electronics

MDPI

Article

A Hybrid and Hierarchical Approach for Spatial Exploration in Dynamic Environments

Qi Zhang [1,†], Yukai Song [1,2,†], Peng Jiao [1] and Yue Hu [1,*]

[1] College of Systems Engineering, National University of Defense Technology, Changsha 410003, China; zhangqiy123@nudt.edu.cn (Q.Z.); songyukai14@163.com (Y.S.); crocus201@163.com (P.J.)
[2] Unit.63880 of PLA, Luoyang 471000, China
* Correspondence: huyue11@nudt.edu.cn
† These authors are both first authors.

Abstract: Exploration in unknown dynamic environments is a challenging problem in an AI system, and current techniques tend to produce irrational exploratory behaviours and fail in obstacle avoidance. To this end, we present a three-tiered hierarchical and modular spatial exploration model that combines the intrinsic motivation integrated deep reinforcement learning (DRL) and rule-based real-time obstacle avoidance approach. We address the spatial exploration problem in two levels on the whole. On the higher level, a DRL based global module learns to determine a distant but easily reachable target that maximizes the current exploration progress. On the lower level, another two-level hierarchical movement controller is used to produce locally smooth and safe movements between targets based on the information of known areas and free space assumption. Experimental results on diverse and challenging 2D dynamic maps show that the proposed model achieves almost 90% coverage and generates smoother trajectories compared with a state-of-the-art IM based DRL and some other heuristic methods on the basis of avoiding obstacles in real time.

Keywords: spatial exploration; hierarchical framework; deep reinforcement learning; intrinsic motivation; path planning; obstacle avoidance

Citation: Zhang, Q.; Song, Y.; Jiao, P.; Hu, Y. A Hybrid and Hierarchical Approach for Spatial Exploration in Dynamic Environments. *Electronics* **2022**, *11*, 574. https://doi.org/10.3390/electronics11040574

Academic Editor: Jeha Ryu

Received: 21 December 2021
Accepted: 10 February 2022
Published: 14 February 2022

1. Introduction

Spatial cognitive behaviour modelling is the basic content of human cognitive behaviour modelling, and is one of the hottest topics in the field of neuroscience and computer science. At its core, the agent in an AI system needs to explore the environment to gain enough information about the spatial structure. The possible applications include, for example, search and rescue (SAR) missions, intelligence, surveillance and reconnaissance (ISR), and planetary exploration. Therefore, it is important to design an efficient and effective exploration strategy in unknown spaces.

At present, autonomous spatial exploration falls into two main categories: traditional rule-based exploration and intelligent machine-learning-based exploration. The rule-based exploration methods such as frontier-based method [1] is simple, convenient and efficient. This kind of approach rely on an expert feature of maps, expanded the exploration scope by searching for the next optimal frontier point which is between free points and unknown points according to the explored map. However, the locomotion of the agent driven by this method is mechanical and rigid, and it is also difficult to balance between exploration efficiency and computational burden. As an effective tool for autonomous learning strategies, deep reinforcement learning (DRL) has been more and more widely used in spatial exploration. However, DRL suffers much from the inherent "exploration-exploitation" dilemma, resulting in sampling inefficiency if the extrinsic rewards are sparse or even non-existent. To solve the problem of sparse rewards, many recent DRL approaches incorporate the concept of intrinsic motivation (IM) from cognitive psychology to produce intrinsic rewards to make the rewards denser. However, intrinsic motivation based enhancement

is insufficient for efficient exploration in unknown spaces. The main reason is that IM treats all unseen states indiscriminately and ignores the structural regularities of physical spaces. In addition, it is difficult for end-to-end DRL agent to simultaneously learn obstacle avoidance, path planning and spatial exploration from raw sensor data.

To this end, we extend our previous work [2] and propose a three-tiered hierarchical autonomous spatial exploration model, named Intrinsic Rewards based Hierarchical Exploration with Soft-adaptive Finite-time Velocity Obstacle (IRHE-SFVO), to explore unknown static and dynamic 2D spaces. This model consists of two parts: a Global Exploration Module (GEM) and a Local Movement Module (LMM). GEM is used to learn an exploration policy to produce a sequence of target points that will maximize the information gain about the spatial structure based on the location of the agent, the trace of the agent, and the explored portions as its spatial memory. Specifically, to make the motion pattern of the agent more like human beings, GEM is not concerned with the immediate neighbourhood of the agent, but determines a distant yet reasonably reachable target to be explored next. Selected based on intrinsic rewards, such targets are usually those with a lot of unexplored areas around them.

In the local movement phase, this paper designs a hierarchical framework to control the movement to the target point. We separate this phase into two parts: planning and controlling. In the planning stage, an optimistic A* path planning algorithm, which can conduct self-adaptive path planning in a partially known environment, is used to compute a shortest path between the current location of the agent and the target point. It assumes that unknown areas are freely reachable and decides whether to replan the global path according to the ongoing perception. In the controlling stage, we use the improved Finite-time Velocity Obstacle (FVO), called Self-adaptive Finite-time Velocity Obstacle (SFVO), and design an optimal velocity function to drive the agent to avoid moving obstacles in real-time. This allows the agent to reach the target point quickly while avoiding collision with moving obstacles at the same time.

Working in synergy, the modules in the three levels apply a long-horizon decision-making paradigm instead of the step-by-step or state-by-state way used by some other exploration methods [3]. This segmentation not only reduces the training difficulty, but also tends to generate smooth movements between targets instead of unnatural trajectories. In summary, the main novelties and technical contributions of this paper include: (a) a hierarchical framework for spatial exploration that well exploits the structural regularities of unknown environments, (b) an information-maximal intrinsic reward function for determining the next best target to be explored, (c) a hierarchical framework for local movement that combines the global path planning with the local path planning for reaching the target point rapidly and safely and (d) an optimal velocity function for choosing the best velocity in collision-avoidance velocity set.

This paper is organized as follows. Section 2 describes related works in automatic exploration, the DRL based on IM and real-time obstacle avoidance. Section 3 formulates automatic exploration. Then, we present the details of our proposed algorithm and hyperparameter setting in Section 4. In Section 5, we compare our approach against several popular competitors in a series of simulation experiments, showing that IRHE-SFVO is promising for spatial exploration. Finally, in Section 6, we summarize our work this paper and discuss future work.

2. Related Work

In this section, we will describe and analyse the research status and development trends of autonomous spatial exploration, reinforcement learning based on IM and various velocity obstacle methods in this section.

2.1. Autonomous Spatial Exploration

At present, the research on autonomous spatial exploration mainly includes two categories: traditional rule-based autonomous spatial exploration and intelligent machine-

learning-based autonomous spatial exploration. The mainstream of rule-based method is frontier-based method proposed by Yamauchi in 1997 [1]. This method detects the "frontier", that is, the edges between the free area and the unknown area, then selects the best "frontier point" by some principles, and the agent moves from the current position to the selected "frontier point" by path planning and locomotion, so as to finally achieve the purpose of exploring the whole map. The frontier-based exploration strategy is similar to the NBV (Next Best View) problem in computer vision and graphics. Similarly, there is a lot of literature on the second step of frontier-based exploration strategy, i.e., evaluating and choosing the best frontier. There are generally three types of metrics: (a) cost-based which select the next target based on the path length or time cost [4–7], (b) utility-based which select the next target based on the information gain [8,9] and (c) the mixture [10]. Another typical traditional rule-based method is associated with information theory. These methods leverage some metrics such as entropy [11] or mutual information (MI) [12] to evaluate the uncertainty of the agent's position and the evidence grid map to control the agent to move in the direction of maximizing the information gain. In general, although the rule-based approach is simple and efficient, the movement mode of the agent driven by them is mechanical and rigid, and it is also difficult to balance exploration efficiency with computational burden.

Due to the recent significant advance in DRL, a number of researchers have tried to solve the exploration problem as an optimal control problem. Tai Lei and Liu Ming [13] proposed an improved DQN framework to train robots to master obstacle avoidance strategies in unknown environments through supervised learning based on convolution neural networks (CNN). However, they only solved the collision avoidance problem and failed to finish the spatial exploration task. Zhang et al. [14] trained an Asynchronous Advantage Actor-Critic (A3C) agent that can learn from perceptual information and construct a global map by combining it with a memory module. Similarly, an A3C network in [15] receives the current map, the agent's location and orientation as input, and returns the next visiting direction, given that the space around the agent is equally divided into six sectors. Chen et al. [16] designed a module of spatial memory and used the coverage area gain as an intrinsic reward, and accelerated the convergence of policy through imitation learning. Razin et al. [17] used Faster R-CNN to avoid collision and used double deep Q-learning (DDQN) model to explore unknown space. However, although DRL can solve the problem of limited dimensions, it has difficulty training in end-to-end control.

To solve these problems, Niroui et al. [18] and Shrestha et al. [19] combined DRL with a frontier-based method to enable robots to learn exploration strategies from their own experience. Li et al. [20] proposed a modular framework for robot exploration based on decision, planning and mapping modules. This framework used DQN to learn a policy for selecting the next exploration target in the decision module and used an auxiliary edge segmentation task to speed up training. Chaplot et al. [21] used the Active Neural SLAM module to address the exploration in 3D environments under the condition of perception noises. We draw some inspiration from these two works but are more interested in exploration in 2D environments.

2.2. RL Based on Intrinsic Motivation

To solve the notorious reward-sparse problem, many recent DRL approaches incorporate the intrinsic motivation from cognitive psychology. Intrinsic motivation is produced from human's natural interest in all kinds of activities that can provide novelty, surprise, curiosity, or challenge [22], without any external rewards such as food, money or punishment.

Applying IM to the RL means that the agent generates an "intrinsic reward" by itself during the interaction with the environment. The formulation of intrinsic rewards can be roughly divided into three categories, (a) visit count and uncertainty evaluation-based methods, (b) knowledge and information gain-based methods, and (c) competence-based methods. The first class of methods, based on upper confidence bound (UCB), estimate the counts of state visitation in high-dimensional feature space and large-scale state space, to

encourage the agent to visit poorly known states. This genre includes the density-based methods [23,24], state generalization-based methods [25–28] and inference calculation-based methods [29]. Second, the knowledge and information gain-based methods generally establish a dynamics model of the unknown environment and measures the intrinsic rewards using the model's increased accuracy as the exploration progresses. The specific formal models of this type include predict inconsistencies based model [30–32], prediction error based model [3,33–36], learning process based model [37] and information theory based model [36,38,39]. The third class formulates the intrinsic rewards by measuring the agent's competence to control the environment or the difficulty and cost of completing a task [40]. At present, the DRL based on IM has made great progress relative to the classic RL in applications with complex state spaces and difficult exploration (such as Atari-57 games) [41].

