Learning Manipulation Skills with Demonstrations for the Swing Process Control of Dredgers

Changyun Wei[a,b], Haonan Bai[a,b], Yi Wei[a,b], Ze Ji[c,*] and Zenghui Liu[a,b]

[a]College of Mechanical and Electrical Engineering, Hohai University, China
[b]Engineering Research Center of Dredging Technology of Ministry of Education, Hohai University, China
[c]School of Engineering, Cardiff University, Cardiff CF24 3AA, United Kingdom

ARTICLE INFO

Keywords:
Cutter Suction Dredger
Intelligent Control
Swing Process
Deep Reinforcement Learning

ABSTRACT

The Cutter Suction Dredger (CSD) is one of the key equipment dedicated to the construction and maintenance projects of harbours, ports and navigational channels. Among the dredging manipulations, the swing process is the most tedious and recurring work for human operators, which often leads to accidents because of carelessness or fatigue of the operators. This paper aims at producing a learning approach for the intelligent control of the swing process of a CSD so as to release human operators from such a boring and heavy task. To this end, the swing process control is formulated as a sequential decision making problem, and Deep Reinforcement Learning (DRL) is employed to design the learning approach based on deterministic policy gradient. The novel feature of the proposed approach is that the manipulation skills are obtained via trial-and-error interactions with a predicting network constructed by human demonstration data. In our approach, human demonstrations can provide a channel to predict state transitions, and also can regulate the exploration procedure for the learning agent. In addition, we carry out empirical studies to investigate how to treat the demonstration data with regard to self exploration, and the experimental results show that the proposed approach provides an effective means of controlling the swing process of CSDs.

1. Introduction

The Cutter Suction Dredger (CSD) is a special type of vessels for the construction and maintenance projects of harbours, ports and navigational channels, and it can excavate nearly all kinds of soil (i.e., sand, clay or rock) on the river/sea bed (Tang et al., 2009). The outstanding dredging ability of the CSD depends on a rotating cutter head, mounted in front of the ladder (see Figure 1), which can cut hard soil or rock into fragments. Afterwards, the dredged materials will be sucked up by the dredge pumps from an inlet underneath the cutter head, and then transported to a disposal area through a pipeline.

Currently, all CSDs are manipulated by trained human operators, and their work usually follows a three 8-hour (or four 6-hour) shift rotation system. Among the dredging manipulations, the swing process control is the most tedious and recurring work for the operators, which often leads to accidents due to carelessness or fatigue, especially in a night shift. Thus, this work seeks to produce a learning approach for the swing process control so as to release the operators from such a heavy workload. Of course, it is a challenge to control the swing process in complex ocean environments, since many factors, such as hydrology, meteorology, soil and underwater sundries, are uncertain and dynamic for dredging manipulations (Bai et al., 2019). Even for a trained operator, it is hard to accurately predict the changes of external environments so as to make quick and appropriate responses (Wei et al., 2019). Thus, it is intractable to construct a precise model that can handle all unforeseen situations in dredging various types of sand, clay or rock.

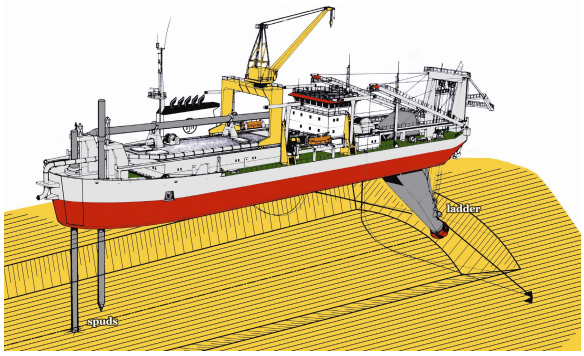

Figure 1: Outline and the swing process of a cutter suction dredger.

