IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 17, 2018, accepted September 16, 2018, date of publication September 28, 2018, date of current version October 29, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872751

UGV Navigation Optimization Aided by Reinforcement Learning-Based Path Tracking

MINGGAO WEI, SONG WANG, JINFAN ZHENG, AND DAN CHEN, (Member, IEEE)
National Engineering Research Center for Multimedia Software, School of Computer Science, Wuhan University, Wuhan 430072, China

Corresponding author: Dan Chen (dan.chen@whu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772380 and in part by the Foundation for Innovative Research Groups of Hubei Province under Grant 2017CFA007.

ABSTRACT The success of robotic, such as UGV systems, largely benefits from the fundamental capability of autonomously finding collision-free path(s) to commit mobile tasks in routinely rough and complicated environments. Optimization of navigation under such circumstance has long been an open problem: 1) to meet the critical requirements of this task typically including the shortest distance and smoothness and 2) more challengingly, to enable a general solution to track the optimal path in real-time outdoor applications. Aiming at the problem, this study develops a two-tier approach to navigation optimization in terms of path planning and tracking. First, a ''rope'' model has been designed to mimic the deformation of a path in axial direction under external force and the fixedness of the radial plane to contain a UGV in a collision-free space. Second, a deterministic policy gradient (DPG) algorithm has been trained efficiently on abstracted structures of an arbitrarily derived ''rope'' to model the controller for tracking the optimal path. The learned policy can be generalized to a variety of scenarios. Experiments have been performed over complicated environments of different types. The results indicate that: 1) the rope model helps in minimizing distance and enhancing smoothness of the path, while guarantees the clearance; 2) the DPG can be modeled quickly (in a couple of minutes on an office desktop) and the model can apply to environments of increasing complexity under the circumstance of external disturbances without the need for tuning parameters; and 3) the DPG-based controller can autonomously adjust the UGV to follow the correct path free of risks by itself.

INDEX TERMS UGV navigation, reinforcement learning, deterministic policy gradient, path tracking.

I. INTRODUCTION

Over the past decades, robotic like unmanned ground vehicle (UGV) systems have gained tremendous successes in various applications, from daily transportations, jungle reconnaissance, to planet exploration. These successes benefit from UGV's fundamental navigation capability of autonomously finding collision-free path(s) to commit mobile tasks in routinely rough and complicated environments. Given the geometry of a UGV and obstacles in a large-scale outdoor environment, a planner needs first to generate a feasible path between the start and end points, then to follow the planned path under the physical constraints of the UGV and the path as well as the uncertain disturbances [1]. Optimization of navigation under such circumstance has long been a grand challenge.

First, one needs to identify an ''optimal path'' in the context of path planning. Optimization of path planning may involve various objectives, but the basic issues under consideration are focused on (1) distance and (2) collision-free. A* based grid methods are simple and commonly used in practice, but it discretizes motion and lacks the capability of distance minimization. Visibility graph (VG) [2], [3] and Voronoi diagram (VD) [4] are two classic methods for this purpose. VG builds a collection of lines connecting a feature of an object to that of another in the free space. In contrast, VD partitions space into cells each consisting of the points closer to one particular object. In large and complicated environments, both methods needs encounter technical barriers in solving NP-hard problems.

In vision of this, numerous suboptimal solutions have been proposed, such as Probabilistic Roadmap method (PRM) [5] and Potential Field method (PFM) [6], [7]. There exist plenty of room to improve these methods, e.g., PRM can get a rough path due to the randomness of the insertion point, the Potential Field can achieve obstacle avoidance in real time but the planned path may get trapped at local minimums.

Nature-inspired methods have been proposed for this purpose, including Genetic algorithm (GA) [8], artificial bee colony (ABC) [9] and particle swarm optimization (PSO) [10]. Elastic band [11]–[13] methods utilize the internal force between adjacent free space around the robot to construct a deformable collision-free path, which have been widely used for optimizing global path planners including PRM and PFM. These methods have no guarantee over the fixedness of the width of path, and the consequent path tracking will be difficult.

Given a path been properly planned in theory, tracking or following the path to commit some mobile tasks remains not less challenging. Methods for this purpose have been extensively explored for decades, salient examples include Fuzzy Logic, Neural Network, and Model Predictive Control (MPC). The Fuzzy Logic method employs a fuzzy logic system (FLS) to estimate the uncertainties of a UGV system [14]. Fuzzy Logic has been further incorporated with Neural Network to improve the effectiveness of uncertainty estimation with a dynamic Petri recurrent fuzzy neural network in path tracking control [15]. MPC enables a controlling strategy considering the error dynamics derived from both the robot states and the path states [16].

These conventional controllers are focused on driving a robot along a track best fitting the path planned previously, i.e., with the objective of a high precision. Note that in practice UGVs are expected to operate in outdoor applications, the influence of uncertain disturbances in the course of path tracking increases the difficulty of problem solving. The latest Learning-based Nonlinear MPC models the various disturbances in a possibly complicated environment as a Gaussian process [17]. These conventional methods generally treat various uncertainties as a (few) deterministic distribution(s) (aka. distributional uncertainty), which is insufficient in practical scenarios usually large in terms of environment and complicated in terms of dynamics. As for robotics applications, this is even more difficult as significant uncertainty may propagate unboundedly temporal- and spatial-wise. There exists a pressing need to introduce computational intelligence to address the challenge.

Deep learning technology excels in the capabilities of directly learning from empirical data to achieve increasingly optimized performance in problem solving. Methods along this direction have recently been widely used to solve artificial intelligence problems especially the control system in robotics [18]. Supervised learning methods heavily rely on knowledge from experts, as a contrast reinforcement learning (*RL*) is salient because it requires no human-labeling based on trial-and-error interactions with the environment [19]–[21]. Furthermore, deterministic policy gradient (*DPG*) algorithms recently gain more and more attentions for its superiority in solving problems concerning continuous action and state spaces [22]. The robot (UGV) path tracking problem is exactly the case. Actually, *RL* methods combined deep learning with *DPG* (*DDPG*) has achieved successes in robot controlling, such as object grabbing [23]–[25], path planning [26], [27] and locomotion skills learning [28], [29]. The success of these applications motivate us to extend this tool to path tracking. However, there still exists a technical gap (1) to generalize the learning model for mobile robot control to adapt to various scenarios and (2) to adapt to external disturbances without the support of sufficient empirical data, which is mandatory for real-time applications in large-scale outdoor environments.

After all, navigation of a UGV routinely needs first to obtain an optimal path between the two points considering distance and collision-free, then to follow the planned path successfully even the uncertain disturbance occurs. This study is aimed at the challenges in this two-tiered problem (1) to meet the critical requirements of this task typically including the shortest distance and smoothness and (2) to enable a general solution to track the optimal path in real-time applications in large-scale outdoor environments:

1) This study first designs a ''rope'' to mimic the deformation of a path in axial direction under external force and the fixedness of the radial plane to contain a UGV in a collision-free space by considering the revolute and collision constraints (Subsection III-A). Given the start and end points in any $2D$ environment, the model can optimize a global path planner by automatically constructing a tube straightforwardly that defines the optimal path with both the shortened distance and enhanced smoothness.
2) Second, this study constructs a Deep Deterministic Policy Gradient algorithm (*DDPG*, Subsection III-B). *DDPG* can be efficiently trained on abstracted structures (Subsection III-B.1) of an arbitrarily derived tube defining any ''safe area'' for UGV traversing. The trained *DDPG* model enables a general policy to control an UGV to track the correct path free of risks by itself. The model can apply to environments of increasing complexity under the circumstance of external disturbances without the need for tuning parameters.

Experiments have been performed over complicated environments of different types (Section IV). The results indicate that (1) the rope model helps in minimizing distance and enhancing smoothness of the path; (2) the *DDPG* can be modelled quickly and the *DDPG*-based controller can autonomously adjust the UGV to follow the correct path. The main contributions of this study are as follows:

1) This study develops an intuitive method to globally optimize path planning for UGVs with the capability of shortening distance and improving smoothness;
2) This study enables a general solution to track the optimal path targeting on real-time outdoor applications.

