Complex & Intelligent Systems (2023) 9:4911–4922
https://doi.org/10.1007/s40747-023-00995-8

ORIGINAL ARTICLE

Efficient state representation with artificial potential fields for reinforcement learning

Hao Jiang[1] · Shengze Li[1] · Jieyuan Zhang[1] · Yuqi Zhu[1] · Xinhai Xu[1] · Donghong Liu[1]

Received: 26 May 2022 / Accepted: 7 February 2023 / Published online: 22 February 2023

Abstract

In the complex tasks environment, efficient state feature learning is a key factor to improve the performance of the agent's policy. When encountering a similar new environment, reinforcement learning agents usually need to learn from scratch. However, humans naturally have a common sense of the environment and are able to use prior knowledge to extract environmental state features. Although the prior knowledge may not be fully applicable to the new environment, it is able to speed up the learning process of the state feature. Taking this inspiration, we propose an artificial potential field-based reinforcement learning (APF-RL) method. The method consists of an artificial potential field state feature abstractor (APF-SA) and an artificial potential field intrinsic reward model (APF-IR). The APF-SA can introduce human knowledge to accelerate the learning process of the state feature. The APF-IR can generate an intrinsic reward to reduce the invalid exploration and guide the learning of the agent's policy. We conduct experiments on PySC2 with different mini-games. The experimental results show that the APF-RL method achieves improvement in the learning efficiency compared to the benchmarks.

Keywords Deep reinforcement learning · Artificial potential field · State representation · Intrinsic reward

Introduction

Deep reinforcement learning (DRL) has been developing rapidly in recent years. It has been widely used in various fields, such as playing Atari games [1,2], robotic motion control [3,4] and recommendation systems [5,6]. Although DRL methods hold promise for automating a wind range of decision-making tasks, these methods encounter the challenge of sample complexity when applied to real-world tasks. The existence of various uncertainties in the real-world tasks [7] greatly increases the difficulty of sampling. Even in some relatively simple tasks, millions of data collection are required. Complex tasks with high-dimensional observations and sparse rewards tend to require much more data. In addition, a large number of random explorations may generate invalid sample data, which further increases the difficulty of effective data collection.

During the learning process of humans, when they encounter a similar environment, they usually use learned common knowledge to extract its state feature. When this extracted feature is applied in the state feature learning, the performance can be improved even if the extracted feature is not complete. The introduction of human knowledge can reduce the invalid exploration by the agent in complex task environments. In addition, integrating human knowledge is promising to improve the efficiency of environmental state feature extraction and reduce the complexity and invalidity of the samples.

There have been studies that combine human knowledge with reinforcement learning. Hu et al. [8] and Fischer et al. [9] introduce logical rules into the deep neural network to improve learning efficiency and model performance. Hester et al. [10] use human knowledge in imitation learning in anticipation of solving sequential decision problems. Zhang

✉ Shengze Li
lsz86591989@163.com

Hao Jiang
haojiang542644@163.com

Jieyuan Zhang
zhangjynudt@163.com

Yuqi Zhu
zhuyuqi1997@126.com

Xinhai Xu
xuxinhai@nudt.edu.cn

Donghong Liu
liu_donghong@sina.cn

[1] Academy of Military Science, Beijing 100000, China

et al. [11] represent human knowledge with the fuzzy logic rules and leverage this knowledge to accelerate the learning process of reinforcement learning agents. To deal with various uncertainties in the environment, Xin et al. [12] designed a probabilistic model to capture uncertainties in the dynamics of complex systems using human knowledge, and Zhuang et al. [13] proposed an iterative online learning method to adapt the dynamics model to an unknown environment. Xie et al. [14] and Noguchi et al. [15] combine the artificial potential field with the reinforcement learning methods to solve the problem of autonomous navigation and obstacle avoidance of the agent. The artificial potential field (APF) method [16] models the agent's motion space as a virtual potential field space by constructing a potential function based on human knowledge about the environment.

Inspired by APF, we propose a novel APF-based RL (APF-RL) method, which integrates human knowledge into RL methods in an end-to-end manner. APF-RL can effectively extract vital state information from a complex and uncertain reinforcement learning environment for agent policy learning. This vital state information allows agent to avoid a lot of invalid exploration and improve the efficiency of sampling, therefore accelerating the learning process. The major contributions of this work can be summarized as follows:

1. **APF-based RL.** We propose an artificial potential field-based reinforcement learning (APF-RL) method which can be combined with any policy-based method. APF-RL consists of two parts: APF-SA and APF-IR.
2. **State representation via APF.** Artificial potential field state feature abstractor (APF-SA) is designed for complex task environments, which incorporates human cognition of the environment into the potential field computation and represents the state feature of the environment as the potential field.
3. **Reward shaping via APF.** We design an artificial potential field intrinsic reward (APF-IR) model to provide the agent with guidance, and thus reduce the agent's invalid exploration in a sparse reward environment. Furthermore, we prove that using the APF-IR model can still preserve the invariance of the learned optimal policies.

We conduct experiments in several popular mini-games in PySC2 [17] and the experiments demonstrate that the APF-RL method outperforms the benchmark methods. Moreover, the effectiveness of each component of the APF-RL method is shown through ablation experiments.

This work is organized as follows. Related work and background knowledge are discussed in "Related work" and "Background" respectively. We elaborate our method in "Method". Experimental results are in "Experiments". Finally, "Conclusion" concludes our work.

Related work

State representation learning (SRL) has been widely used in deep reinforcement learning as an effective method for learning high-dimensional data features to find effective feature from the complex environmental state that contributes to policy learning. Early researchers used manual methods for feature extraction [18]. Although these methods can quickly extract effective features, they require a high level of expertise and understanding of the data feature. To reduce the labor cost, some researchers have tried to learn embedding vectors from complex data through deep neural networks [1,19] to obtain effective features in an end-to-end manner. Moreover, many other researchers used unsupervised auxiliary tasks in DRL to improve data efficiency and robustness of hyperparameter settings. Contrastive unsupervised representations for reinforcement learning (CURL) [20] extracts high-level features from raw pixels using contrastive learning and performs off-policy control on top of the extracted feature. ST-DIM [21] uses temporal and contrastive losses to operate on local features of the intermediate layer within the encoder without data augmentation. Augmented temporal contrast (ATC) [22] is a method that decouples representation learning from policy learning, which trains a convolutional encoder to associate pairs of observations separated by a short time difference, under image augmentations and using a contrastive loss. The APF-SA belongs to the combination of manually constructed features and network embedding.