Learning is an effective methodology for the design of adaptive methods that can satisfy state constraints in nonlinear systems (Pan and Yu, 2015; Liu et al., 2016). Among the learning paradigms, Reinforcement Learning (RL) has the advantage of directly obtaining the optimal control policies, without knowing the system dynamics (Mnih et al., 2015; Lewis et al., 2012). Recent advances in RL have shown great successes in a variety of domains, such as games (Silver et al., 2018), robots (Gu et al., 2017; Xu et al., 2018), unmanned vehicles (Zhang et al., 2017), smart manufacturing (Lin et al., 2019) and smart microgrids (Wang et al., 2016). However, most RL approaches mainly focus on how

*This document is the results of the research project funded by the National Natural Science Foundation of China under Grant 61703138, the International Science and Technology Cooperation Research Funds of Changzhou under Grant CZ20210027, and by the Fundamental Research Funds for the Central Universities under Grant B200202224 and B200204036.
*Corresponding author.
ORCID(s):

to learn the optimal control policies from a virtual environment, where the learning agent can easily make trial-and-error interactions.

Thus, it still remains difficult to apply RL algorithms to real-world industrial applications, because an accurate virtual environment is not ready-made. It is also inefficient and dangerous for an agent to learn from scratch in a physical environment, since the optimal control policy can only be obtained after millions of learning steps with very poor performance. Such a case can be resolved if a perfect simulator or a virtual environment is available. However, it is intractable for many practical applications to provide an acceptable simulator in advance. In this case, the control agent must be able to perform well at the start of learning, taking account of the direct consequences of actions. To this end, the agent can make use of historical data performed by human or other controllers to regulate its initial behaviour, with the aim of performing well at the beginning of learning. Afterwards, the agent can continuously improve its behaviour based on online self-generated actions, which offers the possibility of applying RL algorithms to real industrial applications.

This work aims at developing an intelligent agent that can learn the optimal control policy for manipulating the swing process of a CSD. To this end, we employ the DRL framework to design a novel actor-critic approach based on deterministic policy gradients. In order to regulate the exploration behaviour of the learning agent, our approach incorporates human demonstrations to speed up the learning procedure. Then, the agent's behaviour can also be continuously improved by self-generated actions. The main contributions of this paper are summarized as follows.

1. To our best knowledge, we are the first to present a DRL approach for the swing process control of a CSD in continuous action spaces.
2. The learning agent can directly interact with a predicting network trained by operational data from human experts, instead of a virtual environment, to learn the optimal control policies.
3. In order to speed up the learning procedure, human demonstrations are also used to regulate the policy exploration, which can help the agent perform reasonably at the start of learning, and then continue improving its behaviour by self-generated data.
4. Simulation results show that the proposed approach can imitate the operational behaviour of human experts. It opens up the possibility of applying DRL to many practical industrial problems where demonstration data is accessible but accurate simulators do not exists.

The paper is organized as follows. We discuss the related work in Section 2, and formulate the swing process control as a sequential decision making problem in Section 3. The proposed approach is described in Section 4, and Section 5 discusses the simulation results. Finally, we conclude this work and discuss future plans in Section 6.

2. Related Work

In this section, we will discuss the related work with regard to the intelligent control of a CSD and the state-of-the-art RL methods concerning the optimal control in uncertain and dynamic environments.

2.1. Intelligent Control of a CSD

Designing an intelligent CSD has always been of interest of dredging industrials. IHC Holland, the leading dredger builders in the world, claims that an automatic cutter controller has been integrated into the centralized control system of a CSD built by them. However, we cannot find any scientific research or report on the actual performance of the control system. According to the feedback on its performance from end users, there is still a big gap between the automatic control and manual operation.

