SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2024, Vol. 67, Iss. 7, 172206:1–172206:10
https://doi.org/10.1007/s11432-023-4059-6

Continuous advantage learning for minimum-time trajectory planning of autonomous vehicles

Zhuo LI[1,2], Weiran WU[1,2], Jialin WANG[3], Gang WANG[1,2] & Jian SUN[1,2*]

[1]*School of Automation, Beijing Institute of Technology, Beijing* 100081*, China;*
[2]*Beijing Institute of Technology Chongqing Innovation Center, Chongqing* 401120*, China;*
[3]*China Academy of Launch Vehicle Technology, Intelligent Game and Decision Laboratory, Beijing* 100071*, China*

Received 19 May 2023/Revised 16 August 2023/Accepted 3 February 2024/Published online 25 June 2024

Abstract This paper investigates the minimum-time trajectory planning problem of an autonomous vehicle. To deal with unknown and uncertain dynamics of the vehicle, the trajectory planning problem is modeled as a Markov decision process with a continuous action space. To solve it, we propose a continuous advantage learning (CAL) algorithm based on the advantage-value equation, and adopt a stochastic policy in the form of multivariate Gaussian distribution to encourage exploration. A shared actor-critic architecture is designed to simultaneously approximate the stochastic policy and the value function, which greatly reduces the computation burden compared to general actor-critic methods. Moreover, the shared actor-critic is updated with a loss function built as mean square consistency error of the advantage-value equation, and the update step is performed several times at each time step to improve data efficiency. Simulations validate the effectiveness of the proposed CAL algorithm and its better performance than the soft actor-critic algorithm.

Keywords trajectory planning, continuous advantage learning, stochastic policy, shared actor-critic

1 Introduction

Autonomous vehicles [1] have found wide applications in source seeking [2], field exploration [3], and target tracking [4]. To complete the tasks efficiently, their minimum-time trajectory planning problems have been paid much attention to. Since an autonomous vehicle always has nonlinear, coupled, and uncertain dynamics, model-based methods cannot be directly applied, such as model predictive control [5], sliding mode control [6], and backstepping control [7]. To overcome the challenges, model-free reinforcement learning (RL) has been greatly developed in recent years [8–10], which requires no prior knowledge of a vehicle's dynamics and can learn an optimal policy only by interactions with environments.

Existing RL algorithms can be classified into three categories: value-based, policy-based, and actor-critic algorithms. Q-learning algorithms and their extension, deep Q-network (DQN) algorithms, are typical ones of value-based RL algorithms that learn a value function from which the policy is derived [11]. A variant of DQN called Dueling DQN is developed in [12] based on the advantage-value equation, which leads to better policy evaluation in the presence of many similar-valued actions. Nevertheless, a value-based RL algorithm is mainly applied to a Markov decision process (MDP) with a discrete action space. Thus, the discretization technique has to be employed for a trajectory planning problem with a continuous action space [13, 14], which results in a high-dimensional action space and heavy computational burden for training an optimal policy.

To deal with the tasks with continuous action spaces, policy-based RL algorithms have been thoroughly investigated, which do not need to derive a value function and can learn continuous policies directly [15,16]. However, they might be highly unstable, inefficient, and difficult to implement, especially in the cases with intractable policy gradients. Combining advantages of value-based and policy-based algorithms, actor-critic RL algorithms employ an actor to take an action under a state by the current policy and a critic to approximate the value function for evaluating this policy [17, 18]. In this work, we aim to design an actor-critic RL-based trajectory planning algorithm to learn a continuous policy.

* Corresponding author (email: sunjian@bit.edu.cn)

Under the framework of RL, the minimum-time trajectory planning problem needs to be remodeled as an MDP. Unfortunately, there always exist similar-valued actions, especially with the constraint of obstacle avoidance. To resolve this issue, this work leverages advantages of the dueling architecture based on the advantage-value function, and proposes a continuous advantage learning (CAL) algorithm for the minimum-time trajectory planning, which can be seen as a continuous version of dueling DQN. To further improve the data efficiency and accelerate the learning process, we design the update step for policy training to be performed several times at each time step, and thus a data-efficient CAL algorithm can be obtained for online trajectory planning of vehicles in practice.

Another challenge of learning the continuous planning policy is the exploration in an infinite dimensional action space. In existing studies, deep deterministic policy gradient (DDPG) [19] and its extended algorithm called twin delayed deep deterministic policy gradient (TD3) [20] employ deterministic policy gradients for the tasks with continuous action spaces. In particular, TD3 is one of state-of-the-art (SOTA) RL algorithms for continuous control tasks, which utilizes two sets of actor-critic structures. Normalized advantage function (NAF) algorithms are proposed in [21], where a parameterized advantage function is adopted to deal with continuous control problems. It is worth noting that DDPG, TD3, and NAF algorithms require well-designed exploration strategies, leading to the problem of inefficient exploration.

To motivate exploration of the planning policy, a stochastic policy is designed in the form of Gaussian distribution for the CAL algorithm of this work. A stochastic actor has been designed in soft actor-critic (SAC) [22, 23] with an entropy maximization objective, which provides a substantial improvement in exploration. Nonetheless, there are at least three neural networks in the SAC algorithm for approximations of a state value function, a soft Q-function, and a tractable policy, implying high computational cost and low data efficiency. To save computations, this work adopts a shared actor-critic architecture for the proposed algorithm, such that the shared parameters of neural networks can be simultaneously updated and the computational burden can be reduced.

Our main contributions can be summarized as follows.

• An RL-based algorithm called CAL is devised to learn a continuous policy for the minimum-time trajectory planning problem, which can be seen as a continuous version of dueling DQN [12] and can handle similar-valued actions. The proposed CAL is more applicable to the vehicle with a continuous action space than the discrete algorithms in [11–14].