2.3. Velocity Obstacle

A crucial problem in exploration is how to avoid static and dynamic obstacles in real time. The known static obstacles are usually considered in global path planning, while unknown or dynamic obstacles are the focus of local path planning. The common collision avoidance methods include artificial aperture method (APF) [42], dynamic window method (DWA) [43] and behaviour method [44]. These methods have strong adaptability and high efficiency, so many researchers often combine the intelligent control algorithms with these methods for obstacle avoidance [45,46]. Besides, lazy rapidly-exploring random tree method (RRT) [47] method is also used for local path planning. However, these methods above cannot avoid collisions completely with moving obstacles or have certain randomness which leads to low efficiency of obstacle avoidance such as [47]. Alternatively, Velocity Obstacle (VO), first proposed by Fiorini et al. [48], is a simple and efficient algorithm that can avoid static and moving obstacles completely. It generates a conical velocity obstacle space in the agent velocity space. As long as the current velocity vector is outside the VO space, the agent will not collide with obstacles at any time in the future. However, the basic VO has many disadvantages. First, if the agent and moving obstacles or other agents use VO for local path planning at the same time, it will lead to oscillatory motion on both sides [49]. Secondly, the VO space excludes every velocity that may lead to collision, that is, a velocity that can cause a collision after a long time will also be excluded. This leads to the reduction of the range of optional collision-avoidance velocities in some scenarios, or even no optional velocity. To overcome these problems, Abe and Matsuo [50] proposed a common velocity obstacle (CVO) method, which provides collision detection between moving agents and enables agents to share collision information without explicit communication. This information allows agents to use the general VO method for implicit cooperation, so as to achieve the effect of avoiding collision. Guy et al. [51] proposed the finite time velocity obstacle algorithm (FVO), which expands the optional velocity vector of the traditional VO algorithm by calculating the collision velocity cone within a certain time. In order to solve the local oscillation problem Fulgenzi et al. [49] proposed reciprocal velocity obstacles (RVO) by considering the velocity change of both sides of the agents.

The proposed model in this paper combines the DRL, intrinsic motivation and velocity obstacle. Due to the features of exploration in 2D dynamic spaces, we reshape the generating paradigm of intrinsic reward. In order to ensure the safe and fast movement of the agent, we propose another hierarchical approach that combines a variation of the A* path planning method (called optimistic A*) and improved FVO (called self-adaptive FVO, SFVO).

3. Problem Formulation

Before giving the details of the proposed model, this section first formulates the exploration problem in a 2D environment.

Definition 1. *A Working Space, denoted as* $\boldsymbol{WS_M}$*, represents a 2D grid map of the size* $M \times M$*. Any element in* $\boldsymbol{WS_M}$ *can be represented as* (x, y)*,* $1 \leq x, y \leq M$*. Each cell in the grid is represented by* $T(x, y)$*:* $T(x, y) = 0$ *means a free cell while 1 is for a location occupied by an obstacle. Besides, we assume that the area of each cell is 1.*

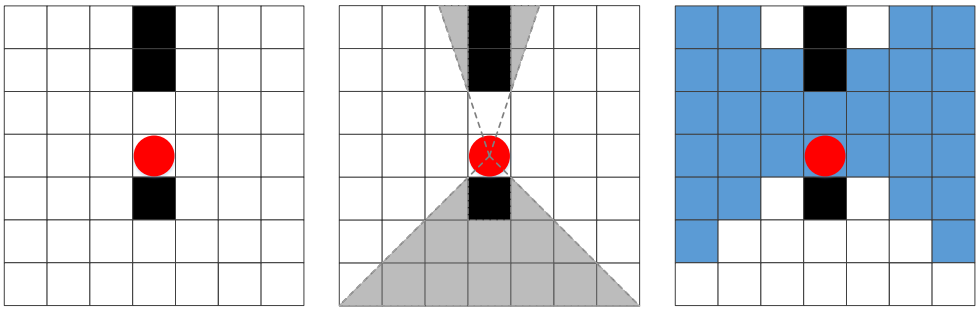
Definition 2. *Definition 2 Observation Range (**ObsR**) of an agent is the set of any point whose vertical and horizontal distance to the current position of the agent is not more than the observation radius (n):*

$$ObsR(x_i, y_i) = \{(x, y) | (|x - x_i| \leq n, |y - y_i| \leq n)\} \tag{1}$$

Definition 3. *Exploration Range (**ExpR**) of an agent is the set of any point whose vertical and horizontal distance to the current position of the agent is not more than the exploration radius(m), and it can be covered more than half of the area by the 'radar wave' emitted by agent:*

$$ExpR(x_i, y_i) = \{(x, y) | (|x - x_i| \leq m, |y - y_i| \leq m), S((x_i, y_i) \to (x, y)) > \frac{1}{2}\} \tag{2}$$

$S((x_i, y_i) \to (x, y))$ means the covered area by the "radar wave" emitted from (x_i, y_i) to (x, y). A specific example is shown in Figure 1.



(**a**) (**b**) (**c**)

Figure 1. An example of Exploration Range. (**a**) shows the obstacles around the agent, where the red solid circle represents the agent, and the black squares represent two obstacles. (**b**) shows the range that can be covered by the "radar wave" emitted from the agent. The gray shaded areas indicate that these areas are not covered by the "radar wave". (**c**) shows whether each cell in this scenario can be regarded as an explored area under Definition 3 when $m = 3$. The blue cells are the areas that the agent has been explored, while the agent has not explored the white areas.

Note that the region observed by the agent does not represent where it has been explored.As a simple example, imagine that we are searching for gold that cannot be seen from the earth's surface, so that we should use a gold detector to explore the region as far as it can extend into. We cannot find gold using our eyes, but the detector can. In general, the "detection range" (m) should not be greater than the "length of field of view" (n), i.e., $m \leq n$ and $ExpR(x_i, y_i) \subseteq ObsR(x_i, y_i)$.

4. The Proposed Model

This paper combines the advantages of DRL algorithms, traditional non-learning planning algorithms and real-time collision avoidance algorithms, and propose a novel approach to solve the exploration problem in the 2D dynamic grid. The proposed model is modular and hierarchical so that it cannot only exploit the structural regularities of the environment but also improve the training efficiency of DRL methods. The overall structure of our model is shown in Figure 2. GEM determines the next long-term target point to be explored based on a spatial map m_t maintained by the agent. LMM takes the next target point as input and computes the specific action to reach the target point. We use t_g to index the step of selecting the next target in only GEM. For example, we select a target point at

initial time, $t = t_g = 1$, and we assume the agent takes 10 steps to reach this target and select a next target point, then $t = 11$ and $t_g = 2$ at this moment.

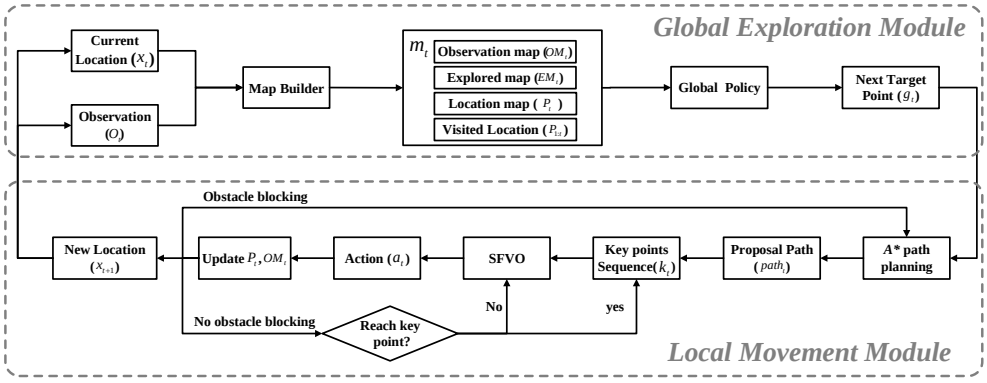


Figure 2. The overview of our IRHE-SFVO. In Global Exploration Module (GEM), the agent uses the current location and observation to build a spatial map m_t, then input m_t into Global Policy and output the next target point that will be explored. The Local Movement Module (LMM) determines the specific action to reach the target point quickly and safely based on the agent's current location, the next target point, and the obstacle map maintained by the agent.

4.1. Global Exploration Module

We want to learn an exploration policy π_g that enables the agent to select a location to explore so that the information gain about the environment can be maximized. For this purpose, we design an intrinsic reward function, favouring states where the agent can increase its exploration range at a fastest speed. Proximal Policy Optimization (PPO) [52] is used for training π_g. Importantly, π_g is learned on a set of training maps and tested on another set of unseen maps. This setting is to demonstrate the desirable generalization of our method across different environments.

4.1.1. Spatial Map Representation

First, as shown in the top block in Figure 2, GEM maintains a four-channel spatial map, m_t, as the input of the global policy. Then, the policy network outputs a next target point (g_{t_g}) that will be explored. To be specific, the spatial map contains four matrices of the same size, i.e., $m_t = \{0,1\}^{4\times M\times M}$, where M is the height and width of the explored maps. Each element in the first channel represents whether the location is an obstacle (OM_t): 0 is for a free cell and 1 is for a blocked one. In the beginning, $OM_0 = \{0\}^{M\times M}$ based on the assumption of free space. Each element in the second channel represents whether the location has been explored (EM_t). The third channel encodes the current location (P_t) in a one-hot manner, i.e., the element corresponding to the agent location is set to be 1, and the others are 0. The fourth channel labels the visited locations ($P_{1:t}$) from the initial time to the current time. The rationality of establishing these four channels is that the agent can fully exploit all spatiotemporal information useful for target decision-making. In particular, this elegant design is: (a) to enable the agent to use the structural regularities of the spatial environment to make correct decisions, (b) to prevent the agent from selecting the points that have already been explored when choosing the next target point, and (c) to make the agent select the best next target point based on the current location, considering the time cost and exploration utility comprehensively.