II. RELATED WORK

Recent trend on optimization of path planning focuses on (1) convex optimization and (2) nature-inspired methods.

Convex optimization aims to plan a continuous trajectory to directly meet the dynamics constraints over UGVs. A typical example is the CHOMP method proposed by

Zucker *et al.* [30]. CHOMP optimizes a cost function that makes trades-off between smoothness and obstacle avoidance to gain high-quality trajectory of a predetermined duration via a gradient descent technique. Schulman *et al.* [31] utilizes a sequential convex optimization procedure to find continuous-time safety path by penalizing violated constraints.

Nature-inspired methods mainly consider how to shorten the distance whole ensuring collision-free. Davoodi *et al.* [32] apply a genetic algorithm with NSGA-II framework to intensify explorative power in complicated path planning problem in the context of grid environment model. Mac *et al.* [33] utilized constrained multi-objective particle swarm optimization (*PSO*) to optimize Dijktra's algorithm with the Visibility Graph model. Contreras-Cruz *et al.* [34] proposed using the artificial bee colony (*ABC*) algorithm for local search and the evolutionary programming algorithm for refining the feasible global path.

Besides traditional conventional controllers for path tracking, Reinforcement Learning (*RL*) -based methods have also been examined. Baltes and Lin [35] utilized the Function Approximator to fit the state space in order to approximate the desired path. The action space was discretized that resulted in rough steering.

Zuo *et al.* [36] attempted to apply *RL* as a feedback control over *PD*. The Laplacian-based hierarchical approximate policy iteration (GHAPI) applied to decompose the state space to smaller subspaces as exploration of subspace was much easier. But the action space is also discretized, and this made *PD* operate in a limited number of parameter settings.

Abbeel *et al.* [37] proposed an ''inaccurate'' model that followed human driver's experiences to obtain tracking strategy without the need for excessive training. The resulted controller used an abstracted UGV model for policy search in both continuous state space and action space. The model always required a number of training cases for each individual tracking task.

Liu and Tong [38] pointed reinforcement learning can achieve the optimal control performance for a class of multiple-input multiple-output nonlinear discrete-time systems. Inspired of this study, Liu *et al.* [39] integrated adaptive reinforcement learning into the fault tolerant controller (FTC) for tracking problem. Their efforts focused on reducing the number of training parameters thus to alleviate the computational load for online parameters tuning at each iteration. However, the learning time of the neural network can still be excessively long.

Inspired by the above work, this study mainly focuses on: (1) optimizing any initial path planner to shorten the distance while ensuring obstacle clearance and (2) a general solution to tracking the optimal path with a low training computational cost towards real-time outdoor applications.

III. METHOD FOR UGV NAVIGATION OPTIMIZATION

This section first introduces the ''rope'' model that optimizes a global path planner. It then presents the *DDPG* algorithm that enables a general policy to control UGV to track the correct path free of risks.

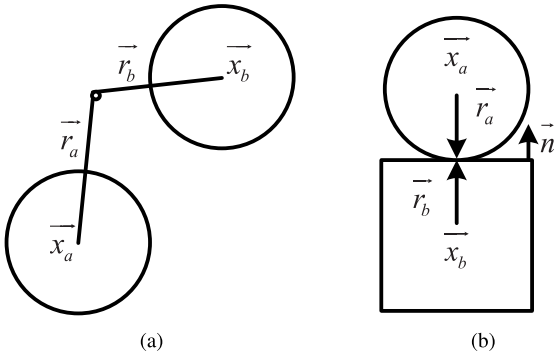


FIGURE 1. Two kinds of constraints. (a) revolute constraint. (b) collision constraint.

A. ROPE MODEL FOR OPTIMIZATION OF PATH PLANNING

1) BACKGROUND FOR CONSTRAINT DYNAMICS

Consider a common constraint between two objects like Figure 1. When the physical constraint is satisfied, the constraint equation can be defined as:

$$C_{constraint}(\vec{x_a}, R_a, \vec{x_b}, R_b) = 0 \tag{1}$$

where $\vec{x}_a$ is the center of mass of object a, $\vec{R}_a$ is the rotation of a. Take the derivative with respect to time, then get the velocity constraint:

$$\frac{d\,(C_{constraint})}{d\,(t)} = \frac{d\,(C_{constraint})}{d\,(x)}\vec{v} = J\vec{v} = 0 \tag{2}$$

According to the principle that constraint force do not work, Jacobian determinants indicate the direction of constraint force, so constraint force can be represented as (3).

$$f_{constraint} = J^T\lambda \tag{3}$$

where λ denotes the magnitude of force, the final velocity of the system is

$$\vec{v}_{i+1} = \vec{v_i} + \Delta t\,\vec{a} = \vec{v_i} + \Delta t M^{-1}\left(\vec{f}_{ext} + \vec{f}_{constraint}\right)$$

If external force $\vec{f}_{ext}$ is integrated in advance, the equation changes to

$$\vec{v}_{i+1} = \vec{v_i} + \Delta t M^{-1}\vec{f}_{constraint} \tag{4}$$

Now substitute equation (3) into (4):

$$\vec{v}_{i+1} = \vec{v_i} + \Delta t M^{-1}J^T\lambda \tag{5}$$

substitute equation (5) into $\dot{C}_{constraint} = Jv = 0$. Then get:

$$J\left(\vec{v_i} + \Delta t M^{-1}J^T\lambda\right) = 0$$
$$-J\vec{v_i} = JM^{-1}J^T\Delta t\lambda \tag{6}$$

Now, the λ in (6) can be solved, then the velocity of the system in the next step $\vec{v}_{i+1}$ is calculated according to (5).

2) UTILIZED CONSTRAINTS

Two kinds of constraints are utilized to simulate the rope model, revolute constraint and collision constraint. As shown in Figure 1(a), revolute constraint arises due to the fact that components rotate freely around a common point, which can keep a certain distance between components to construct an approximate fixed-size tube. In Figure 1(b), collision constraint exists between obstacles and components to prevent mutual penetration, which can ensure that the constructed tube is safe. According to the rule [40], the Jacobian determinants of the two constraints are alternately deduced. The first step is to write out an equation that describes the position constraint. Since the adjacent components rotate around a common point for revolute constraint, the corresponding position constraint is described in (7).

$$\begin{aligned} &C_{revolute}\left(\overrightarrow{x_a},\ R_a,\ \overrightarrow{x_b},\ R_b\right) \\ &\quad = \overrightarrow{p_a} - \overrightarrow{p_b} \\ &\quad = \overrightarrow{x_a} + R_a\overrightarrow{r_a} - \overrightarrow{x_b} - R_b\overrightarrow{r_b} \end{aligned} \tag{7}$$

where R_a and R_b are the rotation matrices, $\overrightarrow{r_a}$ and $\overrightarrow{r_b}$ are the vectors from the centers of the component to the common point. The equation indicates that $\overrightarrow{p_a}$ and $\overrightarrow{p_b}$ must be the same at any step, which allows the components to translate freely about a common point.