• A stochastic policy is designed in the CAL algorithm with a form of multivariate Gaussian distribution, which can provide more efficient exploration than the deterministic policies in [19–21].

• A shared actor-critic architecture is built in the CAL algorithm to simultaneously approximate the policy and the value function, which can reduce the computational cost than those under a separate actor-critic architecture [17, 18].

The rest of this paper is organized as follows. Section 2 formulates the vehicle's minimum-time trajectory planning problem and provides its MDP modeling. The CAL algorithm is proposed in Section 3, and the experimental validations are presented in Section 4. Finally, Section 5 draws the conclusion.

2 Problem formulation

2.1 Minimum-time trajectory planning problem

Consider an autonomous vehicle to reach a target position, and let $\boldsymbol{p}(t) \in \mathbb{R}^3$ and $\boldsymbol{p}_{\rm t} \in \mathbb{R}^3$ denote the vehicle and the target positions in a three-dimensional (3D) environment. We aim to design the control input $\boldsymbol{u}(t) \in \mathbb{R}^m$ for the vehicle to reach the target position as soon as possible. This minimum-time trajectory planning problem can be mathematically given by

$$\underset{\boldsymbol{u}(t)}{\text{minimize}}\ t_{\rm f} \tag{1a}$$

$$\text{subject to } \boldsymbol{p}(t_0) = \boldsymbol{p}_0, \boldsymbol{p}(t_{\rm f}) = \boldsymbol{p}_{\rm t}, \tag{1b}$$

$$\dot{\boldsymbol{p}}(t) = f(\boldsymbol{p}(t), \boldsymbol{u}(t)), \tag{1c}$$

$$\boldsymbol{p}(t) \in \Omega,\ \forall t \in [t_0, t_{\rm f}], \tag{1d}$$

$$\boldsymbol{u}(t) \in \mathcal{U},\ \forall t \in [t_0, t_{\rm f}], \tag{1e}$$

where $t_{\rm f}$ denotes the final time that the vehicle reaches the target position $\boldsymbol{p}_{\rm t}$ from the initial position $\boldsymbol{p}_0$ at time t_0, $f(\cdot, \cdot)$ denotes the vehicle's dynamical model, $\Omega \subset \mathbb{R}^3$ denotes a free space for the vehicle

without any obstacles, and $\mathcal{U} \subset \mathbb{R}^3$ denotes a bounded set for the control input.

Note that an explicit expression of the dynamical constraint (1c) is unknown and might be nonlinear, coupled, and uncertain. As a result, general nonlinear programming methods cannot be directly applied to problem (1). To overcome it, this work aims to design a model-free RL-based trajectory planning algorithm, which requires no prior knowledge of the vehicle's dynamical model.

2.2 MDP modeling

Under the framework of RL, the minimum-time trajectory planning problem in (1) is modeled as an infinite-horizon discounted MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{R}$ is a reward space, and $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$ is the transition probability distribution.

By discretization of the vehicle's dynamical model in (1c), the control input $\boldsymbol{u}(k)$ is naturally selected as the action $\boldsymbol{a}_k \in \mathcal{A}$, and the state is set as $\boldsymbol{s}_k = \boldsymbol{p}(k) - \boldsymbol{p}_\mathrm{t} \in \mathcal{S}$. For the objective of minimum final time, we design the reward function as

$$r_k = r(\boldsymbol{s}_k, \boldsymbol{a}_k) = w_0^k r_\mathrm{t} + w_1 r_\mathrm{p}(k) + w_2 r_\mathrm{o}(k), \tag{2}$$

where the weight parameters $w_0 > 1$, $w_1, w_2 > 0$, a constant $r_\mathrm{t} < 0$ denotes the time cost, $r_\mathrm{p}(k)$ denotes the reward for approaching the target position, and $r_\mathrm{o}(k) \leqslant 0$ is the cost for approaching an obstacle.

We define $r_\mathrm{p}(k)$ by the variation of distance between the vehicle and the target position, i.e.,

$$r_\mathrm{p}(k) = \|\boldsymbol{p}(k-1) - \boldsymbol{p}_\mathrm{t}\| - \|\boldsymbol{p}(k) - \boldsymbol{p}_\mathrm{t}\|. \tag{3}$$

It implies that only the relative distance is measured for computing the reward, and no absolute positions of the vehicle and the target are required in the inertial coordinate. The cost term $r_\mathrm{o}(k)$ for obstacle avoidance is designed as

$$r_\mathrm{o}(k) = \begin{cases} 0, & \text{if } \|\boldsymbol{p}(k) - \boldsymbol{p}_\mathrm{o}\| > d_\mathrm{s}, \\ -1/\|\boldsymbol{p}(k) - \boldsymbol{p}_\mathrm{o}\|, & \text{otherwise}, \end{cases} \tag{4}$$

where $\boldsymbol{p}_\mathrm{o}$ is the position of the obstacle closest to the vehicle, and d_s denotes a safe distance between the vehicle and the obstacle. By the reward in (4), the vehicle is motivated to deviate from the minimum-time trajectory if there exist some obstacles. Moreover, if d_s is too large, the vehicle might take a long time to arrive at the target position. In the case of too small d_s, the vehicle is at great risk of hitting obstacles.

The parameter w_0 in (2) is usually set to be slightly larger than 1; i.e., the weight of r_t increases exponentially over the time step k, which emphasizes the objective of minimum-time trajectory planning. The parameter w_1 can be decreased when the vehicle approaches the target position, which can accelerate the process of policy training. In practice, we usually set w_1 to be a constant for simplicity, and set w_2 to be a constant with a large value since the vehicle's safety must be ensured all the time.

3 Continuous advantage learning algorithm

This section proposes the CAL trajectory planning algorithm based on the advantage-value equation to solve the MDP in Subsection 2.2. Thus, the vehicle can learn an optimal policy π^* that maximizes the expected return $\mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{k+1}]$, where $\gamma \in (0, 1]$ is a discount factor.