4.1.2. Network Architecture

The policy network takes m_t as input and outputs a $g_{t_g} \mapsto \pi_g(m_t;\theta_g)$, where θ_g are the parameters of the global policy. As shown in Figure 3, the spatial map m_t is first passed through an embedding layer and the layer outputs a four-dimensional tensor of size $4 \times N \times M \times M \times M$, where N represents the length of each embedding vector. Then, add the four constituent 3D tensors along their first dimension and we get a tensor with

rich information whose size is $N \times M \times M \times M$. Then, this 3D tensor is passed through three convolution layers and three fully connected layers successively, and finally outputs a next target point: g_{t_g}. Note that the embedding layer is essential for preserving information embedded in the input because its input is $0-1$ matrices of size $4 \times M \times M \times M$, which are all very sparse. Although the convolutional and pooling operations can extract spatial structure information, they will result in loss of many valuable information, and ignore the association between the overall and part as well if we send a matrix to the CNN and pooling layer directly. Therefore, to ensure the integrity of the information, it is necessary to map the m_t to a higher-dimensional vector first.

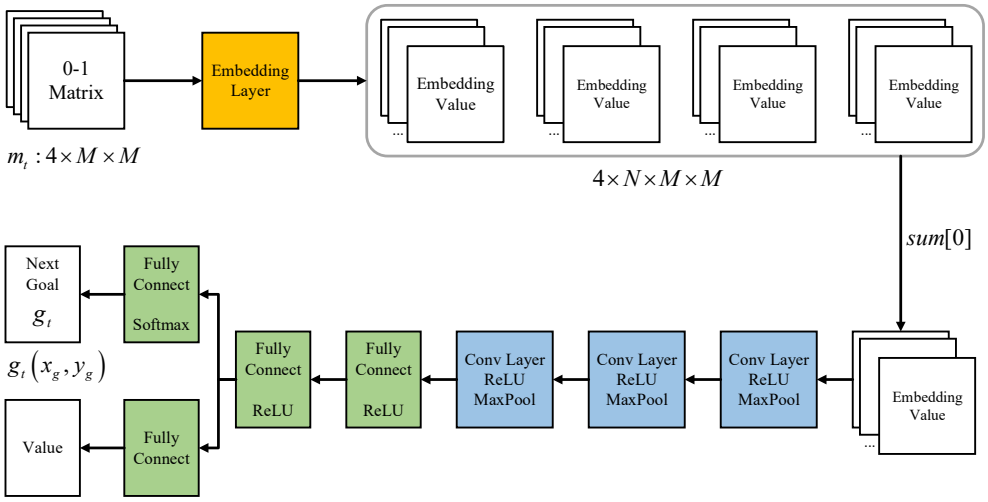


Figure 3. The structure of the actor-critic network in GEM. N represents the size of each embedding vector.

4.1.3. Intrinsic Reward

The effectiveness of DRL relies on rewards heavily. However, the exploration task is a reward-sparse RL problem. To alleviate the problem, we design an intrinsic reward (denoted by $r^i_{t_g}$) and combine it with the external rewards (denoted by $r^e_{t_g}$) given by the environment, i.e., $r_{t_g} = r^i_{t_g} + r^e_{t_g}$, so that the rewards along the exploration trajectory becomes denser. This is critically helpful to speed up the convergence of the policy and for the emergence of directed exploration. In literature, possible IM formulations include "curiosity" [34], "novelty" [53] or "empowerment" [40] to generate intrinsic rewards as described in Section 2. However, these approaches use blackbox models that cannot be initialized at each episode because the weights of neural networks cannot be reset in different episodes, resulting in the intrinsic reward getting smaller and smaller after each episode under the same scenario. To solve this problem, we design a simple yet effective intrinsic reward function that resets r^i at each episode. We use the increase of the explored area deduced from EM_t when the agent arrives at a new target point as the intrinsic rewards $r^i_{t_g}$.

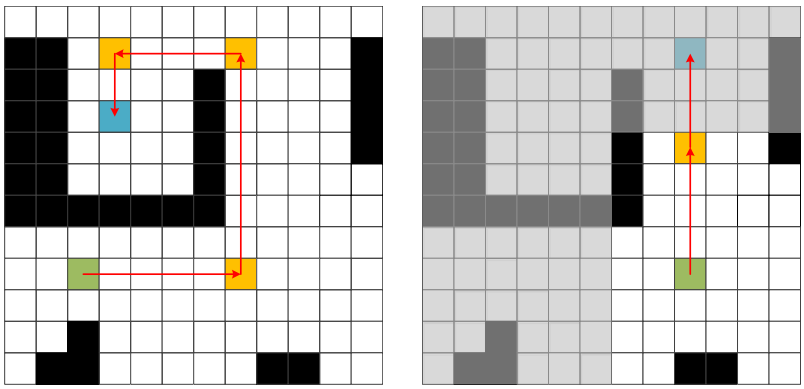
4.2. Local Movement Module

To be able to explore in dynamic spaces, the agent needs both to reach the target point quickly and avoid colliding with moving obstacles. To achieve this goal, we design another hierarchical framework in local movement module including two levels: planning and controlling. In the planning stage, we use optimistic A* algorithm to plan an optimal path under partial observability, and then divides the path into several segments according to some rules. The end point of each segment is called a key point. In the controlling stage, we design an SFVO (Self-adaptive FVO) for the agent to reach these key points sequentially, and finally completes the movement along the path.

4.2.1. Planning Stage

There are many global path planning algorithms such as breadth first search , depth first search and Dijkstra. Instead of using the less efficient Generalized Dijkstra's algorithm to solve the Shortest Path Problem (SPP) in [54], we use A* algorithm which has better search efficiency to plan the optimal global path. The basic A* algorithm performs well in fully observable environments, but it does not work directly in our task since the OM_t dose not reflect the whole map. So we use a variation of A*, called optimistic A* algorithm. We assume that all unknown cells of the obstacle map are traversable and then plan a path between the current position of the agent and the target point. If the agent observes some static obstacles while moving, then it will replan the path using A* algorithm.

Once an optimal path is computed, we select several key points on this path to guide the agent reach the target point. For the motion controller, presented below, to drive the agent to move between them. As shown in Figure 4, this paper categorizes three types of key points: (a) turning points on the path, (b) boundary points on the path that crosses the known and unknown region and (c) the destination point of the path, i.e., the target point.



(**a**) (**b**)

Figure 4. Examples of key points. The left figure shows the first and third types of key points, while the right figure shows the second and third types of key points. The green squares are the current locations of the agent. The blue squares represent the target points, which is also the third type of key points. The red lines represent the optimal path that was generated by A* algorithm and the orange squares are the first or second type of key points. The shaded area represents the unknown range of the agent while the other area represents the known range that has been observed by the agent.

Note that, the second type of key points are selected in the known area. Otherwise, if we select the boundary point in unknown area (the neighbour square above the orange square in Figure 4b, an obstacle might be selected as the key point.

In particular, the rationality of the selection strategy is that: (a) each segment of the path between key points is straight without considering dynamic obstacles, so that it is convenient for the controller to control the movement; (b) it is applicable to unknown spatial exploration problems under the partial observation conditions. We always choose the locations known for the agent as the key point, making its performance more similar to human exploration behaviour.

4.2.2. Controlling Stage

To avoid colliding with moving obstacles, we propose Self-adaptive Finite-time Velocity Obstacle algorithm (SFVO) built on FVO. Let A be the agent and B be an obstacle. For ease of calculation, we assume the agent and obstacles are dish-shaped. We use $D(\mathbf{p}, r)$ to represent a circular region with center $\mathbf{p}$ and radius r:

$$D(\mathbf{p}, r) = \{\mathbf{q} | \parallel \mathbf{q} - \mathbf{p} \parallel < r\} \tag{3}$$

The finite-time velocity obstacle $FVO^{\tau}_{A|B}$ represents the set of relative velocity values between A and B that will cause the collision in time τ in the future:

$$FVO^{\tau}_{A|B} = \{\mathbf{v}|\exists t \in [0,\tau], t\mathbf{v} \in D(\mathbf{p}_B - \mathbf{p}_A, r_A + r_B)\} \tag{4}$$

The collision-avoidance velocity (CA) of the agent is:

$$CA^{\tau}_{A|B}(V_B) = \{\mathbf{v}|\mathbf{v} \notin FVO^{\tau}_{A|B} \oplus V_B\} \tag{5}$$

According to the features of autonomous spatial exploration in dynamic environments, we change the fixed time τ into into adaptive dynamic time τ^d, i.e., $\tau^d_0 = \tau_{max}$ and $\tau^d(n) = \tau_{max} - \Delta\tau \cdot n$, n represents the number of rounds of a cycle, $\Delta\tau$ represents the reduction of finite time. This method is called Self-adaptive Finite-time Velocity Obstacle (SFVO). Figure 5 tells that the larger τ^d we set, the larger range of $FVO^{\tau^d}_{A|B} \oplus V_B$, and the smaller range of $CA^{\tau^d}_{A|B}(V_B)$. Therefore, we will adaptively adjust the velocity obstacle range of the agent by decreasing τ^d gradually. Specifically, at the beginning, the agent calculates the $CA^{\tau^d}_{A|B}(V_B)$ under the condition of $\tau^d = \tau_{max}$. If $CA^{\tau^d}_{A|B}(V_B) = \emptyset$, decrease the τ^d by a fix time interval $\Delta\tau$, and then calculate the collision-avoidance velocity again. If there is still no alternative velocity when $\tau^d = 0$, the agent stays idle until the next time step to continue the process above. The pseudo-code of SFVO is shown in Algorithm 1.

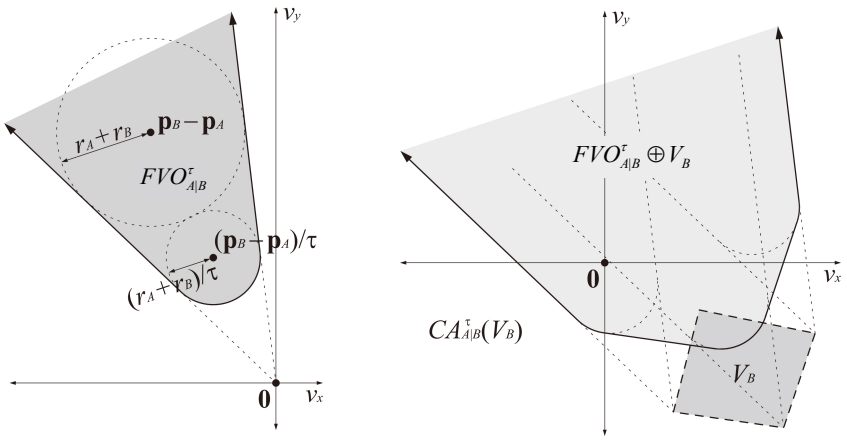


Figure 5. FVO algorithm diagram. In the left figure, the shaded area shows the relative velocity that will cause collision in time in the future. The right figure shows the collision velocity (shaded area) and collision-avoidance velocity (white area) of the agent given the velocity of the obstacle.

