Hierarchical Reinforcement Learning Considering Stochastic Wind Disturbance for Power Line Maintenance Robot

Xiaoliang Zheng (✉ zxlxiaoyi@163.com)
Wuhan University

Gongping Wu
Wuhan University

Research Article

Keywords: Hierarchical Reinforcement Learning, Stochastic Wind Disturbance, power line maintenance robot

DOI: https://doi.org/10.21203/rs.3.rs-783306/v1

License: ☑️ This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Hierarchical Reinforcement Learning Considering Stochastic Wind Disturbance for Power Line Maintenance Robot

Xiaoliang Zheng¹ · Gongping Wu¹

¹School of Power and Mechanical Engineering, Wuhan University, Wuhan, China

Corresponding authors

Xiaoliang Zheng, Wuhan University, Wuhan 430072, China.

Email: zxlpiaoyi@163.com

Gongping Wu, Wuhan University, Wuhan 430072, China.

Email: gpwu@whu.edu.cn

Abstract

Robot intelligence includes motion intelligence and cognitive intelligence. Aiming at the motion intelligence, a hierarchical reinforcement learning architecture considering stochastic wind disturbance is proposed for the decision-making of the power line maintenance robot with autonomous operation. This architecture uses the prior information of the mechanism knowledge and empirical data to improve the safety and efficiency of the robot operation. In this architecture, the high-level policy selection and the low-level motion control at global and local levels are considered comprehensively under the condition of stochastic wind disturbance. Firstly, the operation task is decomposed into three sub-policies: global obstacle avoidance, local approach and local tightening, and each sub-policy is learned. Then, a master policy is learned to select the operation sub-policy in the current state. The dual deep Q network algorithm is used for the master policy, while the deep deterministic policy gradient algorithm is used for the operation policy. In order to improve the training efficiency, the global obstacle avoidance sub-policy takes the random forest composed of dynamic environmental decision tree as the expert algorithm for imitation learning. The architecture is applied to a power line maintenance scenario, the state function and reward function of each policy are designed, and all policies are trained in an asynchronous and parallel computing environment. It is proved that this architecture can realize stable and safe autonomous operating decision for the power line maintenance robot subjected to stochastic wind disturbance.

Keywords Hierarchical Reinforcement Learning · Stochastic Wind Disturbance · power line maintenance robot
1 Introduction

In the electric power field with large coverage and complex network, the power transmission lines (PTLs) often need to be maintained to ensure the stability and reliability of the power supply system. Electric workers are usually in high altitude, high voltage working environment where contains safety risks, and high requirements on the technical level of operators. Therefore, in such a dangerous and complex environment, it is appropriate to use robots instead of human to carry out live operation. Autonomous operation is a promising grid robotics technology that improves the safety and efficiency of live operations, reduces energy consumption, and liberates human operators. In the architecture of autonomous operating robot, decision–making is the key component of autonomous operation. So far, most decision algorithms can be divided into two broad categories: rule-based approaches and imitation-based approaches. The former is usually addressed by manually coding rules for robot behavior, while the latter focuses on using supervised learning techniques to mimic the operations of electrical workers. In order to make safe and efficient decisions under different operating conditions, both rely on a large amount of natural operational data to cover all possible operating scenarios for testing or training.

The rule-based approach has been widely studied in the field of Power Line Maintenance Robot (PLMR). Expliner robot [1] is designed to cross overhang clips on PTLs in an "acrobatic mode" and can be controlled in a semi-automatic manner. After the engineer inputs the position data of the encoder into the memory of the controller, it can be mapped into the motion sequence by certain rules, and the staff can make the robot reach the correct predetermined position only by simple operation of the manipulator. LineScout robot [2] uses unified strategies and rules to safely and controllably coordinate all motors through visual perception. In order to avoid human error and the possibility of accidentally damaging Linescout, three safety interlocking rules are implemented in the control software, including speed limit, mode lock, and linkage limit. The sequencer plans the movement of the robot according to these rules and assists the operator in correctly crossing obstacles. Lioutikov and Maeda et al. [3] intuitively codes the hierarchical architecture by using formal grammar based on the nature of rules, and constrains and effectively searches the grammatical space by using the physical properties of robot motion primitives. The adaptability of simple motion primitives and the continuity of complex motion sequences are verified by the proposed 7-DOF lightweight manipulator. Ananas and Moreno et al. [4] proposes a robot grasping method based on probabilistic logic framework, which uses semantic object components to improve its grasping ability. The framework semantically interprets the pre-grab configurations associated with the expected task and uses object-task implications and object/task ontologies to encode rules that generalize similar object parts and object/task classes. Muni and Parhi et al. [5] take the sensor's output on the distance of obstacles as the input of the rule base model, and obtains the turning angle as the output. A rule-based analysis method is used to train the fuzzy controller with membership function. The output of the rule-base model and other conventional inputs are provided to the Sugeno fuzzy model, and the effective steering angle is used as the final output to avoid obstacles existing in the environment and safely guide the humanoid robot to the target point. Gordic and Jovanovic et al. [6] develops two complementary decision rules to identify conflicts. The first rule, based on the absolute difference between matched samples after comparison, uses a statistically determined threshold to perform a quick detection of clear conflicts. The second rule is based on the eigenvalue of the matched sample covariance matrix, which uses its higher sensitivity to detect collisions with lower intensity.
The method based on learning is an effective way to solve the problem of PLMR. Yan Yu et al. [7], combining the nonlinear mapping and approximation characteristics of neural network, abstracts the evaluation of insulator cleaning effect into a nonlinear approximation process from the actual cleaning effect to the ideal cleaning effect. An evaluation method of robot insulator cleaning effect based on neural network is proposed. Through the training of the neural network, the cleaning control parameters of the robot can be obtained and used in the online operation control of the robot, so that better cleaning effect can be obtained. Jiang Wei et al. [8], based on the nonlinear approximation characteristics of neural network, uses neural network to solve the problem of fault diagnosis of the system. In the control method of PLMR, the tilt angle of the robot caused by wind load and space field strength is the input signal of the neural network. Through the training and learning of the neural network, the output control variables are used to compensate the actual operating posture of the robot. Compared with the traditional proportional integral differential (PID) control, this algorithm has high real-time performance and good stability. Li Hongjun et al. [9] derives the state space expression of joint motion control under the condition of disturbance and uncertainty, and establishes the trajectory tracking control model of the manipulator for sliding mode variable structure. The neural network learning method is used to compensate the influence of the perturbation control parameters on the motion control of the robot, and the stability of the controller is analyzed by using Lyapunov theory. The control method has the characteristics of good generality, strong adaptability and easy extension. Xiang and Su [10] proposes an effective model-free off-policy actor-critic algorithm by integrating task reward and task-oriented guiding reward, and applies it to the skill acquisition and continuous control of the robot. The agent can explore the environment more consciously so as to realize the sampling efficiency. It can also leverage the experience more effectively, resulting in significant performance gains. Sasabuchi and Wake et al. [11] propose a body role division method suitable for different robots. It guides the configuration of the robot to complete a given task, which helps to teach a series of task sequences. Using a single human demonstration to combine task and movement knowledge, the method starts from the fact of human movement and decomposes the robot's body configuration into different roles by using the structure analogy of human body. Zuo and Zhao et al. [12] proposes a new off-policy AIL method — Robust Adversarial Imitation Learning (RAIL). The hindsight idea of variable reward (VR) is first incorporated into the off-policy AIL framework in order to make the agent significantly better than the suboptimal expert who provided the instruction. In the AIL framework, the hindsight copy (HC) strategy designed by the new method maximizes the use of different teaching methods and speeds up the learning. In recent years, Hierarchical Reinforcement Learning (HRL) is regarded as a reinforcement learning algorithm for more complex environments. Ahn and Song et al. [13] proposes a method for learning decision ability in a messy environment, enabling the robot to grasp the target object after rearranging the objects that hinder the target. Learning takes a hierarchical architecture, consisting of high- and low-level motion selectors. The former determines the grab or push actions, and the latter determines how to perform those selected actions. Saha and Gupta et al. [14] uses HRL to learn the optimal strategy to operate at different time steps in order to achieve user goals in the hierarchy of dialogue exchange. The dialog manager consists of a high-level domain meta-policy, a mid-level intent meta-policy for selecting from multiple seed-tasks or options, and a low-level control policy for selecting basic actions to complete subtasks given by high-level meta-policies in different intents and domains. Duan and Li et al. [15] proposes a HRL method for the decision-making of autonomous vehicles, which does not rely on a large number of marked driving data. This method takes into account both horizontal and vertical
directions of high-level control selection and low-level motion control. Firstly, the driving task is divided into lane driving, right lane change and left lane change, and the sub-policies of each maneuver are studied. Then, a master policy is learned to select the manipulation policy in the current state. Li and Srinivasan et al. [16] proposes a layered approach that relies on traditional model-based controllers at the lower level and learning policies at the middle level. Low level controllers can robustly perform different operating primitives (pose, slide, roll). The middle level policy orchestrates these primitives. The method is robust to the inaccuracy of the target model and observation noise. Pang and Liu et al. [17] studies a layering approach in which layering involves two levels of abstraction. One is a macro-operation extracted from an expert demo trajectory that can shrink the action space by an order of magnitude, but still be effective. The other is a two-level, hierarchical architecture that is modular and easy to extend. The agent is trained from easy to hard through a curriculum transfer learning method. In order to solve the performance problem in path planning, Yu and Su et al. [18] uses neural network to enable robots to perceive the environment and perform feature extraction, which enables them to have state-action functions adapted to the environment. The path planning model of mobile robot is constructed by mapping the current state of these actions by HRL.