The position constraint for collision constraint is represented in (8), $\overrightarrow{r_a}$ and $\overrightarrow{r_b}$ locate the contact points on component a and b, $\overrightarrow{n}$ is the normal vector from b to a. The position constraint measures whether the penetration will occur, in the way the constraint force will separate the components if the value is negative. On the other hand, if the value is greater than zero, the interaction does not exist.

$$\begin{aligned} &C_{collision}\left(\overrightarrow{x_a},\ R_a,\ \overrightarrow{x_b},\ R_b\right) \\ &\quad = \left(\overrightarrow{p_a} - \overrightarrow{p_b}\right)\overrightarrow{n} \\ &\quad = \left(\overrightarrow{x_a} + R_a\overrightarrow{r_a} - \overrightarrow{x_b} - R_a\overrightarrow{r_b}\right)\overrightarrow{n} \end{aligned} \tag{8}$$

The next step after defining the position constraint is to take the derivative with respect to time. This will yield the velocity constraint. The result of time derivation is as follows. In (9) $\omega_a \times \overrightarrow{r_a} = \begin{bmatrix} -\omega_a r_{ay} \\ \omega_a r_{ax} \end{bmatrix} = \begin{bmatrix} -r_{ay} \\ r_{ax} \end{bmatrix}\omega_a, R_{ra} = \begin{bmatrix} -r_{ay} \\ r_{ax} \end{bmatrix}$. After isolating the velocity $\overrightarrow{v_i} = \begin{bmatrix} \overrightarrow{v_a} & \omega_a & \overrightarrow{v_b} & \omega_b \end{bmatrix}^T$, the Jacobian is obtained easily, $J_{revolue} = \begin{bmatrix} 1 & R_{ra} & -1 & -R_{rb} \end{bmatrix}$ and $J_{collision} = \begin{bmatrix} \overrightarrow{n^T} & (r_a \times n)^T & -\overrightarrow{n^T} & -(r_b \times n)^T \end{bmatrix}$.

$$\begin{aligned} \dot{C}_{revolute} &= \overrightarrow{v_a} + \omega_a \times \overrightarrow{r_a} - \overrightarrow{v_b} + \omega_b \times \overrightarrow{r_b} \\ &= \begin{bmatrix} 1 & R_{ra} & -1 & -R_{rb} \end{bmatrix} \begin{bmatrix} \overrightarrow{v_a} \\ \omega_a \\ \overrightarrow{v_b} \\ \omega_b \end{bmatrix} \end{aligned} \tag{9}$$

$$\begin{aligned} \dot{C}_{collision} &= \left(\overrightarrow{v_a} + \omega_a \times \overrightarrow{r_a} - \overrightarrow{v_b} + \omega_b \times \overrightarrow{r_b}\right)\overrightarrow{n} \\ &= \begin{bmatrix} \overrightarrow{n^T} & (r_a \times n)^T & -\overrightarrow{n^T} & -(r_b \times n)^T \end{bmatrix} \begin{bmatrix} \overrightarrow{v_a} \\ \omega_a \\ \overrightarrow{v_b} \\ \omega_b \end{bmatrix} \end{aligned} \tag{10}$$

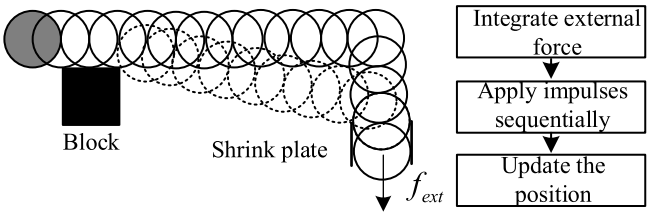


FIGURE 2. rope model.

Substituting $J_{revolute}$ and $J_{collision}$ into (5) and (6), we'll get the updated formula directly. For detailed derivation process, refer to appendix A.

$$\begin{aligned} \overrightarrow{v}_{a_{i+1}} &= \overrightarrow{v}_{a_i} + \overrightarrow{P}/m_1 \\ \omega_{a_{i+1}} &= \omega_{a_i} + I_a^{-1}\overrightarrow{r_a} \times \overrightarrow{P} \\ \overrightarrow{v}_{b_{i+1}} &= \overrightarrow{v}_{b_i} - \overrightarrow{P}/m_2 \\ \omega_{b_{i+1}} &= \omega_{b_i} - I_b^{-1}\overrightarrow{r_b} \times \overrightarrow{P} \end{aligned} \tag{11}$$

For revolute constraint, the impulse is calculated as $\overrightarrow{P}_{revolute} = -K^{-1}\left(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r_a} - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r_b}\right)$ and $K = \left(\frac{1}{m_a} + \frac{1}{m_b}\right)E_{2\times 2} + R_{ra}I_a^{-1}R_{ra}^T + R_{rb}I_b^{-1}R_{rb}^T$. As to collision constraint $\overrightarrow{P}_{collision} = P_{collision}\overrightarrow{n}$, $P_{collision} = \frac{-\left(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r_a} - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r_b}\right)\cdot\overrightarrow{n}}{K}$, $K = m_a^{-1} + m_b^{-1} + I_a^{-1}\left(\overrightarrow{n} \times \overrightarrow{r_a}\right)^2 + I_b^{-1}\left(\overrightarrow{n} \times \overrightarrow{r_b}\right)^2$, $P_{collision} \geq 0$ so that collision constraint force separate the contact.

3) CONSTRUCTION FOR ROPE MODEL

The above discussion provides two concrete solutions to two constraint force. In this part, the method to construct a rope model based on these two constraints is discussed. As shown in Figure 2, the solid circle component is initialized to cover the original path, and the common rotation point of the adjacent circle is designated as the center of the prior circle, which makes the next component rotate only about the center of the prior component to maintain a certain distance. The first component described as gray is fixed so that the chain can be tightened under external forces. For another hand, in order to ensure the correctness of the convergence direction, a shrink plate is constructed at the end of the original chain, and the direction of the shrink plate is consistent with the original direction of the end. Then, an external force is applied to the end in the same direction as the shrink plate so that the chain will slowly contract. The length of the chain is chosen as the judgment condition of convergence. Under the force, the distance between the last component and the end point will be farther and farther. When the distance between them exceeds a certain threshold, the component is deleted and the force is applied to the new end. By continually removing the components, the length of the chain will be constant when the chain is tightened. However, since the two constraints under consideration are not conspicuous for the energy consumption of the entire chain, jitter exists when the chain converges to a fixed length, which leads to the undesirable shape for

the tube. Therefore, it is necessary to increase the air damping effect

$$v_{t+1} = kv_t, \quad k \in [0, 1] \tag{12}$$

where k is the decay ratio for velocity per step.

The update process for each component per step is same as [41], as shown in the right of Figure 2. Integration of external forces is applied at first, where only the end component receives the external force. Then impulses in (11) are calculated iteratively to correct the velocity until the velocity has converged or the iterations have been exhausted. At last, each component's position is added with the corrected velocity. In addition, due to the inaccuracy of the calculations, Baumgarte Stabilization is also used to prevent the drift and penetration of the constraints. Thus the modified impulse equation is

$$\begin{aligned} \overrightarrow{\mathrm{P}}_{m\_revolute} &= \overrightarrow{\mathrm{P}}_{revolute} - \frac{\beta K^{-1}(\overrightarrow{p_a} - \overrightarrow{p_b})}{\Delta t} \\ P_{m\_collision} &= \max\left\{P_{collision} + \frac{\beta\delta}{\Delta t K}, 0\right\} \end{aligned} \tag{13}$$

where $\overrightarrow{p}_a - \overrightarrow{p}_b$ is the drift error, δ is the penetration error, $\beta > 0$ is the decaying bias. More detailed analysis can be seen in [41].

B. DEEP REINFORCEMENT LEARNING FOR PATH TRACKING

Path tracking is feasible with the implementation of deep reinforcement learning. The model can be trained on simulated environment that abstracts the paths in real-world scenarios with extremely simple structures. The trained *DDPG* algorithm then commits the tracking tasks.

1) SIMULATED SCENARIO FOR TRAINING

In order to effectively simulate the complexities of tracking as car-like robots in reality, only four configurations need to be extracted here: I, L, Z, U. This is a reasonable practice for training a great policy because the car-like robot comprised by nonholonomic mechanical systems is constrained by its rate of rotation, like the highway for cars in real world. Shown in Figure 3 are the typical cases which are dealt with. The training scenario is constructed to contains these four configurations.

In this study, a common four-wheel mobile robot with two steered front wheels and two fixed-heading rear wheels is focused on as shown in Figure 4. The following dynamic models are described as [42].