Algorithm 1 SFVO

$\tau^d \leftarrow \tau_{max}$
for $\tau^d > 0$ **do**
  $FVO^{\tau}_{A|B} = \{\mathbf{v}|\exists t \in [0,\tau], t\mathbf{v} \in D(\mathbf{p}_B - \mathbf{p}_A, r_A + r_B)\}$
  $CA^{\tau}_{A|B}(V_B) = \{\mathbf{v}|\mathbf{v} \notin FVO^{\tau}_{A|B} \oplus V_B\}$
  if $CA^{\tau^d}_{A|B}(V_B) = \emptyset$ **then**
    $\tau^d \leftarrow \tau^d - \Delta\tau$
    continue
  else
    return $CA^{\tau}_{A|B}(V_B)$
  end if
end for
return $\emptyset$

Based on the SFVO, the agent can avoid static and dynamic obstacles in real time, and its collision-avoidance velocity is $\mathbf{v}_A \in CA^{\tau^d}_{A|B}(V_B)$. However, if there is more than one element in set $\mathbf{v}_A$, how can we choose an optimal velocity that not only drives the agent to reach the target point quickly, but also minimizes the risk of collision.

Inspired by Kim et al. [55], we design an optimal velocity evaluation function which consists of two parts: Expected Velocity Direction Evaluation Function (f_v) and Relative Vertical Distance Evaluation Function (f_d). As shown in Figure 6. The target point of the agent is known, so the direction of its expected velocity ($\mathbf{v}_v$) is the direction from the agent's current position points to the target point. So, f_v can be expressed as Equation (6).

$$f_v = k_v|\mathbf{v}_v - \mathbf{v}_A| \Rightarrow k_v cos\langle \mathbf{v}_v, \mathbf{v}_A \rangle = k_v \frac{\mathbf{v}_v \cdot \mathbf{v}_A}{|\mathbf{v}_v||\mathbf{v}_A|} \tag{6}$$

Note that, the action space of the agent in our task is discrete, and the agent moves one unit at each step, i.e., the length of its velocity is fixed. So $cos\langle \mathbf{v}_v, \mathbf{v}_A \rangle$ is equivalent to $|\mathbf{v}_v - \mathbf{v}_A|$.

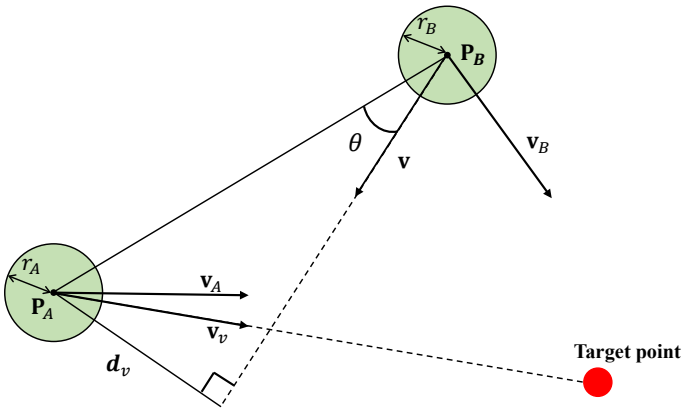


Figure 6. Schematic diagram of the expected velocity and the relative vertical distance. The green disks represent the agent and the obstacle, respectively, and the red point represents the target point.

The relative vertical distance (d_v) is defined as Equation (7).

$$d_v = |\mathbf{P}_A - \mathbf{P}_B| sin\theta = \frac{\mathbf{v} \times (\mathbf{P}_A - \mathbf{P}_B)}{|\mathbf{v}|} \tag{7}$$

$\mathbf{v}$ is the relative velocity of the agent and the obstacle, i.e., $\mathbf{v} = \mathbf{v}_B - \mathbf{v}_A$. A smaller d_v means that the obstacle is prone to collide with the agent. Note that d_v can be negative according to Equation (7), indicating that the obstacle is moving away from the agent and there is no danger for the agent. And the large $|d_v|$ is, the safer the agent is. According to the analysis, f_d is designed as Equation (8):

$$f_d = \begin{cases} -d_v & d_v \leq 0 \\ -\frac{1}{d_v} & d_v > 0 \end{cases} \tag{8}$$

Finally, an overall evaluation function can be defined to be a weighted sum of f_v and f_d, as shown in Equation (9). The importance of each part can be regulated by the weights k_1 and k_2.

$$f = k_1 f_v + k_2 f_d \tag{9}$$

When increasing k_1 and decreasing k_2, f_v dominates and the agent is more inclined to approach the target. In contrast, the agent takes priority in obstacle avoidance.

4.2.3. The Hybrid Algorithm

The optimistic A* algorithm can calculate the shortest global path in the partially known environment, but it cannot timely avoid the moving obstacles in dynamic environments. The SFVO with optimal velocity evaluation function can avoid collision in real time, but it lacks global guidance and has only one expected direction. More specifically, in a space with many moving obstacles, it is easy to fall into local minima, resulting in planning longer paths or even failing to plan. With this problem in mind, we combine the planning and controlling methods described above. The workflow of the hybrid algorithm is shown in Figure 7.

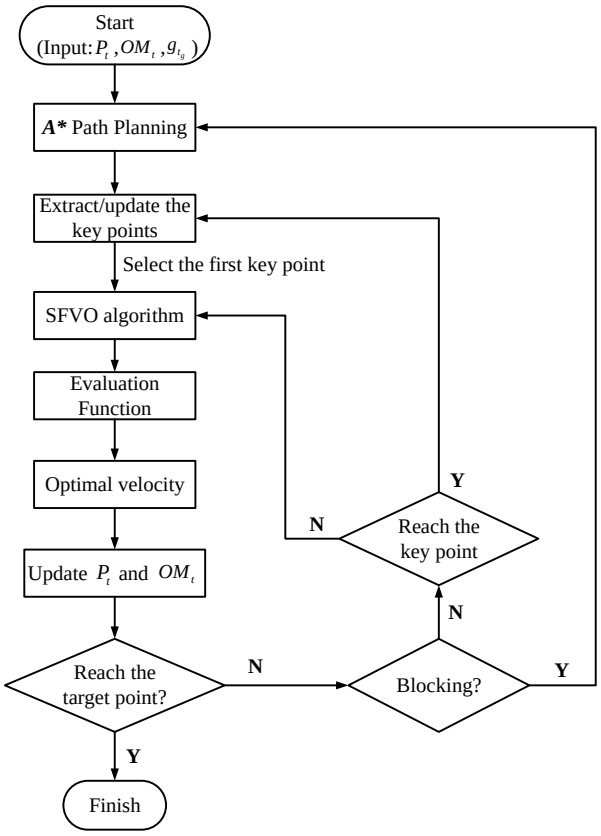


Figure 7. Flowchart of the hybrid algorithm.

At the time t, the algorithm puts the target point (g_{t_g}), which is produced by the Global Exploration Module P_t and OM_t into A* algorithm, plans a global path: $path_t = f_{A*}(P_t, g_{t_g}|OM_t)$. And extract the key points on the $path_t$: $\{K_1, K_2, ..., K_n\}$. Then, put the set of collision-avoidance velocities V_A which is calculated by the SFVO algorithm (Algorithm 1) $V_A = CA^{\tau}_{A|B}(V_B)$, the observation ($O_t$) of the agent and the sequence of key points into evaluation function (f), and calculate an optimal velocity of the agent $\mathbf{v}_{opt} \leftarrow f(V_A, K_1, K_2, ..., K_n|O_t, OM_t, P_t)$. Then, the agent moves one step at this velocity, and updates $OM_t \rightarrow OM_{t+1}$ and $P_t \rightarrow P_{t+1}$ at the same time. At time $t+1$, if there is an obstacle on $path_t$, i.e., $\exists(i,j) \in path_t, OM[i,j] = 1$, conduct the A* path planning again: $path_{t+1} = f_{A*}(P_{t+1}, g_{t_g}|OM_{t+1})$, and continue with the above process. Otherwise, $path_{t+1} = path_t$ and determine whether the agent has reached the key point K_1. If so, the sequence of key points is updated. Otherwise, the agent continues to use SFVO algorithms for local movement control. When the agent reaches the target point (g_{t_g}), the LMM stops running. Then the GEM chooses a new next target point (g_{t_g+1}).

In addition, to make the motion trajectory smoother and reduce unnecessary local oscillation, the agent can be regarded as reaching the key points as long as it reaches the eight adjacent cells around the key point.

These are all the functional modules of the IRHE-SFVO above. As we can see in Figure 2, we use the Global Policy in GEM for generating next target points which the agent will go to, then plan a best path between the current position and the target point and extract the key points in the planning stage. Then, in the controlling stage, we use the SFVO algorithm (Algorithm 1) and evaluation function (Equation (9)) to decide a specific action of the agent, and then update the current knowledge of the spatial structure which decides whether to replan the A* algorithm or update the key points sequence. We run the above functional modules sequentially until the exploration is completed.

5. Empirical Evaluation

The goal of this paper is to build agents that can autonomously explore novel 2D dynamic environments with moving obstacles. To verify the effectiveness of the proposed method, we implement mentioned components for performance evaluation.