The influence of stochastic wind load on mechanism/cable systems, especially in the high-altitude environment, cannot be ignored. Alhassan and Zhang et al. [19] develops a lightweight, dual-arm robot and studies its robustness to wind disturbances on a lab-scale PTL structure. At the same time, the design of the robot components ensures the low drag coefficient of the airflow, and the mechanism of the wind force acting on the robot-line coupling system is given. Lu and Liu et al. [20] proposes a finite frequency H∞ controller based on prior information for active control of double telescopic boom. The robustness of the control strategy with adaptive estimator to stochastic wind field is studied. Although the stochastic wind field greatly aggravates the fluctuation of contact force, the control strategy has a good effect. In order to study the dynamic response of transmission tower line system under the action of wind and rain load, Fu and Li et al. [21] proposes the calculation method of wind and rain load acting on PTLs. Modeling and dynamic analysis are carried out for three-span conductors and pylon system, and the effect of wind attack angle on the response of pylon system under wind and rain load is studied. Pombo and Ambrosio [22] develops a multi-body/finite element coupled computational model. In the co-simulation environment, the stroke field forces act as nonlinear external forces that vary with time and are distributed on the finite element grid to exert on each object. These wind field forces can be characterized by computational fluid dynamics or wind tunnel experiments. Zdziebko and Martowicz et al. [23] uses the fluid-structure-interaction (FSI) method with the Altair AcuSolve solver to calculate the additional lift force caused by airflow at high speeds. The Spalart-Allmaras turbulence model is considered in the steady-state simulation. A rule-based controller is used to output a suitable torque taking into account the actual speed, aerodynamic force and electromagnetic force. Song and Liu et al. [24] uses empirical spectrum to simulate the fluctuating wind speed and stochastic wind field along the catenary. The chattering force acting on the contact line and the suspension cable is deduced. In order to improve the current quality of pantograph system under strong stochastic wind field, a proportional differential (PD) sliding mode controller for high-speed active pantograph is proposed.

The main contribution of this research is to propose a HRL architecture considering stochastic wind disturbance for the decision-making of the PLMR with autonomous operation. This architecture utilizes the prior information of mechanism knowledge (fluid-rigid-flexible coupling dynamics) and empirical data (random forest composed
of decision tree) to improve the safety and efficiency of PLMR operation. In this architecture, the high-level policy selection and the low-level motion control at global and local levels are considered comprehensively under the condition of stochastic wind disturbance. Firstly, the task is decomposed into three sub-policies: global obstacle avoidance, local approach and local tightening, and each sub-policy is learned. Then, a master policy is learned to select the operation sub-policy in the current state. All policies (including master policy and operation policy) are represented by fully connected neural networks (FCNNs). The master policy uses dual deep Q network (DDQN) algorithm, while the operation policy uses deep deterministic policy gradient (DDPG) algorithm. In order to improve the training efficiency, the global obstacle avoidance sub-policy takes the random forest composed of dynamic environmental decision tree (rapid-exploration random tree or rapid search tree) as the expert algorithm for imitation learning. The architecture is applied to a PTL maintenance scenario, and the state function and reward function of each policy are designed respectively. All policies in asynchronous parallel computing environment (APCE) to complete the training.

The rest of this paper is organized as follows: Section 2 proposes a hierarchical reinforcement learning architecture for decision-making of the robot. Section 3 describes the design of the experiment, the state space, and the reward function. Section 4 discusses the training results of each operation policy and task. Section 5 is the conclusion of this paper.

2 Methodology
2.1 Preliminaries
The PLMR operates autonomously using the Markov Decision Process (MDP): for the step \( i \) of running, the robot observes a state vector \( S_i \), takes an action vector \( A_i \), receives a scalar reward \( R_i \), and then reaches the next state \( S_{i+1} \). Reward \( R_i \) is usually designed by human experts, with a lower value of \( R_i \) when the robot is in a bad state (such as a collision) and a higher value of \( R_i \) when it is in a normal state (such as keeping a safe distance from PTL). This process continues until the robot reaches a termination state, such as a collision or goal completion. In this investigation, the reinforcement learning goal is: for each state \( S_i \), it finds a deterministic policy \( \pi(s) \) that maximizes the value function \( v_\pi(s) = E_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k R_{i+k} | S_i = s \right] \) representing the expected return, where \( \gamma \in (0,1] \) is a discount factor that weighs the importance of current and future rewards.

The decision-making process of robots utilizes deep neural networks to approach the value function \( v_\pi(s) \) and deterministic policy \( \pi(s) \) in Q learning and actor-critic architecture. The value network corresponding to the value function \( v_\pi(s) \) maps the state to the expected future cumulative rewards, while the policy network corresponding to the deterministic policy \( \pi(s) \) maps the state to the action that control the robot. Deterministic policies are improved based on value functions. At the same time, the value function always expects to converge to the value of the current policy. If both processes are stable, then the optimal policy must be achieved. General policy iteration (GPI) framework to describe the architecture, and the Bellman Optimality Equation to iteration.

DDQN and DDPG algorithms are used in the HRL mentioned in this study for the decision-making characteristics of the PLMR. DDQN algorithm is an unbiased estimation version of DQN. In order to reduce the influence of overestimation as much as possible, the algorithm separates the work of selecting and estimating the optimal action, and reduces the bias of judging whether the action is good or bad, so the stability of the algorithm is improved. DDPG algorithm absorbs the ideas of DQN and DPG and a lot of improvement schemes, so that the
model can play a better effect in the high dimensional continuous space, and has the guarantee in the training speed and convergence.

