018.

K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Y. Guo, J. Choi, M. Moczulski, S. Feng, S. Bengio, M. Norouzi, and H. Lee. Memory based trajectory-conditioned policies for learning from sparse rewards. *Advances in Neural Information Processing Systems*, 33:4333–4345, 2020.

S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017.

N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*, 2018.

J. Z. Kolter and A. Y. Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th annual international conference on machine learning*, pages 513–520, 2009.

H. Küttler, N. Nardelli, A. Miller, R. Raileanu, M. Selvatici, E. Grefenstette, and T. Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33: 7671–7684, 2020.

D. Y. Little and F. T. Sommer. Learning and exploration in action-perception loops. *Frontiers in neural circuits*, 7:37, 2013.

J. M. Loomis, J. A. Da Silva, N. Fujita, and S. S. Fukusima. Visual space perception and visually directed action. *Journal of experimental psychology: Human Perception and Performance*, 18(4): 906, 1992.

A. Mirchev, B. Kayalibay, M. Soelch, P. van der Smagt, and J. Bayer. Approximate bayesian inference in spatial environments. *arXiv preprint arXiv:1805.07206*, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

S. A. Mobin, J. A. Arnemann, and F. Sommer. Information-based learning by agents in unbounded state spaces. *Advances in Neural Information Processing Systems*, 27, 2014.

M. Narasimhan, E. Wijmans, X. Chen, T. Darrell, D. Batra, D. Parikh, and A. Singh. Seeing the un-scene: Learning amodal semantic maps for room navigation. In *European Conference on Computer Vision*, pages 513–529. Springer, 2020.

G. Ostrovski, M. G. Bellemare, A. Oord, and R. Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

S. Parisi, V. Dean, D. Pathak, and A. Gupta. Interesting object, curious agent: Learning task-agnostic exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

R. Raileanu and T. Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.

S. K. Ramakrishnan, D. Jayaraman, and K. Grauman. An exploration of embodied visual exploration. *CoRR*, abs/2001.02192, 2020. URL `http://arxiv.org/abs/2001.02192`.

J. J. Rieser, D. H. Ashmead, C. R. Talor, and G. A. Youngquist. Visual perception and the guidance of locomotion without vision to previously seen targets. *Perception*, 19(5):675–689, 1990.

J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE transactions on autonomous mental development*, 2(3):230–247, 2010.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

P. Shyam, W. Jaśkowski, and F. Gomez. Model-based active exploration. In *International conference on machine learning*, pages 5779–5788. PMLR, 2019.

B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

J. Storck, S. Hochreiter, J. Schmidhuber, et al. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the international conference on artificial neural networks, Paris*, volume 2, pages 159–164. Citeseer, 1995.

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. Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR, 2021.

D. Zha, W. Ma, L. Yuan, X. Hu, and J. Liu. Rank the episodes: A simple approach for exploration in procedurally-generated environments. *arXiv preprint arXiv:2101.08152*, 2021.

T. Zhang, H. Xu, X. Wang, Y. Wu, K. Keutzer, J. E. Gonzalez, and Y. Tian. Bebold: Exploration beyond the boundary of explored regions. *arXiv preprint arXiv:2012.08621*, 2020a.

T. Zhang, P. Rashidinejad, J. Jiao, Y. Tian, J. E. Gonzalez, and S. Russell. Made: Exploration via maximizing deviation from explored regions. *Advances in Neural Information Processing Systems*, 34, 2021a.

T. Zhang, H. Xu, X. Wang, Y. Wu, K. Keutzer, J. E. Gonzalez, and Y. Tian. Noveld: A simple yet effective exploration criterion. *Advances in Neural Information Processing Systems*, 34, 2021b.

X. Zhang, Y. Ma, and A. Singla. Task-agnostic exploration in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:11734–11743, 2020b.

A Implementation Details

A.1 Task-driven exploration

Implementations of ViewX, NovelD, RIDE, and RND are built on the official codebase of NovelD[Zhang et al., 2021b]. For MADE, we directly take the results reported in the paper[Zhang et al., 2021a]. The intrinsic reward functions of the rest are as follows:

- ViewX: $(x_{t+1} - x_t + \lambda)/\sqrt{N(o_{t+1})}$, where x_t is the area of our constructed map and the map is reset at the beginning of each episode.
- NovelD: $\max[novelty(o_{t+1}) - \beta * novelty(o_t), 0] * \mathbb{1}\{N_e(o_{t+1} = 1)\}$, where $novelty(o_t) = \|\phi(o_t) - \hat{\phi}(o_t)\|^2$ - the difference between a random fixed target network $\hat{\phi}$ and a trainable predictor network ϕ. ϕ is trained to be close to $\hat{\phi}$ with the objective of minimizing $\|\phi(o_{t+1}) - \hat{\phi}(o_{t+1})\|^2$.
- RIDE: $\|\phi(o_t) - \phi(o_{t+1})\|^2/\sqrt{N(o_{t+1})}$, where ϕ is the state embedding network trained to minimize the prediction error of both an inverse and a forward dynamics model, $N(o_{t+1})$ is reset at the beginning of each episode.
- RND: $\|\phi(o_{t+1}) - \hat{\phi}(o_{t+1})\|^2$, where $\hat{\phi}$ is a fixed random network, ϕ is the state embedding network trained to minimize $\|\phi(o_{t+1}) - \hat{\phi}(o_{t+1})\|^2$.