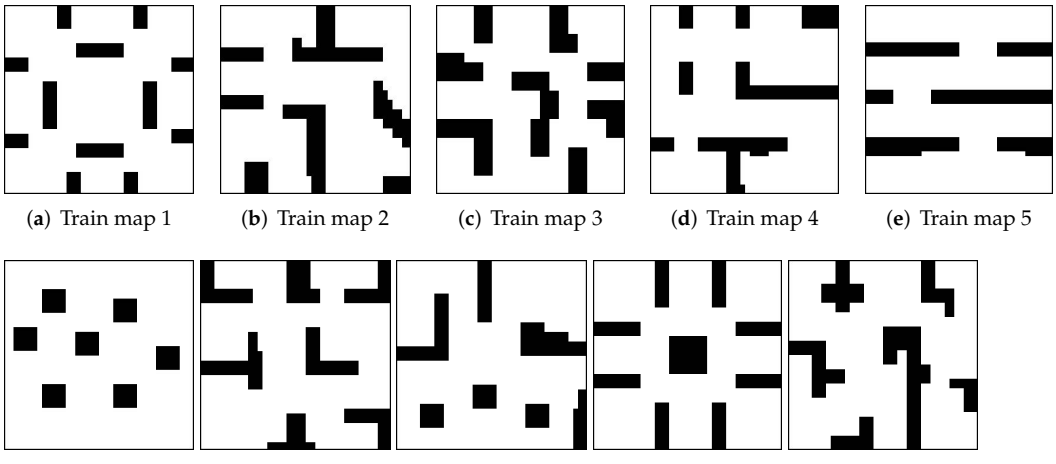
5.1. Experimental Setup

In order to evaluate the effect of the proposed model, we construct 2D grid maps by referring to reference [56] to represent the layout of indoor scenes such as offices. The maps are sized of $M = 40$, as shown in Figure 8. The first six maps make up the training set, and the rest are test maps. These maps have different spatial layouts and there is no intersection between the training set and the test set.

We use the ratio of the explored region as the metrics, which is calculated by dividing the coverage area by the total area that can be explored. It is defined as:

$$ExpRatio = \frac{C(EM_t)}{\sum_{x,y=0}^{x,y=M}(1 - T(x,y))} \tag{10}$$

where $C(EM_t)$ represents the total area that is explored.

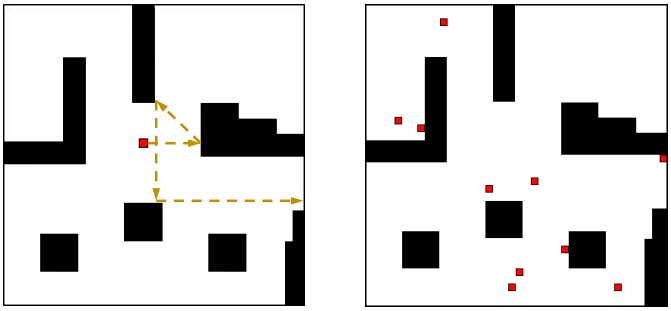


(**a**) Train map 1 (**b**) Train map 2 (**c**) Train map 3 (**d**) Train map 4 (**e**) Train map 5

(**f**) Train map 6 (**g**) Test map 1 (**h**) Test map 2 (**i**) Test map 3 (**j**) Test map 4

Figure 8. The different 2D grid maps without moving obstacles.

In these 2D grid maps, we set several obstacles which move independently of each other. The initial positions and moving directions of the moving obstacles are randomly selected, their movement mode is similar to that of the intelligent sweeping robots. As shown in Figure 9, the obstacles move straight until they collide with the obstacle or touch the boundary of the map, and then change the moving direction randomly.



(**a**) (**b**)

Figure 9. The diagram of the initial distribution of moving obstacles and their movement trajectory. The red cells represent the dynamic obstacles, and the yellow dotted line represents the possible movement trajectory of this obstacle in left figure. When the environment is initialized, moving obstacles are randomly generated in the blank area of the map, and each obstacle is independent of each other. The right figure shows the initialization of the moving obstacles.

To simplify the calculation, both the agent and the moving obstacle are regarded as circles with a radius of 0.5. Table 1 shows the parameters used in this experiment.

Table 1. Dynamic environment parameter details.

Parameter	Value
The weight/height of grid maps (M)	40
Number of moving obstacles (i)	10
Observation range of the agent (n)	5
Exploration range of the agent (m)	2
Physical radius of the agent (r_A)	0.5
Physical radius of the moving obstacles (r_B)	0.5
The maximum of finite time in SFVO (τ_{max})	2
The reduction of finite time in SFVO ($\Delta\tau$)	1
Total steps the agent moves (T)	800

Training Details. We use multi-process paralleled PPO to train the global policy in GEM, with a different process for each map. The hyperparameters of PPO and global policy network are shown in Tables 2 and 3, respectively.

Table 2. PPO hyperparameter details.

Hyperparameter	Value
Number of parallel environment	6
Number of minibatches	12
Number of episodes	100,000
Number of optimization epochs	4
Learning rate	0.0001
Optimization algorithm	Adam
Entropy coefficient	0.001
Value loss oefficient	0.5
λ	0.95
γ	0.99
ϵ/Clip range	0.1/[0.9, 1.1]
Max norm of gradients	0.5

Table 3. Global policy network details.

Layer	Parameters
Embedding	Size of embedding vector $= 16$
Conv1	Output $= 32$, Kernel $= 3$, Stride $= 1$, Padding $= 1$
Conv2	Output $= 64$, Kernel $= 3$, Stride $= 1$, Padding $= 1$
Conv3	Output $= 16$, Kernel $= 3$, Stride $= 1$, Padding $= 1$
MaxPool	Kernel size $= 2$
Linear1	Output size $= 64$
Linear2	Output size $= 32$

Baselines. We use some classical methods and end-to-end DRL methods as baselines, and all methods were tested 15 times on four test maps with random initial positions:

- RND-PPO: A popular IM based DRL approach. We adapt the source code from [3] to the problem settings in this paper. RND is a SOTA (state-of-the-art) DRL method based on prediction error, which has outstanding performance in Atari games. The network of PPO is similar to the proposed model, and an LSTM module [57] is added. The intrinsic discount factor $\gamma_i = 0.999$ and the other hyperparameters as the same as the proposed model. The target and prediction network consist of 3 fully connected layers and the learning rate of optimizing the prediction network $lr_{RND} = 0.0025$. In addition, we design an external reward that is given a negative reward (-10) when the agent collides with an obstacle or moves out of the map;
- Straight: This method is widely used in intelligent sweeping robots. It works by moving the agent in a straight line and performing a random turn when a collision will occur in next time step [58];
- Random: The agent takes a sequence of random actions to exploration.
- Frontier: A method which is based on geometric features to decided its next best frontier, drives the agent always goes to unknown spaces [59].

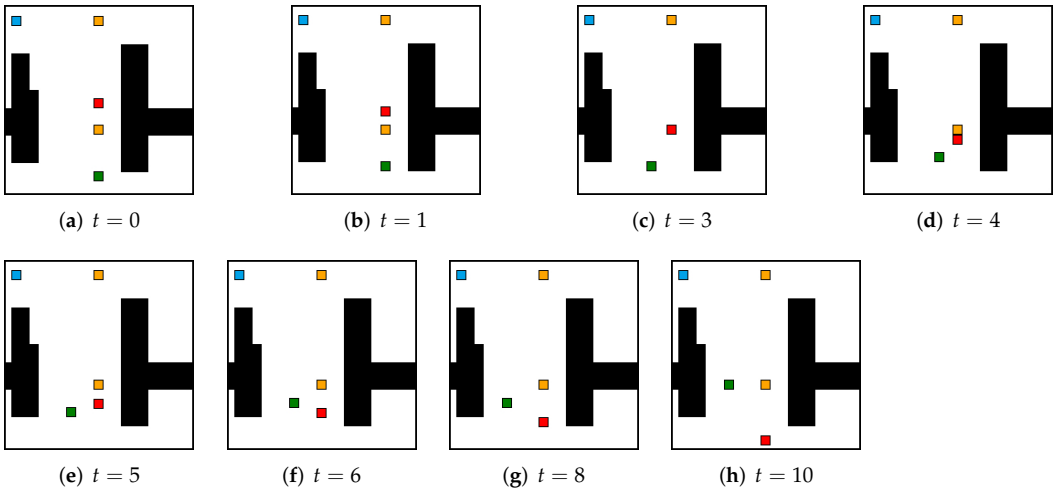
The RND-PPO is an end-to-end method, taking the observation as input and outputting a specific action of the agent. This kind of methods are hard to train for a desirable policy. Compared with RND-PPO and Random, the Straight is more stable, as it changes the velocity of the agent only when the agent will collide with an obstacle. Besides, the frontier-based method is also hierarchical as ours, whose workflow is still to select a position and then move to it, and we find that the SOTA DRL exploration methods are also difficult to achieve its performance in terms of exploration ratio [16].

5.2. Local Real-Time Obstacle Avoidance

We first verify the effectiveness of SFVO and the optimal velocity evaluation function for real-time obstacle avoidance, as shown in Figures 10–12. The green squares represent the positions of the agent at the current time, and the blue squares represent the target points. The red squares represent the moving obstacle with downward velocity, and the orange squares represent key points. The task of the agent is to reach the target point quickly while avoiding static and dynamic obstacles at the same time.

Because of the existence of key points and the four-direction (up, down, left and right) action space, each part of the path that between two key points forms in a straight line, so the agent has to move perpendicular to the line or in the opposite direction in order to avoid the moving obstacles. As a result, the expected velocity direction evaluation function f_v of the agent during obstacle avoidance is not greater than 0. Then, the agent completely depends on the relative vertical distance evaluation function f_d to select the optimal velocity. On other words, if the theory described above is correct, no matter how large k_2 is, it will always play a role in obstacle avoidance. Then, after obstacle avoidance, the agent needs to change its velocity to approach the key point, and the velocity selection

at this time completely depends on the expected velocity direction evaluation function f_v. That is to say, no matter how large k_1 is, it always plays a role in the velocity selection of approaching the key point after completing obstacle avoidance. Therefore, the combination of weighting coefficients in the evaluation function set in this experiment (Equation 9) is relatively single. We set three groups of different weighting coefficients: $k_1 = 0,\ k_2 = 1$; $k_1 = 1,\ k_2 = 0$; $k_1 = 1,\ k_2 = 1$. The trajectories of the agent under the three groups of coefficients are shown in Figures 10–12.