Their action policy output \( A \) can be respectively expressed as:

\[
A = \pi(S) = \begin{cases} 
\arg \max_A Q(S, A|\theta_Q) & \text{for DDQN} \\
\mu(S|\theta_\mu) + N & \text{for DDPG} 
\end{cases}
\]

\( Q(S, A|\theta_Q) \) and \( \mu(S|\theta_\mu) \) are value and policy networks respectively; \( \theta_Q \) and \( \theta_\mu \) are their parameter vectors respectively; \( N \) is the stochastic noise vector from the noise model.

Their value function approximates \( L \), which can be uniformly described as minimizing the following loss function:

\[
L = \frac{1}{K} \sum_{i=1}^{K} [y_i - Q(S_i, A_i|\theta_Q)]^2
\]  
(2)

where \( K \) is the sampling number of experience tuple \( (S, A, R, S') \). The objective of value function can be written as follows:

\[
y_i = \begin{cases} 
R_i + \gamma Q' \left( S'_i, \arg \max_A Q(S'_i, A'|\theta_Q) \right) & \text{for DDQN} \\
R_i + \gamma Q' \left( S'_i, \mu(S'_i|\theta_\mu) \right) & \text{for DDPG} 
\end{cases}
\]

(3)

For the DDPG algorithm, the action policy improvement \( \nabla_{\theta_\mu} J \) can be described as:

\[
\begin{aligned}
\nabla_{\theta_\mu} J &\approx \frac{1}{K} \sum_{i=1}^{K} G_{ai} G_{\mu i} \\
G_{ai} &= \nabla_A Q(S_i, \mu(S_i|\theta_\mu)|\theta_Q) \\
G_{\mu i} &= \nabla_{\theta_\mu} \mu(S_i|\theta_\mu)
\end{aligned}
\]

(4)

2.2 Hierarchical Reinforcement Learning

2.2.1 Overall Architecture

Previous studies on reinforcement learning usually use end-to-end networks to make decisions for autonomous operating robots. In order to complete a task that consists of a variety of tasks, human experts need to design complex reward functions. Inspired by the option framework, this study proposes a robot learner based on HRL architecture, which decomposes the task decision process into two levels to enable the robot to obtain the ability of autonomous operation, as shown in Fig. 1. The architecture consists of two abstraction levels: (a) high-level policy selection and (b) low-level motion control. At the policy selection level, a master policy (DDQN agent) is used to select the operation policy to be executed in the current state. At the motion control level, the corresponding operation policy (DDPG agent) will be activated and output torques from the actuators (components of the motor and reducer). Multiple combinations of operation behaviors can form diversified operation tasks, which means that the trained policies can also be applied to other tasks. Therefore, hierarchical architecture has better portability in the field of operation decision-making than end-to-end reinforcement learning.
In order to learn the policies of HRL, the task should be decomposed into several operations, namely, global obstacle avoidance, local approach and local tightening. The robot can then learn a task policy consisting of three sub-policies, each driven by an independent sub-goal, as shown in Fig. 2. The reward function of sub-policy learning only considers the corresponding task, but does not consider the whole task. The robot then learns a master policy and activates certain sub-policies based on operation tasks. Although the entire operation task
needs to be considered when training the master policy, the associated reward function can be very simple because it doesn’t have to worry about how to control the actuator to achieve each operation. In addition, if the input to each policy is the entire perceptual information, then the learning algorithm must determine which information is relevant. Therefore, this study suggests that different meaningful indicators be designed as state representations of different policies. In addition, the movement of the robot is realized by a plurality of controllers jointly controlling the actuators, which are relatively independent of each other. It is noted that the robot dynamics used in this study still takes into account the coupling between the controllers. In summary, if an operation policy is selected by the master policy, the relevant controllers will work simultaneously to enforce control.

2.2.2 Network Architecture and Hyperparameters

As the emphases of master policy $\pi^m_\theta(s)$ and operation policy (global obstacle avoidance $\pi^g_\theta(s)$, local approach $\pi^a_\theta(s)$, and local tightening $\pi^t_\theta(s)$) are different, the corresponding network architecture is also different. $\theta_g$, $\theta_a$, and $\theta_t$ are network parameters of the three operation policies respectively. The master policy needs to uniquely select the appropriate operation policy according to the current state of the robot/PTL system. The action space is the discrete code representing the policy, so DDQN agent is used to execute it. The action space of the operation policy is continuous torque of each joint of the robot, so DDPG agent is used to realize it. All agents are trained in an Asynchronous Parallel Computing Environment (APCE). When training an agent using parallel computing, the algorithm sends copies of the agent and the environment to each parallel worker. Each worker simulates the agent’s activity in the environment and sends its experiential data back to the master agent. The master agent learns from the data sent by the worker and sends updated policy parameters back to the worker.

**Master policy:** this policy uses DDQN agent, including a value network to evaluate the merits of each operation policy and a value best-first policy for operation policy selection. The input of the value network is the position and torque of each joint of the robot, the position of the PTL (assuming the posture remains constant) and the error metrics between the end of the manipulator and the task target. In order to make the value network have the expansibility to the visual image, the input end uses the image layer. The hidden layer consists of two fully-connected layers, the first layer has 128 nodes and the second layer has 256 nodes, all of which are Rectified Linear Units (ReLUs). The output end of the network is coded for the selected policy and adopts the full connection layer.

**Global obstacle avoidance:** this sub-policy uses a DDPG agent, including a critic representing a value network to evaluate the merits of the current action and an actor representing a policy network to output the action in the current state. Compared with the value network of the master policy (referred to as the master value network), the value network only evaluates for the current action, that is, the input end of the network is the combined input of the current state of the robot/PTL system and the current action of the robot. The hidden layer of the current state of the robot/PTL system is the same as that of the main value network. The hidden layer of the current action of the robot is a fully connected layer with 256 nodes, all of which adopt ReLUs. The output end of the network is the expected return of the current action, using the full connection layer. The input layer and the hidden layer of the policy network are the same as that of the master value network.
network is continuous torque of each joint of the robot, which adopts full connection layer. In order to facilitate the observation of critics of DDPG agents, the output of the policy network is normalized and hyperbolic tangent (tanh) layer is used. The actor needs to explore in the global space, so the output signal is scaled up to a certain extent to get the appropriate coverage.

**Local approach and local tightening:** the network model of these two sub-policies is the same, and the architecture of the DDPG agent is roughly the same as that of the global obstacle avoidance, except that the critic has been redesigned to fully meet the positioning accuracy for the end tool of the manipulator during the local operation. The velocity of each joint of the robot is taken into account, and the pose and coaxiality error between the end tool and the task target are also paid attention to, but the position error of the wrist joint is not considered. For the local task, the output signal of the policy network is limited to a reasonable range, that is, the amplification ratio is reduced to half.

### 2.2.3 Imitative Learning for Global Obstacle Avoidance

For the global obstacle avoidance sub-policy, the PLMR to finish in the relatively complex, narrow space obstacle avoidance operation, needs more exploration and trial and error in the process of training, in order to avoid the local optimal value generated from the policy into obstacles. Therefore, even if multi-agent learning is adopted, it is difficult to learn appropriate policies in a relatively short time. The actors of the global obstacle avoidance sub-policy are initialized using a deep neural network to make the training more efficient. The deep neural network has previously been trained using supervised learning. The expert algorithm of this actor is the motion planner \[25\] of the PLMR previously studied by our team. The planner can efficiently navigate the robot under various dynamic constraints in a slightly disturbed environment. These dynamic constraints include target arrival time, actuator energy consumption and ultimate torque, etc.