As mentioned previously, it is a challenge to design an intelligent control system of a CSD, due to many uncertain factors. Early studies have presented an expert system (Tang et al., 2009, 2008), with the aim of maximizing the productivity under safety constraints. This system integrates specific knowledge of several dynamics models of a CSD, such as diesel engines, pumps and pipeline transportations. Thus, it cannot be easily migrate to the control of other CSDs, and the model-based control method cannot adapt well to the changes of dynamic ocean environments. The work (Bai et al., 2019) investigates four machine learning algorithms to predict the productivity of a CSD, but it does not discuss how to control a CSD in an intelligent manner. Recently, RL has also been employed to study the swing process control problem in (Wei et al., 2019), but it only consider the simple case in discrete state and action spaces. Notably, such a method suffers from the curse of dimensionality. Moreover, the discretization of the action space can, in particular, throw away some essential information about the structure of the action domain. For the swing process control, the action needs finer grained adjustments in response to dynamic changes.

2.2. Reinforcement Leaning for Optimal Policy

Recent advances in RL often take games as a benchmark to evaluate an algorithm's performance, as the progress of a game can be easily measured repeatedly. Moreover, the goal state of a game can be defined clearly, such as Go (Silver et al., 2017), Atari (Oh et al., 2015) and Starchraft (Shao et al., 2018). In games, the agent usually has to interact with a virtual environment via trial-and-error, instead of directly interacting with a physical environment. Comparatively, in many real-world applications, it is hard to specify a good reward function in practice (Quillen et al., 2018; Yahya et al., 2017), and a fair amount of domain knowledge is required to be considered when designing an appropriate reward function. Moreover, the desired goal state of practical control problems is usually unclear and cannot be predefined by a scalar value. In addition, for the sake of repeatability and efficiency, the RL agent has to interact with an affordable environment or an accurate simulator which does not exist

in many real-world industrial applications.

RL is also related to the control theory of optimal control and adaptive control. To control nonlinear systems, optimal controllers require complete knowledge about the system dynamics to solve Hamilton-Jacobi-Bellman equations (Lewis et al., 2012), whereas adaptive controllers can use online data to control unknown systems (Chan et al., 2014). RL techniques are used to design adaptive controllers without having any prior knowledge about the system dynamics (Liu et al., 2014; Cui et al., 2017).

For the swing process control studied in this work, we also face the problem of lacking suitable reward functions. Even an expert operator does not have an explicit understanding of the objective or the goal state of this problem, so we need domain-dependent knowledge to design such a reward function. Moreover, we do not have a virtual environment or a simulator that can allow the agent to learn from scratch. Thus, we need to consider how to utilize the historical data generated by human operators to construct a predicting network that can work as the state transition function for the learning agent.

2.3. Learning from Demonstration

RL algorithms typically require a huge amount of training runs to obtain good results, which is reasonable for virtual simulated environments. For real-world applications, it is intractable and expensive to allow the agent to learn in the real environments (Hester et al., 2018). Learning from demonstrations is often treated as a supervised learning problem, in which the control policies are obtained by regressing expert's actions (Brys et al., 2015). Imitation learning focuses on how to match the performance of demonstrations, e.g., DAGGER (Ross et al., 2011) and its extension (Sun et al., 2017) require that an expert has to be available so as to provide feedback or a value function to the agent. Most importantly, such an imitation learning cannot learn to improve upon the expert. Usually, most real applications can produce historical data operated by human or other controllers, and such demonstrations can be used to pre-train the agent at the start of learning. In this work, human demonstrations will also be applied to regulate the policy exploration, which can speed up the learning procedure. Afterwards, the agent can continuously improve its behaviour by self-generated data.

3. Problem Statement

The CSD is an efficient tool to excavate nearly all kinds of soils, and the swing process of a CSD is depicted in Figure 2. Two spud poles are essential for the control of the swing process. The main spud pole, mounted on a movable spud carriage, moves lengthwise along the vessel, while the auxiliary spud pole is set out of the centerline, usually on the starboard side of the stern of the pontoon. The auxiliary spud pole is used to keep the CSD in position when the working spud pole is raised, and the spud carrier is move back to its initial position. The cutter head is used to cut hard soil or rock by rotating around the main spud, producing an arc trajectory. Such a movement is called swing, and the associated control action is to adjust the swing speed.