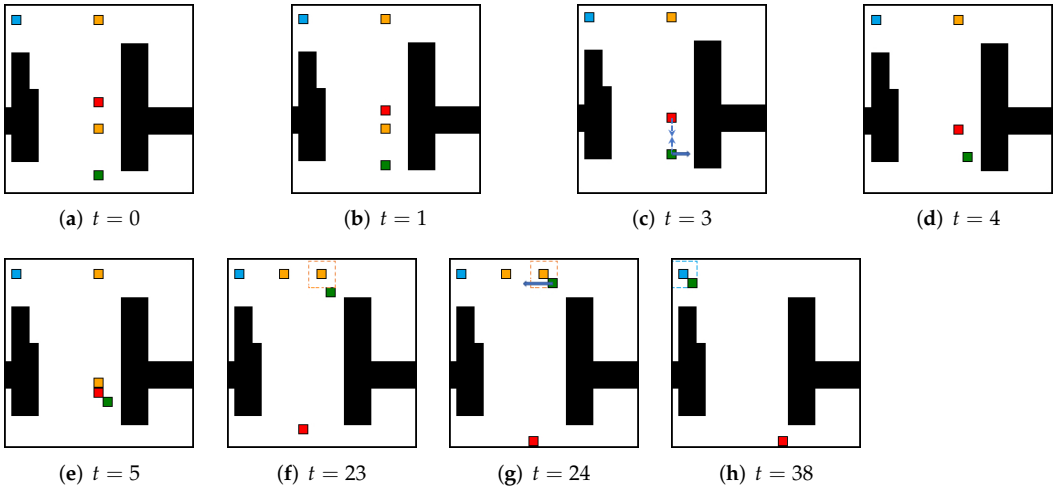


(**a**) $t = 0$ (**b**) $t = 1$ (**c**) $t = 3$ (**d**) $t = 4$

(**e**) $t = 5$ (**f**) $t = 6$ (**g**) $t = 8$ (**h**) $t = 10$

Figure 10. The trajectory of the agent under the condition of $k_1 = 0, k_2 = 1$ set in f.

As shown in Figure 10, when $k_1 = 0,\ k_2 = 1$, the moving trajectory of the agent is more and more away from the obstacle, but does not move towards key points. In essence, the agent is still taking random movements on the basis of avoiding collision with obstacles.



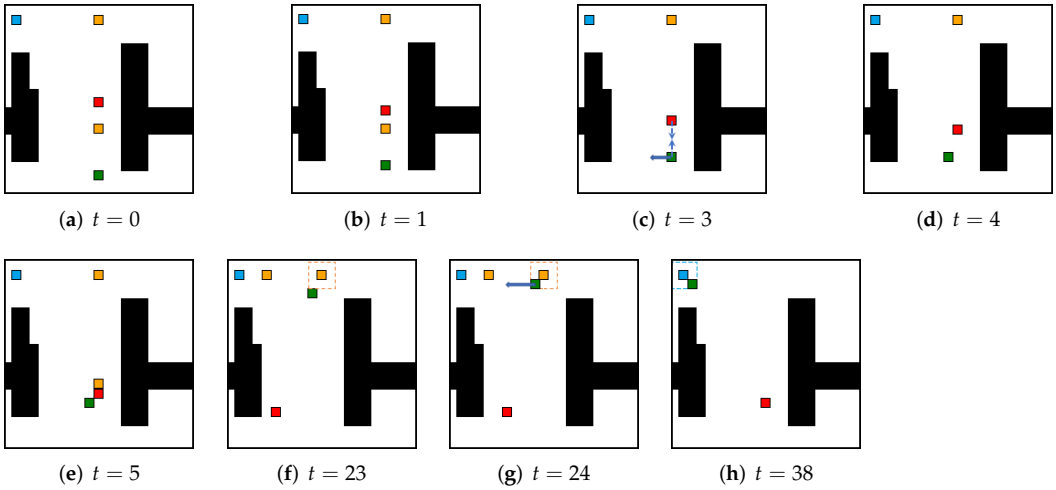
(**a**) $t = 0$ (**b**) $t = 1$ (**c**) $t = 3$ (**d**) $t = 4$

(**e**) $t = 5$ (**f**) $t = 23$ (**g**) $t = 24$ (**h**) $t = 38$

Figure 11. The trajectory of the agent under the condition of $k_1 = 1, k_2 = 0$ set in f.

When $k_1 = 1,\ k_2 = 0$, the agent will ignore the risk of collisions while moving. As shown in Figure 11, at $t = 3$, the agent judges that it will collide with the moving obstacle in the next two steps with the current motion direction through SFVO algorithm, so that it needs to make obstacle avoidance action. The agent is close to the static obstacle on the right and far from the one on the left. Therefore, the best obstacle avoidance action of the agent at this time should be to move left, which can avoid colliding with the moving obstacles and reduce the risk of collision with other obstacles, as shown in Figure 12. However, the agent does not consider the relative vertical distance d_v from the obstacle under the condition of

$k_1 = 1$, $k_2 = 0$, nd has a 50% probability of moving right, as shown in Figure 11, which increases the risk of collision with other obstacles.



(**a**) $t = 0$ (**b**) $t = 1$ (**c**) $t = 3$ (**d**) $t = 4$

(**e**) $t = 5$ (**f**) $t = 23$ (**g**) $t = 24$ (**h**) $t = 38$

Figure 12. The trajectory of the agent under the condition of $k_1 = 1, k_2 = 1$ set in f.

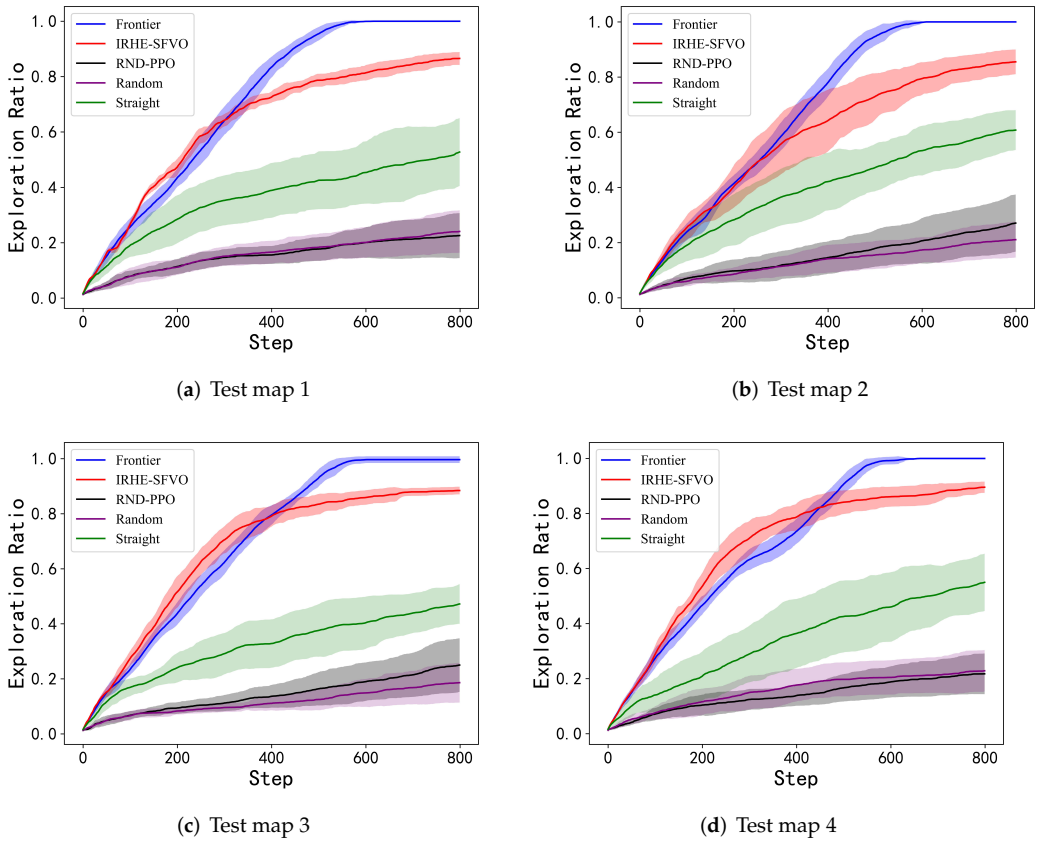
Figures 11 and 12 not only show the effectiveness of SFVO algorithm and evaluation function for obstacle avoidance, but also demonstrate the efficaciousness of the hybrid algorithm for path planning. At $t = 0$, the orange square which close to the agent is the second type of key points, the square that is far from the agent is the first type, and the blue square is the third type. When the agent reaches a key point or one of its adjacent eight squares, it is deemed to have reached the key point, so it continues to select the subsequent key points for local path planning. Finally, it guides the agent to the target point.

5.3. Comparison with Baselines on Spatial Exploration

We test the IRHE-SFVO with weighting coefficient $k_1 = 1$, $k_2 = 1$ of the evaluation function and compare it with the baselines on the test maps. The results are shown in Figure 13 and Table 4. It is worth noting that, from the perspective of safety, when the agent will collide with an obstacle, it should stop moving or change the direction immediately. However, the vanilla frontier-based strategy has no such specific design.

Table 4. The average exploration ratios of the proposed method and baselines on the four test maps. The brackets indicate the average number of times when the agent driven by the frontier-based strategy collides with moving obstacles in 15 spatial explorations on different test maps.

	IRHE-SFVO	**RND-PPO**	**Random**	**Straight**	**Fronteir**
Test map 1	0.8656	0.2258	0.2406	0.5276	0.9943 (4.53)
Test map 2	0.8552	0.2707	0.2107	0.6078	0.9992 (3.06)
Test map 3	0.8842	0.2501	0.1861	0.4721	0.9966 (5.13)
Test map 4	0.8953	0.2177	0.2287	0.5498	0.9997 (4.13)



(**a**) Test map 1 (**b**) Test map 2

(**c**) Test map 3 (**d**) Test map 4

Figure 13. Coverage Performance. Policies are tested in 800 steps (15 random start locations on each of the 4 testing maps). Darker line represents mean exploration ratio and shaded area represents the standard deviation across the 15 runs.