Policy and value function The policy network and value function network are shared across different approaches. Given an input of dimension $7 \times 7 \times 3$ on MiniGrid, we fed it into three convolutional layers with kernel size 3×3, padding 1, number of channels 32, 128, 512, stride 1, 2, 2 respectively. Each convolutional layer is followed by a ELU activation function. After the last convolutional layer, the feature is flattened and processed by two linear layers with 1024 units and a ReLU activation function. The output from linear layers are put into an LSTM layer with 1024 units. After that, two separate fully-connected layers with 1024 units output action distribution as policy and value estimation as value function, respectively.

State embedding The state embedding network ϕ is learned for NovelD, RIDE, and RND. The input is the observation in MiniGrid with dimension $7 \times 7 \times 3$. It is passed through three convolutional layers with kernel size 3×3, padding 1, stride 1,2,2, number of channels 32, 128, 512 respectively. Each convolutional layer is followed by a ELU layer. The output is then passed to two linear layers of 2048 and 1024 units with ReLU activation.

Dynamics model RIDE requires training of the dynamics models to learn the state embedding network ϕ. For the forward dynamics model, the input is the state embedding and action. It is passed through two fully-connected layers with 256 and 128 units. The non-linear activation is ReLU activation function following each fully-connected layer. As for the inverse dynamics model, the input is state embeddings of two consecutive steps. The input is fed into two fully-connected layers with 256 units and a ReLU activation function.

Count dictionary In MiniGrid, the observation is of shape $7 \times 7 \times 3$ with values in the range of $[0, 10]$ as integers. We directly convert the observations o to tuples and store them as the keys in a dictionary to record the visitation counts $N(o)$.

Hyper-parameters Table 2 shows the values of hyper-parameters shared across different methods.

Parameter name	Value
Batch size	32
#Steps for LSTM	100
Optimizer	RMSProp
Learning rate	0.0001
Discount factor γ	0.99
Weight of policy entropy loss	0.0005
Weight of value function loss	0.5

Table 2: The hyper-parameters for experiments in task-driven setting.

For ViewX, we set the λ in Equation 1 as 0.01 for all tasks. For MultiRoom and KeyCorridor, the value of α is set to 0.01 and for ObstructedMaze environments it is set to 0.02.

For NovelD, we set $\alpha = 0.05$ for all the environments as is suggested in their official codebase.

Following [Raileanu and Rocktäschel, 2020], we use $\alpha = 0.1$ on KeyCorridor-S3R3 and $\alpha = 0.5$ on all other environments for RIDE and $\alpha = 0.1$ on all the environments for RND.

Compute resource Each job is run with an Nvidia TITAN X GPU and 40 CPUs. One job of ViewX takes around 4 hours for 10M steps.

In the supplementary material, we provide the code for experiments in the task-driven setting. More details can be found in the code.

A.2 Task-agnostic exploration

For a fair comparison, we implement ViewX for task-agnostic exploration on the codebase of [Parisi et al., 2021]. We compare exploration methods (ViewX, C-BET, RIDE, RND) with different intrinsic reward functions in the same codebase. The basic RL algorithm is IMPALA, the same for all these methods. The intrinsic reward functions are as follows:

- ViewX: $(x_{t+1} - x_t + \lambda)/\sqrt{N(o_{t+1})}$, where x_t is area of our constructed map and the map is reset at the beginning of each episode, the count $N(o_{t+1})$ is reset with a probability of 0.001 at each step.
- C-BET: $1/(N(change) + N(o_{t+1}))$, where $change$ is the environment panorama change of a transition $\{o_t, a_t, o_{t+1}\}$. The count N of $change$ and o_{t+1} is reset with a probability of 0.001 at each step.
- COUNT: $1/\sqrt{N(o_{t+1})}$, where the count $N(o_{t+1})$ is never reset.
- RIDE: $\|\phi(o_t) - \phi(o_{t+1})\|^2/\sqrt{N(o_{t+1})}$, where ϕ is the state embedding network trained to minimize the prediction error of both an inverse and a forward dynamics model, $N(o_{t+1})$ is reset at the beginning of each episode.
- RND: $\|\phi(o_{t+1}) - \hat{\phi}(o_{t+1})\|^2$, where $\hat{\phi}$ is a fixed random network, ϕ is the state embedding network trained to minimize $\|\phi(o_{t+1}) - \hat{\phi}(o_{t+1})\|^2$.

A.2.1 MiniGrid

Environment setting The episode length is the default maximum steps in each environment. In the task-agnostic setting, task is unknown so the episode is not terminated when the goal is achieved. We only terminate an episode when the number of steps reaches the maximum.

Policy and value function The network architecture for the policy and value function are the same for all methods. The input is the partial observations from the environment, with the shape of $7 \times 7 \times 3$. The input is fed into the convolutional neural network with three convolutional layers, and each layer has 32 filters, kernel size 3×3, stride 2 and padding 1. The non-linear activation function is ELU following each convolutional layer. The output of the convolutional network is flattened and then passed through two linear layers with 1024 hidden units and ReLU activation function. After the linear layers, the output is processed by LSTM layer with 1024 units. Finally, two separate fully-connected layers of 1024 units are used for output of policy and value function, respectively.