Reward shaping is a method of modifying the original reward by adding a shaping reward function to the original reward. This shaping reward function incorporates the prior expert knowledge or domain knowledge. There is very early work on reward shaping in reinforcement learning. Dorigo et al. [23] translated expert instructions into rewards for the agent and used these rewards to guide the agent to perform the task. These early works focused on how to design the shaping reward function but ignores that the shaping rewards may change the optimal policy. Ng et al. [24] first proposed the potential-based reward shaping (PBRS) method. PBRS constrains the shaping reward to have the form of a difference of a potential function of the transitioning states and guarantees the so-called policy invariance property. PBRS method has led to more researchers focusing on the shaping of rewards. The potential-based advice (PBA) method [25] extended PBRS to state-action advice potentials, which defines the state-action space for providing advice on actions. The dynamic PBRS method [26] introduces a time parameter into the potential function for allowing dynamic potentials. Harutyun et al. [27] proposed a DPBA method that learns an auxiliary value function for transforming an arbitrary reward function into a dynamic potential function. Hu et al. [28] utilized a bi-level optimization method to solve the problem

of adaptively utilizing a given shaping reward function. The APF-IR is a potential-based reward shaping method.

Background

Deep reinforcement learning

In a standard reinforcement learning framework [29], the agent generally learns by interacting with its Markovian environment at discrete time steps $t = 1, 2, \ldots$. Formally, a Markov Decision Process (MDP) [30] is a tuple $\mathcal{G} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P} = \{P_{sa}(\cdot) \mid s \in \mathcal{S}, a \in \mathcal{A}\}$ are the next state transition probabilities with $P_{sa}(s')$ specifying the probability of state s' occurring upon taking action a from state s, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the expected reward function with $\mathcal{R}(s, a)$ giving the expected value of the reward that will be received when the agent takes an action a in the state s, and $\gamma \in [0, 1)$ is a discount factor.

A policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a probability distribution over actions at each state that determines the actions the agent will take in each state. The goal of the agent is to find the policy π mapping states to actions that maximize the expected discounted total reward over the agent's lifetime. The state value function $V^{\pi}(s)$ is an estimate of the expected future reward that can be obtained from state s when following policy π.

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right], \tag{1}$$

where r_{t+1} denote the components of $\mathcal{R}$ at time t. The state-action value function $Q^{\pi}(s, a)$ is the expected reward following policy π after taking action a at state s.

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a\right] \tag{2}$$

Advantage actor-critic (A2C)

Several effective policy-based methods [31,32] have been proposed in the literature. In a policy-based method, the training objective is to find a policy π that maximizes the expected reward J over all possible dialogue trajectories given a starting state. Generally, the policy π can be represented by a parameterized function (e.g., a neural network). In this work, we denote a policy by π_θ, where θ is the parameter of the policy function. The objective J can be written as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)]. \tag{3}$$

Since gradients of this form have potentially high variance, a baseline function is often introduced to reduce the variance without changing the estimated gradient. The common baseline function is the value function $V^{\pi_\theta}(s)$. In this work, we also use the value function as the baseline function, and then Eq. 3 can be written as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s, a)]. \tag{4}$$

Here, $A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$ is the advantage function, which represents the advantage of the state-action value function over the current state value function. If the advantage function is greater than 0, then a is better than average, otherwise, a is worse. The Eq. 4 can be viewed as a special case of the actor-critic, where π_θ is the actor and $A^{\pi_\theta}(s, a)$ is the critic. Temporal difference (TD) errors [33] can be used to approximate the advantage function [34].

$$A^{\pi_\theta} = r_t + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t) \tag{5}$$

Artificial potential fields

The artificial potential field [16] is a virtual force method for agent motion planning. The method specifies the effect of the target and the obstacle on the agent as an artificial potential field through a potential field function U. The potential energy is low at the target and high at the obstacle. This potential difference generates the gravitational force of the target on the agent and the repulsive force of the obstacle on the agent, whose combined effort controls the agent's motion toward the target point along the negative gradient direction of the potential field. The potential field function consists of the attractive and repulsive potential functions:

$$U(p) = U_{\text{att}}(p) + U_{\text{rep}}(p), \tag{6}$$

where p is the location of the unit, U_{att} and U_{rep} denote the attractive potential and repulsive potential function, respectively. The classical U_{att} and U_{rep} [16] can be written as:

$$U_{\text{att}}(p) = \frac{1}{2}\psi d^2, \tag{7}$$

and

$$U_{\text{rep}}(p) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{d} - \frac{1}{d^*}\right), & d \le d^* \\ 0, & d > d^* \end{cases}, \tag{8}$$

where d^* is the distance threshold, d is the distance between p and obstacles/ goals, ψ and η are the attraction and repulsion gain respectively. Unlike the classical potential field function, in this work, we will use the kernel density estimation function [35] as the potential field function, as described in "Artificial potential field state feature abstractor".