3.1 Advantage equation

Under a policy π, the value function of a state $\boldsymbol{s}_k$ is defined as the expected return when starting in state $\boldsymbol{s}_k$, which is defined as

$$V^\pi(\boldsymbol{s}_k) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i r_{i+k+1} | \boldsymbol{s}_k \right], \tag{5}$$

and the action-value function is defined as the expected return for taking action $\boldsymbol{a}_k$ in state $\boldsymbol{s}_k$, i.e.,

$$Q^\pi(\boldsymbol{s}_k, \boldsymbol{a}_k) = \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i r_{i+k+1} | \boldsymbol{s}_k, \boldsymbol{a}_k \right]. \tag{6}$$

An advantage function represents the degree to which the expected return is increased by performing action $\boldsymbol{a}_k$ in state $\boldsymbol{s}_k$ (followed by current policy π thereafter) relative to the action $\pi(\boldsymbol{s}_k)$ directly followed by policy π. That is,

$$A^{\pi}(\boldsymbol{s}_k, \boldsymbol{a}_k) = Q^{\pi}(\boldsymbol{s}_k, \boldsymbol{a}_k) - Q^{\pi}(\boldsymbol{s}_k, \pi(\boldsymbol{s}_k)) = Q^{\pi}(\boldsymbol{s}_k, \boldsymbol{a}_k) - V^{\pi}(\boldsymbol{s}_k). \tag{7}$$

Together with (5) and (6), it holds that

$$Q^{\pi}(\boldsymbol{s}_k, \boldsymbol{a}_k) = r_{k+1} + \gamma V^{\pi}(\boldsymbol{s}_{k+1}). \tag{8}$$

Then, we substitute (8) into the advantage function in (7), and obtain the advantage-value equation that

$$A^{\pi}(\boldsymbol{s}_k, \boldsymbol{a}_k) = r_{k+1} + \gamma V^{\pi}(\boldsymbol{s}_{k+1}) - V^{\pi}(\boldsymbol{s}_k), \tag{9}$$

which builds a direct relation between the advantage and the value functions, and makes it feasible to iteratively approximate the value function. Moreover, the advantage-value equation (9) is independent of any expectation, which simplifies the sampling process.

Since advantage-value equation (9) is satisfied for any policy π and its corresponding V^{π} and A^{π}, it might be unstable to simultaneously approximate π, V^{π}, and A^{π} by independent neural networks; i.e., the convergence of learning the advantage function cannot be ensured. A feasible solution is to reformulate advantage function A^{π} to be associated with policy π, instead of directly approximating it by a neural network. To achieve it, we design an approximation of policy π to reformulate advantage function A^{π}.

A stochastic policy π is adopted in the following form of multivariate Gaussian distribution as

$$\pi : \boldsymbol{a}_k \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{s}_k), \Sigma(\boldsymbol{s}_k)), \tag{10}$$

where $\boldsymbol{\mu}(\boldsymbol{s}_k)$ and $\Sigma(\boldsymbol{s}_k)$ are the expectation vector and covariance matrix of the Gaussian distribution in state $\boldsymbol{s}_k$. Thus, all actions will be selected with a probability greater than zero to motivate the vehicle to explore in the continuous action space and to produce robust behaviors under disturbances. In contrast, a deterministic policy is always less efficient in exploration, e.g., the deterministic policy $\boldsymbol{\mu}(\boldsymbol{s}_k) + \boldsymbol{n}$ with an additive stochastic noise $\boldsymbol{n}$ in the continuous Q-learning algorithm NAF [21].

Now, we reformulate the advantage function A^{π}, which evaluates the quality of executing an action. It follows from its definition in (7) that it must satisfy the following basic properties:

(a) $A^{\pi^*}(\boldsymbol{s}_k, \pi^*(\boldsymbol{s}_k)) = 0,\ \forall\ \boldsymbol{s}_k \in \mathcal{S}$.

(b) $A^{\pi^*}(\boldsymbol{s}_k, \boldsymbol{a}_k) \leqslant 0,\ \forall\ (\boldsymbol{s}_k, \boldsymbol{a}_k) \in (\mathcal{S}, \mathcal{A})$.

To achieve the advantage function associated with the policy in (10), our idea is to represent A^{π} such that the maximum of Q-function in (8) can be determined as $\boldsymbol{\mu}(\boldsymbol{s}_k)$ in (10) at each Q-learning update. Thus, by jointly considering the above properties and the Gaussian policy in (10), we parameterize advantage function $A^{\pi}(\boldsymbol{s}_k, \boldsymbol{a}_k)$ as a quadratic function of expectation $\boldsymbol{\mu}(\boldsymbol{s}_k)$ and covariance $\Sigma(\boldsymbol{s}_k)$, i.e.,

$$A^{\pi}(\boldsymbol{s}_k, \boldsymbol{a}_k) = -\frac{1}{2}(\boldsymbol{a}_k - \boldsymbol{\mu}(\boldsymbol{s}_k))^{\mathrm{T}} \Sigma^{-1}(\boldsymbol{s}_k)(\boldsymbol{a}_k - \boldsymbol{\mu}(\boldsymbol{s}_k)). \tag{11}$$

Clearly, this quadratic form of advantage function (11) is a more restrictive representation than general neural networks, and can be iteratively approximated. Moreover, this formulation can be seen as the Mahalanobis distance between $\boldsymbol{a}_k$ and $\boldsymbol{\mu}(\boldsymbol{s}_k)$. Substituting (11) into advantage-value equation (9), we can yield the recursive relation that

$$V^{\pi}(\boldsymbol{s}_k) = r_{k+1} + \gamma V^{\pi}(\boldsymbol{s}_{k+1}) + \frac{1}{2}(\boldsymbol{a}_k - \boldsymbol{\mu}(\boldsymbol{s}_k))^{\mathrm{T}} \Sigma^{-1}(\boldsymbol{s}_k)(\boldsymbol{a}_k - \boldsymbol{\mu}(\boldsymbol{s}_k)), \tag{12}$$

which is the core equality in our proposed CAL algorithm in Subsection 3.3. Moreover, policy π and value function $V^{\pi}(\boldsymbol{s}_k)$ are coupled and can be learned, simultaneously.

