electronics

MDPI

*Article*

# Intrinsic Motivation Based Hierarchical Exploration for Model and Skill Learning

**Lina Lu, Wanpeng Zhang *, Xueqiang Gu and Jing Chen**

College of Intelligence Science and Technology, National University of Defense Technology, Changsha, Hunan, 410073, China; lulina16@nudt.edu.cn (L.L.); xqgu_nudt@163.com (X.G.); chenjing001@vip.sina.com (J.C.)

* Correspondence: wpzhang@nudt.edu.cn

Received: 6 January 2020; Accepted: 8 February 2020; Published: 11 February 2020

**Abstract:** Hierarchical skill learning is an important research direction in human intelligence. However, many real-world problems have sparse rewards and a long time horizon, which typically pose challenges in hierarchical skill learning and lead to the poor performance of naive exploration. In this work, we propose an algorithmic framework called surprise-based hierarchical exploration for model and skill learning (Surprise-HEL). The framework leverages the surprise-based intrinsic motivation for improving the efficiency of sampling and driving exploration. It also combines the surprise-based intrinsic motivation and the hierarchical exploration to speed up the model learning and skill learning. Moreover, the framework incorporates the reward independent incremental learning rules and the technique of alternating model learning and policy update to handle the changing intrinsic rewards and the changing models. These works enable the framework to implement the incremental and developmental learning of models and hierarchical skills. We tested Surprise-HEL on a common benchmark domain: Household Robot Pickup and Place. The evaluation results show that the Surprise-HEL framework can significantly improve the agent's efficiency in model and skill learning in a typical complex domain.

**Keywords:** intrinsic motivation; robot intelligence; model learning; skill learning; reinforcement learning

## 1. Introduction

In the field of psychology and neuroscience, one of the core objectives is to understand the formal structure of behavior [1,2]. It has been shown that the behavior displays a hierarchical structure in humans and animals. Simple actions are combined into coherent subtasks, and these subtasks are further combined to achieve higher-level goals [3]. The hierarchical structures are observed readily in our daily life: opening a faucet is the first step of the task 'cleaning vegetables', which is part of the bigger task 'cooking a dish'. The detailed formal analysis of the hierarchical structure of behavior can be found in Whiten's paper [4].

Hierarchical structure-based learning and planning are one of the most important research fields in artificial intelligence. Compared with flat representation, hierarchical representation is more efficient and compact, which is computationally simpler for encoding the complex behaviors at the neural level [5]. Moreover, the hierarchical representation can accelerate the discovery of new hierarchical behaviors through planning and learning [6–9].

Exploration is the fundamental of hierarchical learning and planning. However, exploration in environments with sparse rewards is challenging, as exploration over the space of primitive action sequences is unlikely to result in a reward signal. For humans, the problem of sparse rewards is solved by intrinsic motivation based exploration [10–12]. Intrinsic motivation is defined as the doing of an activity for its own sake, rather than to directly solve some specific problems [10]. When people

are intrinsically motivated, they act for fun or challenge, not for external products, stress or rewards [11]. Intrinsic motivation based exploration can explore the environment and discover hierarchical skills for its own sake. Moreover, the learned skills will be reused to solve more complex and unfamiliar problems in later life [12]. In artificial intelligence, such intrinsic motivation based exploration is efficient and scalable to facilitate skill learning, and it can eventually help the agent solve the problems posed by the environment in their lifelong time.

In this work, we adopted surprise as an effective and robust intrinsic motivator to drive exploration. Surprise has played crucial roles in human behavior. The theories of intrinsic motivation propose that surprise is one of the main factors that motivate exploration, arouse interest, and drive learning [13]. There is a comprehensive agreement that surprise is an emotion caused by a mismatch between actual observations or experiences and expectations [14]. In this paper, we employed the dictionary definition of "surprise" as "an agent is excited to see outcomes that run contrary to its understanding of the world".

Moreover, we formulated the surprise with the divergence between the true dynamic transition probabilities and the learned transition probabilities, in other words, the relative-entropy of the true transition model with respect to the learned model. Furthermore, we employed a function of the surprise to reshape intrinsic rewards. In the learning process, we used the learned skills to bootstrap the learning of new skills, not only achieving incremental skill learning, but also speeding up model learning and skill learning. In this framework, the incremental skill learning enables agents to solve harder problems with less cognitive effort over time, which is essential in many complex real-world tasks such as rescue robots in cluttered and unknown urban search and rescue (USAR).

The main contributions of this paper are as follows:

1. In the MDP (Markov decision process) task environment, we investigate the incentive mechanisms based on intrinsic motivation, and formalize the surprise-based intrinsic motivation with information theory. The framework leverages the intrinsic motivation both for improving sample efficiency and driving exploration. We demonstrate empirically that the surprise-based intrinsic motivation can result in more efficient exploration.

2. We integrate the surprise-based intrinsic motivation and hierarchical exploration into the framework for the first time. This work not only improves the efficiency of model learning and skill learning, but also improves the ability of incremental learning of more complex skills.

3. The framework of Surprise-HEL incorporates the reward independent incremental learning rules and the technique of alternating model learning and policy update. This work enables the framework to not only handle changing intrinsic rewards and changing models, but to also implement the incremental and developmental learning of models and hierarchical skills.

The rest of this paper is organized as follows. Section 2 describes the related work. Section 3 introduces the necessary background of hierarchical skill learning and information theory. Section 4 describes the framework of surprise-based hierarchical exploration for model and skill learning in detail. Section 5 evaluates the performance in the domain of household robot. Section 6 concludes and discusses future research.

## 2. Related work

Intrinsic motivation has many forms of expression including innate preferences, empowerment [15–17], novelty [18–20], surprise [21,22], predictive confidence, habituation [23], and so on [13,14]. Mohamed et al. [17] used mutual information to form the definition of an internal drive, which is called empowerment. Christopher et al. [12] employed the function of an agent's confidence to formulate intrinsic rewards. Burda et al. [24] performed the study of purely curiosity-driven learning across 54 standard benchmark environments. Singh et al. [25] adopted an evolutionary perspective to optimize the reward framework, and designed a primary reward function by capturing the pressure, which leads to a notion of intrinsic and extrinsic motivation.

In this work, we used intrinsic motivation to drive exploration and skill learning. Substantial works have been done in this field.

Kulkarni et al. [26] presented a hierarchical-DQN (h-DQN) framework. The framework integrated hierarchical action-value functions and intrinsic motivation. The focus of their work is to make a decision at different temporal scales, but the intrinsic motivation is independent of the current performance of the agent.

McGovern et al. [27] and Menache [28] searched for states that act as bottlenecks to generate skills. Tomar et al. [29] used successor representation to generalize the bottleneck approach to continuous state space. These works mainly focus on building options, rather than improving skill learning performance from intrinsic motivation based exploration.

Achiam et al. [22], Alemi et al. [30], and Klissarov et al. [31] used the approximation of KL-divergence to form intrinsic rewards. Frank et al. [32] empirically demonstrated that artificial curiosity exists in humanoid robots and is capable of exploring the environment through information gain maximization. Fu et al. [33] proposed a discriminator to differentiate states from each other in order to judge whether a state was sufficiently visited. Still et al. [34] defined a curiosity-based, weighted-random exploration mechanism. This exploration mechanism enabled agents to investigate unvisited regions iteratively. Emigh et al. [35] proposed a framework called Divergence-to-Go to quantify the uncertainty of each action-state pair. These works are inefficient in problems with large state and action space due to the underlying models functioning at the level of primitive actions.

In this work, our framework combines the intrinsic motivation-driven exploration and hierarchical exploration to accelerate model learning and hierarchical skill learning. Moreover, the framework motivates the agent to reuse the learned skills to improve its capabilities.

## 3. Background

### *3.1. Semi-Markov Decision Process (SMDP)*

The semi-Markov decision process (SMDP) is the mathematical model of the hierarchical methods, and provides the theoretical foundation for hierarchical reinforcement learning.

SMDP is a generalization of the Markov decision process (MDP). MDP is defined as a four-tuple $(S, A, R, P)$, where $S$ is the set of states, $A$ is the set of actions, $P$ is the transition function, and $R$ is the reward function. $P(s'|s,a)$ is the transition probability of executing the action $a$ in the state $s$ and terminating at the state $s'$, and $R(s'|s,a)$ is the reward function for transitioning from $s$ to $s'$ under the action $a$. $\pi : S \times A \to [0,1]$ is the stationary stochastic policy. $\pi(a|s)$ is the probability of selecting action $a$ in state $s$.