State embedding RIDE and RND requires learning of the state embedding network ϕ. The input is the observation in MiniGrid with dimension $7 \times 7 \times 3$. It is passed through three convolutional layers with kernel size 3×3, stride 2, padding 1, number of channels 32, 32, 128 respectively. Each convolutional layer is followed by a ELU layer. The output feature is flattened as state embedding.

The architecture of the dynamics network for RIDE, and the way to record count $N(o)$ are the same as we explained in Appendix A.1.

Count dictionary For C-BET, $N(change)$ is the count of any change in the panoramic view with dimension $7 \times 7 \times 3 \times 4$. The change in panoramic view is directly converted to tuples and stored as the key in the count dictionary.

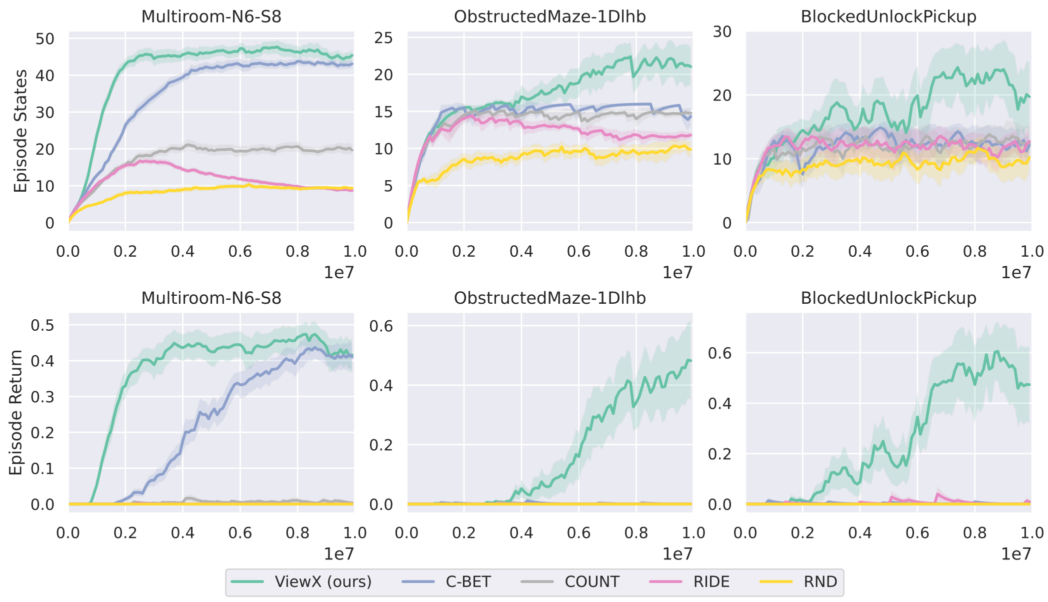


Figure 11: Training process of exploration policies in task-agnostic setting on MiniGrid

Hyper-parameters The values of hyper-parameters are listed in Table 3. The choice of the hyper-parameters follows the prior work [Parisi et al., 2021]. Most of the hyper-parameters are the same across different methods. The constant term λ in ViewX is set as 1 for simplicity without any hyper-parameter tuning.

Parameter name	Value
Batch size	8
#Steps for LSTM	100
Optimizer	RMSProp
Learning rate	0.0001
Discount factor γ	0.99
Gradient clip max norm	40
Weight of intrinsic reward α	0.005 (ViewX,C-BET, COUNT) 0.1(RIDE, RND)
Weight of policy entropy loss	0.0005
Weight of value function loss	0.5
Weight of forward dynamics loss	1.0 (RIDE)
Weight of inverse dynamics loss	0.1 (RIDE)
Weight of random network loss	0.1 (RND)
Constant term λ	1.0 (ViewX)

Table 3: The hyper-parameters for experiments in task-agnostic setting.

Compute resource Each job is run with an Nvidia TITAN X GPU and 10 CPUs. We set IMPALA using 10 actors. One job of ViewX takes around 20 hours for 10M steps.

We add more experiment details in Figure 11. We present the learning curves of the exploration policies on the training environments. The statistics we recorded during training is the number of visited cells and total extrinsic rewards in an episode. The episode rewards are measured only for evaluation, not for policy learning. The test performance of the learned exploration policy are presented in Figure 5,6,7 in the paper.

A.2.2 Habitat

Environment setting At each step, the agent can move forward by 0.25 meter or turn left/right by 10°. The episode length is 500 steps and each episode terminates at 500 steps.

Policy and value function The input is an environment partial observation with dimension $64 \times 64 \times 3$. It is passed through five convolutional layers and each layer has 32 filters, kernel size 3×3, stride 2 and padding 1. The non-linear activation function is ELU. The output of the convolutional layers is flattened and fed into two fully-connected layers with 1024 hidden units and ReLU activation. Then the output is processed by LSTM layer with 1024 units. Finally, two separate linear layers of 1024 units are after LSTM and used for policy and value function respectively.

Count dictionary The observations are RGB images of dimension $64 \times 64 \times 3$, with integer values in the range of $[0, 255]$. We convert the observations to 128 bits with a hash function, the same as [Parisi et al., 2021]. The hash encoding is then used as the key of the count dictionary.

Compute resource Each job is run with an Nvidia TITAN X GPU and 10 CPUs. We set IMPALA using 10 actors. One run of ViewX tasks around 20 hours for 2M steps.