3.2 Shared actor-critic architecture

It follows from (12) that $\pi(\boldsymbol{s}_k)$ and $V^{\pi}(\boldsymbol{s}_k)$ might have identical information in hidden layers of two networks due to the same input $\boldsymbol{s}_k$. Thus, we design a shared actor-critic architecture with a shared full-connected network in Figure 1, where the input layer and the front part of hidden layers are shared. The actor and the critic are approximated by $\pi(\boldsymbol{s}_k|\omega_{\mathrm{s}}, \omega_{\mathrm{a}})$ and $V(\boldsymbol{s}_k|\omega_{\mathrm{s}}, \omega_{\mathrm{c}})$ with ω_{s} as the parameters of

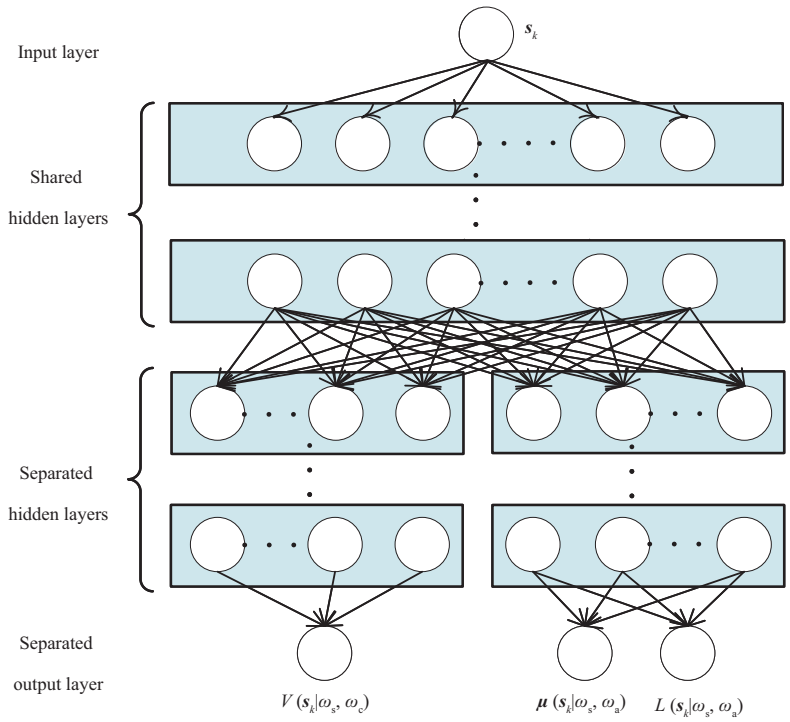

Figure 1 (Color online) Shared actor-critic architecture based on a full-connected neural network.

shared layers, $\omega_{\rm a}$ and $\omega_{\rm c}$ as the parameters of separated layers. Hence, the shared actor-critic architecture requires lower computational cost with fewer parameters than general actor-critic architectures.

According to the policy in (10), it holds that

$$\pi(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a}):\boldsymbol{a}_k\sim\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a}),\Sigma(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})). \tag{13}$$

Since the covariance $\Sigma(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})$ is a positive semi-definite matrix, it can be decomposed into a lower triangular matrix $L(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})$ and an upper triangular matrix $L(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})^{\rm T}$ by Cholesky factorization, which gives that

$$\Sigma(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})=L(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})^{\rm T}L(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a}). \tag{14}$$

Remark 1. The Cholesky factorization is adopted mainly for saving computations. For example of $\boldsymbol{a}_k\in\mathbb{R}^{n_{\rm a}}$, the critic neural network outputs $n_{\rm a}(n_{\rm a}+1)/2$ elements for $L(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})$ to compute $\Sigma(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})$ by (14). Otherwise, it has to output $n_{\rm a}^2$ elements to obtain $\Sigma(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})$ directly.

Jointly considering (13) and (14), we can deduce the advantage function in (11) that

$$A(\boldsymbol{s}_k,\boldsymbol{a}_k|\omega_{\rm s},\omega_{\rm a})=-\frac{1}{2}Y(\boldsymbol{s}_k,\boldsymbol{a}_k)^{\rm T}Y(\boldsymbol{s}_k,\boldsymbol{a}_k), \tag{15}$$

where $Y(\boldsymbol{s}_k,\boldsymbol{a}_k)=L^{-1}(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})(\boldsymbol{a}_k-\boldsymbol{\mu}(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a}))$. Thus, the actor only needs to output the elements of expectation $\boldsymbol{\mu}(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})$ and lower triangular matrix $L(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a})$.

3.3 The CAL algorithm

To learn an optimal policy π^*, the shared actor-critic is trained to minimize the soft consistency error of CAL equation (12) with off-policy experience sampled from a replay buffer. Moreover, a target critic is employed to get consistent target value $V'(\boldsymbol{s}_{k+1}|\omega'_{\rm s},\omega'_{\rm c})$ to ensure stable learning. Thus, the soft consistency error is defined as a function of parameters $\omega_{\rm s}$, $\omega_{\rm a}$, and $\omega_{\rm c}$ as

$$c(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c})=r_{k+1}+\gamma V'(\boldsymbol{s}_{k+1}|\omega'_{\rm s},\omega'_{\rm c})-V(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm c})-A(\boldsymbol{s}_k,\boldsymbol{a}_k|\omega_{\rm s},\omega_{\rm a}). \tag{16}$$