An SMDP extends an MDP, in which the action can take multiple time steps [36,37]. Let the random variable $k \in \mathbb{N}+$ denote the number of time steps that action $a$ executes from state $s$ and terminates at state $s'$. The state transition probability function can be extended to the joint probability distribution $P(s',k|s,a)$, where action $a$ executes from state $s$ and terminates at state $s'$ after $k$ time steps. The reward function is changed from $R(s'|s,a)$ to $R(s',k|s,a)$ accordingly.

### *3.2. Option*

There are three general approaches to formalize hierarchical skills: option [38], MAXQ [36], and HAM [39,40]. In this paper, we worked with the framework of option (temporally extended actions) in skill learning.

An option is a short-term skill that consists of a policy for the state space in a specified region and a termination function for leaving the region. Formally, an option $o$ is defined as a three-tuple $o = \langle I, \pi, \beta \rangle$, where $I \subseteq S$ is an initiation state set, $\beta : S \to [0,1]$ is a termination condition function, and $\pi$ is an intra-policy. The action space of the intra-policy contains primitive actions and low-level options. If an option is selected, then this option will be executed according to $\pi$ until its termination.

The low-level options can be regarded as the primitive actions embedded within a high-level option. Moreover, an option can also be regarded as a sub-MDP embedded in a larger MDP. Therefore, all the mechanisms associated with MDP learning also apply to the learning options [41].

### *3.3. Information Theory*

**Entropy** (Information Entropy) Entropy measures uncertainty between random variables. The entropy $H(X)$ of the discrete random variable $X$ is defined as:

$$H(X) = -\sum_{x \in \chi} q(x) \log q(x) \tag{1}$$

where $\chi$ is the value space of random variable $X$ and $q(x) = \Pr\{X = x\}, x \in \chi$ is the probability mass function [42]. Moreover, the entropy of the random variable $X$ can also be interpreted as the expected value of the random variable $\log \frac{1}{q(x)}$:

$$H(X) \overset{def}{=} H(q) = \mathbb{E}_q \log \frac{1}{q(x)} \tag{2}$$

The entropy has the following property: $H(q) \geq 0$ $H(q) = 0$ if and only if $q$ is deterministic, and $0 \log 0 = 0$ since $x \log x \to 0$ as $x \to 0$.

**Relative-Entropy** The relative entropy $\mathrm{D}(q \parallel p)$ is a measure of the inefficiency of assuming that the distribution is $p$ when the true distribution is $q$ [42]. The relative entropy (Kullback–Leibler divergence) between two probability mass functions $q(x)$ and $p(x)$ is defined as

$$\begin{aligned} \mathrm{D}_{KL}(q \parallel p) &= \sum_{x \in \chi} q(x) \log \frac{q(x)}{p(x)} \\ &= \mathbb{E}_q \log \frac{q(x)}{p(x)} \end{aligned} \tag{3}$$

In the definition, it uses the convention that $0 \log \frac{0}{0} = 0$ and $0 \log \frac{0}{p} = 0$ $q \log \frac{q}{0} = \infty$. The relative entropy has the following property: $\mathrm{D}_{KL}(q \parallel p)$ is non-negative and $\mathrm{D}_{KL}(q \parallel p) = 0$ if and only if $p = q$. $\mathrm{D}_{KL}(q \parallel p)$ is not in general symmetric: $\mathrm{D}_{KL}(q \parallel p) \neq \mathrm{D}_{KL}(p \parallel q)$.

## 4. Surprise-Based Hierarchical Exploration for Model and Skill Learning

### *4.1. Surprise-Based Intrinsic Motivation*

Surprise is one of the intrinsic motivations that arouse interest and drive learning. In this section, we formalized the surprise-based intrinsic motivation with information theory, and used it to weigh the tradeoff between exploration and exploitation.

The surprise is used to measure the uncertainty of a state-action pair. Let surprise denote a measure for the exploration and learning progress when visiting a state-action pair. The surprise changes with the new experience. That is, as the agent continues to explore the environment, the true transition model $Q$ and the learned transition model $P$ in the context of $(s,a)$ gets closer, and the surprise gets smaller.

Exploration and exploitation are two different ways of acting, and is an inherent challenge in reinforcement learning problems [43]. Exploitation is about using the current information to make the best decisions, and exploration is about collecting more information. The tradeoff between exploration and exploitation is that the best long-term policy may include short-term sacrifices to gather enough information to make the best overall decision.

The update step of surprise-based exploration policy makes progress on an approximation to the optimization problem:

$$\max V_{\pi} + \eta \underset{s,a\sim\pi}{\mathbb{E}}[D_{KL}(Q \,\|\, P)] \tag{4}$$

where $V_{\pi}$ is the performance measure, and $\eta > 0$ is an exploration–exploitation trade-off coefficient. The left-hand term is to maximize exploitation, and the right-hand term is to drive exploration. In the right-hand term, we formalized surprise in the context of $(s,a)$ with the divergence between the true transition model $Q$ and the learned transition model $P$, in other words, the relative-entropy of $Q$ with respect to $P$.

$$\begin{aligned} Surprise(s,a) &= \underset{s,a}{\Lambda}(Q,P) \\ &= \underset{s,a}{\mathrm{D}_{KL}}(Q \,\|\, P) \\ &= \underset{s'\sim Q(\cdot|s,a)}{\mathbb{E}} \log\frac{Q(\mathrm{s},\mathrm{a})}{P(\mathrm{s},\mathrm{a})} \quad (P(\mathrm{s},\mathrm{a}) \neq 0) \end{aligned} \tag{5}$$

Equation (5) shows that in regions of the transition state space, the more times the region is visited, the closer *P* to *Q* are in this region. In other words, the surprise of *P* and *Q* is more significant in unfamiliar regions, and tends to be 0 in familiar regions.

The exploration incentive in Equations (4) and (5) is intended to minimize the agent's surprise about its experience. If the surprise is significant, it implies that the agent's learned model of the environment was insufficient to predict the true environment dynamics and the state transition. The region should be repeatedly investigated until the surprise decreases. If the surprise is low, the learned transition model is approximating the true transition model well. The agent understands the part of the environment well, thus it can move on to explore more uncertain parts to acquire more knowledge and improve its performance.

According to the surprise incentive, we proposed the reshaped form of intrinsic reward as follows:

$$\begin{aligned} r'_{\mathrm{int}}(\mathrm{s},\mathrm{a}) &= r_{\mathrm{int}}(\mathrm{s},\mathrm{a}) - \alpha Surprise(s,a) \\ &= r_{\mathrm{int}}(\mathrm{s},\mathrm{a}) - \alpha\left[\underset{s'\sim Q(\cdot|s,a)}{\mathbb{E}} \log\frac{Q(\mathrm{s},\mathrm{a})}{P(\mathrm{s},\mathrm{a})}\right] \quad (P(\mathrm{s},\mathrm{a}) \neq 0) \end{aligned} \tag{6}$$

where $\alpha \in (0,1)$ is a positive step-size parameter; $r(\mathrm{s},\mathrm{a})$ is the original reward; and the initial values of intrinsic rewards of all state-action pairs are a constant (e.g., 1). The intrinsic rewards will gradually decrease in the learning process, and the rate of decline is gradually decreasing. Here, the intrinsic rewards are the surprise in the context of $(s,a)$. As above-mentioned, the intrinsic reward in unfamiliar regions will be higher than that of familiar regions. Moreover, after full exploration, the variance between the true model and the learned model is so small that the intrinsic reward will decrease to zero. Essentially, this encourages the agent to select less-selected actions in the current state to go where it is unfamiliar.

However, the intrinsic reward (Equation (6)) cannot be implemented directly, and the true transition model *Q* is unknown. In order to solve this problem, we introduce the following theorems.

**Theorem 1** (*Jensen's inequality* [42]). If $f$ is a convex function and $X$ is a random variable,

$$\mathbb{E}f(X) \geq f(\mathbb{E}X) \tag{7}$$

Moreover, if $f$ is strictly convex, the equality in Equation (7) implies that $X = \mathbb{E}X$ with probability 1 (i.e., $X$ is a constant).

**Theorem 2** (*Convexity of relative entropy* [42]) $\mathrm{D}_{KL}(Q \,\|\, P)$ is convex in the pair $(Q,P)$, that is, if $(Q_1,P_1)$ and $(Q_2,P_2)$ are two pairs of probability mass functions, then

$$D(\lambda Q_1 + (1-\lambda)Q_2 \,\|\, \lambda P_1 + (1-\lambda)P_2) \leq \lambda D(Q_1 \,\|\, P_1) + (1-\lambda)D(Q_2 \,\|\, P_2) \tag{8}$$

for all $0 \leq \lambda \leq 1$.