The network architecture of the state embedding network, forward dynamics model, inverse dynamics model, and the hyper-parameters are the same as experiments in MiniGrid.

In the supplementary materials, we provide some examples of videos showing the exploration behavior of different methods on training and test environments.

B Ablation Study

In this section, we provide more experiment details for Section 4.4.

B.1 Task-driven exploration

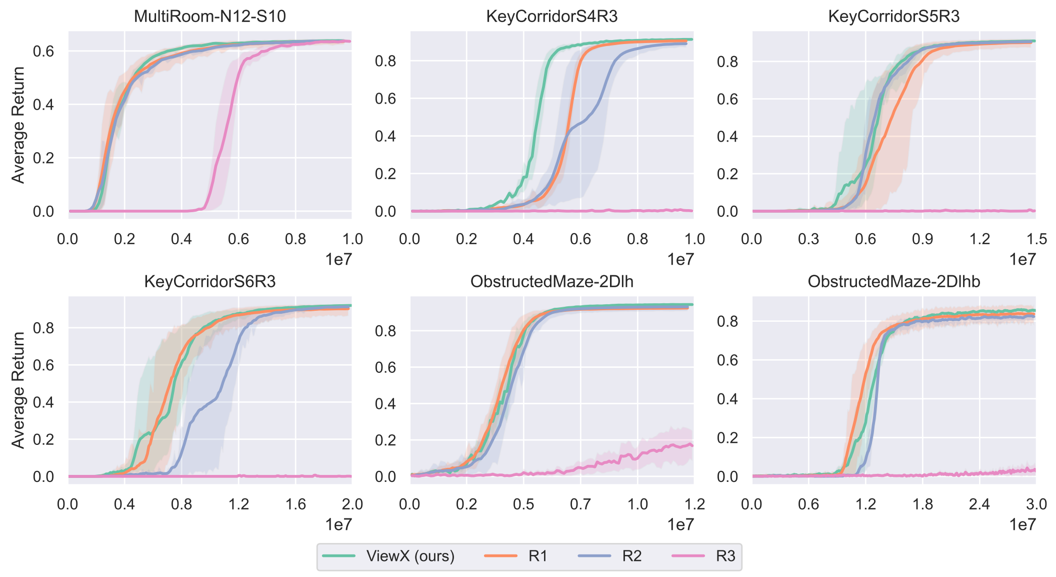


Figure 12: Learning curves of ViewX and the ablations.

Setting	ViewX	R1	R2	R3
KC-S4R3 (10M)	**0.9134** (±0.0029)	0.9057(±0.0024)	0.8969(±0.0029)	0.0023(±0.0008)
KC-S5R3 (15M)	**0.924** (±0.0031)	0.9185 (±0.0011)	0.9234 (±0.0047)	0.0031 (±0.0028)
KC-S6R3 (20M)	**0.9332**(±0.0024)	0.9245 (±0.0037)	0.9313 (±0.0024)	0.0000 (±0.0000)
MR-N12-S10(10M)	**0.6402**(±0.0029)	0.6401 (±0.0010)	0.6336 (±0.0086)	0.6399 (±0.0054)
OM-2Dlh (12M)	**0.9461** (±0.0007)	0.9239 (±0.0068)	0.9434 (±0.0014)	0.1359 (±0.1280)
OM-2Dlhb (20M)	**0.8598** (±0.0038)	0.8402 (±0.0143)	0.8281 (±0.0106)	0.0113 (±0.0015)

Table 4: Mean episodic rewards and standard deviations after training with a specific timesteps

Figure 12 shows the learning curves of ViewX and the three intrinsic reward designs we proposed in section 4.4 in task-driven setting. The performance of learned policy after task-driven training are summarized in Table 4. R1 has comparable sample efficiency with ViewX, but usually converges to sub-optimal policies. R2 has slightly worse sample efficiency than ViewX and R1, and R3 fails to learn in hard environments like KC-S6R3.

B.2 Task-agnostic exploration

In Figure 13, we show the learning process of exploration policies with different intrinsic reward functions. The policies are trained only with intrinsic rewards, and ViewX agent is about to visit more cells in the grid in an episode on the training environment. When trained on a difficult task like BlockedUnlockPickup (BUP), ViewX agent explores environments better than the variations R1, R2, and R3. In Table 5, we list the test performance of exploration policies in various environments, including training environment BUP and test environments. R1 and R2 agents fail to solve the BUP task because the view coverage maximization reward is mostly 0 before opening the door. Simply doing random exploration without any reward makes it challenging to open the door through a tedious process of picking up the key and moving around the obstacle. The agent has a slight chance to correctly interact with objects or find passages to leave the current room via random exploration. R3 agent takes the indicator of coverage increase for bonus reward, but this bonus is less informative compared to ViewX. At the same time, the use of indicator may make the R3 agent prefer to only see a bit more in each step, and thus, it explores much worse than ViewX in complex environments.

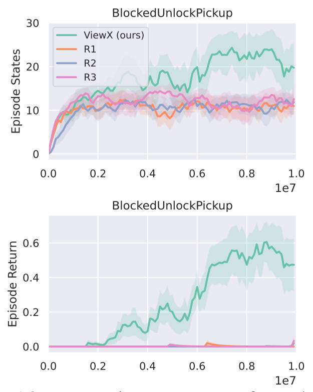


Figure 13: Learning curves of exploration policies in task-agnostic setting.