Algorithm 1 Data-efficient CAL trajectory planning problem

1: Initialize a full-connected neural network for the shared actor-critic network with random weights $\omega_{\rm s}$, $\omega_{\rm a}$, and $\omega_{\rm c}$. Let $\omega'_{\rm s}=\omega_{\rm s}$ and $\omega'_{\rm c}=\omega_{\rm c}$, and a replay buffer $\mathcal{B}$ be an empty set;
2: **for** episode = 1, Max-epi **do**
3: Randomly generate an observation state $\boldsymbol{s}_0$;
4: **for** $k=0,K$ **do**
5: **if** $\|\boldsymbol{s}_k\|\leqslant\epsilon$ **then**
6: Break;
7: **end if**
8: Sample an action $\boldsymbol{a}_k$ according to $\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a}),\Sigma(\boldsymbol{s}_k|\omega_{\rm s},\omega_{\rm a}))$;
9: Execute $\boldsymbol{a}_k$ and receive r_k, $\boldsymbol{s}_{k+1}$;
10: Store transition $(\boldsymbol{s}_k,\boldsymbol{a}_k,r_k,\boldsymbol{s}_{k+1})$ into $\mathcal{B}$;
11: **for** iteration = 1, I **do**
12: Sample a random mini-batch of N transitions $(\boldsymbol{s}_k,\boldsymbol{a}_k,r_k,\boldsymbol{s}_{k+1})$ from $\mathcal{B}$;
13: Calculate (15)–(17);
14: Update the network parameters $\omega_{\rm s}$, $\omega_{\rm a}$, and $\omega_{\rm c}$ according to (18);
15: Update the network parameters $\omega'_{\rm s}$ and $\omega'_{\rm c}$ according to (19);
16: **end for**
17: **end for**
18: **end for**

Accordingly, the loss function for the shared actor-critic network is defined as the mean square soft consistency error among N sampled transitions from the replay buffer, i.e.,

$$\mathrm{Loss}(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c})=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}c_i(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c})^2, \tag{17}$$

where c_i is the soft consistency error $c(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c})$ of the i-th sampled transition $(\boldsymbol{s}_i,\boldsymbol{a}_i,r_i,\boldsymbol{s}_{i+1})$ in the replay buffer. The update rules for parameters $\omega_{\rm s}$, $\omega_{\rm a}$, and $\omega_{\rm c}$ are derived by the gradient of the loss function in (17) as follows:

$$\begin{aligned}
\omega_{\rm s}&\leftarrow\omega_{\rm s}+\frac{1}{N}\sum_{i=1}^{N}c_i(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c})\nabla_{\omega_{\rm s}}c_i(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c}),\\
\omega_{\rm a}&\leftarrow\omega_{\rm a}+\frac{1}{N}\sum_{i=1}^{N}c_i(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c})\nabla_{\omega_{\rm a}}c_i(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c}),\\
\omega_{\rm c}&\leftarrow\omega_{\rm c}+\frac{1}{N}\sum_{i=1}^{N}c_i(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c})\nabla_{\omega_{\rm c}}c_i(\omega_{\rm s},\omega_{\rm a},\omega_{\rm c}),
\end{aligned} \tag{18}$$

and the weights of the target critic are softly updated as

$$\omega'_{\rm s}\leftarrow\epsilon\omega_{\rm s}+(1-\epsilon)\omega'_{\rm s},\ \omega'_{\rm c}\leftarrow\epsilon\omega_{\rm c}+(1-\epsilon)\omega'_{\rm c}, \tag{19}$$

where $\epsilon\in[0,1]$ is a scalar to control the update rate.

In implementations, a data-efficient training scheme is applied to make full use of previous transitions or experiences. Specifically, I mini-batches of transitions are sampled to update the shared actor-critic network in each time step. That is, the policy training is performed several times within a single time step to achieve improved data efficiency, which however requires more computations. Thus, the number I needs to take a small value. The data-efficient CAL for minimum-time trajectory planning is summarized in Algorithm 1, which solves an optimal policy for problem (1) by updating the shared actor-critic network. In addition, we set a terminal condition of an episode in line 5 with a small value of $\epsilon>0$.

4 Simulation

We conduct simulations to validate the effectiveness of the proposed CAL trajectory planning algorithm, and its better performance than those of the SAC [22] and the TD3 [20] algorithms.

Table 1 Values of the hyperparameters for the three algorithms

Hyperparameter	TD3	SAC	CAL
Minibatch size	256	256	80
Number of networks[a)]	3	3	1
Number of hidden layer in each network	2	2	2
Number of neurons in the two hidden layers	256, 256	256, 256	150, 100
Learning rate	0.0003	0.0003	0.0003
Target smoothing coefficient τ	0.005	0.005	0.005
Discount γ	0.99	0.99	0.99

a) The target critic networks are not counted here, since their network parameters adopt soft updates.

Table 2 Performance of learned policies by the three algorithms

Index	TD3	SAC	CAL
Success rate	0.74	0.91	**0.99**
$t_{\rm f}(s)$	17.98 ± 3.44	21.74 ± 7.74	$\mathbf{16.84 \pm 4.29}$
$\|\boldsymbol{p}(t_{\rm f}) - \boldsymbol{p}_{\rm t}\|$(m)	0.46 ± 0.03	0.46 ± 0.03	0.48 ± 0.03

4.1 Simulation settings

The Dubins dynamical model in the 3-D space is adopted for the autonomous vehicle, i.e.,

$$\begin{cases} \dot{x}(t) = \nu \cos\theta(t)\cos\phi(t), \\ \dot{y}(t) = \nu \sin\theta(t)\cos\phi(t), \\ \dot{z}(t) = \nu \sin\phi(t), \\ \dot{\theta}(t) = \omega(t), \\ \dot{\phi}(t) = u(t), \end{cases} \tag{20}$$

where $\boldsymbol{p}(t) = [x(t), y(t), z(t)]^{\rm T}$ denotes the vehicle's position, $\theta(t)$ and $\phi(t)$ denote the heading angle and the flight path angle, $\nu = 1$ m/s is the constant forward velocity, and $\boldsymbol{u}(t) = [\omega(t), u(t)]^{\rm T}$ is the control input with $\omega(t) \in [-\pi/6, \pi/6]$ and $u(t) \in [-\pi/6, \pi/6]$. We discretize the dynamical model with a sampling interval 0.1 s for the MDP modeling in Subsection 2.2, and the weight parameters of the reward function are selected as $w_0 = 1.0015, w_1 = 3$, and $w_2 = 0.004$ in (2).