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ sin\phi & 0 \\ cos\phi & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{14}$$

where υ is the robot's forward speed and ω is the rate of rotation. The detailed parameter settings related to the robot movement will be introduced in Section IV. Robot is surrounded by five laser sensors with uniform angle from center

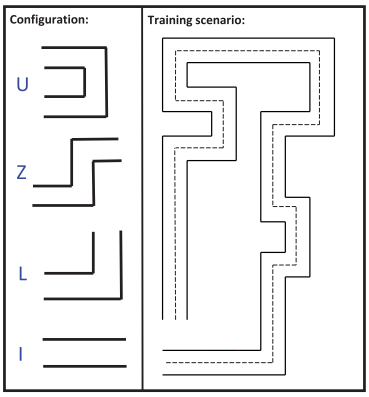


FIGURE 3. The training scenario. As shown in left is the four configurations: I, L, Z, U. The right is the training scenario consisting of the four configurations. The dotted line is the desired following path.

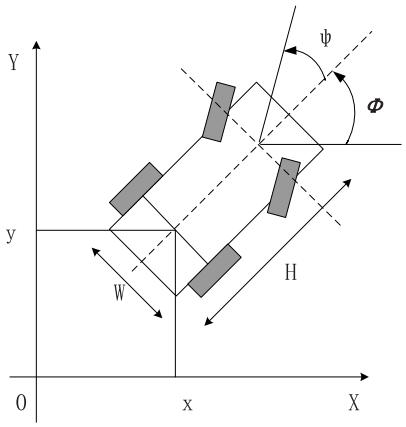


FIGURE 4. The model of the car-like robot. W, H are defined as the width and height of robot body respectively. (x, y) is the location of the robot. ϕ is the robot's heading direction. ψ is the steering angle of the robot decided by the rate of rotation.

of the robot which can form a simple bounding box to detect the collision effectively. The state is

$$S_t = (sensor_1, sensor_2 ... sensor_n) \tag{15}$$

where $sensor_n$ represents the distance information of the obstacle detected by the nth sensor. Reward function is designed on the principle of collision avoidance:

$$r(s_t, a_t) = \begin{cases} -1 & if\ min(sensor_1, sensor_2 ... sensor_n) < W/2 \\ 0 & else \end{cases} \tag{16}$$

A negative reward -1 is arranged while the collision is detected, otherwise a positive reward is arranged.

2) PATH TRACKING BASED ON THE *DDPG* ALGORITHM

A standard reinforcement learning use Markov decision processes (MDP) to describe the environment. The MDP is a tuple $\langle S, A, P, R, \gamma \rangle$. At each time step t, the robot observe the state $s_t \in S$, choose the action $a_t \in A$ according to the potential policy $\pi(a|s)$ mapped s to a and receive a

feedback scalar reward $r_{t+1} \in R$ and a new state s_{t+1} from the environment, $\gamma \in [0, 1]$ is the discount factor for future reward. The cumulative reward at t is represented as $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$, the action-value function is $Q_\pi(s_t, a_t) = E_\pi[R_t; s_t, a_t]$, the goal of reinforcement learning is to learn the optimal policy $\pi = argmax_\pi Q_\pi(s, a)$, which obeys the Bellman optimality equation $Q_\pi(s_t, a_t) = E_{s_{t+1}}[r + \gamma max_{a_{t+1}} Q_\pi(s_{t+1}, a_{t+1})|s_t, a_t]$. When the state space is continuous, value function approximation is introduced, that is the action-value function is approximated by θ^Q, so the loss function is given by:

$$L_i(\theta^Q) = E_{s_t, a_t, r, s_{t+1}}[(y_i - Q_\pi(s_t, a_t; \theta^Q))^2] \tag{17}$$

where $y_i = r + \gamma Q(s_{t+1}, a_{t+1}; \theta^Q)$ denotes the expected cumulative reward at (s_t, a_t), where $Q_\pi(s_t, a_t; \theta^Q)$ is the real cumulative reward and i is the ith iteration.

While the action space is also continuous, DPG [22] based actor-critic algorithm maintains a parameterized actor function $\mu(s|\theta^\mu)$, which maps state to a specific action deterministically. The critic function is updated by Bellman equation similar to Q-learning. Obtain the actor's iterative equation through derivation from θ^μ according to the chain rule:

$$\begin{aligned} \nabla_{\theta^\mu} &\approx E_{s_t, a_t, r, s_{t+1}}\left[\delta_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t|\theta^\mu)}\right] \\ &= E_{s_t, a_t, r, s_{t+1}} \\ &\quad \times \left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|s = s_t\right] \end{aligned} \tag{18}$$

The DDPG [42] used neural networks as the parameters of DPG under the continuous action domain. The DDPG algorithm adapted for path tracking is shown in Algorithm 1. Similar to [43], the experience replay and separate target network are also utilized to enhance the stability of the algorithm. Random noise $\mathcal{N}$ is added to the continuous action space in order to perform more efficient exploration, which diminishes step by step to imitate the $\epsilon - greedy$ strategy. *MAX_EP_STEPS* is the threshold indicating whether exploration is enough for each trying. In this paper, neural network is used to learn the policy. Figure 5 shows the structure of the network. The Actor and Critic Network both contain two full-connected layers, which are 100 nodes and 20 nodes respectively. The input vector for actor is the detected distance from obstacles by the five sensors. Moreover, the tanh is used as activation function of the output layer to constrain the rate of rotation. As to critic, the action is merged with state as action-state input, where $Q(s, a)$ value is calculated by the linear activation function.

Algorithm 1: DDPG for Path Tracking (Adapted From [42])

Initialize critic network $Q(s, a|\theta^Q)$, actor $\mu(s|\theta^\mu)$, target critic Q' and actor μ' with weights θ^Q,θ^μ,$\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$, replay buffer M.
for *episode* = 1, *MAX_EPISODES* **do**
  Reset the environment and receive the initial observation state s_t.
  for t = 0, *MAX_EP_STEPS* **do**
    Choose the action a_t according to:$\mu(s_t; \theta^\mu) + \mathcal{N}$
    Execute a_t, receive the feedback reward r_t, a new state s_{t+1} and the condition of end *terminal*.
    Store the tuple$\langle s_t, a_t, r_t, s_{t+1}\rangle$ in M.
    if *size(M) == capacity(M)* **then**
      Sample a random *batch* transitions (s_i, a_i, r_i, s_{i+1}) from M.
      Set $\mathcal{Z}_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
      Update critic by minimizing the loss: $L = \frac{1}{batch}\sum_i(\mathcal{Z}_i - Q(s_i, a_i|\theta^Q)^2)$
      Update the actor policy using the sample gradient: $\nabla_{\theta^\mu}\mu|s_i \approx \frac{1}{batch}\sum_i \nabla_\alpha Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$
    end
    if *episode* % *replace_iteration* == 0 **then**
      Update the target networks: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$ $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
    end
    if *terminal* == *true* **then**
      break
    end
  end
end

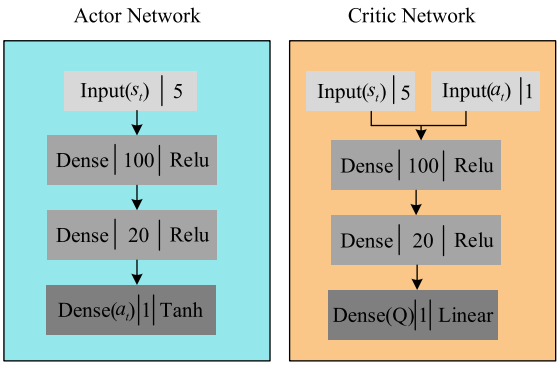


FIGURE 5. The structure of networks. From left to right in each small square is type of layer, number of node, activation function.

IV. EXPERIMENT AND RESULTS

Two sets of experiments have been performed to evaluate the proposed approach: (1) global path optimization with the rope model; (2) path tracking based on *DDPG*, and models were trained and examined with Tensorflow on a single NVIDIA GeForce GTX 1060 and Intel Core i7-4790 with 24GB RAM.