Setting	ViewX	R1	R2	R3
BUP	**0.6788** (±0.1669)	0.0131 (±0.0129)	0.0123 (±0.0041)	0.0049 (±0.0039)
OM-1Dl	**0.5798** (±0.2270)	0.0257 (±0.0243)	0.0296 (±0.0169)	0.0078 (±0.0051)
OM-2Dl	**0.4154** (±0.1222)	0.0444 (±0.0212)	0.0629 (±0.0151)	0.0125 (±0.0034)
DK-6x6	**0.1920** (±0.1318)	0.1077 (±0.0254)	0.1473 (±0.0466)	0.0629 (±0.0137)
DK-8x8	**0.2617** (±0.1420)	0.0901 (±0.0370)	0.1349 (±0.0584)	0.0408 (±0.0182)
DK-16x16	**0.1405** (±0.0598)	0.0328 (±0.0174)	0.0555 (±0.0307)	0.0127 (±0.0043)

Table 5: Mean episode rewards and standard deviation after during with 10M timesteps.

C Map construction process

In Figure 14 and Figure 15, we provide an illustrative example to explain our process to construct the map m_t. At each step t, we suppose the partial observation o'_t from top-down viewpoint is available in (or can be learned from) the environment partial observation o_t. We assume the agent's relative position and orientation in any two consecutive steps are available. At the start of each episode, we assign a simple position and orientation p_0 (e.g. (0, 0, up) in MiniGrid environment) and then update p_t step by step based on the relative position and orientation. The map m_0 is reset as empty at the start of each episode. The top-down view observations o'_t are filled into our map according to the assumed position, and thus, m_t is incrementally accumulated in the episode.

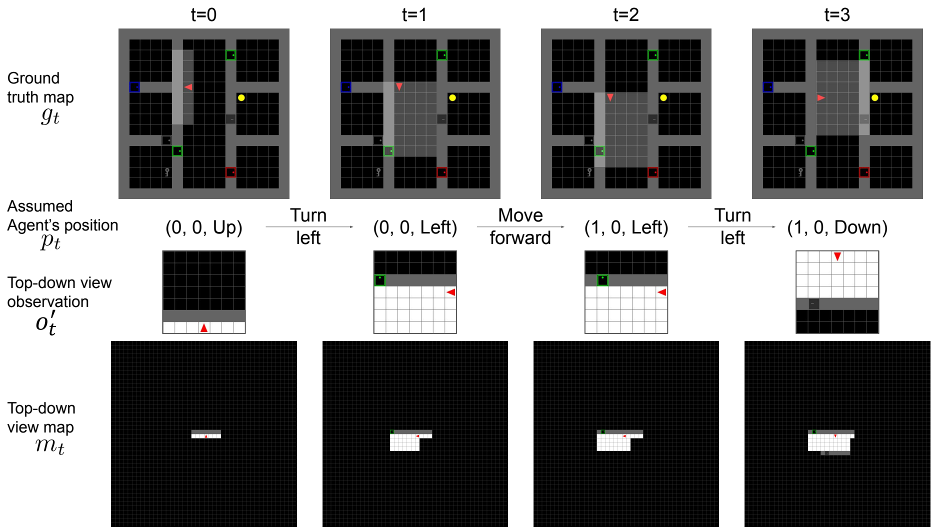


Figure 14: Illustrative example of map construction on MiniGrid Environment. The small red triangle annotates the agent's position and orientation, but this agent information is not available in observation o_t from the environment. We show it here only for better visualization and clearer explanation.

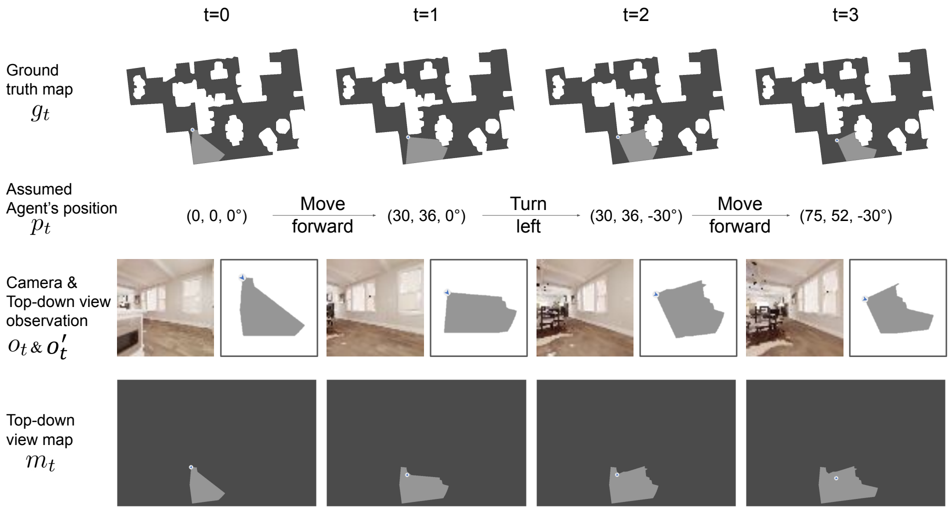


Figure 15: Illustrative example of map construction on Habitat. For more clear visualization, we make the action 'move forward', 'turn left' and 'turn right' to move more aggressively than the default setting (see Appendix A), so the observation difference in consecutive steps is more obvious.