The target of the global obstacle avoidance sub-policy is to transfer the current state \(\mathbf{S}\) to the target state \(\mathbf{S}^*\), if there is no collision. The robot's state transition can be expressed as \(p(\mathbf{S}_{i+1}|\mathbf{S}_i, \mathbf{u}_i)\). In these states, the output torque of the actuator is \(\mathbf{u} = \pi^g(\mathbf{s})\) and the expected value of the torque is written as \(\mathbf{u}^*\). Expectations come from expert algorithms or real robots operated by electrical engineers. Note that since \(\mathbf{u}_i^*\) and \(\mathbf{u}_i\) are different and that \(\mathbf{u}_i\) may violate the kinematic or dynamic constraints of robots, \(\mathbf{S}_i^*\) and \(\mathbf{S}_{i+1}\) are often different. The global obstacle avoidance sub-policy \(\pi^g(\mathbf{s})\) optimizes the performance by improving the learnable parameter \(\theta^g\). This policy can be optimized according to the actual situation of known information in one of two forms: \(\mathbf{S}^*\) is known or \(\pi^g(\mathbf{s})\) is known, the latter being selected in this study. In this context, the imitative learning problem that the global obstacle avoidance sub-policy hopes to solve can be described as the minimization loss function \(L_g = \frac{1}{K_g} \sum_{i=1}^{K_g} D_i\), in which \(K_g\) is the number of samples of experience pair (\(\mathbf{S}, \mathbf{u}^*\)) and:

\[
D_i = \begin{cases} 
|\mathbf{S}_i^* - \mathbf{S}_i|^2 & \text{known } \mathbf{S}^* \\
|\pi^g(\mathbf{S}_i) - \pi^g(\mathbf{S}_i)|^2 & \text{known } \pi^g(\mathbf{s}) 
\end{cases}
\]

\(\pi^g(\mathbf{s})\) provides optimal control action by expert. The expert is following the baseline planner algorithm defined in \[25\] and is also readily available in the typical human-in-loop task. In order to find the initial parameters \(\theta^{init}_g\) of the global obstacle avoidance policy, the random forest \[26\] is used as a potential learnable expert policy in the imitation learning framework. The random forest sets a large number of decision trees with random initial
configuration (initial velocity is 0) as the root, and the decision trees grow exploratory or objective into any collision-free state space by adopting the rapid-exploration random tree (RRT) or rapid search tree (RST) algorithm under dynamic environment (DE). The simulation learning method can be decomposed into two sub-steps: dataset sampling and policy optimization. The first step is to sample a dataset \( D = \{(S, u^*)\} \), in which each sample is a combination of state observations and the best action. The second step is to train the initialization policy \( \pi^\theta_g(s) \) for global obstacle avoidance, based on the random forest with given \( D \), as shown in the upper left of Fig. 1.

### 2.2.4 Stochastic Wind Disturbance

The PLMR operates in a special extreme environment. Among many external disturbance factors, wind load is the main influencing factor due to its randomness, time-varying and non-linearity. Due to the stochastic high-altitude wind load, the rigid robot sways and vibrates on the flexible PTL, and the eddy current falls off. The turbulence generated in the boundary layer near the robot body excites the arm and end-effector of the manipulator, which will affect the positioning accuracy for its end tool. Therefore, the research on HRL of the PLMR under wind load has strong theoretical significance and practical application value. According to the rigid-flexible coupling dynamics of robot and PTL without wind, the mathematical model with stochastic wind disturbance is further established by mapping the wind force to the joint of the robot and the finite element nodes of PTL in this section. The robot can be used for reinforcement learning in the simulation environment with stochastic wind disturbance, which improves the robustness of the control for the manipulator end-effector.

In order to better study the influence of wind field on robot/PTL system, it is necessary to apply stochastic time-varying nonlinear aerodynamic forces on rigid robot and flexible PTL. Assuming the robot/PTL system is in a streamlined wind field, the relative velocity between the object and the wind needs to be calculated first, and then the drag and lift forces on the object are obtained. The direction of these forces depends on the direction of the relative velocity, as shown in the upper right of Fig. 1. The aerodynamic drag and lift force can be respectively written in the following form [27]:

\[
\begin{bmatrix}
F_d \\
F_l
\end{bmatrix} = \frac{1}{2} \rho \bar{A} |\bar{v}_r|^2 \begin{bmatrix}
C_d(\sigma) \cdot \bar{v}_d \\
C_l(\sigma) \cdot \bar{v}_l
\end{bmatrix}
\]  

(6)

where \( F_d \) and \( F_l \) represent the drag and lift force vectors of the robot/PTL system under the wind load in the ground coordinate system respectively; \( \rho \) is the fluid density of air; \( C_d \) and \( C_l \) are respectively the drag and lift coefficients per unit area related to the wind attack angle (collectively referred to as the aerodynamic coefficient, dimensionless). \( \bar{A} \) is the area of aerodynamic action; \( \bar{v}_r \) is the vector of the relative velocity between the object and the wind; \( \bar{v}_d \) and \( \bar{v}_l \) are unit vectors in the direction of drag and lift, respectively.

If the geometrical shape of the object is certain and the influence of wind speed is not considered under the breeze condition, the aerodynamic coefficient is a function of the wind attack angle. The flow field around the robot/PTL system is simulated in Computational Fluid Dynamics (CFD) software to calculate the aerodynamic coefficients of the robot and the PTL fittings at different wind attack angles. In order to accurately model the environment of reinforcement learning agents, it is necessary to accurately understand the aerodynamic forces acting on PTL, because they significantly affect the positioning accuracy for the end tool of the manipulator under the action of rigid and flexible coupling. In order to meet this requirement, the CFD model of PTL.
segment is established, and the aerodynamic coefficients of PTL under different wind attack angles are calculated by using the FSI method. The geometry of the robot/PTL system is simplified to facilitate modeling. The aerodynamic coefficients of PTL are calculated according to the CFD model, as shown in Fig. 3.

**Fig. 3** Calculation of the PTL aerodynamic coefficients based on the CFD model

Based on the previous research work [25, 28] and Equation (6), the fluid-rigid-flexible coupling dynamical equation of the robot/PTL system under wind load can be obtained:

\[
\begin{bmatrix}
Q_r
\end{bmatrix} + \begin{bmatrix}
Q_{dr} & Q_{tr}
\end{bmatrix} \begin{bmatrix}
1
\end{bmatrix}
= \begin{bmatrix}
\hat{M}_r(\psi) & 0
\end{bmatrix} \begin{bmatrix}
\dot{\psi}
\end{bmatrix} + \begin{bmatrix}
S_r(\psi, \dot{\psi})\dot{\psi} + G_r(\psi) + D_r(\dot{\psi})
\end{bmatrix}
\begin{bmatrix}
C_p^T \lambda_p
\end{bmatrix}
\]

in which:

\[
\begin{bmatrix}
Q_{dr} & Q_{tr}
\end{bmatrix} = \begin{bmatrix}
\sum_{i=1}^{N_r} j_{ri}^T F_{dr}^i & \sum_{i=1}^{N_r} j_{ri}^T F_{tr}^i
\end{bmatrix}
\begin{bmatrix}
\sum_{i=1}^{N_p} j_{pi}^T F_{dp}^i & \sum_{i=1}^{N_p} j_{pi}^T F_{lp}^i
\end{bmatrix}
\]

where, for the robot, \(Q_r\) is the generalized force vector of the robot except wind load, \(Q_r = F_r + J_{a}^T F_{a}\), in which \(F_r\) is the control torque directly acting on each joint of the robot, \(F_r = [0_{1\times 6} u^T]^T\); \(Q_{dr}, Q_{tr}\) are respectively the generalized drag and lift force vector of the robot under wind load in the robot coordinate system; \(N_r\) is the number of rigid-body mesh nodes of the robot; The Jacobian matrix of the robot's position field can be expressed as \(J_r = dr_r/\partial \psi; J_a\) is the Jacobian matrix of the operation or walking manipulator end-effector; \(F_\alpha\) is the force and moment exerted on it by PTL (including fittings); \(\hat{M}_r\) is the inertia matrix of the robot; \(\dot{\psi}\) is defined as the generalized coordinate of the robot, including the position vector \(r_b\), Euler angle vector \(e_b\) of the robot body and the position vector \(q_a\) of manipulator joint in the ground coordinate system, \(\psi = [r_b^T e_b^T q_a^T]^T; S_r\) is the stiffness matrix. \(S_r(\psi, \dot{\psi})\dot{\psi}\) contains the force and moment caused by centrifugal and Coriolis acceleration. \(G_r\) is the gyro vector; \(D_r\) is the damping vector.