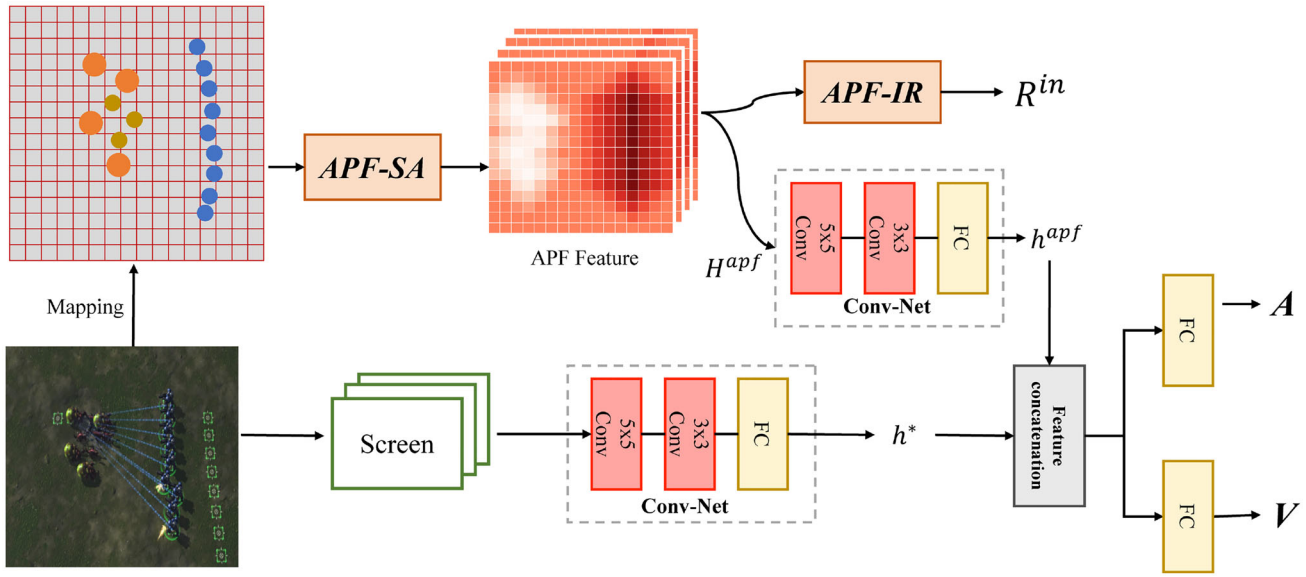

Fig. 1 Overall architecture of APF-RL

Methods

In this section, we propose a novel end-to-end learning method to integrate human knowledge into DRL. This proposed method is called APF-based reinforcement learning (APF-RL), and the overall architecture of APF-RL is shown in Fig. 1. First, we construct the artificial potential field state feature abstractor (APF-SA) using a priori domain knowledge, which can convert the original environmental state feature into a potential field feature. Second, we design the artificial potential field intrinsic reward model (APF-IR) based on the potential field feature, which generates an intrinsic reward for guiding the learning of the agent's policy. Third, the fully convolutional neural (FullyConv) network is used to encode the potential field feature and the original environment feature. and then connect the encoded feature. Finally, the encoded feature is connected and used to compute value function and policy.

In "Artificial potential field state feature abstractor" and "Artificial potential field intrinsic reward", we describe the architecture of the APF-SA and APF-IR, respectively. In "APF-RL overview", we demonstrate how to combine the APF-SA and APF-IR, forming a complete APF-RL method.

Artificial potential field state feature abstractor

APF-SA is an environmental state feature extraction method. It converts the original environmental state feature (e.g., images) into the potential field feature by the potential field function. The classical APF method is generally used in the field of robot navigation and obstacle avoidance. It first calculates the potential field value between the robot and the target/obstacle unit. And then guides the robot closer to the target unit (away from the obstacle unit) based on the change in the potential field value.

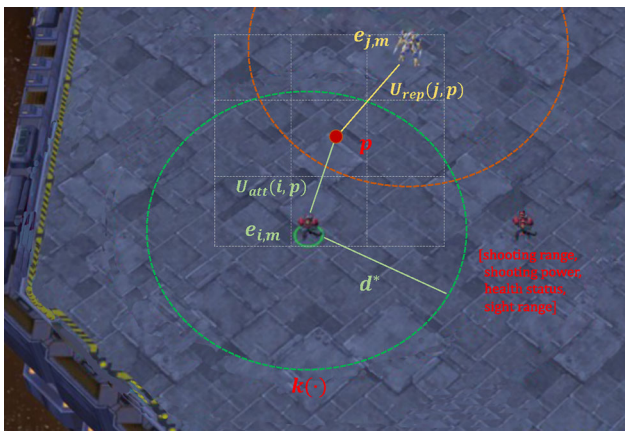

Fig. 2 Calculate the potential field value of point p

In contrast, in this work, we utilize the APF method to quantify the degree of influence of the unit at the spatial point p. The specific calculation of the potential field value at p is shown in Fig. 2. We first zone the environment, converting the whole environment into a $C \times C$ discrete environment. The spatial point p is the center point of one of the regions, and the potential field values are the same everywhere in the same region. Then, we design the potential field function U to calculate the potential field value of each unit at point p. The U consists of an attractive function U_{att} and a repulsive function U_{rep}. The attractive function is designed to increase as our unit i approaches the spatial point p, i.e. the closer our unit i is to the point p, the greater its positive influence

at the point p. While the repulsive function is negative, its absolute value increases as the enemy unit j approaches the spatial point p, i.e. the closer the enemy unit j is to the point p, the greater its negative influence at the point p. Formally, the attractive function U_{att} and repulsive function U_{rep} can be described as follows:

$$U_{\text{att}}(i, p) = \begin{cases} \frac{3}{\pi}\left[1 - \frac{\hat{d}(i)}{d^*}\right]^2 \lambda_i(\hat{e}_{i,b_{\text{pro}}}), & \hat{d}(i) \le d^* \\ 0, & \hat{d}(i) > d^* \end{cases} \tag{9}$$

and

$$U_{\text{rep}}(j, p) = \begin{cases} \frac{3}{\pi}\left[\frac{\check{d}(j)}{d^*} - 1\right]^2 \kappa_j(\check{e}_{j,b_{\text{pro}}}), & \check{d}(j) \le d^* \\ 0, & \check{d}(j) > d^* \end{cases}. \tag{10}$$

Here, $\hat{d}(i)$ and $\check{d}(j)$ are the Euclidean distances from the current positions of our unit i and the enemy unit j to the point p, respectively. d^* is the distance threshold. $\hat{e}_{i,b_{\text{pro}}}$ and $\check{e}_{j,b_{\text{pro}}}$ are the intrinsic properties of our unit i and the enemy unit j, such as shooting range, shooting power, health status, and sight range, etc., respectively. $\lambda_i(\hat{e}_{i,b_{\text{pro}}})$ is the b_{pro} property value of the our unit i and $\kappa_j(\check{e}_{j,b_{\text{pro}}})$ is the b_{pro} property value of the enemy unit j.

According to Eqs. 9 and 10, we can calculate the influence value of our unit i or the enemy unit j at the point p. Finally, the influence of all units in the environment at point p is superimposed to obtain the final potential field value. Formally, the final potential field value can be described as follows:

$$f_{b_{\text{pro}}}(p) = \sum_{i=1}^{N}[U_{\text{att}}(i, p)] + \sum_{j=1}^{K}[U_{\text{rep}}(j, p)], \tag{11}$$

where N and K are the numbers of our units and enemy units in the environment, respectively. For the different properties of the unit at p, we use the Eq. 11 to calculate the final potential field values of different properties, which can be described as follows:

$$\mathcal{F} = [f_{b_1}(p), \ldots, f_{b_M}(p)], \tag{12}$$

where the M is the number of the units' property. The original environment state feature is transformed into a discrete potential field feature $H^{\text{apf}} : M \times C \times C$.