For Algorithm 1, we set Max-epi = 5000, $K = 500$, and $I = 2$. Without loss of generality, a right-handed coordinate is constructed with $\boldsymbol{p}_{\rm t}$ as the origin. The vehicle's initial position $\boldsymbol{p}_0$ is randomly selected to be far away from the desired position in $[10, 15]$ m, and random initial orientation angles θ_0 and ϕ_0 are in $[0, \pi]$ rad. In addition, the obstacles are randomly generated in the 3-D environment.

All the simulations are performed on Apple M1 Pro CPU @3.20 GHz with RAM 16.00 GB, and our implementation employs PyTorch with Python 3.10.0. and torch 1.13.1 + CPU.

4.2 Comparisons with RL algorithms

We compare the proposed CAL algorithm with two SOTA RL alternatives: an SAC-based and a TD3-based trajectory planning algorithms, for continuous control tasks. While TD3 takes a deterministic policy, SAC adopts a similar stochastic actor to that in the proposed CAL algorithm. Moreover, SAC has an entropy maximization objective and has been proven to provide a substantial improvement in exploration. The hyperparameters of the three algorithms are selected as in Table 1, which implies that our proposed CAL has a less lightweight network architecture and takes up fewer memory resources than the other two algorithms.

Figure 2 provides the learning curves of the SAC, the TD3, and the CAL algorithms in five Monte Carlo experiments. Clearly, the CAL algorithm converges more rapidly and becomes much more stable during the later stage of training than the other two algorithms. In particular, the cumulative reward of the CAL is much larger than that of TD3. These results validate the higher learning efficiency of the CAL than the SAC and the TD3 algorithms, and the more efficient exploration of the stochastic policy in the CAL and the SAC than the deterministic one in the TD3 algorithm.

In the testing phase, Table 2 lists some performance indices of the learned policies by 100 Monte Carlo experiments, where the boldface denotes the best performance by the three algorithms. An experiment

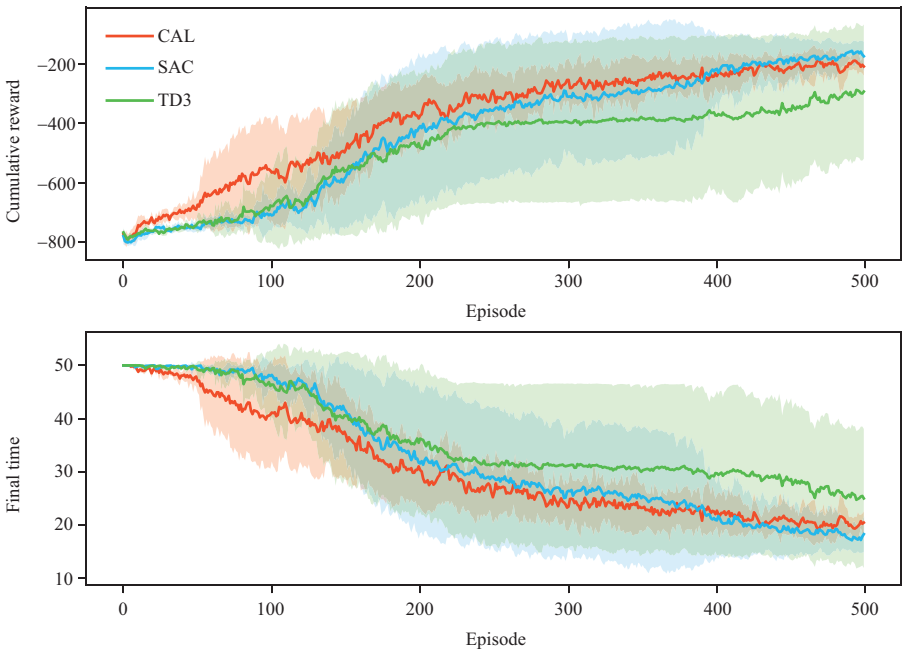

Figure 2 (Color online) Learning curves of the three algorithms.