For PTL, \(Q_p\) is the elastic force vector of PTL; \(Q_{dp}\) and \(Q_{lp}\) are respectively the generalized drag and lift
vectors in the ground coordinate system; \( N_p \) is the number of flexible-body mesh nodes of PTL; the Jacobian matrix of the position field of PTL can be expressed as \( J_p^i = \frac{d\mathbf{r}_p^i}{d\mathbf{q}_p} = S_p \); \( M_p \) is the total mass matrix of PTL; \( \mathbf{q}_p \) represents the generalized coordinate vector of PTL nodes; \( S_p \) is the shape function matrix associated with the aerodynamic ANCF element; \( C_p \) is the Jacobian matrix of PTL for the constraint; \( \lambda_p \) represents the Lagrange multiplier vector corresponding to the constraint.

3 Experiment Design

3.1 Operation Task Description
This paper mainly studies the operation problem of the dampers on PTL. PTL has four conductors, and the conductors are fixed by spacers at certain distances, which are arranged in square corners from the cross-section. The robot will initialize and operate in any non-collision configuration between the four cables of PTL. The robot will not collide with the cables, dampers and bolts mounted on them during the operation. The objective of the operation is to tighten the damper bolts on the random cable. The damper models include FD-5 and FDZ-5F. The autonomous operating robot needs to learn how to complete a target task as quickly as possible without colliding with itself or obstacles. In order to simulate the real operation situation, the wind disturbance is presented on the four cables and randomly initialized at the beginning of the task. A continuous stochastic flow of air means the robot operates in an unstable environment. The airflow model is established by finite element software, and several typical operation conditions are calculated and randomly assigned to the robot simulation environment. The robot dynamics described in section 2.2.4 is approximately a series manipulator model coupled with cables and wind. The running frequency of the simulation system is 10Hz.

3.2 Action Space
The robot motion is controlled jointly by multiple controllers to actuators, and each actuator is relatively independent. The tasks of the PLMR can be divided into three operation policies: global obstacle avoidance, local approach and local tightening. In order to improve the learning performance, the output actions of the master policy are normalized. Therefore, the action space of the master policy is defined as the following ternary value:

\[
A_m = \begin{cases} 
-1 & \text{Global obstacle avoidance sub-policy} \\
0 & \text{Local approach sub-policy} \\
1 & \text{Local tightening sub-policy}
\end{cases}
\] (9)

For operation policy, previous reinforcement learning studies on decision-making of autonomous operating robot usually do not consider the physical limitations of actuators, and often take actuator speed as the output of action policy. In order to prevent the sudden change of the actuator speed, the output torque of the actuator is constrained, namely \( |u^{(i)}| \leq u^{(i)}_{\text{limit}} \). Torque constraint \( u_{\text{limit}} \) refers to the maximum continuous torque of the actuator, and the superscript represents the \( i \)th element of the torque vector.

3.3 State Representation and Reward Function
Four states are proposed to represent the operation scenario and task: the state associated with the robot, the state associated with the cable, the state associated with the damper, and the state associated with the bolt. A total of
17 indicators are presented to represent operation state, and a complete list of these states is shown in Table 1. Each state is added to a stochastic wind disturbance before being observed by the robot learner. Some states are selected from the list as input to the relevant value networks and corresponding policies. The state space and reward function for each network are varied and depend on the operation policy. In order to better set the reward function, four errors of the evaluation state are given, namely the position error $\varepsilon_{wp}$ for the wrist joint, the position error $\varepsilon_{tp}$ for the end tool, the posture error $\varepsilon_{ta}$ for the end tool, and the coaxiality error $\varepsilon_{tc}$ for the end tool. The formula is as follows:

$$\varepsilon_{wp} = |r_w^* - r_w| \quad \varepsilon_{tp} = |r_t^* - r_t|$$

$$\varepsilon_{ta} = |e_t^* - e_t| \quad \varepsilon_{tc} = \max(d_{tl}, d_{tr})$$

in which:

$$d_{t\#} = \frac{|(r_{tl}^* - r_{tr}^*) \times (r_{t\#}^* - r_{tt}^*)|}{|r_{t\#}^* - r_{tt}^*|}$$

$r_w$ and $r_w^*$ are respectively the position of coordinate system for the wrist joint and its expected value; $r_t$ and $r_t^*$ are respectively the position of coordinate system for the end tool and its expected value; $e_t$ and $e_t^*$ are respectively the posture of coordinate system for the end tool and its expected value; $d_{t\#}$ is the distance between the end point of the end tool and the center axis of the bolt, and the subscript $\# \in \{l, r\}$ represents the left and right ends of the end tool. Note that due to the flexibility of PTL, variations in $r_w^*$, $r_t^*$, and $e_t^*$ occur that can be solved by the system coupling dynamics model during robot operation. In order to normalize the error, the error metric function is defined as $M_\# = \text{metr}(\varepsilon_\#)$, and the subscript $\# \in \{wp, tp, ta, tc\}$ represents error type.

**Table 1.** State representation

| State type | Feature | Policy | Collision detection |
|------------|---------|--------|---------------------|
| Mobile platform & walking wheel | $P_m$ | $\pi^m_{\theta_m}(s)$ | $\bullet$ |
| | $V_m$ | $\pi^d_{\theta_m}(s)$ | $\bullet$ |
| | | $\pi^a_{\theta_m}(s)$ | $\bullet$ |
| | | $\pi^l_{\theta_m}(s)$ | $\bullet$ |
| | | $L_m$ | $\bullet$ |
| | | $L_b$ | $\bullet$ |
| | | $L_s$ | $\bullet$ |
| | | $L_e$ | $\bullet$ |
| | | $L_w$ | $\bullet$ |
| | | $L_\#$ | $\bullet$ |
| Base | $P_b$ | $\bullet$ |
| | $V_b$ | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
| | | $\bullet$ |
In the table, $P$ denotes joint position; $V$ denotes joint velocity; $C$ denotes node coordinate; $L$ denotes geometric solid. The subscript $\# \in \{1, \cdots, N_p\}$ for PTL represents a mesh node number within a limited range.