The intrinsic reward is:

$$\begin{aligned} r'_{\text{int}}(\text{s},\text{a}) &= r_{\text{int}}(\text{s},\text{a}) - \alpha Surprise(s,a) \\ &= r_{\text{int}}(\text{s},\text{a}) - \alpha\left[\mathop{\mathbb{E}}_{s'\sim Q(\cdot|s,a)} \log\frac{Q(\text{s},\text{a})}{P(\text{s},\text{a})}\right] \\ &\le r_{\text{int}}(\text{s},\text{a}) + \alpha \log \mathop{\mathbb{E}}_{s'\sim Q(\cdot|s,a)} \frac{P(\text{s},\text{a})}{Q(\text{s},\text{a})} \\ &= r_{\text{int}}(\text{s},\text{a}) + \alpha \log \sum_{s'\sim Q(\cdot|s,a)} Q(\text{s},\text{a})\frac{P(\text{s},\text{a})}{Q(\text{s},\text{a})} \\ &= r_{\text{int}}(\text{s},\text{a}) + \alpha \log \sum_{s'\sim Q(\cdot|s,a)} P(\text{s},\text{a}) \qquad \sum_{s'\sim Q(\cdot|s,a)} P(\text{s},\text{a}) \in (0,1] \end{aligned} \tag{9}$$

According to the above equations, the update rule of intrinsic reward is approximated as:

$$r'_{\text{int}}(\text{s},\text{a}) \leftarrow r_{\text{int}}(\text{s},\text{a}) + \alpha \log \sum_{s'\sim Q(\cdot|s,a)} P(\text{s},\text{a}) \tag{10}$$

Ideally, we would like the intrinsic rewards to approach zero in the limit of $P \to Q$, because in this case, the agent should have sufficiently explored the state space. Moreover, with continuous exploration and learning, the value of $\alpha$ is gradually decreasing. Ideally, we could expect that $r_{\text{int}}^{t+1}(\text{s},\text{a}) \in (0, r_{\text{int}}^{t}(\text{s},\text{a}))$. Based on these constraints, the value range of $\alpha$ is limited to:

$$\alpha = \begin{cases} 0.05 & \text{if } r_{\text{int}}(\text{s},\text{a}) > 0.1 - 0.05 \sum_{s'\sim Q(\cdot|s,a)} P(\text{s},\text{a}) \\ \dfrac{\sum_{s'\sim Q(\cdot|s,a)} P(\text{s},\text{a}) - 1}{\log \sum_{s'\sim Q(\cdot|s,a)} P(\text{s},\text{a})} * r_{\text{int}}(\text{s},\text{a}) & otherwise \end{cases} \tag{11}$$

where $r_{int}^{ini}(s,a)$ is the initial intrinsic reward in the context of $(\text{s},\text{a})$. The update progress of $r_{int}(s,a)$ is a monotonically decreasing process.

### *4.2. Reward Independent Hierarchical Skill Model*

In this work, we used the framework of option to represent skill. However, the intrinsic rewards are continually changing as the process of exploration and learning in this work, so the traditional option model is not applicable in this case. Therefore, we needed to adopt a reward-independent option model. Currently, Yao et al. [44] have proposed the linear universal option model (UOM) to deal with these problems. In this section, based on the incremental learning rule [41] and UOM [44], we propose reward independent incremental learning rules for hierarchical skill learning to deal with these problems. Moreover, we describe the formal expression of the reward-independent option model.

Formally, an option model of the long-term effect is defined as $(R^o, P^o)$. This model comprises two parts: the reward model $R^o$ and the transition model $P^o$. $R^o$ and $P^o$ are equivalent to the reward function and transition function in the MDP problem, respectively.

The reward model $R^o$ gives the expected discounted reward, which is received after executing the option *o* in state *s* at time *t* until its termination:

$$R_s^o = \mathbb{E}\{r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{k-1} r_{t+k}\} \tag{12}$$

where $k$ is the random execution time for option *o* from start to termination. $r_{t+k}$ is the reward at the time $t+k$. $\gamma \in [0,1]$ is a discount factor, which affects how much weight it gives to future rewards in the reward model.

The transition model $P^o$ gives the discounted probability of executing option $o$ from state $s$ and terminating at state $s'$:

$$P_{ss'}^o = \sum_{k=1}^{\infty} \gamma^k p(s',k) \tag{13}$$

where $p(s',k)$ is the probability that option $o$ terminates at state $s'$ after $k$ time steps.

The option model has the Bellman equation the same as the value function. For a Markov option $o = \langle I, \pi, \beta \rangle$, the model is:

$$\begin{aligned} R_s^o &= \sum_{a \in A_s} \pi(s,a) \mathbb{E}\{r_s^a + \gamma(1-\beta(s'))R_s^o\} \\ &= \sum_{a \in A_s} \pi(s,a) \sum_{s' \in S} p_{ss'}^a [r_{ss'}^a + \gamma(1-\beta(s'))R_{s'}^o)] \end{aligned} \tag{14}$$

where $\pi(.)$ is a distribution that is consistent with $o = \langle I, \pi, \beta \rangle$; $s'$ is the next state in the context of $(s,a)$; and $r_{ss'}^a$ is the reward in the context of $(s' \mid s,a)$.

$$\begin{aligned} P_{ss'}^o &= \sum_{a \in A_s} \pi(s,a) \mathbb{E}\{\gamma[(1-\beta(s_k))P_{s_k s'}^o + \beta(s_k)\mathbb{I}_{\{s_k = s'\}}]\} \\ &= \sum_{a \in A_s} \pi(s,a) \sum_{s_x \in S} p_{ss_k}^a \{\gamma[(1-\beta(s_k))p_{s_k s'}^o + \beta(s_k)\mathbb{I}_{\{s_k = s'\}}]\} \end{aligned} \tag{15}$$

For all $s, s_k, s' \in S$, $\mathbb{I}_{\{\cdot\}}$ is an indicator function. If its condition is satisfied, it equals 1; otherwise, it equals 0. According to Equations (14) and (15), the temporal-difference update rules are:

$$R_{s_t}^o \leftarrow R_{s_t}^o + \eta\left(r_{t+1} + \gamma(1-\beta(s_{t+1}))R_{s_{t+1}}^o - R_{s_t}^o\right) \tag{16}$$

$$P_{s_t s'}^o \leftarrow P_{s_t s'}^o + \eta\{\gamma[(1-\beta(s_{t+1}))P_{s_{t+1} s'}^o + \beta(s_{t+1})\mathbb{I}_{\{s_{t+1} = s'\}}] - P_{s_t s'}^o\} \tag{17}$$

where $\eta$ is a positive step-size parameter.

From the perspective of a linear setting, the temporal-difference approximation of $R^o$ is defined as:

$$R_s^o = f^\top(U^o \phi(s)) \tag{18}$$

where $f$ is the least-squares approximation of the expected one-step reward under the option $o$. $U^o$ is the discounted state occupancy function (reward-independent function). $\phi(s)$ is the feature vector, which maps any state $s \in S$ to its n-dimensional feature representation.

Combining the Bellman equation and the linear setting of the option model, we propose the reward independent incremental learning rules as follows:

$$U^o \leftarrow U^o + \eta\left(\phi(s_t) + \gamma(1-\beta(s_{t+1}))U^o\phi(s_{t+1}) - U^o\phi(s_t)\right)\phi(s_t)^\top \tag{19}$$

$$P^o \leftarrow P^o + \eta\{\gamma[(1-\beta(s_{t+1}))P^o + \beta(s_{t+1})\mathbb{I}_{\{s_{t+1} = s'\}}]\phi(s_{t+1}) - P^o\phi(s_t)\}\phi(s_t)^\top \tag{20}$$

The learning of the option model is divided into two parts: the learning of the reward independent function and the learning of the transition mode. In the high-level options, action selection is performed on options rather than primitive actions. Moreover, the modeling of option selection and the update of option models are the same as those of the primitive actions. Thus, we can implement the incremental learning of hierarchical skill models in lifelong learning. In the following section, we detail the specific implementation of the incremental learning framework for hierarchical skills.

### *4.3. Learning Algorithm*

In this work, our framework allows agents to explore the environment and learn skills in a developmental way, and the exploring trajectories can be multi-step. Furthermore, the learned model can be reused to perform complex specific tasks later.

Algorithm 1 gives an overview of the algorithm framework. The algorithm contains two parts. The first part is surprise-based hierarchical exploration and learning (Algorithm 1, lines 2–11). The second part is regularly updating the basic surprise-based exploration policy (Algorithm 1, lines 12–13) as well as the model and the intra-policy of the options (Algorithm 1, lines 14–16).