A. GLOBAL PATH OPTIMIZATION

Ten complicated maps with different layouts and obstacles were built to validate the performance of the rope model by referring to [44] (illustrated in Figure 6). The classic algorithm A* was utilized to plan the initial global path with one unit searching radius. After that, the proposed rope model

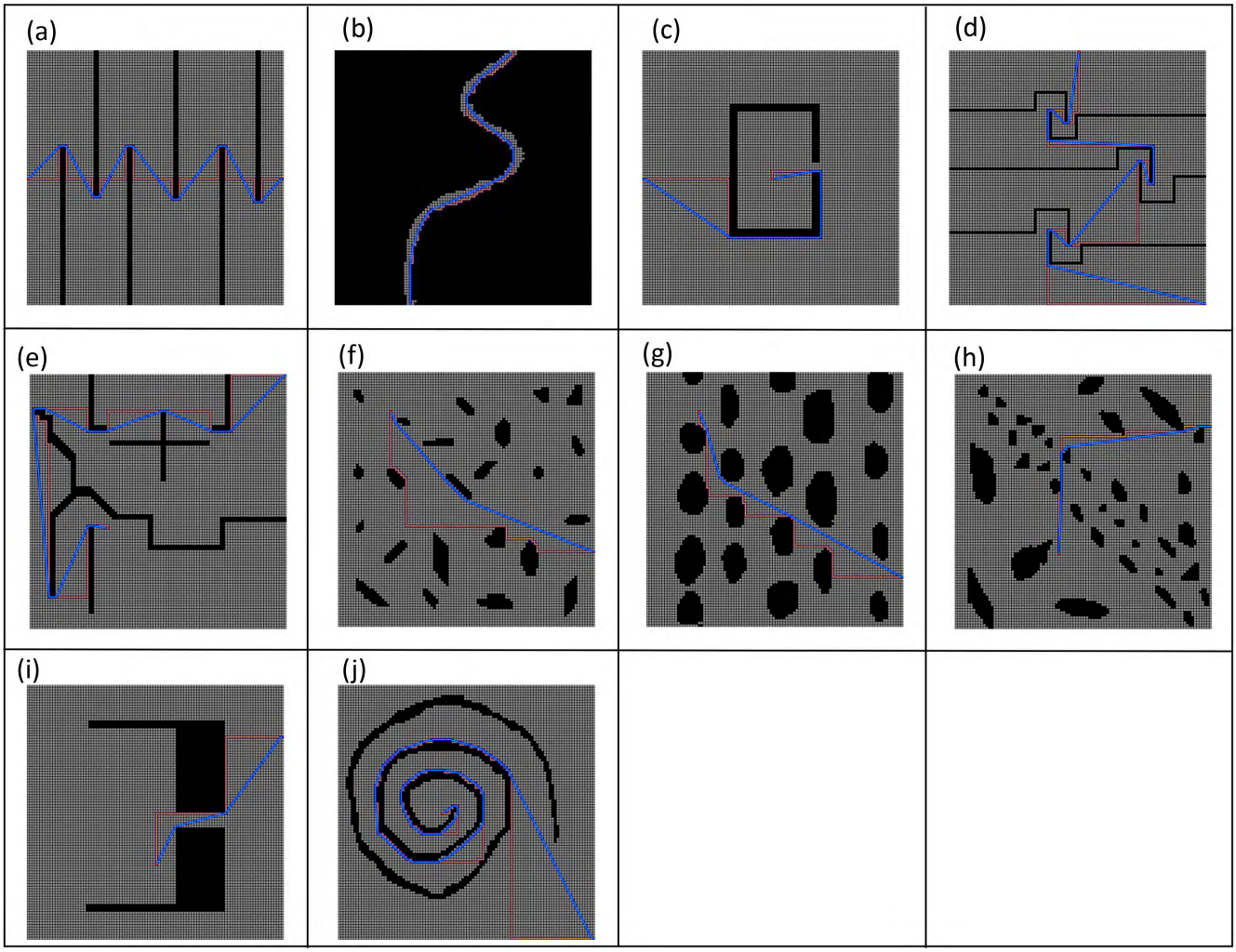


FIGURE 6. Result of proposed optimized method. The black is occupied, the grey is feasible for robot.

TABLE 1. The improvement between initial path and final path.

	Map_1	Map_2	Map_3	Map_4	Map_5	Map_6	Map_7	Map_8	Map_9	Map_{10}
Ratio of improved distance	0.258	0.223	0.136	0.160	0.177	0.253	0.243	0.083	0.253	0.216
Improved smoothness	0.085	0.819	0.0155	0.025	0.062	0.203	0.230	0.0786	0.006	0.483

planned the final optimized path on its basis. The initial and final paths were marked as blue and brown red lines separately. Apparently, redundancy exists in the initial path in terms of distance due to the discrete movement setting. The optimized path significantly improved the initial path in both distance and smoothness. The improvement was measured as $D_{improvedRadio} = 1 - \frac{D_{optimized}}{D_{initial}}$, $S_{improved} = S_{optimized} - S_{initial}$, where D and S were calculated (19) according to [44].

$$\begin{aligned} Distance &= \sum_{p=1}^{N-1} ||x_{p+1} - x_p|| \\ Smoothness &= \frac{1}{N-2} \sum_{p=2}^{N-1} cos^{-1}\left(\frac{(x_p - x_{p-1})(x_{p+1} - x_p)}{||x_p - x_{p-1}||\ ||x_{p+1} - x_p)||}\right) \end{aligned} \tag{19}$$

where x_p is the location of pth point, N is the number of component.

As shown in Table 6, distances were significantly reduced in almost all maps. Only the ratio for $Map8$ was not very obvious due to the key point set have little changes. The ratios for $Map1$, $Map6$ and $Map9$ were closed to the optimum one $1 - \frac{\sqrt{2}}{2} \approx 0.293$ calculated in the scene where the start and goal points were located on the diagonal line in the square blank map.

It should also be noted that the smoothness for all maps have significantly increased. The largest improvements were witnessed in $Map2$ and $Map10$ where the number of deforming points decreased the most. Observations from Figure 6 implied that the proposed method could minimize the redundant distances while maximizing smoothness by simulating

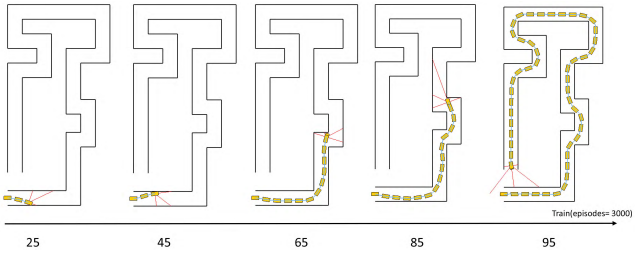


FIGURE 7. The training process. The yellow object represents the UGV; The red object denotes the sensor; The horizontal axis represents the number of episodes.

the deformation characteristics of a chain-like object along the direction of the force. It was applicable in various complicated maps. The method avoided local optimum such as being trapped in U shape obstacle (the typical problem with *BFS* and potential field).

In addition, when overlapping the adjacent circles properly, a collision-free tube could be constructed as shown in Figure 6, where the size of grid is enlarged compared to Figure 6. When the distance between the adjacent circles is small enough, a tube of fixed size was theoretically obtained. The experiments assumed that the robot occupied one grid cell, settings of more grid cells could be easily planned through setting a larger radius in A* for global planner [32].

B. EVALUATION OF PATH TRACKING

Path tracking models should be first trained on *DDPG* before the tracking strategies might be examined.

1) PARAMETER SETTING FOR TRAINING

This study used a four-wheel mobile robot similar to those described in [36] and [45]. The parameters of the model were set as $v = 0.34m/s$, $\omega_{max} = 60°/s, H = 2W = 0.68m$, the time step was 0.1s. The width of the proposed scenario was large enough to allow the robot going through easily. The parameters for training were as follows: $MAX\_EP\_EPISODES = 200$, $MAX\_EP\_STEPS = 3000$, $\gamma = 0.9$, *learning rate* for actor and critic $= 0.0001$, $batch\_size = 32$, $replay = 3000$, $\mathcal{N}$ is a normal distribution whose *mean* was equal to the a_t and *variance* diminished step by step. The training procedure of the model was illustrated in Figure 7.