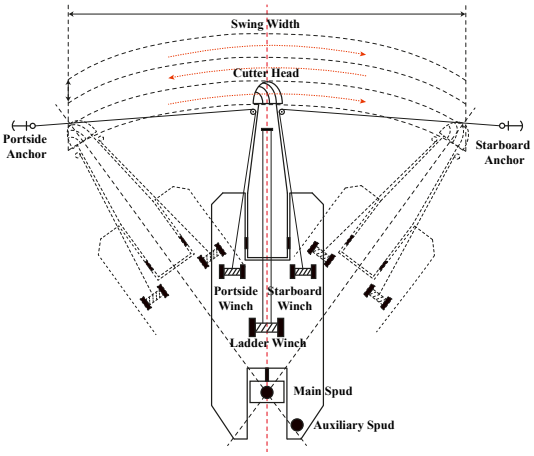

Figure 2: The swing process of a CSD.

For the swing process control, an agent has to determine the swing speed to adjust the amount of materials to be cut. The swing movement around the main spud is achieved by slacking (or pulling) the cable of the starboard anchor (or the portside anchor). Deck winches are connected to those anchors by cables, so if the deck winches pull or slack the cables, the cutter head will produce an arc trajectory around the working apud pole. Thus, we can conclude that the swing speed control is achieved by manipulating the deck winches.

In swing control, the agent should ensure that the production rate of dredged materials must be maintained in an appropriate range. Otherwise, the slurry transportation pipeline can be blocked if the slurry density is too high. Usually, the CSD has to use a nuclear-based gamma densitometer to measure the slurry density of the pipeline. Because the flow state inside the stern pipeline is relatively stable, the densitometer has to be installed at the stern of a CSD. As a consequent, the measured slurry density has a time lag with respect to the swing speed control, as depicted in Figure 3.

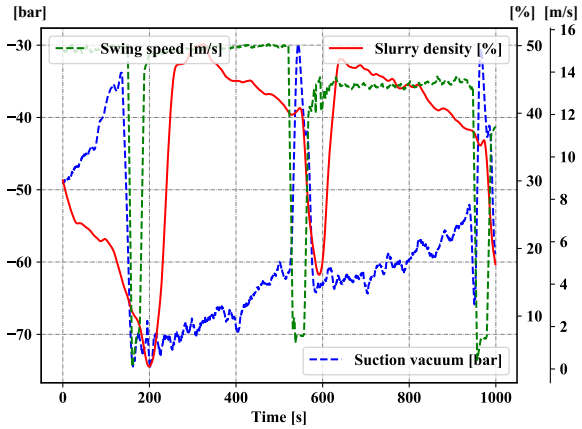

Figure 3: The measured slurry density has a time lag, but we can still infer the performance of the swing operation by other signals, i.e., the suction vacuum.

As mentioned before, the swing speed determines the extent to which the soil is cut by the cutter head, we still can infer the real-time performance of the swing speed from other correlated signals, e.g., the suction vacuum (see Figure 3). In other words, the immediate responses of the changes of the swing speed can be evaluated by those correlated signals, and we still need to look at the resulting slurry density in the long run. All of those measurements will constitute the state space for decision making, which will be discussed later. With regard to an intelligent control, we seek to find the optimal policy for the swing speed control so as to stabilize the slurry density in complex and dynamic situations of the pipeline.