Artificial potential field intrinsic reward

To guide the agent to learn an effective policy and reduce the agent's invalid exploration in a sparse reward environment, we proposed an APF-IR model which generates an intrinsic reward from the potential field feature. This intrinsic reward turns the original sparse reward into a dense reward. When the agent has performed an action, it causes a change in the potential field value. If the value of the potential field increases, it means that the action is positive. In this case, we give the agent positive feedback in addition to the environment's original reward. Conversely, we give negative feedback when the value of the potential field reduces. Based on the above analysis, we can define an intrinsic reward function r^{in} and add it to the original reward r^{origin} to get the proxy reward. Formally, the proxy reward can be described as follows:

$$\mathcal{R}_t^{\text{proxy}} = r_t^{\text{origin}} + \xi \mathcal{R}_t^{\text{in}}, \tag{13}$$

where ξ is a hyper-parameter that balances the original reward and distinct intrinsic reward. The final potential field value of different properties $\mathcal{F}$ can be obtained by Eq. 12. To obtain the integrated potential field values for the whole environment, we sum up the potential field values for each region in the discrete environment. Formally, the integrated potential field values $f_{b_{\text{pro}}}^{\text{all}}$ can be described as follows:

$$f_{b_{\text{pro}}}^{\text{all}} = \sum_{c=1}^{C \times C} f_{b_{\text{pro}}}(p_c), \tag{14}$$

where p_c denotes the center point of region c. The intrinsic reward function is designed as the potential field difference between the t and $t-1$. According to Eq. 14, the intrinsic reward function can be written as follows:

$$r_{b_{\text{pro}},t}^{\text{in}} = f_{b_{\text{pro}},t}^{\text{all}} - f_{b_{\text{pro}},t-1}^{\text{all}}, \tag{15}$$

where we consider only one property of the unit. To simplify the notation, we write $r_{b_{\text{pro}},t}^{\text{in}}$ as $r_{b_{\text{pro}}}^{\text{in}}$. The intrinsic reward of different properties can be written as follows:

$$\mathcal{R}_t^{\text{in}} = [r_{b_1}^{\text{in}}, \ldots, r_{b_M}^{\text{in}}]. \tag{16}$$

The different properties of the unit have different influences on the task. To balance the influence of each property, we introduce trainable weights $\beta_1, \ldots, \beta_M$. Therefore, the $\mathcal{R}_t^{\text{in}}$ can be rewritten as follows:

$$\mathcal{R}_t^{\text{in}} = \beta_1 r_{b_1}^{\text{in}} + \cdots + \beta_M r_{b_M}^{\text{in}}. \tag{17}$$

In this work, these trainable weights are set according to prior knowledge. For example, if a property is more relevant to the unit's task, the intrinsic reward for this property can be assigned a higher weight. In practice, intrinsic rewards are normalized.

Moreover, it needs to notice that adding the intrinsic reward to the original reward of the Bellman equation may change the optimal policy $\pi^*(a|s)$. To address this problem,

Ng et al. [24] proposed a Potential-based reward shaping (PBRS) method, which guarantees the policy invariance property. PBRS defines the Theorem 1 and proves that adding PBRS function can guarantees the learned policy maintains its optimally. More detailed proof of the Theorem 1 is given in [24].

Theorem 1 *Let any $\mathcal{S}$, $\mathcal{A}$, ζ, and any shaping reward function $\mathcal{F} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \to \mathcal{R}$ be given. We say $\mathcal{F}$ is a potential-based shaping function, if there exists a real-valued function $\Phi : \mathcal{S} \to \mathcal{R}$ such that for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $s' \in \mathcal{S}'$*

$$\mathcal{F}(s, a, s') = \zeta \Phi(s') - \Phi(s). \tag{18}$$

Then, that $\mathcal{F}$ is a potential-based shaping function is a necessary and sufficient condition for it to guarantee consistency with the optimal policy (when learning from $\mathcal{G}' = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}+\mathcal{F}, \gamma \rangle$ rather than from $\mathcal{G} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$).

We define the reward function $\Phi(s')$ to be the APF value $f^{\text{all}}_{b_{\text{pro}},t}$. According to the definition of $F(s, a, s')$, the $F(s, a, s')$ can be rewritten as follows:

$$\mathcal{F}(s, a, s') = \zeta f^{\text{all}}_{b_{\text{pro}},t} - f^{\text{all}}_{b_{\text{pro}},t-1} = r^{\text{in}}_{b_{\text{pro}},t}, \tag{19}$$

where the ζ is 1. According to Theorem 1 and Eq. 19, the Eq. 17 can be rewritten as follow:

$$\begin{aligned} \mathcal{R}^{\text{in}}_t = &[\beta_1 r^{\text{in}}_{b_1,t} + \cdots + \beta_M r^{\text{in}}_{b_M,t}] \\ &-[\beta_1 r^{\text{in}}_{b_1,t-1} + \cdots + \beta_M r^{\text{in}}_{b_M,t-1}]. \end{aligned} \tag{20}$$

Therefore, by adding $\mathcal{R}^{\text{in}}_t$ to the learning process, the optimal policy learned by the agent remains invariance.

APF-RL overview

To leverage human knowledge, we use the potential field feature H^{apf} extracted by APF-SA as input for downstream policy learning. However, the H^{apf} may lose some state feature information, so the original state feature is used as a supplement to the potential field feature, as shown in Fig. 1. First, the FullyConv Network is used to obtain the original state feature h^*. FullyConv model has the characteristics of full convolution and maintaining resolution. And then, we use the H^{apf} as input of the FullyConv network and output hidden potential field feature h^{apf}. Finally, we aggregate the two features of h^* and h^{apf} and input them into a fully connected layer to calculate the value V or action A. The overall algorithm, APF-RL, is summarized as Algorithm 1.

APF-based RL is an end-to-end policy framework, it can be combined with any policy-based algorithm. In this work, we use A2C as our basic RL algorithm, and the training objective Eqs. 4 and 5 can be rewritten as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) A(s, a)], \tag{21}$$

where

$$A = \mathcal{R}^{\text{proxy}}_t + \gamma V(s_{t+1}) - V(s_t). \tag{22}$$

Algorithm 1 APF-RL

```
Initialize Actor θ, replay memory D
for episode = 1 to M do
    Initialize a random process N for action exploration, and receive state information s
    for t = 1 to max-episode-length do
        Execute actions a and receive reward r_t^{origin} and new observation s_{t+1}
        Calculate the discrete potential field feature H^{apf} // Eq.12
        Calculate the intrinsic reward r^{in} // Eq.17
        Calculate the proxy reward R_t^{proxy} // Eq.13
        Store (s_t, a_t, R_t^{proxy}, s_{t+1}) in replay buffer D
        Sample a random of minibatch b from D
        set advantage function: A = R_t^{proxy} + γV(s_{t+1}) − V(s_t)
        set objective function: J(θ) = E[log π_θ(a|s)A(s, a)]
        θ ← θ + α∇_θ J(θ) // Eq.21
    end for
end for
```

Experiments

Settings

For our experiments, we used the Advantage Actor-Critic (A2C) method as the base learner. The A2C RL agent structure with neural networks as function approximators, as implemented by Simonmeister [36]. The experimental environment is PySC2 [17] learning environment. PySC2 is a Python component of the StarCraft II learning environment developed by DeepMind, which provides an interface for reinforcement learning agents to interact with StarCraft II. In the experiment, we consider symmetric battle games in StarCraft II with 3 types of mini-games (see Fig. 3). The rewards are based on the score from the StarCraft II engine against the built-in computer opponent. Therefore the higher the score the higher the reward. Like as [17], the score is used as the performance evaluation indicator. We evaluated the proposed APF-RL method and compared it with the benchmark methods (A2C and PBA), where the potential-based advice (PBA) [25] is a reward reshaping method that provides advice on actions for the agent.