The learned policy were gradually improved with more explorations. It took 7 mins to derive a satisfactory policy at the 95th episode: closely staying in the middle of the tube to fit the desired path in Figure 3.

2) PERFORMANCE EVALUATION

This subsection first presents the objectives of training, then the performance of the tracking policy will be analyzed. Finally the generalization ability of the learned tracking policy will be examined with a very complicated environment.

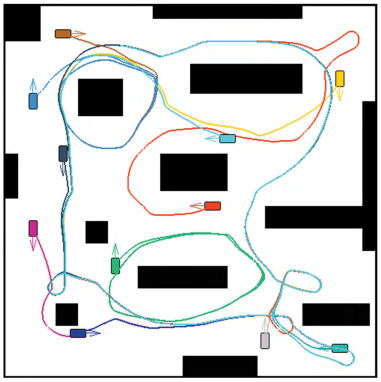


FIGURE 8. Examination of the capability of collision avoidance. The object with the box shape denotes the robot; Each arrow denotes an initial direction of the robot.

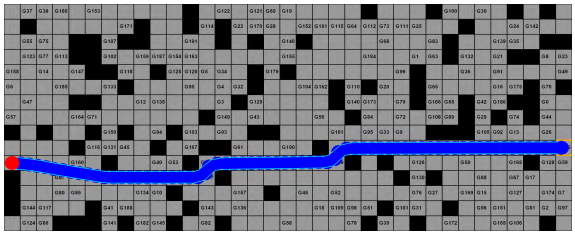


FIGURE 9. Examination of the capability of following. Each number denotes a randomly chosen destination; The red dot denotes the start point; The tube constructed by the rope model is marked blue.

a: LEARNING CAPABILITY

Two scenarios were designed to demonstrate what the model could learn. The first scenario with obstacles of different sizes is shown as Figure 8, in which eleven points were randomly chosen as the start points. The trajectories (driving lines) indicated that each robot can do the right action by evaluating the learned policy to avoid the obstacles successfully even in the dead ends like upper right and lower right corner. The result suggested the learned policy based on the proposed scenario has great reaction ability to avoid obstacles in the complex environment according to its own state.

The second scenario (Figure 9) consisted of a grid world (same cell width in Figure 6 applied). The tube was constructed between a fixed start point (red) and any of 191 random end points, which represented a variety of radians for the robot to explore similar to highways in real world. Experimental results suggested the tracking policy could lead the UGV to reach all destinations without collision and to stay in the middle of the path to the most. The details of the errors between the desired path (defined by the tube constructed by the ''rope'' model) and the actual tracks were analyzed in the later part of this section.

Clearly, the learned policy made use of the constructed tube to enable obstacle avoidance when steering the UGV to track the desired path.

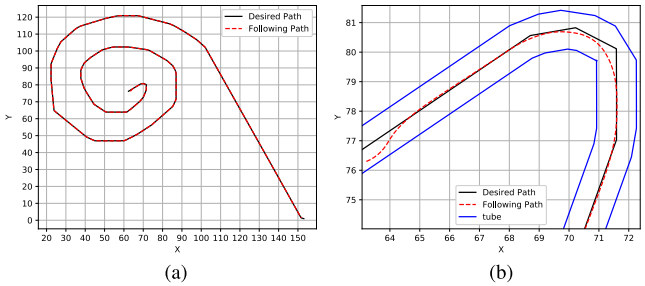


FIGURE 10. Path tracking and performance analysis. (a) following. (b) analysis.

b: PERFORMANCE ANALYSIS

In order to analyze the performance of the learned model, the most complicated map (*Map*10) was selected for examination. The size was set as about $120 \times 150 m^2$ and the velocity of the robot was set as 1.45 m/s. Results (Sub-figure 10(a)) indicated that the model could correctly control the car to pass through the entire scenario. Sub-figure 10(b) was a zoom-in view of Figure 10 (from the beginning to the completion of the first turn), which represented the hardest part of the whole scenario. Although the initialized direction and position of the UGV were not in line with the expectations, the model was able to quickly adapt to the harsh environment with no collisions with the constructed tube and steering the UGV to fitting the desired path to the most, i.e., the center of the tube.

This study only used a very simple reward function for model training. Nevertheless, the resulted tracking policy exhibited excellent performance in path tracking. It significantly outperformed the conventional controllers via minimizing the errors between the desired path and the actual track.

c: CAPABILITY OF ADAPTING TO VARIATIONS OF VELOCITY AND EXTERNAL DISTURBANCES

Three sets of experiments were performed to examine how well the model might adapt to abrupt change of velocity and external disturbances: (1) setting the UGV to operate in a speed four times of the original one; (2) gradually accelerating the UGV to the previous speed; (3) introducing external disturbances while the UGV operated in the original speed, in which noises conforming to a normal distribution (mean: a_t and variance: 4.0) were added to the action of the UGV.

The first results (Sub-figure 11(a)) demonstrated that in spite of the shape increase of speed, the UGV still trespassed the scenario successfully. The second results (Sub-figure 11(b)) showed that the learning policy worked well while the UGV was accelerated with no collision. The third results (Sub-figure 11(c)) exhibited that although the disturbances caused significant jitters abruptly, the UGV quickly responded with rectification to avoid collisions. The tracking policy also worked well with the much more complicated scenario presented in Figure 9. It should be noted that this was achieved without any parameter fine-tuning.

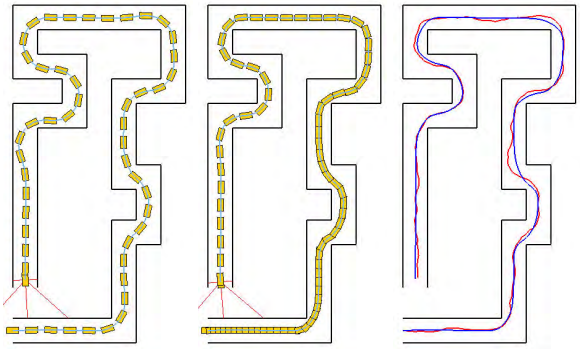


FIGURE 11. Examination of capability of adapting to variations of velocity and external disturbances. (a) quadruple velocity. (b) accelerate. (c) disturbance.

C. DISCUSSIONS

Reinforcement learning has always been an important method both in the control field and machine learning field. In the control field, the stability analysis about systems is always necessary, such as the Lyapunov method. In contrast, the convergence of an algorithm in the context of machine learning is to ensure that the desired knowledge can been learned from the data.

The high-dimensional continuous action space has always been a key issue in traditional RL. In recent years, DDPG has emerged with the problem properly solved, and its convergence has been demonstrated in [42]. This study extends the DDPG method in solving the tracking problem. The experimental results have demonstrated that a proper tracking policy can be learned.

UGV navigation models with generalization ability have also received attentions, such as learning-based and numerical optimization methods [46], [47]:

- A Bayesian learning model [46] aims to gain navigation ability in an unfamiliar environment. The model encode safety constraints as a priori over collision probabilities and can be trained on expert data collected in a separate hallway environment. Expert data and a priori are mandatory otherwise it is impossible for the model to rule out risky behaviors.
- Collision-free path planning can also be formulated as an open-loop pursuit-evasion game [47], which can be solved by the modified fast marching method. However, it remains unclear that these methods can support non-holonomic dynamics associated with car-like robots, i.e., its practicality needs further examination.

Compared with these work, this study shows that deep reinforcement learning can easily apply to the navigation problem without the need for **extra expert data**. The method proposed at the current stage applies in tracking, and for future work deep reinforcement learning will be extended to end-to-end navigation.

V. CONCLUSIONS

This study developed a two-tier approach to navigation optimization in terms of path planning and tracking towards large scale outdoor applications.

A rope model was first designed to optimized any given path planner. It mimic the deformation of a path in axial direction under external force and the fixedness of the radial plane to contain a UGV in a collision-free space. Given the start and end points in any $2D$ environment, the model automatically constructed a tube that defined the optimal path with shortened distance and enhanced smoothness.