As for our experiments in Section 4.3.2 in the Habitat environments, we assume that we have the ground truth top-down view observations o'_t corresponding to the partial observations o_t, as shown in

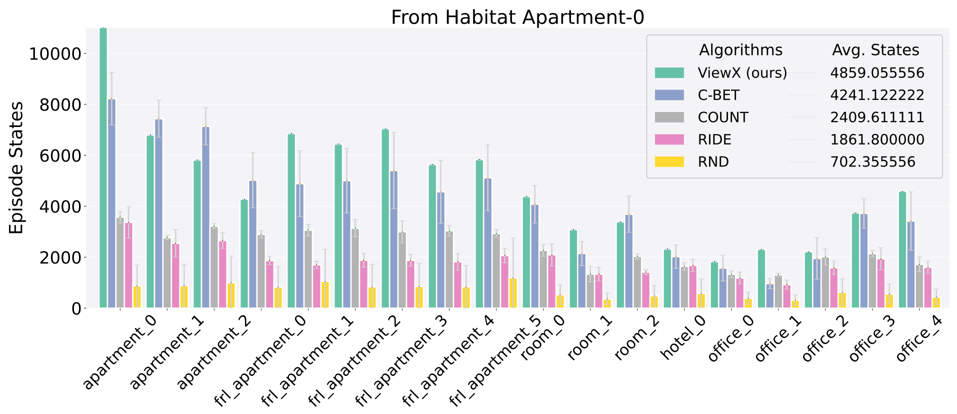


Figure 16: Exploration performance of ViewX with learned top-down view map

Figure 15. We further relax this assumption by utilizing an approach to approximate top-down view map from the first-person view observations [cha]. The performance of our approach and baselines is shown in Figure 16. ViewX with learned top-down view map still outperforms the baselines.

D Additional Experimental Results on Habitat

In this section, we provide results of additional experiments on Habitat environment.

D.1 Ablation Study

Figure 17 shows the exploration performance of ViewX and the intrinsic reward design R1, R2, and R3 in task-agnostic setting. The exploration policy is trained in one scene Apartment 0 with only intrinsic rewards for 2M steps. Here the intrinsic reward design R1 maximizes its view coverage, R2 excludes the constant term λ, and R3 uses indicator of whether view expands instead of raw increment of view coverage. Similar to the experimental results on MiniGrid environments, the performance of ViewX is also better than the view-coverage maximization reward in Habitat environment.

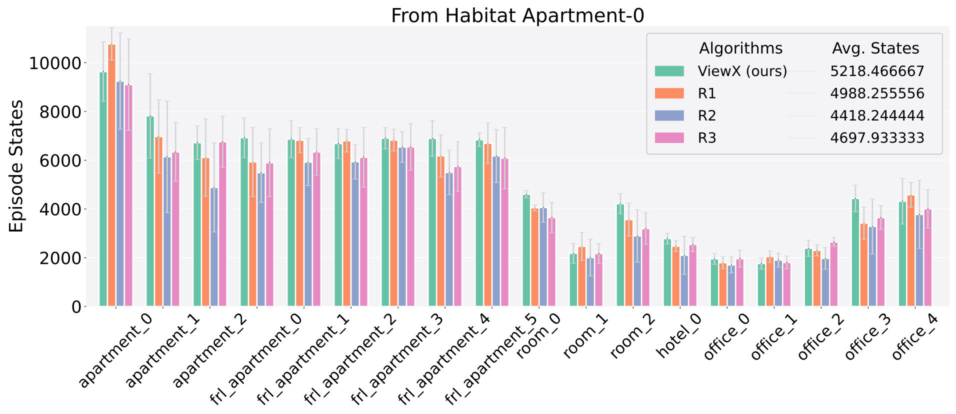


Figure 17: Exploration performance of ViewX and ablation models on scenes in Replica datasets

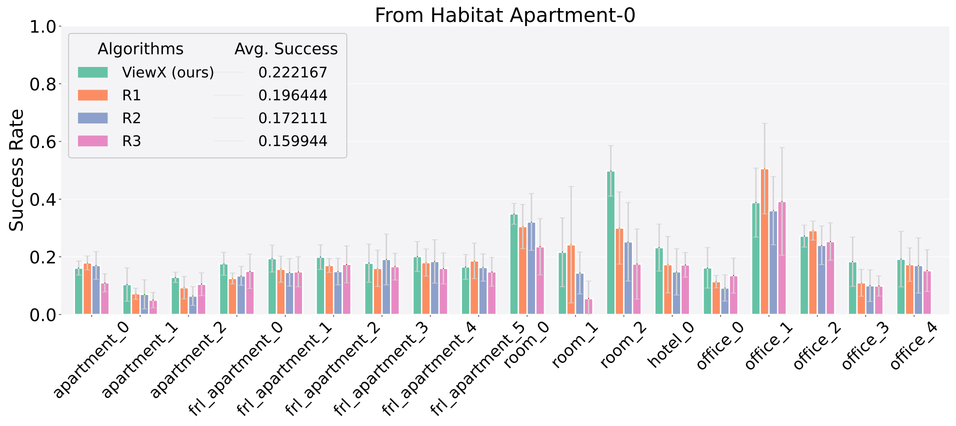


Figure 18: Success Rate of ViewX and ablation models on PointNav task on scenes in Replica datasets