3.1. Reinforcement Learning Model

The swing process control can be considered as a sequential decision making problem, formulated by Markov Decision Process (MDP). Without loss of generality, a MDP is defined by a tuple $\langle S, A, R, P, \gamma \rangle$, where S and A indicate the state space and the action space, R and P represent the reward function and the transition function, and γ denotes the discount factor. At each time-step t, the learning agent has to select an action $a_t \in A$ in state $s_t \in S$ to perform, which will lead to a new state s_{t+1} according to a probability distribution $P(s_{t+1}|s_t, a_t)$, and the agent will receive an immediate reward r_t from the environment. The return from a specific state is defined as the sum of discounted reward

$$R(s_t, a_t) = \sum_{i=t}^{T} \gamma^{(i-t)} r_t, \tag{1}$$

where $\gamma \in [0, 1]$ is the discounting factor. A control policy π maps states to actions, i.e., specifying what action the agent should take in each state. Thus, the goal of the learning agent is to find the optimal control policy that can maximize the expected total future discounted reward, which is defined by an action value function

$$Q^{\pi}(s_t, a_t) = \mathbb{E}^{\pi}[R(s_t, a_t)|s_t, a_t], \tag{2}$$

where $\mathbb{E}^{\pi}$ describes the expected return when following the policy π after performing action a_t in state s_t. For a continuous control problem, we can use neural networks to approximate the action value function, and thus maximizing the expected return from the start distribution can be formulated as to maximize a mean value

$$J(\theta) = \mathbb{E}_{s\sim\mu}[Q^{\pi(.|\theta)}(s_t, \pi(s_t|\theta)], \tag{3}$$

where the policy π is parameterised by θ, and μ denotes the initial state distribution. To maximize the mean value, the parameter θ can be updated by a gradient approach as follows:

$$\theta \rightarrow \theta + \alpha \nabla_{\theta} J(\theta). \tag{4}$$

Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) is an actor-critic algorithm that consists of an actor network $\pi(.|\theta^{\pi})$ and a critic network $Q(.|\theta^{Q})$. The actor network outputs the action to perform based on $a = \pi(s|\theta^{\pi}) + \mathcal{N}$, where $\mathcal{N}$ is used to add random noise for action exploration. Performing actions will produce new transitions $(s_t, a_t, r_t = R(s_t, a_t), s_{t+1} \sim P(.|s_t, a_t))$. All the new transitions will be added into a replay buffer B. The critic network can be updated by minimizing the following loss:

$$L(\theta^{Q}) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1})\sim D}[Q(s_t, a_t|\theta^{Q}) - y_t]^2, \tag{5}$$

where

$$y_t = r_t + \gamma Q'(s_{t+1}, \pi'(s_{t+1}|\theta^{\pi'})|\theta^{Q'}). \tag{6}$$

Here D is the distribution over transitions, and $\pi'(.|\theta^{\pi'})$ and $Q'(.|\theta^{Q'})$ represents the target networks of $\pi(.|\theta^{\pi})$ and $Q(.|\theta^{Q})$, respectively. The actor network can be updated by a gradient step,

$$\nabla_{\theta^{\pi}} J(\theta^{\pi}) \approx \mathbb{E}_{(s_t, a_t)\sim D}[\nabla_a Q(s_t, \pi(s_t|\theta^{Q}) \nabla_{\theta^{\pi}} \pi(s_t|\theta^{\pi})]. \tag{7}$$

We can see that the DDPG algorithm is off-policy, consisting of two neural networks with the same structure but different update frequencies. However, for the swing process control of a CSD, it is intractable to directly apply DDPG to find the optimal control policy because we do not have an accurate simulator for the agent to learn from scratch. Our approach is modified based on DDPG but has to consider the issue of lacking an accurate simulator or a virtual environment for the agent to interact with, as well as the problem of regulating the exploration policy at the start of learning.

4. Learning Approach

To obtain the optimal control policy for the swing process via learning, although an accurate simulator or a virtual environment is not ready-made, we still have access to the historical data operated by experienced humans. The demonstration data can provide us with the possibility of obtaining dynamics of dredging environment, and also let the agent learn as much as possible from human demonstrations before making trial-and-error interactions. The general ideal of our approach is depicted in Figure 4.