A Deep Deterministic Policy Gradient algorithm was used to enable tracking of the resulted optimal path. A simulated scenario was designed that abstracted the paths in real-world scenarios with extremely simple structures. The trained *DDPG* model enables a general policy to control an UGV to track the correct path free of risks by itself in environments of increasing complexity without the need for tuning parameters.

Experiments have been performed over 10 types of complicated environments examined in renowned literatures. The global paths were first planned with A*, the rope model then operated on the paths, and it could significantly shorten the distance and enhance smoothness of the paths in all cases. The *DDPG* can be trained quickly in only a couple of minutes on an office desktop over the simulated scenario. The *DPG*-based controller could then autonomously adjust the UGV model to follow the correct path.

Overall, the proposed method was useful in optimizing the global path with respect to distance and clearance in complicated environments independent of initial path planners. The deep reinforcement learning technique had been able to support a general solution to path tracking free of risks with a low computational cost, which held great potentials in real-time outdoor applications.

APPENDIX A

For revolute constraint, $J_{revolute} = [1\ R_{ra}\ -1\ -R_{rb}]$, according to (6)

$$\begin{aligned} K &= JM_{-1}J^T \\ &= \begin{bmatrix} 1 & R_{ra} & -1 & -R_{rb} \end{bmatrix} \begin{bmatrix} M_a^{-1} & & & \\ & I_a^{-1} & & \\ & & M_b^{-1} & \\ & & & I_b^{-1} \end{bmatrix} \\ &\quad \times \begin{bmatrix} 1 \\ R_{ra}^T \\ -1 \\ -R_{rb}^T \end{bmatrix} \\ &= \begin{bmatrix} M_a^{-1} & R_{ra}I_a^{-1} & -M_b^{-1} & -R_{rb}I_b^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ R_{ra}^T \\ -1 \\ -R_{rb}^T \end{bmatrix} \\ &= \left(\frac{1}{m_a} + \frac{1}{m_b}\right) E_{2\times 2} + R_{ra}I_a^{-1}R_{ra}^T + R_{rb}I_b^{-1}R_{rb}^T \end{aligned} \tag{20}$$

$$\begin{aligned} -J\overrightarrow{v_i} &= -\begin{bmatrix} 1 & R_{ra} & -1 & -R_{rb} \end{bmatrix} \begin{bmatrix} \overrightarrow{v}_{a_i} \\ \omega_{a_i} \\ \overrightarrow{v}_{b_i} \\ \omega_{b_i} \end{bmatrix} \\ &= -(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r_a} - \overrightarrow{v}_{b_i} + \omega_{b_i} \times \overrightarrow{r_b}) \end{aligned} \tag{21}$$

So the $\Delta t\lambda = -K^{-1}(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r_a} - \overrightarrow{v}_{b_i} + \omega_{b_i} \times \overrightarrow{r_b})$. Substitute this formula into (5), then can get:

$$\begin{aligned} \begin{bmatrix} \overrightarrow{v}_{a_{i+1}} \\ \omega_{a_{i+1}} \\ \overrightarrow{v}_{b_{i+1}} \\ \omega_{b_{i+1}} \end{bmatrix} &= \begin{bmatrix} \overrightarrow{v}_{a_i} \\ \omega_{a_i} \\ \overrightarrow{v}_{b_i} \\ \omega_{b_i} \end{bmatrix} - M^{-1}J^TK^{-1} \\ &\quad \times (\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r_a} - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r_b}) \\ &= \begin{bmatrix} \overrightarrow{v}_{a_i} \\ \omega_{a_i} \\ \overrightarrow{v}_{b_i} \\ \omega_{b_i} \end{bmatrix} - M^{-1} \begin{bmatrix} 1 \\ R_a^T \\ -1 \\ -R_b^T \end{bmatrix} K^{-1} \\ &\quad \times (\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r_a} - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r_b}) \\ &= \begin{bmatrix} \overrightarrow{v}_{a_i} \\ \omega_{a_i} \\ \overrightarrow{v}_{b_i} \\ \omega_{b_i} \end{bmatrix} + \begin{bmatrix} \frac{\overrightarrow{P}}{m_a} \\ I_a^{-1}R_{ra}^T\overrightarrow{P} \\ -\frac{\overrightarrow{P}}{m_b} \\ -I_b^{-1}R_{rb}^T\overrightarrow{P} \end{bmatrix} \end{aligned} \tag{22}$$

where $\overrightarrow{P}_{revolute} = -K^{-1}(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r_a} - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r_b})$.

The same process for collision constraint. $J_{collision} = \begin{bmatrix} \overrightarrow{n^T} & (r_a \times n)^T & -\overrightarrow{n^T} & -(r_b \times n)^T \end{bmatrix}$, according to (6)

$$\begin{aligned} K &= JM^{-1}J^T \\ &= \begin{bmatrix} \overrightarrow{n^T} & (r_a \times n)^T & -\overrightarrow{n^T} & -(r_b \times n)^T \end{bmatrix} \begin{bmatrix} M_a^{-1} & & & \\ & \mathrm{I_a^{-1}} & & \\ & & M_b^{-1} & \\ & & & \mathrm{I_b^{-1}} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \overrightarrow{n} \\ (r_a \times n) \\ -\overrightarrow{n} \\ -(r_b \times n) \end{bmatrix} \\ &= \begin{bmatrix} \overrightarrow{n^T}M_a^{-1} & (r_a \times n)^T\mathrm{I_a^{-1}} & -\overrightarrow{n^T}M_b^{-1} & -(r_b \times n)^T\mathrm{I_b^{-1}} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \overrightarrow{n} \\ (r_a \times n) \\ -\overrightarrow{n} \\ -(r_b \times n) \end{bmatrix} \\ &= \overrightarrow{n^T}M_a^{-1}\overrightarrow{n} + (r_a \times n)^T\mathrm{I_a^{-1}}(r_a \times n) + \overrightarrow{n^T}M_b^{-1}\overrightarrow{n} \\ &\quad + (r_b \times n)^T\mathrm{I_b^{-1}}(r_b \times n) \\ &= \frac{1}{m_a} + \frac{1}{m_b} + \mathrm{I_a^{-1}}(n \times r_a)^2 + \mathrm{I_b^{-1}}(n \times r_b)^2 \end{aligned} \tag{23}$$

$$-J\overrightarrow{v_i} = -\begin{bmatrix}\overrightarrow{n^T} & (r_a \times n)^T - \overrightarrow{n^T} & -(r_b \times n)^T\end{bmatrix} \times \begin{bmatrix}\overrightarrow{v}_{a_i} \\ \omega_{a_i} \\ \overrightarrow{v}_{b_i} \\ \omega_{b_i}\end{bmatrix} = -\left(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r}_a - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r}_b\right) \cdot \overrightarrow{n} \quad (24)$$