### 3.3.1 Master Policy:
given the three sub-policy of the operation policy, the robot needs to consider all the states in Table 1 when choosing the low-level sub-policy. The master policy uses the reward function to provide a positive reward that is proportional to the weighted value for the error metric of the end tool pose and coaxiality as it approaches or moves away from the final goal. The master policy should impose a negative penalty representing the actuator’s energy consumption because the robot uses a battery with limited power. Of course, any operation that causes a collision will be penalized with a negative value of -200, accompanied by a stop signal. Different from the learning of operation policy, the master policy requires macroscopic consideration of the achievement of the final goal. Therefore, if the robot completes the final goal, it will receive a positive reward of +100. To sum up, the reward function of the master policy can be expressed as:

$$ R_m = \begin{cases} 
100 & \text{Final goal completion} \\
 a_m \cdot M_{tp} + b_m \cdot M_{tc} - c_m u^T u & \text{Final goal approach} \\
 a_m \cdot M_{tp} + b_m \cdot M_{ta} - c_m u^T u & \text{Exploration and trial} \\
 -200 & \text{Collision} 
\end{cases} $$

where $a_m$ and $b_m$ are the weight coefficients of the error metrics of the end tool position and posture in the reward function of the master policy; $c_m$ is the weight coefficient of actuator energy consumption.

### 3.3.2 Global Obstacle Avoidance:
for the global obstacle avoidance sub-policy, the target task is to enable the manipulator to reach any target point within the operating range without collision in the environment with obstacles. This policy requires the design state to reflect the collision detection between the robot itself and its environment. At this point, the robot only needs to pay attention to the configuration of itself, the surrounding cables and dampers, and assume that the bolt is already enveloped in the damper. Intuitively speaking, each link of the manipulator should consider the self-collision and the collision with the cables and dampers. The relationship between collision detection is shown in Table 1. In order to evaluate the distance between the manipulator end-effector and the target point, the policy sets two indicators: the position error $\varepsilon_{wp}$ and $\varepsilon_{tp}$ for the wrist joint and the end tool respectively, defined as the distance between the target point and the coordinate system origin of the wrist and the end tool. If a collision occurs, a negative penalty of -200 is applied. If the end tool reaches the target position, that is, if the position error is less than a certain threshold, a positive reward of +100 is provided. If sparse returns are used, the training will be seriously slowed down due to falling into local minima, and the global optimal solution cannot be obtained. Specifically, the manipulator can avoid obstacles but cannot reach the target position. In order to prevent this situation, the reward function of the global obstacle avoidance sub-policy can be expressed as segmented dense reward considering the positioning effect of the wrist:

$$ R_g = \begin{cases} 
100 & \text{Sub−goal completion} \\
 a_g \cdot M_{wp} + b_g \cdot M_{tp} & \text{Sub−goal approach} \\
 (a_g \cdot M_{wp} + b_g \cdot M_{tp})/2 & \text{Exploration and trial} \\
 -200 & \text{Collision} 
\end{cases} $$

where $a_g$ and $b_g$ are the weight coefficients of the error metrics of the wrist joint and end tool position in the
3.3.3 Local Approach: for the local approach sub-policy, the robot needs to consider subtler collision detection with cables, dampers and bolts at the same time when making decisions, and finally the end tool needs to reach the specified target pose. Therefore, the three are subdivided into smaller geometric units. The profile of the aluminum strand on the surface of the conductor, the external profile of the damper and the bolt are carefully considered. In order to connect the subsequent local tightening policy, the positioning accuracy for the end tool is strictly required, that is, the top pose of the end tool and bolt head should be consistent (the error is less than a certain threshold value). To evaluate this error, two indicators are introduced: the position error $e_{tp}$ and posture error $e_{ta}$ for the end tool. The position error, as previously mentioned, reflects the distance relation and the posture error is defined as the modulus length of the Euler angular difference vector. The reward function of this policy can be set in the form of dense reward, and the smaller the error, the bigger the reward. If a collision occurs, a negative penalty of -200 is applied. A positive reward of +100 is given when the end tool reaches the specified pose, that is, when the weight of the two errors is less than a certain threshold. Specifically, it can be expressed as:

$$R_a = \begin{cases} 100 & \text{Sub-goal completion} \\ a_a \cdot M_{tp} + b_a \cdot M_{ta} & \text{Exploration and trial} \\ -200 & \text{Collision} \end{cases}$$

(14)

where $a_a$ and $b_a$ are the weight coefficients of the error metrics of the end tool position and posture in the reward function of the local approach sub-policy.

3.3.4 Local Tightening: for the local tightening policy, the robot needs to pay attention to the continuous feeding in the direction coaxial to the central axis of the bolt until tightening while meeting the collision conditions of the local approach sub-policy. In order to evaluate the coaxiality error $\epsilon_{tc}$ between the end tool and the bolt, Equation (11) calculates the distance between the key points on the end tool and the straight line where the bolt is located, and takes the maximum of the two as the evaluation value. At the same time, considering the feed of the end tool along the central axis of the bolt, the position error $e_{tp}$ for the end tool is used to characterize it. The reward function of this policy can be set in the form of dense reward, and the smaller the error, the bigger the reward. If a collision occurs, a negative penalty of -200 is applied. When the bolt is screwed to the bottom, that is, the position error is less than a certain threshold, a positive reward of +100 is given. Specifically, it can be expressed as:

$$R_t = \begin{cases} 100 & \text{Sub-goal completion} \\ a_t \cdot M_{tp} + b_t \cdot M_{tc} & \text{Exploration and trial} \\ -200 & \text{Collision} \end{cases}$$

(15)

where $a_t$ and $b_t$ are the weight coefficients of the error metrics of the end tool position and coaxiality in the reward function of the local tightening sub-policy.

4 Experiment Results
The HRL algorithm uses a discount of $\gamma = 0.99$ for all policies and updates the target network with a smoothing factor of $10^{-3}$. For each policy, the training process ran 16 copies of the asynchronous parallel agents and
environment at a fixed frequency of 10 Hz. In each round of training, each agent has a sampling period of 1 second and a maximum duration of 100 seconds. For all the learning processes, the value function and the action policy learn the neural network parameters at a learning rate of $10^{-3}$ and $10^{-4}$, respectively. Their gradient threshold is 1, which limits the variation of network parameters in the training iteration. The l2norm method is used to deal with the gradient value exceeding the gradient threshold, which reduces the learnable parameters whose $L_2$ modulus of the gradient is longer than the gradient threshold, so that the $L_2$ modulus length is equal to the gradient threshold. The $L_2$ regularization (weight decay) factor is set as $10^{-4}$, which alleviated the over-fitting of the neural network. The standard deviation of the motion noise model and its decay rate and mean attraction constant are 0.2, $10^{-4}$ and 0.15, respectively. For the agents of each policy, no matter the master policy or the operation policy, the learning speed, training stability and policy performance of multi-agent under APCE are much higher than that of single agent.

### 4.1 Training Effect

Fig. 4(a) is comparison of training curves for the master policy. It can be seen that since the algorithm architecture of the master policy is relatively simple, the well-trained and fast-running operation policy is only needed to be invoked when the action is executed, which tends to be stable after about 500 rounds. The fluctuation is caused by the random initialization and free exploration of the autonomous operating robot. Compared with DQN algorithm, the master policy using DDQN algorithm reduces the influence of overestimation and obtains better training performance. Fig. 4(b) shows the comparison of training curves of the global obstacle avoidance policy. It is easy to see that the imitation learning effect of the random forest composed of decision trees as the expert algorithm is far better than the original DDPG algorithm which requires a long time of exploration, trial and error. Since the random forest already contains a lot of empirical data from the expert algorithms, the global obstacle avoidance policy only needs to adapt to the changes caused by wind disturbance on the basis of the original, without the need to exploration, trial and error again.

![Fig. 4](image.png)

**Fig. 4** Training curve of the master and operation policy: (a) Master policy; (b) Global obstacle avoidance sub-policy; (c) Local approach and tightening sub-policy.