In the first part, the agent starts with the basic surprise-based hierarchical exploration policy (Algorithm 1, line 2), which is described in detail (Algorithm 2, lines 1–9). In the learning process, the primitive action models have the same representation and learning approach as the option models. The difference is that the primitive action models are only modeled over a single time step, and the termination functions return one in all next states. Then, the framework executes the option $o_t$ and observes the next state (Algorithm 1, line 3). After a round of action selection and execution, the framework updates the intrinsic rewards and the models of primitive actions with current experience (Algorithm 1, line 4). The update methods of intrinsic rewards and models are described in Sections 4.1 and 4.2, respectively. Next, the framework determines whether to create an option by judging whether the current state is the goal state for the first visit (Algorithm 1, lines 5–8). The new option is made up of a valid goal state, a pseudo-reward function $r_{pse}^{o}$, and a termination function $\beta_o(s)$. $r_{pse}^{o}$ and $\beta_o(s)$ are all defined as an indicator vector $\vec{\mathbb{I}}_{\{\phi(s)\in g_i\}}$; if the state $s$ is the goal state of option $o$, then $r_{pse}^{o} = \beta_o(s) = 1$, else $r_{pse}^{o} = \beta_o(s) = 0$. Finally, the framework updates the executed state stack, the current feature vector and the current time.

**Algorithm 1** Surprise-based Hierarchical Exploration for Model and Skill Learning

**Input**: current time $t$, arbitrary initial base policy $\pi$, current executing option stack $H$, current feature vector $\phi(s_t)$, intrinsic reward $r_{int}^{a} \leftarrow \vec{1}$, primitive action stack $A$, base value function $\theta_{int} \leftarrow \vec{0}$, arbitrary initial-policy for any option $\pi^o$, option model $(U^o, P^o)$, option value function $\theta^o$, option stack $O$ $(A \epsilon O)$, important goal stack $G$, pseudo reward for any option $r_{pse}^{o}$, executed state stack $D$, dataset of transition tuples of the area around the goal sate of option $o$ $D_{area\ o}$

1: **while** $t < t_{max}$ **do**
2: $o_t \leftarrow$ Surprise-based Hierarchical Exploration $(\phi(s_t), H, \pi)$
3: $\phi(s_{t+1}) \leftarrow$ Execute $(\phi(s_t), o_t)$
4: update the intrinsic reward and the model of primitive actions $(H, \phi(s_t), o_t, \phi(s_{t+1}))$
5: **if** $\phi(s_t) \in g_i,\ \forall g_i \in G$ and $o_i \notin O$ **then**
6: $o_i \leftarrow$ create new option $(g_i, r_{pse}^{o}, \beta_{o_i})$
7: $O \leftarrow O \cup o_i$
8: **end if**
9: update executed state stack $(\phi(s_t), D)$
10: $\phi(s_t) \leftarrow \phi(s_{t+1})$
11: $t \leftarrow t + 1$
12: Every $K$ time steps do:
13: $\pi \leftarrow$ Update Surprise-based Exploration Policy $(\theta_{int}, D)$
14: Every $T$ time steps do:
15: Update Intra-policy of Option $(r_{pse}^{o}, O, D)$
16: Update Option Model $(r_{pse}^{o}, O, D)$
17: **end while**

In the second part, every $K$ time steps, the framework updates the basic surprise based exploration policy with current experience (Algorithm 1, lines 12–13), which is described in detail (Algorithm 2, lines 11–18). Every $T$ time steps, the framework updates the model and the intra-policy of the learned options with current experience (Algorithm 1, line 14–16). The update method of the intra-policy is described in line 20 to 29 of Algorithm 2, and the update method of the option model is described in Section 4.2. The advantages of alternate learning and updating are to accelerate the learning process of the option policy and to avoid the bad learning effect when the option policy is not perfect.

Algorithm 2 contains specific implementations of some functions in Algorithm 1. The function of *Surprise-based Hierarchical Exploration* encourages the agent to go where it is unfamiliar to speed up the discovery of new goal states. The agent chooses the primitive action based on

$\arg\max_a \left[ (r_{\text{int}}^a)^\top U^a \phi(s) + \gamma (P^a \phi(s))^\top \theta_{\text{int}} \right]$. Then, it loops over the option stack (exclude primitive actions) to compare with the current option to choose the option that has the max loss value (Algorithm 2, lines 1–9). The function of *Update Surprise-based Exploration Policy* uses value iteration to update the basic policy of surprise-based exploration. Likewise, the function of *Update Intra-policy of Option* uses value iteration to update the intra-policy of option.

**Algorithm 2** Exploration and Update

1: **function** Surprise-based Hierarchical Exploration $(\phi(s_t), H, \pi)$
2: $o \leftarrow \pi(\phi(s_t)) = arg\ max_a[(r_{int}^a)^\top U^a \phi(s_t) + \gamma(P^a \phi(s_t))^\top \theta_{int}]$
3: **For** each $o_i \in (O - A)$ **do**
4: **if** $-\frac{1}{|D_{area}\ o_i|} log\ \Pi_{(s,a)\in D} P(s,a) > -\frac{1}{|D_{area}\ o|} log\ \Pi_{(s,a)\in D} P(s,a)$ *t***hen**
5: $o = o_i$
6: **end if**
7: **end for**
8: Return *o*
9: **end function**
10:
11: **function** Update Surprise-based Exploration Policy $(\theta_{int}, D)$
12: **for** *N* iterations **do**
13: **for** each $\phi(s) \epsilon D$ **do**
14: $\theta_{int} \leftarrow \theta_{int} + \alpha\ max_a[(r_{int}^a)^\top U^a \phi(s) + \gamma(P^a \phi(s))^\top \theta_{int} - \phi(s)^\top \theta_{int}]\phi(s)$
15: **end for**
16: **end for**
17: $\pi(s) \leftarrow arg\ max_a[(r_{int}^a)^\top U^a \phi(s) + \gamma(P^a \phi(s))^\top \theta_{int}]$
18: **end function**
19:
20: **function** Update Intra-policy of Option $(r_{pse}^o, O, D)$
21: **For** each $o_i \epsilon (O - A)$ **do**
22: **for** *N* iterations **do**
23: **for** each $s \epsilon D \cup I^{o_i}$ **do**
24: $\theta^{o_i} \leftarrow \theta^{o_i} + \alpha\ [(r_{pse}^o)^\top U^o \phi(s) + \gamma(P^o \phi(s))^\top \theta^{o_i} - \phi(s)^\top \theta^{o_i}]\phi(s)$
25: **end for**
26: **end for**
27: $\pi^{o_i}(s) \leftarrow \arg\ max_o[(r_{pse}^{o_i})^\top U^o \phi(s) + \gamma(P^o \phi(s))^\top \theta^{o_i}]$
28: **end for**
29: **end function**

### *4.4. Working Example*

In this section, we describe a simple working example to make each definition clearer. Figure 1 is a simplified version of our experimental environment. The robot can be anywhere in the room. We assume that the robot starts at the position $s$. $s1$ and $s2$ are the goal positions. The purpose of the robot is to explore the entire environment aimlessly based on intrinsic motivation and learn the skills to reach the goal positions from any position in the room. The working example is described as follows.

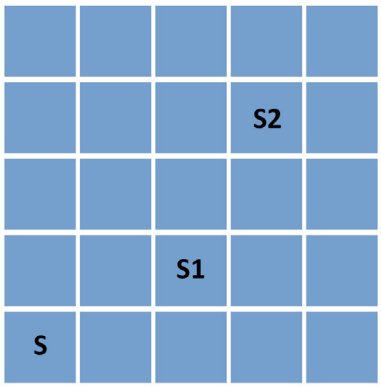



**Figure 1.** A simple environment.

**Step 1:** The robot explores the environment according to the surprise based hierarchical exploration policy (Algorithm 1, line 2; Algorithm 2, lines 1–9), and continuously updates the transition model and intrinsic reward of primitive actions (Algorithm 1, line 4).

**Step 2:** When the robot first reaches the position $s1$, it finds that $s1$ is a goal position whose associated option has not yet been created. Then, the robot creates a new option $o_{s1}$ for reaching the position $s1$, and adds the new option to the option stack. The new option is made up of a valid goal state $s1$, a pseudo-reward function $r^{o}_{pse}$, and a termination function $\beta_o(s)$.

**Step 3:** During the learning and exploration process, the framework updates the surprise based exploration policy every $K$ time steps as follows (Section 4.3, Algorithm 2, line 17).