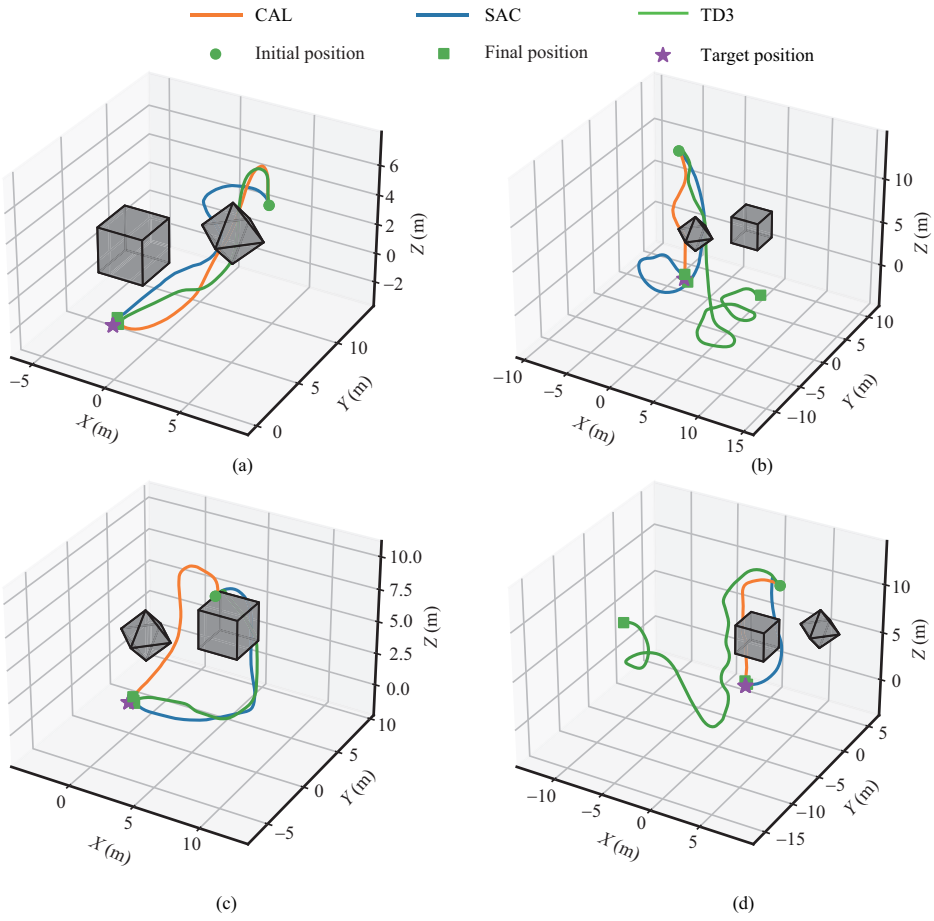

Figure 3 (Color online) Vehicle's trajectories from different initial positions with two obstacles. The final times are (CAL/SAC/TD3) (a) 19.2 s/19.9 s/18.6 s, (b) 14.6 s/32.0 s/failed, (c) 15.2 s/22.5 s/21.1 s, and (d) 14.0 s/15.4 s/failed.

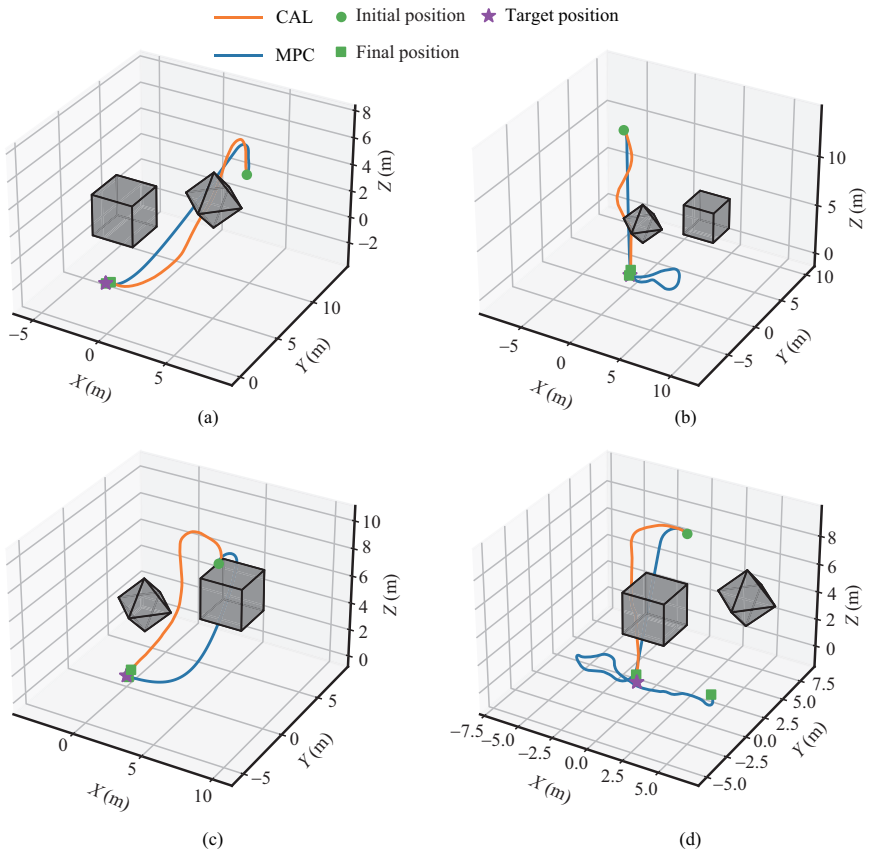

Figure 4 (Color online) Vehicle's trajectories from different initial positions with two dynamic obstacles. The final times are (CAL/MPC) (a) 19.2 s/18.0 s, (b) 14.6 s/25.8 s, (c) 15.2 s/15.4 s, and (d) 14.0 s/failed.

with the final time $t_{\rm f} < 50$ s and the final tracking error $\|\boldsymbol{p}(t_{\rm f}) - \boldsymbol{p}_{\rm t}\| < 0.5$ m is regarded as a success. By Table 2, the CAL has a 33.8% and 9.8% higher success rate than the TD3 and the SAC algorithm, and the vehicle's final time of the CAL is 6.3% and 29.1% smaller than that of the two algorithms. The results validate the effectiveness of the proposed CAL and its better performance than that of the SAC. Moreover, it is worth noting that the final tracking error is only a terminal condition, and its smaller value does not necessarily imply better performance.

To show the performance clearly, we apply the learned policies by the three algorithms to the vehicle, and depict its trajectories from different initial positions. By Figure 3, the vehicle is observed to have acquired the ability to avoid obstacles through the three algorithms, where the CAL usually outperforms the SAC in terms of the final time. Moreover, although the TD3 sometimes takes the vehicle less time to approach the target position than the CAL algorithm, its performance is not stable as in Figures 3(b) and (d) with a success rate 0.74 as in Table 2.