- **DefeatRoaches:** In this map, the agent needs to fight against the enemy army, the enemy army is intact, and it

Fig. 3 The three types of mini-games in PySC2: **a** *DefeatRoaches*, **b** *DefeatZerglingsAndBanelings*, **c** *7m_vs_8m*

is different from the agent, so the agent needs to find the best attack policy to defeat the built-in army while only controlling its own units.

- **DefeatBanelingsAndZerglings:** In this map, the agent has to fight two kinds of enemy units, namely *Banelings* and *Zerglings*. Among them, the *Banelings* will explode when they encounter the agent, so the agent needs to avoid poisonous explosions during the fight against insects.
- **7 M_vs_8M:** In this map, allied units and enemy units have the same type, but there are more enemy units than the agent, so the agent needs to learn effective policy to defeat the enemy army that is stronger than the agent.

Table 1 Hyperparameter settings

Name	Description	Value
lr	Learning rate	0.0001
Optimizer	Type of optimizer	Adam [37]
Optimize α	Adam param	0.99
Optimize ϵ	Adam param	1e−5
Grad clip	Reduce the global norm of gradients	10
γ	Discount factor	0.99
ξ	Temperature	0.01
Starting ϵ	Starting value for exploration rate annealing	1.0
Ending ϵ	Ending value for exploration rate annealing	0.05
Anneal time	Number of steps to anneal exploration rate	50K

We train the agent to control allied units, while the enemy units are controlled by a built-in hand-crafted AI. Proper micromanagement of units during battles is necessary to maximize damage to enemy units while minimizing damage received, which requires a range of coordination skills such as focusing firepower or avoiding overkill. It is a challenge to learn these different behaviors under partial observation conditions.

Each type of unit in the mini-games has its inherent properties. In this work, we focus only on the effect of four inherent properties—*sight range*, *shooting range*, *shooting power* and *health status*—on the policy of the agent. Based on these four properties, we convert the original environmental state feature into the potential field feature with four layers. The specific potential field calculation is shown in Fig. 2. Because of space limitation, here we only give the potential field value calculation for a Marine at point p as an example. The Marine's *shooting power* is 7. We obtain $f_{b_{\text{pro}}(p)}$ by Eq. 11 where $\lambda_i(\hat{e}_{i,b_{\text{pro}}})$ is 7.

The training time is about 24–36 h on these maps (Intel (R) Core (TM) i9-10900K CPU @ 3.70 GHz, 96 GB RAM, GeForce RTX 2080Ti GPU), which is ranging based on the agent numbers and environmental features of each map. The number of total training steps is about 300K. We use the ϵ-greedy policy for exploration. The starting exploration rate is set to 1 and the end exploration rate is 0.05. The exploration rate decays linearly at the first 50K steps. The distance threshold d^* depends on the sight range of the unit. The $\lambda_i(\hat{e}_{i,b_{\text{pro}}})$ depends on our units' inherent properties: *shooting power*, *shooting range*, *health status*, and *moving speed*. The $\kappa_j(\check{e}_{j,b_{\text{pro}}})$ depends on enemy units' inherent properties. The discrete potential field feature $H^{\text{apf}} : M \times C \times C$ is set to $4 \times 32 \times 32$. More hyper-parameters are listed in Table 1.

Evaluation

To evaluate the performance and verify our method, we trained the agent on each mini-game. These experiments are carried out with 5 different random seeds, and the results are shown with a 95% confidence interval. The results are reported in Fig. 4, where the mean scores vs. the training steps on the three mini-games *DefeatRoaches*, *DefeatBanelingsAndZerglings*, *7M_vs_8M* are given.

The results show that our APF-RL method scores higher than the A2C method and PBA method in all three mini-games, which indicates that APF-RL outperforms the baseline methods. In *DefeatRoaches*, the scores of the APF-RL, PBA, and A2C methods reach about 120, 115, and 100 respectively, indicating that these methods learn a good policy for the agent in this scenario. The APF-RL, PBA, and A2C methods achieve scores of about 100, 92, and 80 in *DefeatBanelingsAndZerglings*, respectively. Both methods learn a worse policy in the *DefeatBanelingsAndZerglings* scenario than in the *DefeatRoaches* scenario. This is because the enemy units in *DefeatBanelingsAndZerglings* have different abilities, which requires the agent to explore more different policies. However, in this heterogeneous scenario, our APF-RL method still performs better than the A2C method. In *7M_vs_8M*, our 7 Marine face the 8 Marine. The unequal power gap means that our agent is doomed to fail if they attack directly. It requires our agents to find and attack the weak points of the enemy forces. The A2C and PBA are able to score about 70 and 90 respectively, which means that the agent learns an effective policy. However, our APF-RL method scored about 100, which means that the agent learns a better policy. The experimental results in Fig. 4 show that potential field feature representations and intrinsic rewards can help the agent explore the weak areas of enemy forces, learn a policy to preferentially attack the weak areas, and improve the winning rate.