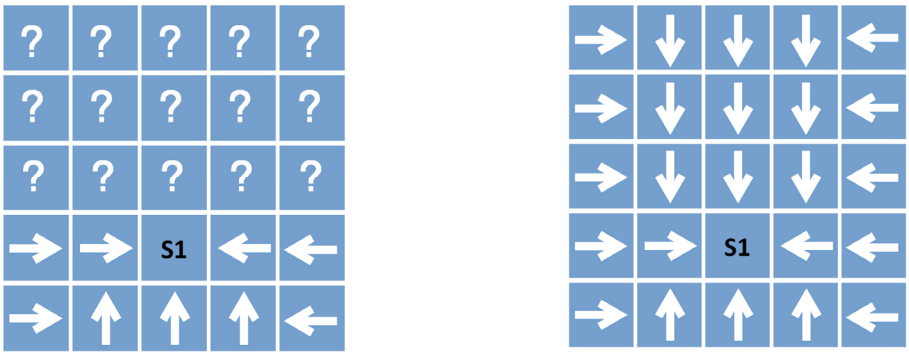
$$\pi(s) \leftarrow \arg\max_a \left[ (r^a_{\text{int}})^\top U^a \phi(s) + \gamma (P^a \phi(s))^\top \theta_{\text{int}} \right] \tag{21}$$

where $r^a_{\text{int}}$ is the intrinsic reward, and is reshaped by surprise, which is described in Equation (10) (Section 4.1). $a$ is a primitive action or an option.

**Step 4:** Every $T$ time steps, the framework optimizes the model and the intra-policy of the learned options with simulation. The framework updates the option model according to Equations (19) and (20) (Section 4.2), and updates the intra-policy of the learned options as follows (Section 4.3, Algorithm 2, line 27). Figure 2 shows that the intra-policy of option $o1$ changes from incomplete (Figure 2a) to optimal (Figure 2b) after several optimizations.

$$\pi^{o_i}(s) \leftarrow \arg\max_o \left[ (r^{o_i}_{pse})^\top U^o \phi(s) + \gamma (P^o \phi(s))^\top \theta^{o_i} \right] \tag{22}$$

**Step 5:** The robot uses the intra-policy of the learned option $o_{s1}$ to reach the unfamiliar area more quickly, which is hierarchical exploration. When the robot first encounters the position *s2*, it constructs the new option $o_{s2}$, and it can use the intra-policy of option $o_{s1}$ to learn the intra-policy of option $o_{s2}$ faster. For example, the intra-policy of option $o_{s2}$ from *s* to *s2* only needs to learn from *s1* to *s2* because it can use the existing intra-policy of option $o_{s1}$ from *s* to *s1*.




(**a**) Incomplete intra-policy (**b**) Optimal intra-policy

**Figure 2.** The incomplete intra-policy and the optimal intra-policy of option $o_{s1}$.

**Step 6:** After multiple time steps, the learned model will get closer to the true model, and the intrinsic rewards will tend to zero in the limit of $P \rightarrow Q$. This means that the robot has sufficiently explored the state space. Moreover, the learned options underlie both the ability to choose to spend effort and time to specialize at particular tasks, and the ability to collect and exploit previous experience to be capable of solving harder problems over time with less cognitive effort.

## 5. Experiments

### *5.1. Household Robot Pickup and Place Domain*

We tested the performance of our framework in the domain of Household Robot Pickup and Place. The problem is modeled as the robot picks up an object and places it at designated positions in the discrete rooms (Figure 3). The test domain is a variant of the hierarchical reinforcement learning household robot [45]. The experiments are designed with increasing state sizes to demonstrate that the proposed methods are scalable to handle large problems.

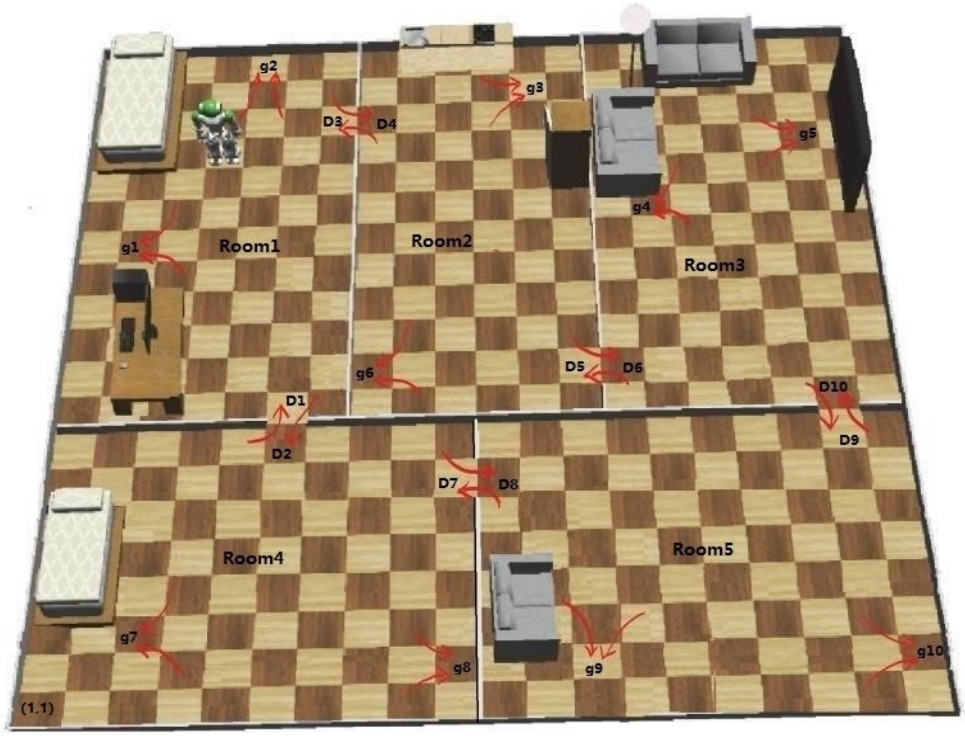



**Figure 3.** Household Robot Pickup and Place.

We preset ten goal positions of g1~g10. The destination position and the initial position of the object were randomly selected from the ten positions. The destination position and the initial position were two different positions. For example, if the destination position is g1, then the initial position of the object should be selected from g2~g10. Moreover, there are also many other notable positions (D1~D10). The goal positions and notable positions were used as sub-goals to create corresponding options (e.g., option $o_{D1}$ is defined as the transition from room 4 to room 1, and option $o_{g1}$ is defined as the transition from any position in room 1 to position g1).

In this work, we split the experiment into separate phases. In the first phase, the robot first explores the environment without a specific task, and it learns the skills in a developmental way. In the course of exploration and learning, the surprise-based hierarchical exploration can improve the exploration efficiency and speed up the model learning and skill learning. The robot also continues to refine the models and intra-policies of skills through simulation. In the second phase, the robot leverages the learned skills to perform the specific task (Robot Pickup and Place) in the framework of hierarchical MCTS (Monte Carlo Tree Search).

### *5.2. Performance Evaluation*

The surprise based intrinsic motivation is defined as the doing of an activity for its own sake rather than to directly solve some specific problems. Therefore, evaluating the benefits of surprised-base hierarchical exploration is not as simple as evaluating the performance of a standard reinforcement learning in a particular task. Therefore, we will evaluate the performance of our framework with model accuracy, loss value, the number of visited states, and the task performance in specific task with the learned model and options.

A typical loss is:

$$\mathcal{L}(P;D) = -\frac{1}{|D|}\log \prod_{(s,a,s')\in D} P(s' \mid s,a) \tag{23}$$

where $P$ denotes the learned transition model, which approximates the true model $Q$. $D$ is the dataset of transition tuples from the environment. In general, learning $P(s,a)$ implies minimizing the loss $\mathcal{L}(P(s,a);D)$. The closer $P$ and $Q$ are, the smaller the loss.

The model accuracy is defined as:

$$\bar{P}(D) = \frac{1}{|D|}\sum_{(s,a,s')\in D} P(s' \mid s,a) \tag{24}$$

To compare the performance of our framework with the performance of other methods, we used three different exploration policies, and these exploration policies guide behavior as the agent learns skills in the domain. The comparison will help us verify whether the proposed framework can accelerate exploration and skill learning.

**Random exploration**. A random exploration chooses a random action from the sets of primitive actions and options. It executes each action or option to completion before choosing another one. It is the baseline approach.

**Exploration with Exemplar Models (EX²)** [33]. We use the heuristic bonus they used in their experiments to reshape the intrinsic rewards to drive exploration.

**Surprise-HEL.** This method employs surprise-base hierarchical exploration policy. The combination of surprise and hierarchical exploration allows the agent to reach the areas with more information faster.

### *5.3. Results*

In this section, we present the evaluation results for Household Robot Pickup and Place domain. In the following content, we first show the performance of model learning and skill learning where there is no external reward. Then, we applied the learned model in the first step to a specific task to evaluate the task performance.