Figure 13 shows the coverage performance of different methods on the four test maps. Combining Table 4, we can see that the order of coverage from low to high is: Random and RND-PPO, Straight, IRHE-SFVO, Frontier. Specifically, we first notice that RND-PPO and Random method have similar poor performance. The reasons are as follows: the core of the dynamics-based curiosity agent such as RND is that if a state is encountered many times, its novelty will continue declining due to network parameter updates during training. So, in these grid maps, the agent needs to come out of its familiar area which is around the initial location at each episode. However, after several episodes of training, the "novelty" of the states around the initial position drop to a low level, and the agent does not touch the high-novelty world outside so that it is difficult for the agent to walk out of its familiar area. Second, the policy learned by the RND agent lacks the ability to explore since the intrinsic reward is to encourage agents to traverse more states rather than teaching the agent to learn how to explore. To be specific, its intrinsic reward will gradually decrease as training times due to the black-box model, so that the learned policy will depend more and more on the external reward, and the policy is dependent on external rewards completely at the end of training. In our experimental setting, RND only has a collision punishment as its external reward, so the agent moves randomly and only learns to avoid collision at the end. This is why the RND-PPO algorithm is slightly better than the Random algorithm.

Then, we notice that the Straight method performs much better than Random and RND-PPO because this method takes random actions only when a collision occurs. It is more stable than the Random and RND-PPO algorithms, which move randomly at every step.

Overall, these methods perform largely worse than IRHE-SFVO because our algorithm has an instructive high-level exploration strategy and an effective local movement module.

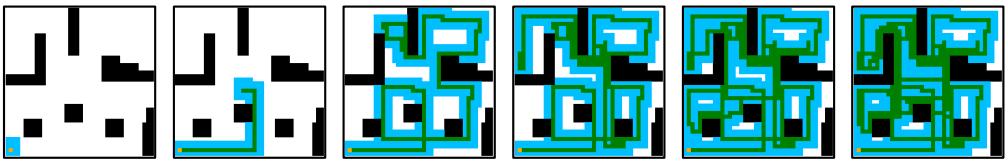
At the beginning of training GEM, the target points are selected almost randomly. As the training proceeds, the agent, driven by the intrinsic rewards which are generated by itself, gradually choose the points that are distant but easily reachable. Intrinsic reward, generated from intrinsic motivation, can make individual feel satisfied psychologically or emotionally because of increase of obtained knowledge or control [60]. In our exploration task, the intrinsic rewards are calculated by the increase of the explored area, so the agent will fill more cheerful when it chooses the point in an unknown region that is further away from it. Furthermore, the agent will reasonably adjust the distance between itself and the selected target point as the training progresses with the fixed number of target points that will be chosen during an exploration task. For example, the distances will be larger when the number of the target points is 10, while they will be smaller when the number of the target points is 20. In addition, LMM can adapt the agent to the planned path and avoid colliding with moving obstacles according to its perception, which can reach the target points quickly and safely.

Finally, we notice that the frontier-based strategy has achieved the highest exploration ratio in all the test maps. This method selects frontier points that lie on the boundary between the known free space and unknown region according to the maps built by the agent, and the experimental environments in this paper are very realistic, without any perceived noise or action errors, which is highly favourable to the frontier-based exploration agent. In the environment with moving obstacles, although this method may miss some "frontier points" at some time, resulting in that the spaces around them are not explored for a period of time. However, at a later time, this method can always select these "frontier points" again for spatial exploration. Because the motion trajectories of moving obstacles are random, it is impossible for them to stay at the positions where the frontier method always misjudges these "frontier points", so the exploration ratio of the frontier-based method is almost unaffected by dynamic obstacles. However, safety is a crucial problem that we must consider in spatial exploration. And Table 4 shows that the frontier-based method has collided with dynamic obstacles many times during exploration, while the others do not.

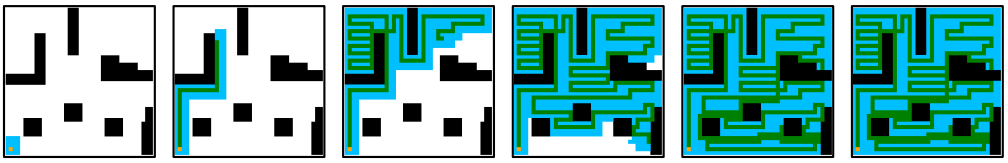
In addition, we visualize the paths and coverage areas of IRHE-SFVO and baselines on test map 2, the initial position of the agent is $(1, 1)$ on the bottom left. As shown in Figure 14, in each row of subfigures from left to right are the trajectories at step 0, step 40, step 200, step 400, Step 600 and step 800 respectively.

We can see that IRHE-SFVO algorithm covers almost all space, and its motion trajectory is relatively smoother and more reasonable than those of its competitors. It can be thought that it produces similar exploration strategies as human beings. Although frontier-based method has high exploration coverage, its motion trajectory is mechanical and very zigzag, such as the upper left corner and the blank area in the middle of the map. In addition, combined with Figure 13 we can also see that the exploration efficiency of IRHE-SFVO is slightly higher than that of the frontier in the initial exploration stage, because IRHE-SFVO aims to maximize the information gain of each step, but this is also the reason for its insufficient local exploration. Every time IRHE-SFVO selects a target point, it tends to select the locations where a large number of unexplored areas around it, so that the agent can quickly obtain a large amount of map information, but the exploration is insufficient for local details.

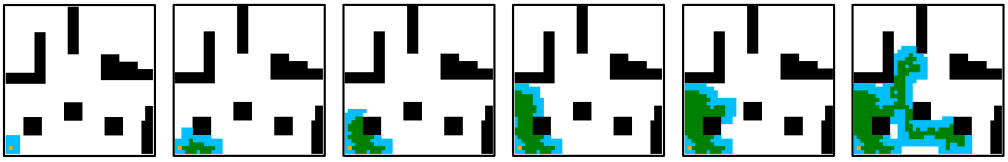
In summary, we use the DRL based on the intrinsic motivation to simulate the human high-level cognitive behavior during exploration, so that the agent always chooses those places that are particularly unknown to explore. And as for the quick and safe movement, we use a hierarchical framework including planning and controlling instead of learning methods that have difficulty in joint training to simulate the human low-level real-time response. Therefore, combining the two modules above, the IRHE-SFVO algorithm could meet the requirements of high efficiency and quasi-humanity of spatial exploration.



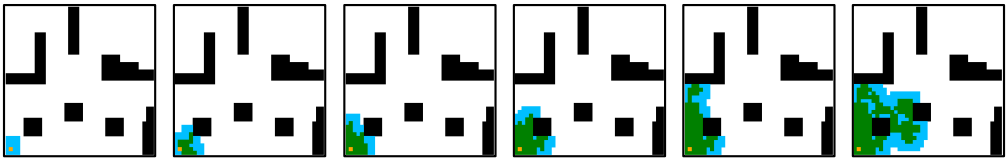
(**a**) The trajectory of IRHE-SFVO



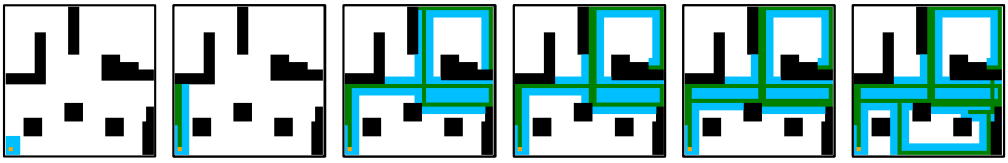
(**b**) The trajectory of Frontier



(**c**) The trajectory of RND-PPO



(**d**) The trajectory of Random



(**e**) The trajectory of Straight

Figure 14. Sample trajectories of the competing approaches along with the coverage region in Test map 2. The orange point represents the initial location of the agent. The solid green lines represent the trajectories that the agent has traversed and the blue shaded region shows the explored area.

6. Conclusions and Future Work

This paper proposed a three-tiered hierarchical autonomous spatial exploration model, IRHE-SFVO, that combines a high-level exploration strategy (GEM) and a low-level module (LMM) including a planning phase and a controlling phase. This decomposition not only overcomes the disadvantage of the end-to-end training difficulty, but also generates smooth movements which makes the behaviours of agent more reasonable and safely. We showed how to design and train these modules and validated them on multiple challenging 2D maps with complex structures and moving obstacles. The results showed that the proposed model has consistently better efficiency and generality than a state-of-the-art IM based DRL and some other heuristic methods. Although the proposed approach tends to revisit explored locations in some time, resulting in the lower coverage performance compared with frontier-based method, IRHE-SFVO still meets the application requirements to a certain extent.

For future work, we would like to extend this work to the following directions. First, in order to further improve the coverage of exploration, we would like to design more complex mechanisms like incorporating spatial abstraction into the framework to improve the efficiency of exploration and the rationality of motion mode. Second, more complex

constraints should be considered, such as uneven terrains, diverse surface features and the energy of the agent. Third, we would like to work on multi-agent collaborative spatial exploration, which faces the problems of non-stationary environments, incomplete observations and inefficient exploration of single agent in complex environments.

Author Contributions: Proposal and conceptualization, Q.Z. and Y.H.; methodology, Y.S.; implementation, Y.S. and P.J.; writing-original draft preparation, Y.S. and Y.H.; writing-review and editing, Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported partially by the National Natural Science Fund of China (Grant NO. 62102432, 62103420, 62103428 and 62103425) and the Natural Science Fund of Hunan Province (Grant NO. 2021JJ40697 and 2021JJ40702).

Institutional Review Board Statement: This study does not involve humans or animals.