So the $\Delta t\lambda = -K^{-1}(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r}_a - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r}_b) \cdot \overrightarrow{n}$. Substitute this formula into (5), then can get:

$$\begin{aligned}\begin{bmatrix}\overrightarrow{v}_{a_{i+1}} \\ \omega_{a_{i+1}} \\ \overrightarrow{v}_{b_{i+1}} \\ \omega_{b_{i+1}}\end{bmatrix} &= \begin{bmatrix}\overrightarrow{v}_{a_i} \\ \omega_{a_i} \\ \overrightarrow{v}_{b_i} \\ \omega_{b_i}\end{bmatrix} - M^{-1}J^TK^{-1} \times \left(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r}_a - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r}_b\right) \cdot \overrightarrow{n} \\ &= \begin{bmatrix}\overrightarrow{v}_{a_i} \\ \omega_{a_i} \\ \overrightarrow{v}_{b_i} \\ \omega_{b_i}\end{bmatrix} - M^{-1}\begin{bmatrix}\overrightarrow{n} \\ (r_a \times n) \\ -\overrightarrow{n} \\ -(r_b \times n)\end{bmatrix} K^{-1} \times \left(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r}_a - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r}_b\right) \cdot \overrightarrow{n} \\ &= \begin{bmatrix}\overrightarrow{v}_{a_i} \\ \omega_{a_i} \\ \overrightarrow{v}_{b_i} \\ \omega_{b_i}\end{bmatrix} + \begin{bmatrix}\frac{\overrightarrow{P}}{m_a} \\ \mathrm{I}_a^{-1}\overrightarrow{r_a} \times \overrightarrow{P} \\ -\frac{\overrightarrow{P}}{m_b} \\ -\mathrm{I}_b^{-1}\overrightarrow{r_b} \times \overrightarrow{P}\end{bmatrix}\end{aligned} \quad (25)$$

where $\overrightarrow{P}_{collision} = P_{collision}\overrightarrow{n}$, $P_{collision} = -K^{-1}\left(\overrightarrow{v}_{a_i} + \omega_{a_i} \times \overrightarrow{r}_a - \overrightarrow{v}_{b_i} - \omega_{b_i} \times \overrightarrow{r}_b\right) \cdot \overrightarrow{n}$

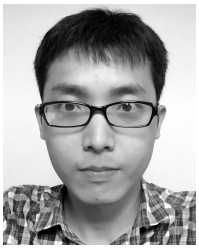
Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 61772380) and the Foundation for Innovative Research Groups of Hubei Province (No. 2017CFA007).

REFERENCES

[1] B. Siciliano and O. Khatib *Springer Handbook of Robotics*, 2nd ed. Berlin, Germany: Springer-Verlag, 2016.
[2] H.-P. Huang and S.-Y. Chung, "Dynamic visibility graph for path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Sep./Oct. 2004, pp. 2813–2818.
[3] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, 1979.
[4] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," in *Proc. 4th Int. Symp. Voronoi Diagrams Sci. Eng. (ISVD)*, Jul. 2007, pp. 38–47.
[5] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Hoboken, NJ, USA: Wiley, 2006, pp. 163–182.
[6] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, Apr. 1991, pp. 1398–1404.
[7] C. W. Warren, "Global path planning using artificial potential fields," in *Proc. Int. Conf. Robot. Automat.*, May 1989, pp. 316–321.
[8] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, vol. 5, Apr./May 2004, pp. 4350–4355.
[9] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: Artificial bee colony (ABC) algorithm and applications," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 21–57, 2014.
[10] R. C. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proc. Congr. Evol. Comput.*, vol. 1. May 2001, pp. 81–86.
[11] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 1993, pp. 802–807.
[12] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *Int. J. Robot. Res.*, vol. 21, no. 12, pp. 1031–1052, 2002.
[13] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. 54th IEEE Conf. Decis. Control (CDC)*, Dec. 2015, pp. 835–842.
[14] T. Das and I. N. Kar, "Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots," *IEEE Trans. Control Syst. Technol.*, vol. 14, no. 3, pp. 501–510, May 2006.
[15] R.-J. Wai and C.-M. Liu, "Design of dynamic petri recurrent fuzzy neural network and its application to path-tracking control of nonholonomic mobile robot," *IEEE Trans. Ind. Electron.*, vol. 56, no. 7, pp. 2667–2683, Jul. 2009.
[16] K. Kanjanawanishkul and A. Zell, "Path following for an omnidirectional mobile robot based on model predictive control," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2009, pp. 3341–3346.
[17] C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking," *J. Field Robot.*, vol. 33, no. 1, pp. 133–152, 2015.
[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, no. 1. Cambridge, MA, USA: MIT Press, 1998.
[20] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2722–2730.
[21] T. Kollar and N. Roy, "Trajectory optimization using reinforcement learning for map exploration," *Int. J. Robot. Res.*, vol. 27, no. 2, pp. 175–196, 2008.
[22] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 387–395.
[23] S. Paul and L. Vig, "Deterministic policy gradient based robotic path planning with continuous action spaces," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 725–733.
[24] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2017, pp. 3389–3396.
[25] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. (2017). "Overcoming exploration in reinforcement learning with demonstrations." [Online]. Available: https://arxiv.org/abs/1709.10089
[26] L. Tai, G. Paolo, and M. Liu. (2017). "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation." [Online]. Available: https://arxiv.org/abs/1703.00420
[27] P. Mirowski *et al.* (2016). "Learning to navigate in complex environments." [Online]. Available: https://arxiv.org/abs/1611.03673
[28] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1329–1338.
[29] X. B. Peng and M. van de Panne, "Learning locomotion skills using Deeprl: Does the choice of action space matter?" in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*, 2017, p. 12.
[30] M. Zucker *et al.*, "CHOMP: Covariant Hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, nos. 9–10, pp. 1164–1193, 2013.
[31] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, 2014.
[32] M. Davoodi, F. Panahi, A. Mohades, and S. N. Hashemi, "Multi-objective path planning in discrete space," *Appl. Soft Comput.*, vol. 13, no. 1, pp. 709–720, 2013.
[33] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization," *Appl. Soft Comput.*, vol. 59, pp. 68–76, Oct. 2017.
[34] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Appl. Soft Comput.*, vol. 30, pp. 319–328, May 2015.

[35] J. Baltes and Y. Lin, "Path tracking control of non-holonomic car-like robot with reinforcement learning," in *Robot Soccer World Cup*. Berlin, Germany: Springer, 1999, pp. 162–173.
[36] L. Zuo, X. Xu, C. Liu, and Z. Huang, "A hierarchical reinforcement learning approach for optimal path tracking of wheeled mobile robots," *Neural Computing Appl.*, vol. 23, nos. 7–8, pp. 1873–1883, 2013.
[37] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 1–8.
[38] Y.-J. Liu and S. Tong, "Optimal control-based adaptive NN design for a class of nonlinear discrete-time block-triangular systems," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2670–2680, Nov. 2016.
[39] L. Liu, Z. Wang, and H. Zhang, "Adaptive fault-tolerant tracking control for MIMO discrete-time systems via reinforcement learning algorithm with less learning parameters," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 299–313, Jan. 2017.
[40] E. Catto, "Iterative dynamics with temporal coherence," in *Proc. Game Developers Conf.*, vol. 2, no. 4, 2005, p. 5.
[41] E. Catto, "Modeling and solving constraints," in *Proc. Game Developers Conf.*, 2009, p. 16.
[42] T. P. Lillicrap *et al.* (2015). "Continuous control with deep reinforcement learning." [Online]. Available: https://arxiv.org/abs/1509.02971
[43] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.
[44] J. Han and Y. Seo, "Mobile robot path planning with surrounding point set and path improvement," *Appl. Soft Comput.*, vol. 57, pp. 35–47, Aug. 2017.
[45] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2014, pp. 4029–4036.
[46] C. Richter, W. Vega-Brown, and N. Roy, "Bayesian learning for safe high-speed navigation in unknown environments," in *Robotics Research*. Cham, Switzerland: Springer, 2018, pp. 325–341.
[47] R. Takei, H. Huang, J. Ding, and C. J. Tomlin, "Time-optimal multi-stage motion planning with guaranteed collision avoidance via an open-loop game formulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2012, pp. 323–329.



MINGGAO WEI received the bachelor's degree from the Dalian University of Technology. He is currently pursuing the M.D. degree with the School of Computer Science, Wuhan University. His main research interests include machine learning and multi-agent systems.

SONG WANG received the bachelor's degree from the China University of Geosciences. He is currently pursuing the M.D. degree with the School of Computer Science, Wuhan University. His main research interests include machine learning and multi-agent systems.

JINFAN ZHENG is a currently pursuing the bachelor's degree with the School of Computer Science, Wuhan University. His main research interests include transfer learning and multitask learning.

DAN CHEN was an HEFCE Research Fellow with the University of Birmingham, U.K. He is currently a Professor with the School of Computer Science, Wuhan University, Wuhan, China. His research interests include data science and engineering, high-performance computing, and modeling and simulation of complex systems.