5.3.1. Performance of Model and Skill Learning

Figures 4a,c,e show the performance of model learning and skill learning in the domain of 10 × 10. Similarly, Figures 4b,d,f show the performance of model learning and skill learning in the domain of 20–20. In this experiment, all of the results were averaged over 100 runs.

From Figures 4a–d, we can see that compared with other methods, Surprise-HEL learns more accurate models when they all take the same number of steps, and can converge to the true transition model faster. Random exploration failed in obtaining convergence in both domains.

While learning relatively accurate models with finite steps, it is more important for the algorithm to explore unfamiliar and useful parts of the domain. Figures 4e,f show the average number of visited states of the three methods for both the 10 × 10 domain and 20 × 20 domain, which demonstrate the performance of the skill learning of different algorithms in these domains. Surprise-HEL performed better than the other two algorithms, and the random exploration failed in obtaining convergence in both domains. The reason is that the Surprise-HEL leverages the learned options and the surprise-based intrinsic motivation to speed up exploring unfamiliar areas. As a result, the more states the robot visits, the more options will be discovered to accelerate skill learning.

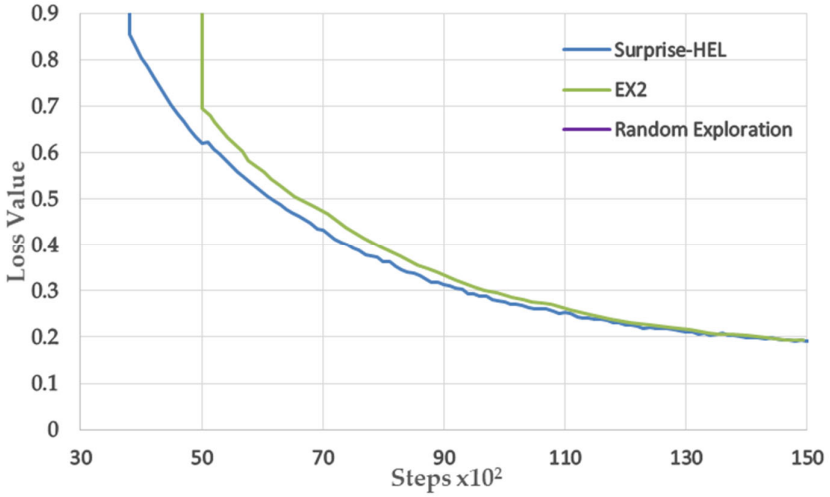
Taking the same number of steps in the domain of 10 × 10, it shows that in the term of model learning, compared with $EX^2$ and random exploration, the maximum differences of model accuracy were 4.73 and 19.64, and the maximum increases were 15.95% and 37.00%, respectively. Similarly, in the domain of 20 x 20, the maximum differences were 7.38 and 23.45, and the maximum increases were 18.80% and 69.17%, respectively.

In the term of skill learning, compared with $EX^2$ and random exploration and taking the same number of steps, the maximum differences of the visited states were 85.46 and 193.53, and the maximum increases were 38.40% and 105.37%, respectively. Similarly, in the domain of 20 × 20, the maximum differences were 361.90 and 790.83, and the maximum increases were 42.57% and 150.25%, respectively. The specific analysis results are shown in Table 1.

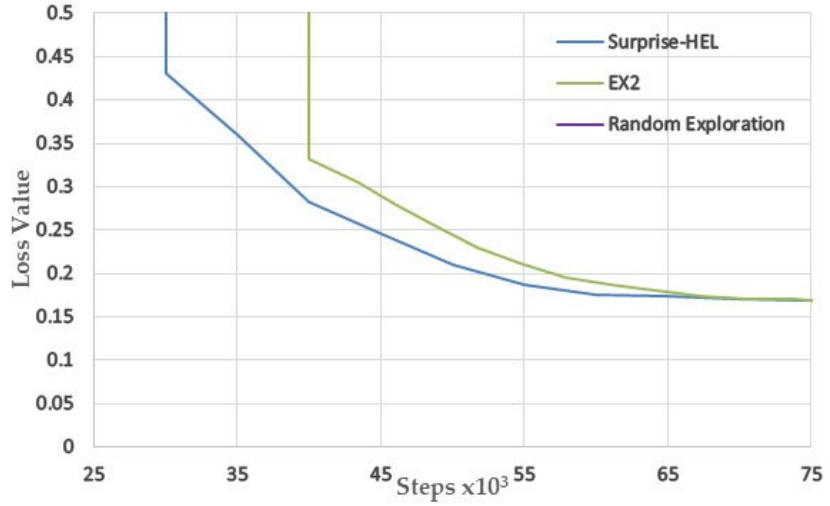
**Table 1.** The performance comparison of model learning and skill learning under the same steps.

| **The comparison in model learning** | | | | |
|---|---|---|---|---|
| **Algorithm** | **10 × 10** | | **20 × 20** | |
| | **Difference** | **Increase** | **Difference** | **Increase** |
| $EX^2$ | 4.72927569301687 | 15.9545090896041% | 7.386379431 | 18.7976065% |
| random | 19.6397000224526 | 37.0010595733331% | 23.44520569 | 69.1739% |
| **The comparison in skill learning** | | | | |
| **Algorithm** | **10x10** | | **20 × 20** | |
| | **Difference** | **Increase** | **Difference** | **Increase** |
| $EX^2$ | 85.456489458935 | 38.4035087719295% | 361.899865560966 | 42.5681606254505% |
| random | 193.533333333334 | 105.372050816698% | 790.833333333327 | 150.248459550786% |

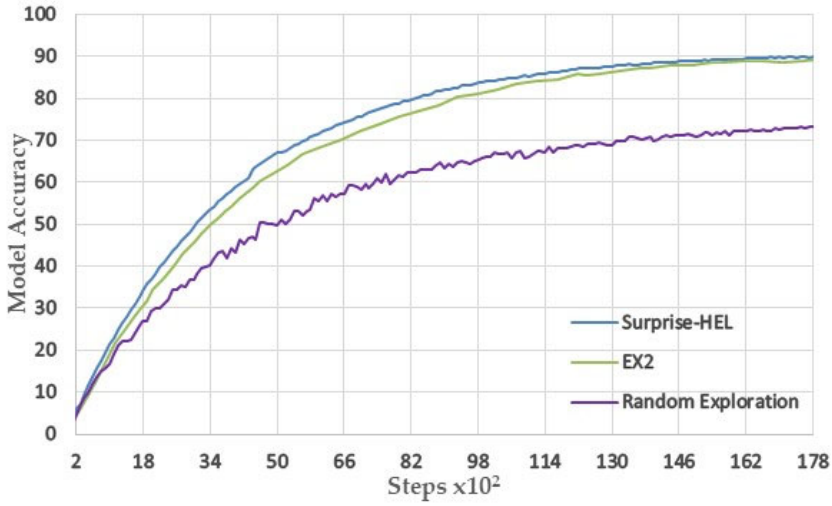
This result shows that the Surprise-HEL consistently outperformed other methods. Moreover, the performance improvements were much more dramatic in both model learning and skill learning in the larger domain. This further demonstrates the efficiency of our framework.



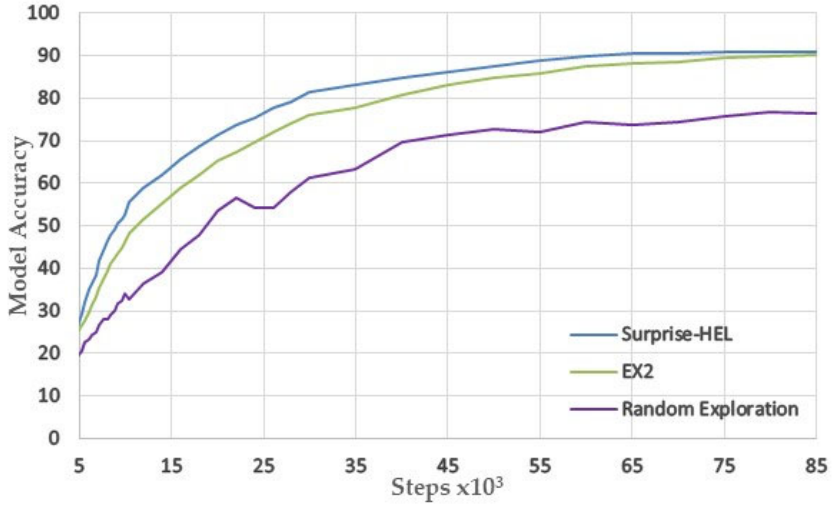

(**a**) Loss value (10 × 10 domain)