Informed Consent Statement: This study does not involve humans or animals.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yamauchi, B. A frontier-based approach for autonomous exploration. In Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation', Monterey, CA, USA, 11–12 July 1997; pp. 146–151.
2. Song, Y.; Hu, Y.; Zeng, J.; Hu, C.; Qin, L.; Yin, Q. Towards Efficient Exploration in Unknown Spaces: A Novel Hierarchical Approach Based on Intrinsic Rewards. In Proceedings of the 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE), Dalian, China, 15–17 July 2021; pp. 414–422.
3. Burda, Y.; Edwards, H.; Storkey, A.; Klimov, O. Exploration by random network distillation. *arXiv prep.* **2018**, arXiv:1810.12894.
4. Wirth, S.; Pellenz, J. Exploration transform: A stable exploring algorithm for robots in rescue environments. In Proceedings of the 2007 IEEE International Workshop on Safety, Security and Rescue Robotics, Rome, Italy, 27–29 September 2007; pp. 1–5.
5. Mei, Y.; Lu, Y.H.; Lee, C.G.; Hu, Y.C. Energy-efficient mobile robot exploration. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, Orlando, FL, USA, 15–19 May 2006; pp. 505–511.
6. Juliá, M.; Gil, A.; Reinoso, O. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Auton. Robot.* **2012**, *33*, 427–444.
7. Oßwald, S.; Bennewitz, M.; Burgard, W.; Stachniss, C. Speeding-up robot exploration by exploiting background information. *IEEE Robot. Autom. Lett.* **2016**, *1*, 716–723.
8. Basilico, N.; Amigoni, F. Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. *Auton. Robot.* **2011**, *31*, 401–417.
9. Niroui, F.; Sprenger, B.; Nejat, G. Robot exploration in unknown cluttered environments when dealing with uncertainty. In Proceedings of the 2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), Ottawa, ON, Canada, 5–7 October 2017; pp. 224–229.
10. González-Banos, H.H.; Latombe, J.C. Navigation strategies for exploring indoor environments. *Int. J. Robot. Res.* **2002**, *21*, 829–848.
11. Whaite, P.; Ferrie, F.P. Autonomous exploration: Driven by uncertainty. *IEEE Trans. Pattern Anal. Mach. Intell.* **1997**, *19*, 193–205.
12. Julian, B.J.; Karaman, S.; Rus, D. On mutual information-based control of range sensing robots for mapping applications. *Int. J. Robot. Res.* **2014**, *33*, 1375–1392.
13. Tai, L.; Liu, M. A robot exploration strategy based on q-learning network. In Proceedings of the 2016 IEEE International Conference on Real-Time Computing and Robotics (RCAR), Angkor Wat, Cambodia, 6–10 June 2016; pp. 57–62.
14. Zhang, J.; Tai, L.; Liu, M.; Boedecker, J.; Burgard, W. Neural slam: Learning to explore with external memory. *arXiv Prep.* **2017**, arXiv:1706.09520.
15. Zhu, D.; Li, T.; Ho, D.; Wang, C.; Meng, M.Q.H. Deep reinforcement learning supervised autonomous exploration in office environments. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 7548–7555.
16. Chen, T.; Gupta, S.; Gupta, A. Learning exploration policies for navigation. *arXiv Prep.* **2019**, arXiv:1903.01959.
17. Issa, R.B.; Rahman, M.S.; Das, M.; Barua, M.; Alam, M.G.R. Reinforcement Learning based Autonomous Vehicle for Exploration and Exploitation of Undiscovered Track. In Proceedings of the 2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 7–10 Janaury 2020; pp. 276–281.
18. Niroui, F.; Zhang, K.; Kashino, Z.; Nejat, G. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robot. Autom. Lett.* **2019**, *4*, 610–617.
19. Shrestha, R.; Tian, F.P.; Feng, W.; Tan, P.; Vaughan, R. Learned map prediction for enhanced mobile robot exploration. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 1197–1204.

20. Li, H.; Zhang, Q.; Zhao, D. Deep reinforcement learning-based automatic exploration for navigation in unknown environment. *IEEE Trans. Neural Net. Learn. Syst.* **2019**, *31*, 2064–2076.
21. Chaplot, D.S.; Gandhi, D.; Gupta, S.; Gupta, A.; Salakhutdinov, R. Learning to explore using active neural slam. *arXiv Prep.* **2020**, arXiv:2004.05155.
22. Barto, A.; Mirolli, M.; Baldassarre, G. Novelty or surprise? *Front. Psychol.* **2013**, *4*, 907.
23. Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; Munos, R. Unifying count-based exploration and intrinsic motivation. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 1471–1479.
24. Ostrovski, G.; Bellemare, M.G.; Oord, A.; Munos, R. Count-based exploration with neural density models. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, NSW, Australia, 6–11 August 2017; pp. 2721–2730.
25. Tang, H.; Houthooft, R.; Foote, D.; Stooke, A.; Chen, X.; Duan, Y.; Schulman, J.; De Turck, F.; Abbeel, P. A study of count-based exploration for deep reinforcement learning. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017; pp. 1–18.
26. Fu, J.; Co-Reyes, J.D.; Levine, S. Ex2: Exploration with exemplar models for deep reinforcement learning. *arXiv Prep.* **2017**, arXiv:1703.01260.
27. Choi, J.; Guo, Y.; Moczulski, M.; Oh, J.; Wu, N.; Norouzi, M.; Lee, H. Contingency-aware exploration in reinforcement learning. *arXiv Prep.* **2018**, arXiv:1811.01483.
28. Machado, M.C.; Bellemare, M.G.; Bowling, M. Count-based exploration with the successor representation. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 Feburary 2020; pp. 5125–5133.
29. Choshen, L.; Fox, L.; Loewenstein, Y. Dora the explorer: Directed outreaching reinforcement action-selection. *arXiv Prep.* **2018**, arXiv:1804.04012.
30. Shyam, P.; Jaśkowski, W.; Gomez, F. Model-based active exploration. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 5779–5788.
31. Pathak, D.; Gandhi, D.; Gupta, A. Self-supervised exploration via disagreement. In Proceedings of the International conference on Machine Learning, PMLR, Los Angeles, CA, USA, 9–15 June 2019; pp. 5062–5071.
32. Ratzlaff, N.; Bai, Q.; Fuxin, L.; Xu, W. Implicit generative modeling for efficient exploration. In Proceedings of the International Conference on Machine Learning, PMLR, Vienne, Austria, 12–18 July 2020; pp. 7985–7995.
33. Stadie, B.C.; Levine, S.; Abbeel, P. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv Prep.* **2015**, arXiv:1507.00814.
34. Pathak, D.; Agrawal, P.; Efros, A.A.; Darrell, T. Curiosity-driven exploration by self-supervised prediction. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–12 August 2017; pp. 2778–2787.
35. Kim, H.; Kim, J.; Jeong, Y.; Levine, S.; Song, H.O. Emi: Exploration with mutual information. *arXiv Prep.* **2018**, arXiv:1810.01176.
36. Ermolov, A.; Sebe, N. Latent World Models For Intrinsically Motivated Exploration. *arXiv Prep.* **2020**, arXiv:2010.02302.
37. Lopes, M.; Lang, T.; Toussaint, M.; Oudeyer, P.Y. *Exploration in Model-Based Reinforcement Learning by Empirically Estimating Learning Progress*; Neural Information Processing Systems (NIPS): Lake Tahoe, NV, USA, 2012.
38. Gregor, K.; Rezende, D.J.; Wierstra, D. Variational intrinsic control. *arXiv Prep.* **2016**, arXiv:1611.07507.
39. Houthooft, R.; Chen, X.; Duan, Y.; Schulman, J.; De Turck, F.; Abbeel, P. Vime: Variational information maximizing exploration. *arXiv Prep.* **2016**, arXiv:1605.09674.
40. Oudeyer, P.Y.; Kaplan, F. What is intrinsic motivation? A typology of computational approaches. *Front. Neurorobotics* **2009**, *1*, 6.
41. Badia, A.P.; Piot, B.; Kapturowski, S.; Sprechmann, P.; Vitvitskyi, A.; Guo, Z.D.; Blundell, C. Agent57: Outperforming the atari human benchmark. In Proceedings of the International Conference on Machine Learning, PMLR, Vienne, Austria, 12–18 July 2020; pp. 507–517.
42. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*; Springer: Berlin/Heidelberg, Germany, 1986; pp. 396–404.
43. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33.
44. Rezaee, H.; Abdollahi, F. A decentralized cooperative control scheme with obstacle avoidance for a team of mobile robots. *IEEE Trans. Ind. Electron.* **2013**, *61*, 347–354.
45. Ali, F.; Kim, E.K.; Kim, Y.G. Type-2 fuzzy ontology-based semantic knowledge for collision avoidance of autonomous underwater vehicles. *Inf. Sci.* **2015**, *295*, 441–464.
46. Cheng, Y.; Zhang, W. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing* **2018**, *272*, 63–73.
47. Celsi, L.R.; Celsi, M.R. On Edge-Lazy RRT Collision Checking in Sampling-Based Motion Planning. *Int. J. Robot. Autom.* **2021**, *36*. doi:10.2316/J.2021.206-0551.
48. Fiorini, P.; Shiller, Z. Motion planning in dynamic environments using velocity obstacles. *Int. J. Robot. Res.* **1998**, *17*, 760–772.
49. Van den Berg, J.; Lin, M.; Manocha, D. Reciprocal velocity obstacles for real-time multi-agent navigation. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 1928–1935.
50. Abe, Y.; Yoshiki, M. Collision avoidance method for multiple autonomous mobile agents by implicit cooperation. In Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180), IEEE, Maui, HI, USA, 29 October–3 November 2001; pp. 1207–1212.

51. Guy, S.J.; Chhugani, J.; Kim, C.; Satish, N.; Lin, M.; Manocha, D.; Dubey, P. Clearpath: Highly parallel collision avoidance for multi-agent simulation. In Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, New Orleans, Louisiana, 1–2 August 2009; pp. 177–187.
52. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv Prep.* **2017**, arXiv:1707.06347.
53. Conti, E.; Madhavan, V.; Such, F.P.; Lehman, J.; Stanley, K.O.; Clune, J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *arXiv Prep.* **2017**, arXiv:1712.06560.
54. Celsi, L.R.; Di Giorgio, A.; Gambuti, R.; Tortorelli, A.; Priscoli, F.D. On the many-to-many carpooling problem in the context of multi-modal trip planning. In Proceedings of the 2017 25th Mediterranean Conference on Control and Automation (MED), Valletta, Malta, 3–6 July 2017; pp. 303–309.
55. Kim, M.; Oh, J.H. Study on optimal velocity selection using velocity obstacle (OVVO) in dynamic and crowded environment. *Auton. Robot.* **2016**, *40*, 1459–1470.
56. Hu, Y.; Subagdja, B.; Tan, A.H.; Yin, Q. Vision-Based Topological Mapping and Navigation with Self-Organizing Neural Networks. *IEEE Trans. Neural Net. Learn. Syst.* **2021**. Available online: https://ieeexplore.ieee.org/document/9459468 (accessed on 20 December 2021).
57. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780.
58. Savva, M.; Kadian, A.; Maksymets, O.; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; et al. Habitat: A platform for embodied ai research. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 9339–9347.
59. Mobarhani, A.; Nazari, S.; Tamjidi, A.H.; Taghirad, H.D. Histogram based frontier exploration. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 1128–1133.
60. Baldassarre, G. What are intrinsic motivations? A biological perspective. In Proceedings of the 2011 IEEE International Conference on Development and Learning (ICDL), Frankfurt am Main, Germany, 24–27 August 2011; Volume 2, pp. 1–8.