Visualizing the APF state feature

To explain the superiority of APF-RL, we show the APF feature representation in Fig. 5 at *DefeatRoaches*. Here, we only show the APF feature for the *shooting power* property

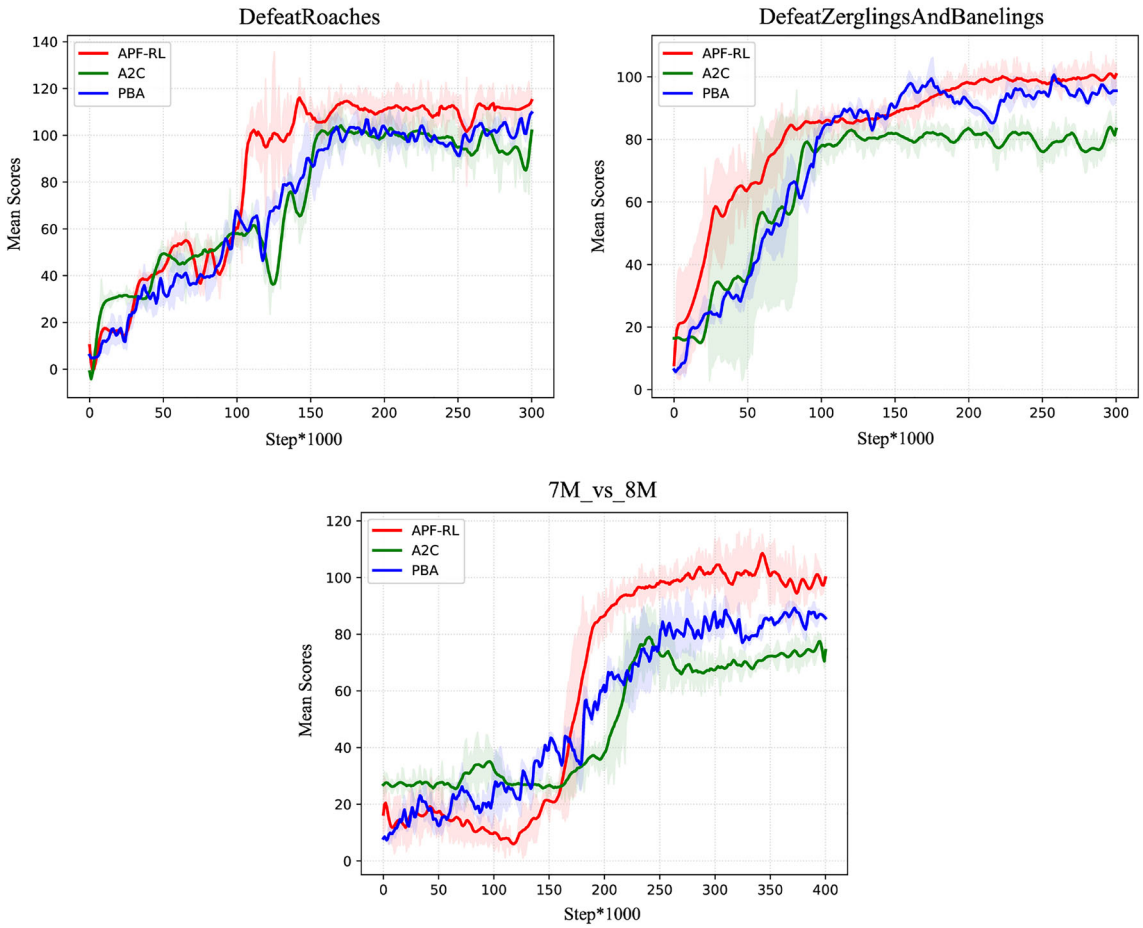

Fig. 4 The mean scores for A2C and APF-RL on three mini-games

layer. Note that when we calculate the potential field, the enemy units always generate a negative potential field, and our units generate a positive potential field. Therefore, the area where the potential field value is zero is the boundary of engagement between the enemy and us. In *DefeatRoaches*, APF-RL learns to prioritize attacking enemy units with the largest potential field value. As shown in Fig. 5, our agent prefers to attack enemy edge units. It is because the potential field value is greater in the area where the unit is located, which means that the threat level of units in that area is relatively low and belongs to the enemy's weak area.

Through the potential field feature, the agent perceives the current situation information of the environment. Based on the situational information, the agent can accurately identify the weak areas of the enemy and learn to prioritize attacking weak areas. Moreover, the APF feature representation not only guides the agent's actions but also conforms to basic human cognition and provides a degree of interpretability for high-performance policies.

Ablation study

To investigate the effect of the two components (APF-SA and APF-IR) on policy performance, we evaluate two variants of APF-RL in an ablation study. These two variants are APF-RL-WSA and APF-RL-WIR. APF-RL-WIR is APF-RL without APF-IR, meaning that the reward is the original sparse reward r^{origin} and not the proxy reward r^{proxy}. APF-RL-WSA is APF-RL without APF-SA, meaning that the potential field feature is only used to generate an intrinsic reward and is not used as input to the neural network. The input to the network uses only the original state feature. We conduct experiments on the mini-games of *DefeatRoaches* and *DefeatZerglingsAndBanelings*.

As shown in Fig. 6 by comparing APF-RL and APF-RL-WSA, we can see that the removal of APF-SA causes a drop in performance. Moreover, when comparing APF-RL and APF-RL-WIR, we can see that removing APF-IR leads to slower convergence of the policy. These experimental results show that APF-SA effectively extracts features for the agent's