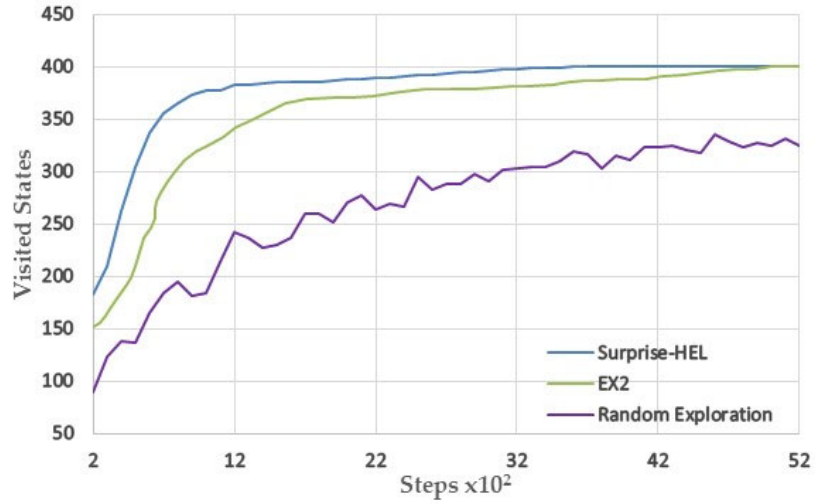

(**b**) Loss value (20 × 20 domain)



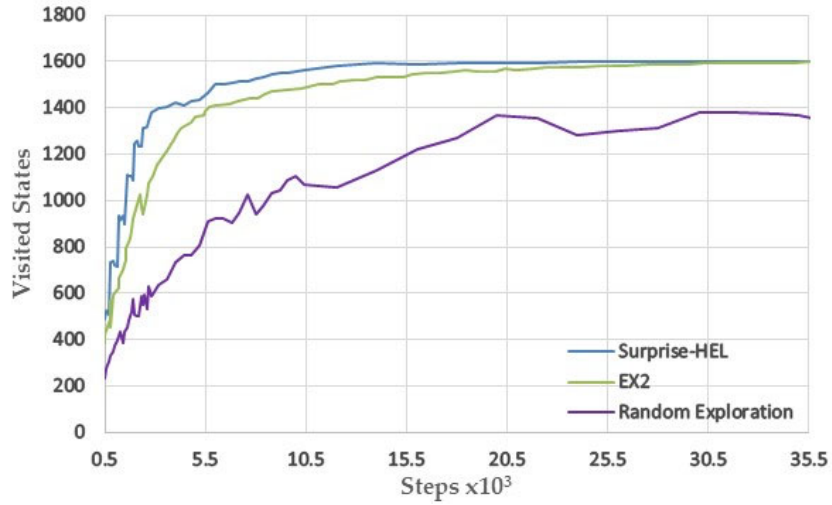

(**c**) Model accuracy (10 × 10 domain)




(**d**) Model accuracy (20 × 20 domain)




(**e**) Visited states (10 × 10 domain)




(**f**) Visited states (20 × 20 domain)

**Figure 4.** The empirical results of the model and skill learning: (**a**) The average loss value of model accuracy in the 10 × 10 domain; (**b**) The average loss value of model accuracy in the 20 x 20 domain; (**c**) The average model accuracy in the 10 × 10 domain; (**d**) The average model accuracy in the 20 × 20

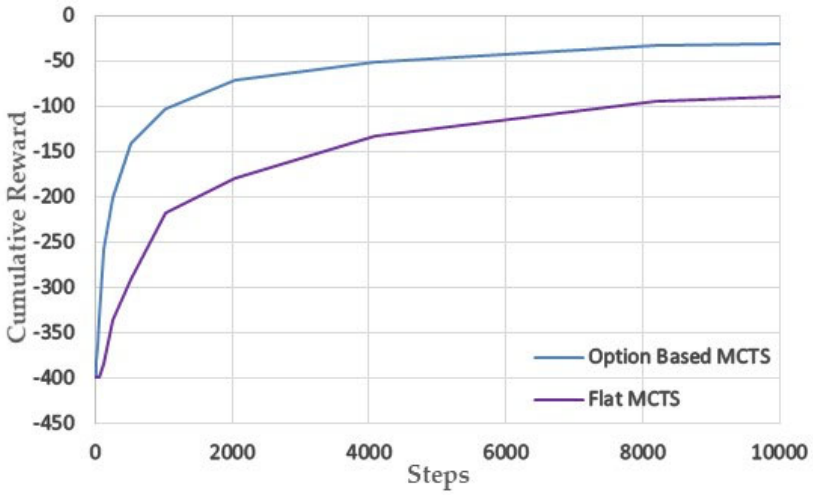
domain; (**e**) The number of visited states in the 10 × 10 domain; (**f**) The number of visited states in the 20 × 20 domain.

#### 5.3.2. Performance of Specific Task

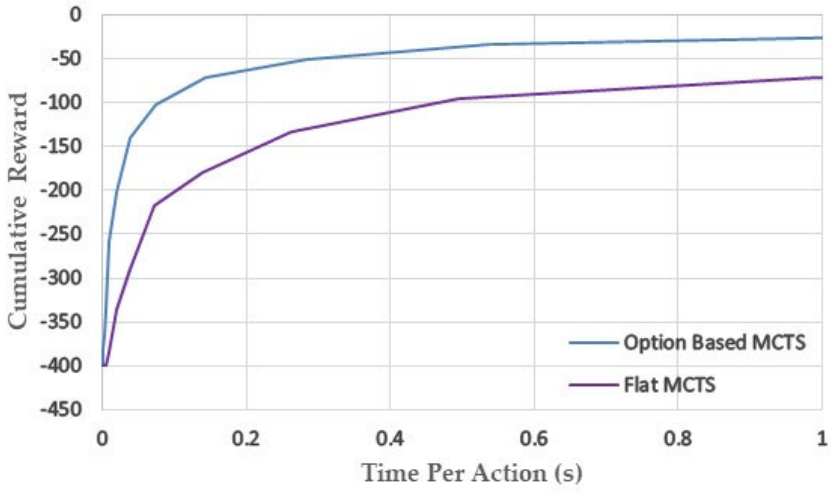
Next, we applied the learned model and skills to a specific task. The specific task was defined as the robot picks up the object in position g5 and places it in position g9. The initial position of the robot was (1, 1). There were three state variables: the position of the robot, the position of the object, and the position of the destination. There were four primitive actions: move up, move down, move left, and move right. Each primitive action had a reward of −1. Reaching the object's initial position and destination position both have a reward of 10. The maximal planning horizon was 400.

The performance was evaluated using the average cumulative return in terms of the number of simulations. Each data point was averaged over 100 runs in terms of the number of simulations.

Figure 5 shows the cumulative reward received by option based hierarchical MCTS and flat MCTS over $2^{18}$ simulations of the task. It shows that the option based hierarchical MCTS significantly outperformed the flat MCTS with limited computational resources. The option based hierarchical MCTS could find the optimal policy faster than flat MCTS. The reason is that the feature of option based hierarchical search can significantly reduce computational cost while speeding up learning and planning. Moreover, the rollouts of option based hierarchical MCTS sometimes terminate early, since the learned options help the rollouts to reach more useful parts of the state regions. Therefore, the option based MCTS can find the strategy to complete the task faster. Intuitively, the effect of option based hierarchical MCTS should become more significant in larger domains with longer planning horizons.




(**a**)




(**b**)

**Figure 5.** The performance of the specific task (picking up the object in position g5 and moving it to position g9) (**a**) The cumulative reward in terms of the number of simulations; (**b**) The cumulative reward in terms of the averaged computation time per action.

Additionally, the primary purpose of our work was to apply this framework to the domain of robot urban search and rescue (USAR) environments in a cluttered environment. However, the state space of the real world is continuous, dynamic, partially observable, and there are many uncertain factors. Applying our framework to this domain is a complex project, and our future work is to solve these problems.

## 6. Conclusions

In this work, we proposed the framework of surprise-based hierarchical exploration for model and skill learning (Surprise-HEL). This framework has three main features: surprise-based hierarchical exploration, reward independent incremental learning rules, and the technique of alternating model learning and policy update. The nature of the framework implements the incremental and developmental learning of models and hierarchical skills. In the experiment of household robot domain, we empirically showed that Surprise-HEL performed much better than other algorithms both for model learning and skill learning, and it performed better in large-scale problems. Moreover, we applied the learned model to a specific task to evaluate the task performance, and the results showed that the skill based hierarchical planning significantly outperformed the flat planning with limited computational resources. In future work, we plan to apply this method to complex real-world application scenarios such as rescue robots in cluttered and unknown urban search and rescue (USAR). We expect that our method can accelerate the robot's exploration and learning in a complex and unknown environment, and improve the robot's search and rescue capabilities based on the learned skills.