D.2 Additional Evaluation Metric

To further explore the capability of ViewX, we apply the learned exploration policy into Point Navigation task in test scenes in Replica datasets. Point Navigation task requires the agent to start from a specific position, and reaches the given goal in limited steps. Both the start and goal position are randomly sampled in the environment in every episode. We regard it as success once the distance between the goal and the agent is less than 0.2m, and the agent has 500 steps at maximum to explore the scene in each episode. On each scene, we report the average success rate in 200 test episodes. Figure 18 shows that ViewX outperforms the intrinsic reward design R1, R2, and R3 on the success rate of point navigation task, which means under the situation that both the start and goal position is unknown to the agent, ViewX helps to better explore more unknown areas in limited steps.

E Relation with Information-Gain Exploration

In this section, we explain the relation between information gain exploration [Storck et al., 1995, Little and Sommer, 2013, Mobin et al., 2014, Charrow et al., 2015, Mirchev et al., 2018] and view coverage maximization method. The information gain exploration method maintains an internal model of the environment and optimizes the agent to achieve largest change in information in the internal model.

In our case, the map m_t is the agent's internal model of the environment. The change in information is quantified as the KL divergence between the map variable distributions at the start and end of an episode $KL(p(m|o_1, o_2, \cdots, o_T)\|p(m))$, where $m \in \mathbb{R}^{H\times W}$ is the environment map with height H and width W, and T is the limit of timesteps in an episode.

The distribution of the map variable at each cell (i, j) is independent from each other. Thus, $p(m) = \prod_{1\le i\le H, 1\le j\le W} p(m^{i,j})$ and $p(m|o_1, o_2, \cdots, o_T) = \prod_{1\le i\le H, 1\le j\le W} p(m^{i,j}|o_1, o_2, \cdots, o_T)$.

For each cell, the initial distribution $p(m^{i,j})$ follows the discrete uniform distribution on $1, 2, \cdots, K$, where K is the total number of possible objects and it is a prefixed constant. If the agent ever sees the cell (i, j) in its view after T steps, $p(m^{i,j}|o_1, o_2, \cdots, o_T) = 1$; if $m^{i,j}$ is the correct object k at the cell (i, j) then $p(m^{i,j}|o_1, o_2, \cdots, o_T) = 0$ otherwise.

According to these assumptions, the change in information can be simplified as:

$$\begin{aligned}
& KL(p(m|o_1, o_2, \cdots, o_T)\|p(m)) \\
= & \sum_{m\in\{1,2,\cdots,K\}^{H\times W}} p(m|o_1, o_2, \cdots, o_T) \log \frac{p(m|o_1, o_2, \cdots, o_T)}{p(m)} \\
= & \sum_{m\in\{1,2,\cdots,K\}^{H\times W}} p(m|o_1, o_2, \cdots, o_T) \log \frac{\prod_{1\le i\le H,1\le j\le W} p(m^{i,j}|o_1, o_2, \cdots, o_T)}{\prod_{1\le i\le H,1\le j\le W} p(m^{i,j})} \\
= & \sum_{m\in\{1,2,\cdots,K\}^{H\times W}} p(m|o_1, o_2, \cdots, o_T) \sum_{1\le i\le H,1\le j\le W} \log \frac{p(m^{i,j}|o_1, o_2, \cdots, o_T)}{p(m^{i,j})} \\
= & \sum_{1\le i\le H,1\le j\le W} \sum_{m\in\{1,2,\cdots,K\}^{H\times W}} p(m|o_1, o_2, \cdots, o_T) \log \frac{p(m^{i,j}|o_1, o_2, \cdots, o_T)}{p(m^{i,j})} \\
= & \sum_{1\le i\le H,1\le j\le W} \sum_{m^{/i,j}\in\{1,2,\cdots,K\}^{H\times W-1}} p(m^{/i,j}|o_1, o_2, \cdots, o_T) \\
& \sum_{m^{i,j}\in\{1,2,\cdots,K\}} p(m^{i,j}|o_1, o_2, \cdots, o_T) \log \frac{p(m^{i,j}|o_1, o_2, \cdots, o_T)}{p(m^{i,j})} \\
= & \sum_{1\le i\le H,1\le j\le W} \sum_{m^{i,j}\in\{1,2,\cdots,K\}} p(m^{i,j}|o_1, o_2, \cdots, o_T) \log \frac{p(m^{i,j}|o_1, o_2, \cdots, o_T)}{p(m^{i,j})} \\
= & \sum_{1\le i\le H,1\le j\le W} KL(p(m^{i,j}|o_1, o_2, \cdots, o_T)\|p(m^{i,j}))
\end{aligned}$$

If the cell (i, j) is never seen at this episode for T steps, the distribution $p(m^{i,j}|o_1, o_2, \cdots, o_T)$ is the same as initial distribution $p(m^{i,j}))$. So the KL divergence for this cell in the map is 0.