As shown in Fig. 4(c), compared with the local tightening policy, the reward obtained by the local approach policy reached a higher and more stable level. There are two main reasons for these apparent differences. First of all, the task of local tightening is difficult, and it needs to meet stricter coaxiality error requirements, which is more complicated than the operation of the end tool using the local approach sub-policy, so it needs more
exploration. Secondly, the random initialization of the autonomous operating robot will inevitably fall into the local optimum temporarily caused by coaxiality error in the process of operation. But with the increase of the episode number of training, the robot will explore the area beyond the local optimal through the noise of the action policy, and get better returns. The fluctuation of the two curves near the peak value after convergence is caused by random initialization.

4.2 Simulation Verification

The performance of the learned master and operation policies is evaluated through a series of simulations. The simulation platform used in this study includes the model module of the robot/PTL system and wind load, which is built in MATLAB. At the beginning of each episode, the stationary autonomous operating robot is initialized on PTL with a random, collision-free configuration and pose. The target task is randomly set to tighten the damper bolt of any cable in PTL. In order to simulate real wind disturbances and verify the robustness of the policies against wind disturbances, continuous wind loads at random speeds and attack angles are set throughout the operating space. The robot/PTL system is divided into a rigid-body and a flexible-body mesh for wind loading. During the training process, the simulation environment adds continuous wind loads at random speeds and attack angles to each mesh node of the system. According to the wind speed of the city, the wind load considered is no more than third-order wind (breeze) and the range of attack angle is within ±180°.

Fig. 5 shows the decision-making process of the master policy for the operation policy under different wind load conditions for a specific target task. The autonomous operating robot enters the global obstacle avoidance policy at the beginning of the simulation (approximately from 0 seconds to 8 seconds). When the robot arrives near the target, it switches to a local approach policy (from about 8 seconds to 9 seconds). It switches to a local
tightly policy (approximately from 9 seconds to 10 seconds) finally when the tightening position is reached. This suggests that our approach can learn a basic value network to evaluate operation states. In order to study the sensitivity of each operation policy to wind disturbance, these policies evaluate the joint positions of the manipulator under different wind loads based on the determined master policy, as shown in Fig. 5(b). It can be seen that when the wind load does not exceed the specified range, the policy is less affected by noise. Interestingly, depending on the choice of the master policy, the robot tended to adopt the global obstacle avoidance policy at the beginning without explicit instruction. This is because the reward is the highest relative to the other local policies. When approaching the targets of the local policies, the control accuracy for the end tool of the manipulator by the neural network is limited under the action of stochastic wind disturbance, so it is easy to cause small positioning error. To some extent, this shows that the master policy, together with the three operation policies, can achieve smooth and safe decisions on PTL, and that the trained policy is capable of being translated into practical applications.

4.3 Hierarchical and Single-policy Architecture

One question that is often encountered with a hierarchical architecture is whether the robot decision problem can be addressed by using a non-hierarchical single-policy architecture. Previous papers have shown the learning effects of using single-policy reinforcement learning algorithms. Obviously, this is not a satisfactory performance. Therefore, this study divides the task into two difficulty levels, and tests the effect of using the hierarchical and single-policy architecture to make decisions under different difficulty levels. The single-policy architecture also uses state representation and reward function similar to those mentioned in Section 3.3 for training, and its network architecture is the same as that of global obstacle avoidance policy.

| Task type   | Difficulty | Architecture | Success rate | Collision rate | Path length | Completion time (s) |
|-------------|------------|--------------|--------------|---------------|-------------|-------------------|
| Damper FD-5 | Low        | Hierarchical | 97.5%±0.2%   | 1.3%±0.6%     | 2.3±0.5     | 58.7±5.7          |
|             |            | Single-policy| 95.9%±1.0%   | 1.8%±0.7%     | 2.6±0.9     | 60.5±7.6          |
|             | High       | Hierarchical | 80.1%±3.3%   | 14.9%±0.6%    | 3.2±0.8     | 81.8±6.6          |
|             |            | Single-policy| 74.5%±4.2%   | 20.5%±0.3%    | 3.7±1.0     | 87.4±7.4          |
| Damper FDZ-5F | Low      | Hierarchical | 88.2%±2.3%   | 5.6%±0.4%     | 2.4±0.6     | 79.5±3.9          |
|             |            | Single-policy| 86.4%±3.9%   | 6.9%±0.9%     | 2.6±0.3     | 82.2±6.8          |
|             | High       | Hierarchical | 72.1%±2.8%   | 22.6%±0.8%    | 3.5±0.7     | 86.6±4.2          |
|             |            | Single-policy| 65.3%±4.5%   | 28.1%±0.2%    | 4.3±1.2     | 93.1±7.1          |

As shown in Table 2, the study is surprised to find that at the lower difficulty levels, the single-policy architecture could be learned almost as well as the hierarchical architecture, with less significant differences; However, when it comes to high difficulty, the hierarchical architecture performs significantly better than the single-policy architecture. This indicates that the designed state representation and reward function are effective for training a good agent regardless of difficulty. As the difficulty of the problem increased, the single-policy architecture did not learn to use the nearest configuration path and the fastest running time to complete the target task, resulting in lower average operation speeds and longer operation times (about 18% slower than the
It is obvious that the local optimal solution is found with the single-policy architecture. This is because in the training process, when the robot operates locally, it often collides with the obstacles on PTL, so it thinks that the policy of choosing the farther configuration path or running at a slower speed is the best.

To sum up, on the one hand, due to the high state dimension, the single-policy architecture cannot learn the appropriate policy; On the other hand, modularization facilitates sub-policy substitution. For example, if the master policy needs to replace an operation policy, the parameters of other networks can still be retained, thus improving the portability of the policy and training efficiency. This is the reason why hierarchical architecture is chosen for the autonomous operating robot.

5 Conclusions
Aiming at the motion intelligence of robot intelligence, a hierarchical reinforcement learning architecture considering stochastic wind disturbance is proposed for the decision-making of the PLMR with autonomous operation. This architecture uses the prior information of the mechanism knowledge and empirical data to improve the safety and efficiency of the PLMR operation. In this architecture, the high-level policy selection and the low-level motion control at global and local levels are considered comprehensively under the condition of stochastic wind disturbance. Firstly, the task is decomposed into three sub-policies: global obstacle avoidance, local approach and local tightening, and each sub-policy is learned. Then, a master policy is learned to select the operation sub-policy in the current state. The dual deep Q network algorithm is used for the master policy, while the deep deterministic policy gradient algorithm is used for the operation policy. In order to improve the training efficiency, the global obstacle avoidance policy takes the random forest composed of dynamic environmental decision tree as the expert algorithm for imitation learning. The architecture is applied to a PTL maintenance scenario, the state function and reward function of each policy are designed, and all policies are trained in an asynchronous and parallel computing environment. It is proved that this architecture can realize stable and safe autonomous operating decision for the PLMR subjected to stochastic wind disturbance. In the future, researchers will test and improve the algorithm in more complex operating environments.

Acknowledgements
The authors thank the Special Robot Research Institute of Wuhan University at School of Power and Mechanical Engineering.

Funding
The authors received no financial support for the research, authorship, and/or publication of this article.

Conflicts of interest
The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.
Authors' Contributions

In this research, X.Z. and G.W. contributed to the conceptualization of this study; X.Z. contributed to the methodology, software, validation, and investigation of this study; X.Z. and G.W. contributed to the resources of this study; X.Z. and G.W. contributed to the writing—original draft preparation; X.Z. contributed to the writing—review and editing; X.Z. contributed to the visualization of this study; G.W. contributed to the supervision and project administration of this study.