**Author Contributions:** Conceptualization, X.G.; Data curation, L.L.; Formal analysis, L.L.; Funding acquisition, X.G.; Supervision, J.C.; Writing – original draft, L.L.; Writing – review & editing, W.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the National Natural Science Foundation of China (61603406, 61806212, 61603403, and 61502516).

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Hebb, D.O. *The Organizations of Behavior: A Neuropsychological Theory*; Lawrence Erlbaum Wiley Mahwah New Jersey: London, 1949.

2. Weiner, I.B.; Craighead, W.E.; Tolman, E.C.; Weiner, I.B.; Craighead, W.E. *The Corsini Encyclopedia of Psychology*, 4th ed.; John Wiley & Sons: Hoboken, NJ, USA, 2010; Volume 4.
3. Botvinick, M.M. Hierarchical models of behavior and prefrontal function. *Trends Cogn. Sci.* **2008**, *12*, 201–208.
4. Whiten, A.; Flynn, E.; Brown, K.; Lee, T. Imitation of hierarchical action structure by young children. *Dev. Sci.* **2006**, *9*, 574–582.
5. Graybiel, A.M. The basal ganglia and chunking of action repertoires. *Neurobiol. Learn. Mem.* **1998**, *70*, 119–136.
6. Barto, A.G.; Konidaris, G.; Vigorito, C. Behavioral hierarchy: Exploration and representation. *Computational and robotic models of the hierarchical organization of behavior*. Springer, Berlin, Heidelberg, 2013. 13–46
7. Boutilier, C.; Dean, T.; Hanks, S. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *J. Artif. Intell. Res.* **1999**, *11*, 1–94.
8. Foster, D.; Dayan, P. Structure in the space of value functions. *Mach. Learn.* **2002**, *49*, 325–346.
9. Botvinick, M.M.; Niv, Y.; Barto, A.C. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition* **2009**, *113*, 262–280.
10. Barto, A.G. Intrinsically motivated learning of hierarchical collections of skills. *Proceedings of the 3rd International Conference on Development and Learning.* **2004**, 112–119. Available link: http://citeseerx.ist.psu.edu (accessed on 8 February 2020)
11. Ryan, R.M.; Deci, E.L. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemp. Educ. Psychol.* **2000**, *25*, 54–67.
12. Vigorito, C.M. Intrinsically Motivated Exploration in Hierarchical Reinforcement Learning. *Congr. Evol. Comput.* **2016**, *38*, 1550–1557.
13. Barto, A.; Mirolli, M.; Baldassarre, G. Novelty or Surprise? *Front. Psychol.* **2013**, *4*, 907.
14. Robertson, S.D.; Freel, J.; Anderson, R.B. *The Nature of Emotion: Fundamental Questions*; Oxford University Press: Oxford, UK, 1994.
15. Klyubin, A.S.; Polani, D.; Nehaniv, C.L. Empowerment: A universal agent-centric measure of control. In proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, 2–5 September 2005.
16. Jung, T.; Polani, D.; Stone, P. Empowerment for continuous agent-environment systems. *Adapt. Behav.* **2011**, *19*, 16–39.
17. Mohamed, S.; Rezende, D.J. Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning. *Advances in neural information processing systems.* **20**15, 2125-2133.
18. Huang, X.; Weng, J. Novelty and Reinforcement Learning in the Value System of Developmental Robots. In proceedings of the 2nd workshop on Epigenetic Robotics, Edinburgh, Scotland, 10–11 August 2002. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.336 (accessed on 8 February 2020)
19. Merrick, K.E. Novelty and beyond: Towards combined motivation models and integrated learning architectures. In *Intrinsically Motivated Learning in Natural and Artificial Systems*; Springer, Berlin, Heidelberg, 2013.
20. Kim, Y.; Nam, W.; Kim, H.; Kim, J.-H.; Kim, G. Curiosity-Bottleneck: Exploration By Distilling Task-Specific Novelty. In proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019.
21. Baldi, P.; Itti, L. Of bits and wows: A Bayesian theory of surprise with applications to attention. *Neural Networks* **2010**, *23*, 649–666.
22. Achiam, J.; Sastry, S. Surprise-Based Intrinsic Motivation for Deep Reinforcement Learning. *arXiv* **2017** arXiv:1703.01732.
23. Rankin, C.H.; Abrams, T.; Barry, R.J.; Bhatnagar, S.; Clayton, D.F.; Colombo, J.; Coppola, G.; Geyer, M.A.; Glanzman, D.L.; Marsland, S.; et al. Habituation revisited: An updated and revised description of the behavioral characteristics of habituation. *Neurobiol. Learn. Mem.* **2009,** *92*, 135–138.
24. Burda, Y.; Edwards, H.; Pathak, D.; Storkey, A.; Darrell, T.; Efros, A.A. Large-Scale Study of Curiosity-Driven Learning. *arXiv* **2018**, *arXiv:1808.04355*.
25. Singh, S.; Lewis, R.L.; Barto, A.G.; Sorg, J. Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective. *IEEE Trans. Auton. Ment. Dev.* **2010**, *2*, 70–82.
26. Kulkarni, T.D.; Narasimhan, K.R.; Saeedi CSAIL, A.; Tenenbaum BCS, J.B.; Saeedi, A.; Tenenbaum, J.B. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In

Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, 4–9 December 2016.
27. McGovern, A.; Barto, A.G. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In proceedings of the 18th International Conference on Machine Learning. Williams College, Williamstown, MA, USA, 2001. Available link: https://scholarworks.umass (accessed on 8 February 2020)
28. Menache, I.; Mannor, S.; Shimkin, N. Q-cut - Dynamic discovery of sub-goals in reinforcement learning. In Proceedings of the *European Conference on Machine Learning*; Springer, Berlin, Heidelberg, 2002; pp. 295–306.
29. Ramesh, R.; Tomar, M.; Ravindran, B. Successor options: An option discovery framework for reinforcement learning. In Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019.
30. Alemi, A.A.; Fischer, I.; Dillon, J.V.; Murphy, K. Deep variational information bottleneck. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, Toulon, France, 24–26 April 2017.
31. Kong, L.; Melis, G.; Ling, W.; Yu, L.; Yogatama, D. Variational state encoding as intrinsic motivation in reinforcement learning. In the Task-Agnostic Reinforcement Learning Workshop at ICLR **2019,** New Orleans, USA, 30 April 2019.
32. Frank, M.; Leitner, J.; Stollenga, M.; Forster, A.; Schmidhuber, J. Curiosity driven reinforcement learning for motion planning on humanoids. *Front. Neurorobot.* **2014**, *7*, 25.
33. Fu, J.; Co-Reyes, J.D.; Levine, S. EX2: Exploration with exemplar models for deep reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2017**, *2017*, 2578–2588.
34. Still, S.; Precup, D. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory Biosci.* **2012**, *131*, 139–148.
35. Emigh, M.; Kriminger, E.; Principe, J.C. A model based approach to exploration of continuous-state MDPs using Divergence-to-Go. *IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), IEEE*, **2015**, *1*, 1–6.
36. Dietterich, T.G. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *J. Artif. Intell. Res.* **2000**, *13*, 227–303.
37. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 1994.
38. Sutton, R.S.; Precup, D.; Singh, S. Between MDPs and Semi-MDPs: Learning, Planning, and Representing Knowledge at Multiple Temporal Scales. *Artif. Intell.* **1998**, *1*, 1–39.
39. Parr, R.; Parr, R. *Hierarchical Control and Learning for Markov Decision Processes*; University of California at Berkeley: Berkeley, CA, USA, 1998.
40. Parr, R.; Russell, S. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, **1997**, *1*, 1043–1049**.**
41. Sutton, R.S.; Precup, D.; Singh, S.P. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.* **1999**, *112*, 181–211.
42. COVER, T.M.J.A.T. *Elements of Information Theory*; John Wiley & Sons: Hoboken, NJ, USA, 1994; Volume 203.
43. Sutton, R.S.; Barto, A.G.; Book, A.B. *Reinforcement Learning: An Introduction*; MIT press, Cambridge ,MA, USA, 2018.
44. Yao, H.; Szepesvári, C.; Sutton, R.; Modayil, J.; Bhatnagar, S.; Szepesv, C.; Sutton, R.; Modayil, J. Universal Option Models. In Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS), Montreal, QC, Canada, 8–13 December 2014; pp. 1–9.
45. Li, Z.; Narayan, A.; Leong, T. An Efficient Approach to Model-Based Hierarchical Reinforcement Learning. In proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 3583–3589.