As we have access to the human manipulation data, we can use the collected data represented by the MDP model to form the demonstration fragments. Small batches of the demonstration fragments can be sampled according to the sample strategy discussed in Section 4.4, along with the samples from the experience replay buffer. This means that human demonstrations can be considered a set of pre-collected experiences for initializing the experience replay buffer. Thus, the agent does not have to learn from scratch when interacting with the external environment in the beginning. Moreover, we can also find that the human manipulation data can be used to construct a neural network that predicts the state transition, which will be detailed in Section 4.2. The predicting neural network works as an interactive environment that reflects the dynamics of the swing process. To be specific, when the agent chooses an action a_t to perform at state s_t, the predicting neural network can output the coming state s_{t+1} and the environmental reward r_t.

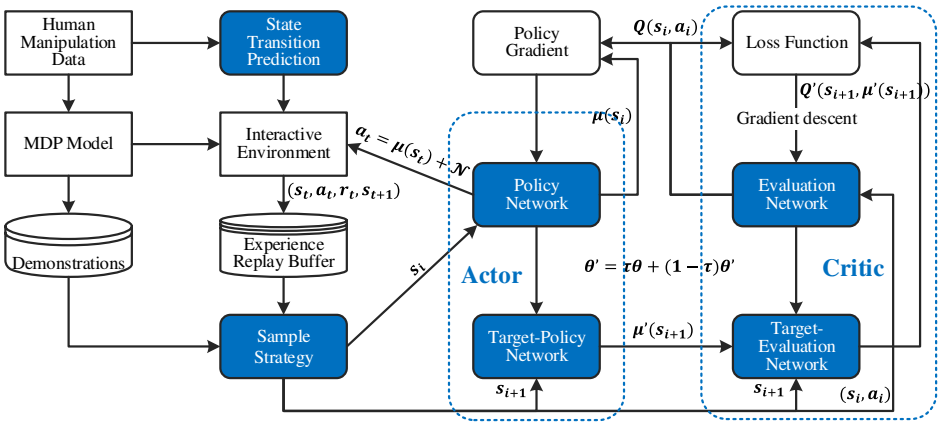

Figure 4: The general framework of our proposed approach.

The right part of Figure 4 depicts the learning mechanism, where the actor networks output the action to perform, and the critic networks will be updated by minimizing the loss function. It should be noted that a copy of the actor and critic networks are used to calculate the target values, and θ' of the target networks can be updated by tracking learned networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$. Having the target networks can ensure that the critic can be trained consistently without divergence.

In addition, in order to speed up the learning period, we require the agent to learn as much as possible from human demonstrations before interacting with the predicting network. The agent determines whether to learn from the demonstrations or the replay buffer based on the sampling priority mechanism. In this way, the human demonstrations can regulate initial policy and decrease the randomness of exploration at the start of learning. In our work, the experience replay mechanism allows off-policy learning, and thus the agent can use the experiences from the replay buffer as well as the demonstration data. Since we cannot sample the experience transitions from the replay buffer in an uniform manner, we need to consider the significance of the experience transitions in our work.

4.1. State and Action Space

In order to learn the manipulation skills of human operators, the state space is defined based on the measured data that can also be observed by human operators. As mentioned above, the learning agent has to make sure that the production rate of dredged materials must be maintained in an appropriate range. The variables of the state space must reflect the dynamics of the swing process. Since the measured slurry density has a time lag between the soil cutting and the measurement at the stern of the CSD, we still can have other correlated signals to infer to what extent the soil by the cutter head. In this work, the state space includes the motor current of the cutter head I_c, the motor current of the underwater pump I_p, the degree of suction vacuum D_v, the flow velocity V_f, and the measured slurry density C_v, that is $s = [I_c, I_p, D_v, V_f, C_v]$.

As the dredging environment is very complex, the reason why those variables are selected to constitute the state space mainly relies on the observations of the human operators. Because the learning agent discussed in this work seeks to imitate the manipulation skills of human operators, we believe that what the human operators can observe in the monitoring screen of a CSD can reflect enough information about the complex and dynamic dredging process.