If the cell (i, j) is seen in the agent's view and the object is $k \in \{1, 2, \cdots, K\}$, we have:

$$
\begin{aligned}
& KL(p(m^{i,j}|o_1, o_2, \cdots, o_T) \| p(m^{i,j})) \\
= & \sum_{m^{i,j} \in \{1,2,\cdots,K\}} p(m^{i,j}|o_1, o_2, \cdots, o_T) \log \frac{p(m^{i,j}|o_1, o_2, \cdots, o_T)}{p(m^{i,j})} \\
= & 1 \cdot \log \frac{1}{1/K} + \sum_{m^{i,j} \neq k, m^{i,j} \in \{1,2,\cdots,K\}} p(m^{i,j}|o_1, o_2, \cdots, o_T) \log \frac{p(m^{i,j}|o_1, o_2, \cdots, o_T)}{p(m^{i,j})} \\
= & \log K
\end{aligned}
$$

Overall, $KL(p(m|o_1, o_2, \cdots, o_T) \| p(m))$ can be written as *number of seen cells* $\times \log K$. In our notations, the number of seen cells after T steps in an episode is x_T.

Therefore, the information gain exploration seeking largest change in information is equivalent to seeking largest seen area in our setup. The view coverage maximization method with intrinsic reward $x_{t+1} - x_t$ for each step t is a special case of the information gain exploration.

F Hyper-parameter Sensitivity

The two major hyper-parameters for ViewX include the intrinsic reward coefficient and the λ from equation 1. For the intrinsic reward coefficient, most previous methods use 0.05, however, our intrinsic reward design includes an area change term which may be greater than 1. Therefore we lower the coefficient to 0.01 to avoid the intrinsic rewards being too large. For λ, we search the hyper-parameter in [0.001, 0.01, 0.05, 0.1]. As is shown in Figure 19, the performance of ViewX is not greatly influenced by λ and we set it to 0.01 across all the task-driven experiments for simplicity.

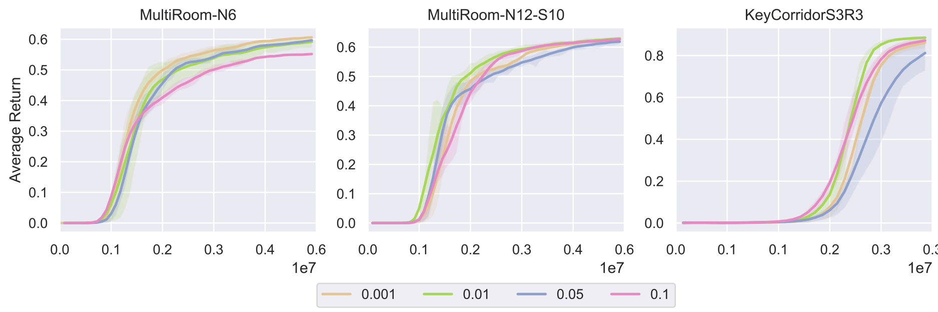


Figure 19: Task-driven experiment results with different λ.

G Comparison with Baselines using Localization Assumption

In this section, we experiment with one more baseline that uses localization assumption. Instead of recording the area that the agent "sees", we record the area where the agent has been to. More specifically, the intrinsic reward is designed as

$$r_t^{int} = \frac{l_{t+1} - l_t + \lambda}{\sqrt{N(o_{t+1})}} \tag{2}$$

where l_t denotes the total area that the agent has been to. The result is shown in Figure 20. We can see that changing view coverage to the coverage of the parts that the agent has physically been to makes the training much slower. The result is reasonable because in this baseline, if the agent sees some empty space, it still goes there for higher intrinsic reward, which prevents it from reaching the goal more efficiently. Therefore, visitation coverage l_t cannot replace view coverage x_t in our intrinsic reward design. Because the view coverage maximization reward in ViewX can be interpreted as an information-gain exploration bonus (see Appendix E), this comparison also implicitly verifies the benefit of pursuing information gain in ViewX intrinsic reward.

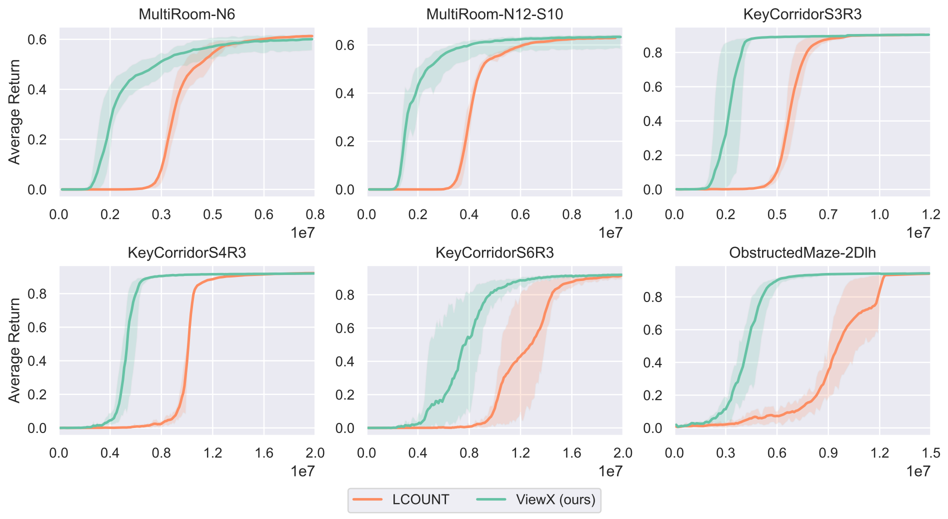


Figure 20: Compare ViewX with Location Count.