4.3 Comparisons with a model-based method

In this subsection, the proposed model-free CAL algorithm is compared with a commonly-used model-based trajectory planning method, model predictive control (MPC) [5], to show the effectiveness and advantages. For the MPC method, we need to minimize the following objective function at each time t,

$$\underset{\omega(k),u(k)}{\text{minimize}} \quad \sum_{k=0}^{h} \frac{1}{2}(\boldsymbol{p}(k) - \boldsymbol{p}_{\rm t})^{\rm T} Q(\boldsymbol{p}(k) - \boldsymbol{p}_{\rm t}),$$

where $\boldsymbol{p}(0) = \boldsymbol{p}(t)$, the vehicle's state $\boldsymbol{p}(k), \theta(k), \phi(k)$ and control input $\omega(k), u(k)$ are subject to (20), (1d), and (1e) in discrete time. In the simulations, we select $Q = \text{diag}(5, 5, 5)$ and $h = 5$, and set the same terminal condition for the MPC method as that in the CAL. Applying the MPC method to the same scenarios in Figure 3, we obtain the vehicle's trajectories in Figure 4. Clearly, the final times of the CAL algorithm are about as large as or smaller than those of the MPC method. The results validate

that the proposed model-free CAL algorithm has similar performance as the MPC method that requires the dynamical model of the vehicle. In addition, while the learned policy by the CAL can be applied in real time, the MPC method needs more time for solving the optimization problem at each time step.

5 Conclusion

This paper has proposed the CAL algorithm for the minimum-time trajectory planning problem of autonomous vehicles based on the advantage-value equation. The continuous and stochastic planning policy was designed to motivate the exploration, and the shared actor-critic architecture was constructed with fewer parameters to reduce the computational cost than general actor-critic ones. Moreover, the proposed algorithm was data-efficient by setting the training step to be performed several times at each time step. Simulations verified the effectiveness of the proposed algorithm by comparison of the MPC method, and showed its better performance than those of the SAC-based and the TD3-based trajectory planning algorithms. In the future, we would like to consider more constraints and objectives for the trajectory planning of a vehicle, such as energy assumption. A distributed CAL algorithm will be proposed for multi-vehicle systems, which is data-efficient in policy training and applicable in the networks with low bandwidth.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2022ZD0119302) and National Natural Science Foundation of China (Grant Nos. 61925303, 62173034, 62088101, 62303054, U20B2073).

References

1 Chen J, Sun J, Wang G. From unmanned systems to autonomous intelligent systems. Engineering, 2022, 12: 16–19
2 Li Z, You K Y, Song S J. Cooperative source seeking via networked multi-vehicle systems. Automatica, 2020, 115: 108853
3 Cheng S, Paley D A. Optimal guidance and estimation of a 2D diffusion-advection process by a team of mobile sensors. Automatica, 2022, 137: 110112
4 Li Y F, Wang X, Sun J, et al. Data-driven consensus control of fully distributed event-triggered multi-agent systems. Sci China Inf Sci, 2023, 66: 152202
5 Ji J, Khajepour A, Melek W W, et al. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. IEEE Trans Veh Technol, 2016, 66: 952–964
6 Liu Y, Wang Y, Guan X, et al. Direction and trajectory tracking control for nonholonomic spherical robot by combining sliding mode controller and model prediction controller. IEEE Robot Autom Lett, 2022, 7: 11617–11624
7 Chen Z, Helian B, Zhou Y, et al. An integrated trajectory planning and motion control strategy of a variable rotational speed pump-controlled electro-hydraulic actuator. IEEE ASME Trans Mechatron, 2022, 28: 588–597
8 Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge: MIT Press, 2018
9 O'Connell M, Shi G Y, Shi X C, et al. Neural-fly enables rapid learning for agile flight in strong winds. Sci Robot, 2022, 7: eabm6597
10 Dong L, He Z, Song C, et al. A review of mobile robot motion planning methods: from classical motion planning workflows to reinforcement learning-based architectures. 2021. ArXiv:2108.13619
11 Zhu J W, Zhang H, Zhao S B, et al. Multi-constrained intelligent gliding guidance via optimal control and DQN. Sci China Inf Sci, 2023, 66: 132202
12 Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning, New York, 2016. 1995–2003
13 Wang B Y, Liu Z, Li Q B, et al. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. IEEE Robot Autom Lett, 2020, 5: 6932–6939
14 Wei Y, Zheng R. A reinforcement learning framework for efficient informative sensing. IEEE Trans Mobile Comput, 2020, 21: 2306–2317
15 Xu Y, Wu Z-G, Che W-W, et al. Reinforcement learning-based unknown reference tracking control of HMASs with nonidentical communication delays. Sci China Inf Sci, 2023, 66: 170203
16 Song Y L, Scaramuzza D. Policy search for model predictive control with application to agile drone flight. IEEE Trans Robot, 2022, 38: 2114–2130
17 Dong L, Yuan X, Sun C Y. Event-triggered receding horizon control via actor-critic design. Sci China Inf Sci, 2020, 63: 150210
18 Zhu B, Bedeer E, Nguyen H H, et al. UAV trajectory planning in wireless sensor networks for energy consumption minimization by deep reinforcement learning. IEEE Trans Veh Technol, 2021, 70: 9540–9554
19 Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning. Comput Sci, 2015, 8: A187
20 Ackermann J, Gabler V, Osa T, et al. Reducing overestimation bias in multi-agent domains using double centralized critics. 2019. ArXiv:1910.01465
21 Gu S, Lillicrap T, Sutskever I, et al. Continuous deep Q-learning with model-based acceleration. In: Proceedings of the 33rd International Conference on Machine Learning, New York, 2016. 2829–2838
22 Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the 35th International Conference on Machine Learning, Stockholm, 2018. 1861–1870
23 Zhang L X, Zhang R X, Wu T, et al. Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles. IEEE Trans Neural Netw Learn Syst, 2021, 32: 5435–5444