Besides, the constitution of the state space can also be explained based on the working principle of a CSD. We usually use a nuclear-based gamma densitometer to directly measure the slurry density C_v of the pipeline, but this value has a time lag because the densitometer has to be installed at the stern of a CSD. The slurry density can directly reflect the solids level in the pipeline, and the control objective of the learning agent is to track the desired slurry density. During the swing process, the cutter head has to rotate around the spud pole by slacking (or pulling) the cable of anchors, and, thus, the motor current of the cutter head I_c can reflect the reaction forces. The increase of the amount of the dredged materials will lead to the augmentation of this value. At the same time, dredged materials determine the slurry density in the pipeline, and the underwater pump needs to suck up all the materials to improve the pressure for pipeline transportation. Thus, the motor current of the underwater pump I_p is also related to the amount of the dredged materials. If the underwater pump needs to suck up the dredged materials with high density, the degree of suction vacuum D_v will also increase. Since the dredged materials need to be transported to the disposal area through a pipeline, the energy for carrying the dredged materials is supplied by the flow velocity V_f.

The action space is a set of permissible values to continuously control the swing speed V_s, which has to be kept in a safety range. At each time-step t, the learning agent has to change the swing speed to determine the amount of soil to be dredged. The swing speed is restricted by the permitted torque on the swing winches.

4.2. State Transition Prediction

Since an accurate simulator or a virtual environment of a CSD is not ready-made for the learning agent to interact with, we will develop a neural network to predict the state transitions of the swing process. For a stochastic MDP, the state transition function should specifics the distribution over the possible next states. Thus, the neural network $f_\phi(s, a)$ should be able to output the coming state based on the current state s and action a. Here we will first cover the considerations of how to build a one-step model for predicting the state transitions. Recurrent Neural Networks (RNNs) can establish the connections between time series issues via cycles in a network, where the action at time-step t can affect the state transition at time-step $t+1$. Thus, we can use RNNs to construct a model to predict what the coming state will be when the agent performs an action. As typical RNNs has the difficulty to handle the problem with long-term temporal correlations, we borrow the idea of the Deep Long Short-Term Memory (DLSTM) as in (Sagheer and Kotb, 2019) to predict the state transitions of the dredging dynamics. The DLSTM is a variety of the LSTM structure (Hochreiter and Schmidhuber, 1997), containing multiple LSTM layers to learn the nonlinearity and complexity of time-series data.

4.2.1. Single LSTM Structure

The basic structure of a single LSTM block is shown in Figure 5. We can see that the cell state is the key unit that runs along the entire straight line. The cell state is regulated by the gates to ensure that the information can be inserted or removed from it.

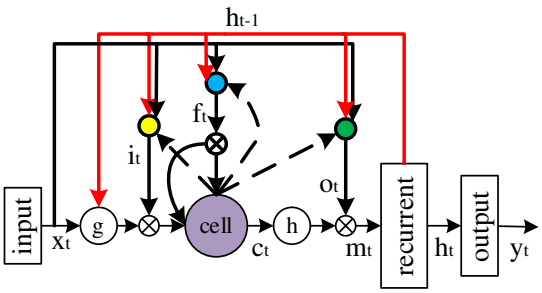

Figure 5: The basic structure of a single LSTM block.

The input of a single LSTM structure is a sequence of data points x_t indexed by time-step t, and in this work, the input consists of the current state and action, i.e., $x_t = [s_t, a_t]$. The target of a LSTM is demoted by y_t, we have $y_t = s_{t+1}$ for the swing process. In order to maintain temporal correlations, memory blocks with input and output gates are used in the recurrent hidden layer. In the LSTM structure, the forget gate f_t is used to determine which information should be discarded from the cell state. The basic idea is that it will cut off the flow if the value is zero and will allow all flow to pass through if the value is one. Moreover, the cell state can be updated by the input gate i_t, and the new optional values $\widetilde{c}_t$ are generated by an activation function. The gates and activation vectors are updated by the following formulas,