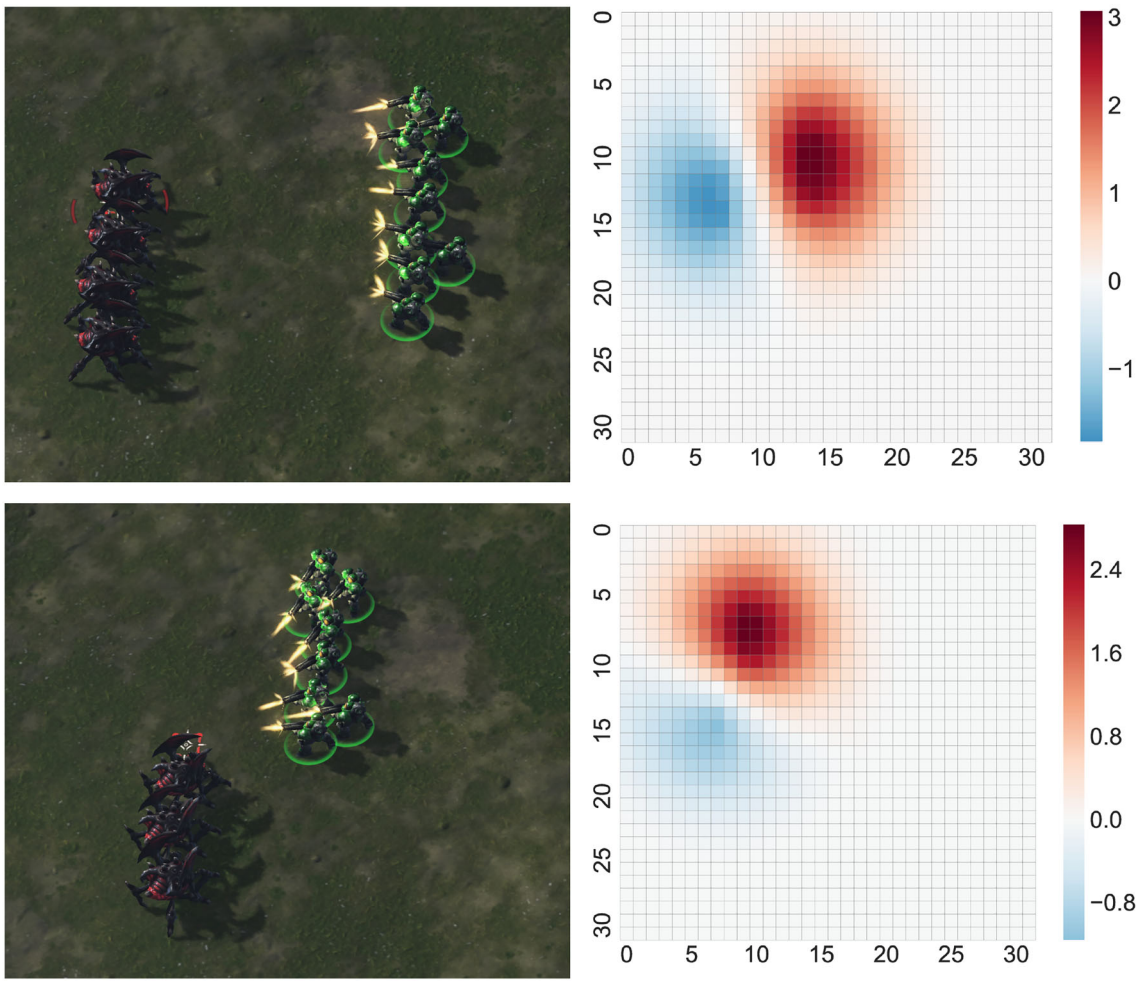

Fig. 5 APF feature representation on *DefeatRoaches*

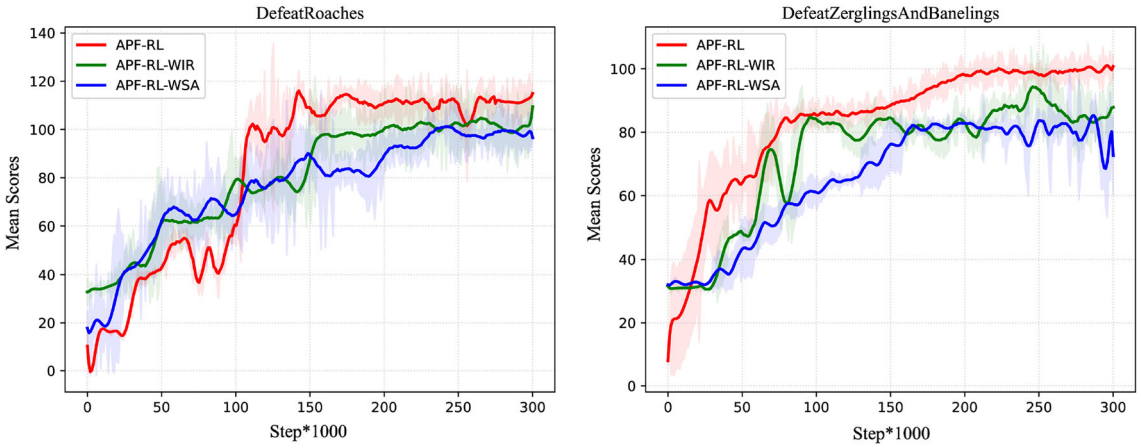

Fig. 6 Mean scores for APF-RL and ablations on *DefeatRoaches* and *DefeatZerglingsAndBaneling*

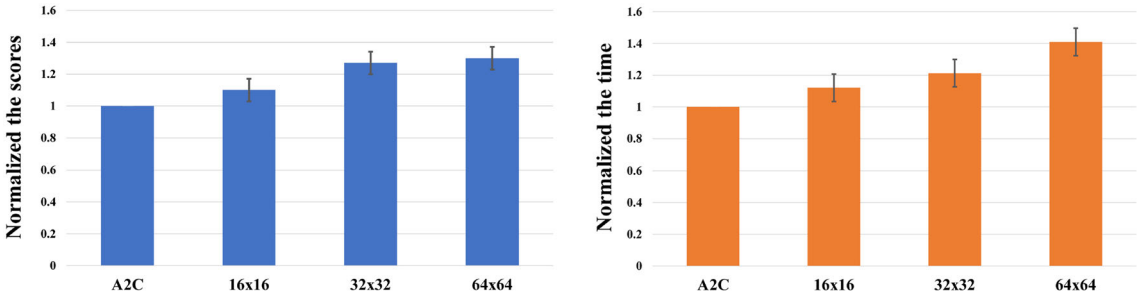

Fig. 7 Comparison between A2C and the APF-RL method with different discrete potential field feature H^{apf} on the mini-games of *7M_vs_8M*. We scale the results in terms of the time consumed by the A2C algorithm. **a** Each bar shows the scaled scores. **b** Each bar shows the scaled time consumption

policy learning, and the intrinsic reward can guide policy learning and accelerate policy convergence.

To explore the impact of different granularity of potential field feature on policy performance and time consumption, we set the discrete potential field feature H^{apf} to $4 \times 64 \times 64$, $4 \times 32 \times 32$ and $4 \times 16 \times 16$ respectively. The comparison experiments are conducted on the mini-games of *7M_vs_8M* with 5 different random seeds. We obtained the time consumed and the score of the policy by the A2C method and the APF-RL method with different H^{apf} settings for running 300K steps. The results show that in Fig. 7a, as the granularity of the discrete potential field feature H^{apf} gets finer ($16 \rightarrow 32 \rightarrow 64$), the performance of the policies learned by the agent gets better. This is because the finer the granularity of the potential field feature, the more information they contain. Conversely, computing fine-grained potential field feature also requires more time and computational resources. We can see from Fig. 7b that as the granularity of the discrete potential field features gets finer, the time consumed by the computation grows faster.

To balance performance and time-consuming, we believe it is more suitable to set H^{apf} to 32×32.