References

1. Debenest, P., Guarnieri, M., Takita, K., Fukushima, E. F., Hirose, S., Tamura, K., Kimura, A., Kubokawa, H., Iwama, N., Shiga, F.: Expliner – robot for inspection of transmission lines. IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA, USA, May 19-23(2008)
2. Pouliot, N., Latulippe, P., Montambault, S., HQRI, Tremblay, S. Averna.: Reliable and intuitive teleoperation of LineScout: a mobile robot for live transmission line maintenance. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, USA, October 11-15(2009)
3. Lioutikov, R., Maeda, G., Veiga, F., Kersting, K., Peters, J.: Learning attribute grammars for movement primitive sequencing. The International Journal of Robotics Research. 2, 1-18(2019)
4. Antanas, L., Moreno, P., Neumann, M., De Figueiredo, R. P., Kersting K., Santos-Victor, J., De Raedt, L.: Semantic and geometric reasoning for robotic grasping: a probabilistic logic approach. Autonomous Robots. 43(6), 1393-1418(2019)
5. Muni, M. K., Parhi, D. R., Kumar, P. B., Rath, A. K.: Navigational analysis of multiple humanoids using a hybridized rule base-sugeno fuzzy controller. International Journal of Humanoid Robotics. 17(4), 2050017(2020)
6. Gordin, Z., Jovanovi, K.: Collision detection on industrial robots in repetitive tasks using modified dynamic time warping. Robotica. 38(10), 1717-1736(2019)
7. Yan, Y., Jiang, W., Zhang, A., Li, Q. M., Lei, Y. F.: Research on configuration design and operation effect evaluation for ultra high voltage (UHV) vertical insulator cleaning robot. Industrial Robot: the international journal of robotics research and application. 47(1), 90-101(2020)
8. Jiang, W., Peng, M. H., Yan, Y., Wu, G. P., Zhang, A., Yu, L. Q., Li, H. J.: Motion posture control for power cable maintenance robot in typical operation conditions. Industrial Robot: the international journal of robotics research and application. 46(5), 631-641(2019)
9. Li, H. J., Jiang, W., Zou, D. H., Yan, Y., Zhang, A., Chen, W.: Robust motion control for multi-split transmission line four-wheel driven mobile operation robot in extreme power environment. Industrial Robot: the international journal of robotics research and application. 47(2), 219-229(2020)
10. Xiang, G. F., Su, J. B.: Task-oriented deep reinforcement learning for robotic skill acquisition and control. IEEE transactions on cybernetics. 51(2), 1056-1069(2019)
11. Sasabuchi, K., Wake, N., Ikeuchi, K.: Task-oriented motion mapping on robots of various configuration using body role division. IEEE Robotics and Automation Letters. 6(2), 413-420(2020)
12. Zuo, G. , Zhao, Q. , Chen, K. , Li, J. , & Gong, D.: Off-policy adversarial imitation learning for robotic tasks with low-quality demonstrations. Applied Soft Computing. 97(B), 106795(2020)
13. Ahn, K. H., Song, J. B.: Image preprocessing-based generalization and transfer of learning for grasping in cluttered environments. International Journal of Control, Automation and Systems. 18(9), 2306-2314(2020)
14. Saha, T., Gupta, D., Saha, S., Bhattacharyya, P.: Towards integrated dialogue policy learning for multiple domains and intents using hierarchical deep reinforcement learning. Expert Systems with Applications. 162, 113650(2020)
15. Duan, J., Li, S. E., Guan, Y., Sun, Q., Cheng, B.: Hierarchical reinforcement learning for self-driving decision-making without
16. Li, T. G., Srinivasan, K., Meng, Q. H., Yuan, W. Z., Bohg, J.: Learning hierarchical control for robust in-hand manipulation. IEEE International Conference on Robotics and Automation (ICRA). Online. May 31-August 31(2020)
17. Pang, Z. J., Liu, R. Z., Meng, Z. Y., Zhang, Y., Yu, Y., Lu, T.: On Reinforcement Learning for Full-length Game of StarCraft. 33rd AAAI Conference on Artificial Intelligence. Honolulu, HI, USA. January 27-February 01(2019)
18. Yu, J. L., Su, Y. C., Liao, Y. F.: The path planning of mobile robot by neural networks and hierarchical reinforcement learning. Frontiers in Neurorobotics. 14, 63(2020)
19. Alhassan, A. B., Zhang, X., Shen, H., Jian, G., Xu, H., Hamza, K.: Investigation of aerodynamic stability of a lightweight dual-arm power transmission line inspection robot under the influence of wind. Mathematical Problems in Engineering. 2019, 2139462(2019)
20. Lu, X. B., Liu, Z. G., Jing, Z., Wang, H. R., Yang, S., Duan, F. C.: Prior-information-based finite-frequency control for active double pantograph in high-speed railway. IEEE Transactions on Vehicular Technology, 66(10), 8723-8733(2017)
21. Fu, X., Li, H. N.: Dynamic analysis of transmission tower-line system subjected to wind and rain loads. Journal of Wind Engineering and Industrial Aerodynamics, 157, 95-103(2016)
22. Pombo, J., Ambrosio, J., Pereira, M., Rauter, F., Collina, A., Facchinetti, A.: Influence of the aerodynamic forces on the pantograph–catenary system for high-speed trains. Vehicle System Dynamics. 47(11), 1327-1347(2009)
23. Zdziebko, P., Martowicz, A., Uhl, T.: An investigation on the active control strategy for a high-speed pantograph using co-simulations. Proceedings of the Institution of Mechanical Engineers Part I Journal of Systems and Control Engineering. 233(4), 370-383(2019)
24. Song, Y., Liu, Z., Ouyang, H., Wang, H., Lu, X.: Sliding mode control with pd sliding surface for high-speed railway pantograph-catenary contact force under strong stochastic wind field. Shock and Vibration. 2017, 4895321(2017)
25. Zheng, X. L., Wu, G. P.: Kinodynamic planning with reachability prediction for PTL maintenance robot. Proceedings of the Institution of Mechanical Engineers Part I Journal of Systems and Control Engineering. doi: 10.1177/0959651820987927(2021)
26. Jia, B., Pan, Z., Hu, Z., Pan, J., Manocha, D.: Cloth manipulation using random-forest-based imitation learning. IEEE Robotics and Automation Letters. 4(2), 2086-2093(2019)
27. Kulkarni, S., Pappalardo, C. M., Shabana, A. A.: Pantograph/Catenary Contact Formulations. Journal of Vibration and Acoustics-Transactions of the ASME. 139(1), 011010(2017)
28. Zheng, X. L., Wu, G. P., Jiang, W., Fan, F., Zhu, J. L.: Rigid–flexible coupling dynamics with contact estimator for robot/PTL system. Proceedings of the Institution of Mechanical Engineers Part K-Journal of Multi-Body Dynamics. 234(4), 635-649(2020)

Xiaoliang Zheng is currently pursuing his Ph.D degree within the Special Robot Research Institute of Wuhan University at School of Power and Mechanical Engineering, Wuhan University, Wuhan, China. He received his bachelor's and master's degrees from Wuhan University of Technology. His research interests include robotics and robot intelligence.

Gongping Wu is a professor and academic leader within the Special Robot Research Institute of Wuhan University at School of Power and Mechanical Engineering, Wuhan University, Wuhan, China. He received a master's degree from Huazhong University of Science and Technology and a doctor's degree from Wuhan University. His research interests include special robots and mechanical engineering.

-Ethics approval

Not applicable
- Consent to participate

Not applicable

- Consent for publication

Not applicable