Conclusion

In this work, we propose a novel method called APF-RL to accelerate the learning process of state features and reduce invalid exploration. This method consists of two parts: (1) APF-SA introduces human knowledge to convert environmental features into potential field features. (2) APF-IR generates an intrinsic reward to reduce the invalid exploration and guide the learning of the agent's policy based on the potential field feature. We evaluate our method in PySC2 with some mini-games. The experimental results show that our APF-RL method can improve learning efficiency. The drawback of this work is that the potential field function needs to be specific to the scenario or task, which requires the designer to have a good understanding of the environment or task. In future work, we will design a more general and pervasive method to automatically generate potential fields and apply the potential field-based representation methods to more challenging tasks, such as the full game of StarCraft II.

Acknowledgements This paper is supported by the National Natural Science Foundation of China (Grant no. 11901578).

Data availability statement The data that support the findings of this study are available from the corresponding author, [Shengze LI], upon reasonable request.

Declarations

Conflict of interest We all declare that we have no conflict of interest in this paper.

References

1. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
2. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M et al (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489
3. Zhang F, Leitner J, Milford M, Upcroft B, Corke P (2015) Towards vision-based deep reinforcement learning for robotic motion control. arXiv preprint arXiv:1511.03791
4. Chen J, Yuan B, Tomizuka M (2019) Model-free deep reinforcement learning for urban autonomous driving. In: 2019 IEEE intelligent transportation systems conference (ITSC). IEEE, pp 2765–2771
5. Chen X, Li S, Li H, Jiang S, Qi Y, Song L (2019) Generative adversarial user model for reinforcement learning based recommendation system. In: International conference on machine learning. PMLR, pp 1052–1061
6. Chen M, Beutel A, Covington P, Jain S, Belletti F, Chi E.H (2019) Top-k off-policy correction for a reinforce recommender system. In: Proceedings of the twelfth ACM International conference on web search and data mining, pp 456–464
7. Stojanovic V, He S, Zhang B (2020) State and parameter joint estimation of linear stochastic systems in presence of faults and non-gaussian noises. Int J Robust Nonlinear Control 30(16):6683–6700
8. Hu Z, Ma X, Liu Z, Hovy E, Xing E (2016) Harnessing deep neural networks with logic rules. arXiv preprint arXiv:1603.06318
9. Fischer M, Balunovic M, Drachsler-Cohen D, Gehr T, Zhang C, Vechev M (2019) Dl2: training and querying neural networks with logic. In: International conference on machine learning. PMLR, pp 1931–1941
10. Hester T, Vecerik M, Pietquin O, Lanctot M, Schaul T, Piot B, Sendonaris A, Dulac-Arnold G, Osband I, Agapiou JP, Leibo JZ, Gruslys A (2017) A Learning from demonstrations for real world reinforcement learning. arXiv:1704.03732
11. Zhang P, Hao J, Wang W, Tang H. Ma Y, Duan Y, Zheng Y (2020) Kogun: accelerating deep reinforcement learning via integrating human suboptimal knowledge. arXiv preprint arXiv:2002.07418
12. Xin X, Tu Y, Stojanovic V, Wang H, Shi K, He S, Pan T (2022) Online reinforcement learning multiplayer non-zero sum games of continuous-time Markov jump linear systems. Appl Math Comput 412:126537
13. Zhuang Z, Tao H, Chen Y, Stojanovic V, Paszke W (2022) An optimal iterative learning control approach for linear systems with nonuniform trial lengths under input constraints. IEEE Trans Syst Man Cybern Syst. https://doi.org/10.1109/TSMC.2022.3225381
14. Xie L-J, Xie G-R, Chen H-W, Li X-L (2008) Solution to reinforcement learning problems with artificial potential field. J Cent South Univ Technol 15(4):552–557
15. Noguchi Y, Maki T (2019) Path planning method based on artificial potential field and reinforcement learning for intervention auvs. In: 2019 IEEE underwater technology (UT). IEEE, pp 1–6
16. Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. Int J Robot Res 5(1):90–98
17. Vinyals O, Ewalds T, Bartunov S, Georgiev P, Vezhnevets A.S, Yeo M, Makhzani A, Küttler H, Agapiou J, Schrittwieser J et al (2017) Starcraft ii: a new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782
18. François-Lavet V, Henderson P, Islam R, Bellemare M.G, Pineau J (2018) An introduction to deep reinforcement learning. arXiv preprint arXiv:1811.12560
19. Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T et al (2020) Mastering atari, go, chess and shogi by planning with a learned model. Nature 588(7839):604–609
20. Srinivas A, Laskin M, Abbeel P (2020) Curl: contrastive unsupervised representations for reinforcement learning. arXiv preprint arXiv:2004.04136
21. Anand A, Racah E, Ozair S, Bengio Y, Côté M-A, Hjelm RD (2019) Unsupervised state representation learning in atari. arXiv preprint arXiv:1906.08226
22. Stooke A, Lee K, Abbeel P, Laskin M (2021) Decoupling representation learning from reinforcement learning. In: International conference on machine learning, pp 9870–9879
23. Dorigo M, Colombetti M (1998) Robot shaping: an experiment in behavior engineering. MIT Press, Cambridge
24. Ng AY, Harada D, Russell S (1999) Policy invariance under reward transformations: theory and application to reward shaping. Icml 99:278–287
25. Wiewiora E, Cottrell GW, Elkan C (2003) Principled methods for advising reinforcement learning agents. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp 792–799
26. Devlin S.M, Kudenko D (2012) Dynamic potential-based reward shaping. In: Proceedings of the 11th international conference on autonomous agents and multiagent systems. IFAAMAS, pp 433–440
27. Harutyunyan A, Devlin S, Vrancx P, Nowé A (2015) Expressing arbitrary reward functions as potential-based advice. In: Proceedings of the AAAI conference on artificial intelligence, vol 29
28. Hu Y, Wang W, Jia H, Wang Y, Chen Y, Hao J, Wu F, Fan C (2020) Learning to utilize shaping rewards: a new approach of reward shaping. Adv Neural Inf Process Syst 33:15931–15941
29. Sutton RS, Barto AG et al (1998) Introduction to reinforcement learning, vol 135. MIT Press, Cambridge
30. Puterman ML (2014) Markov decision processes: discrete stochastic dynamic programming. Wiley, New York
31. Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M (2014) Deterministic policy gradient algorithms. In: International conference on machine learning. PMLR, pp 387–395

32. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347
33. Tesauro G et al (1995) Temporal difference learning and td-gammon. Commun ACM 38(3):58–68
34. Schulman J, Moritz P, Levine S, Jordan M, Abbeel P (2015) High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438
35. Kim J, Scott CD (2012) Robust kernel density estimation. J Mach Learn Res 13(1):2529–2565
36. Simonmeister (2018) pysc2-rl-agents. https://github.com/simonmeister/pysc2-rl-agents. Accessed 28 Apr 2018
37. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980