$$\begin{cases} i_t = \sigma\left(U_{ix}x_t + U_{im}m_{t-1} + U_{ic}c_{t-1} + b_i\right) \\ f_t = \sigma\left(U_{fx}x_t + U_{fm}m_{t-1} + U_{fc}c_{t-1} + b_f\right) \\ o_t = \sigma\left(U_{ox}x_t + U_{om}m_{t-1} + U_{oc}c_{t-1} + b_o\right) \\ \widetilde{c}_t = g\left(U_{cx}x_t + U_{cm}m_{t-1} + b_c\right) \\ c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t \\ m_t = o_t \odot h\left(c_t\right) \\ y_t = \phi\left(U_{ym}m_t + b_y\right). \end{cases} \tag{8}$$

Here i_t, f_t and o_t denote the input gate, forget gate and output gate, respectively, and c_t and m_t have the same size, representing the cell activation vectors. The notable weight matrices are denoted by U terms, in which U_{ix} indicates the weights from the input gate to the input node, and the other terms (e.g., U_{ic}, U_{fc}), and U_{oc} represent the diagonal weight matrices. We also have the b terms that denote the bias vectors, e.g., b_f means the forget gate bias vector. The cell input and output activation functions are denoted by g and h, representing the tanh activation function and the softmax activation function. We use σ, ϕ and the symbol $\odot$ to denote the logistic sigmoid function, the softmax output activation function, and the element-wise product of vectors, respectively.

4.2.2. Deep LSTM Architecture

In order to improve the overall performance of a neural network, the depth of the LSTM structure can be expanded into a deeper architecture. As depicted in Figure 6, the DLSTM recurrent network consists of multiple LSTM blocks. The input of the first LSTM block is a sequence of data points

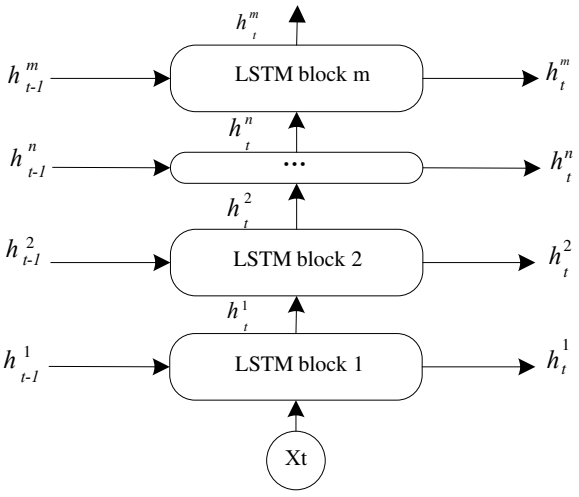

Figure 6: The architecture of the DLSTM recurrent network.

x_t indexed by time-step t and previous hidden state h^1_{t-1}. Afterwards, its output will be sent to the second LSTM block as the input, and so forth. Consequently, the m-th LSTM block can output the predicted target. In this work, we use the DLSTM network to predict the state transitions of the swing

process based on historical collected data. Afterwards, this network can work as a virtual environment that provides the feedback about the coming state when the agent performs an action.

4.3. Reward Function

In order to define the reward function for the learning agent, we have to take account of the safety concerns and the control objective. Specifically, the dredged materials should be able to maintain at a high level, but the other parameters should not exceed the permitted values. Here we use r_t^{dis} to denote the tendency towards the desired goal state s_{goal},

$$r_t^{\text{dis}} = \begin{cases} 0 & \text{if } \|s_t - s_{\text{goal}}\| < \|s_{t-1} - s_{\text{goal}}\| \\ -1 & \text{otherwise.} \end{cases} \tag{9}$$

Here $\|s_t - s_{\text{goal}}\|$ indicates the distance to the desired goal state s_{goal} from the current states_t. The basic idea is that if the agent moves farther away from the desired goal state in comparison with the last step, it will get a penalty. Then, the reward function at time step t is defined by

$$r_t = r_t^{\text{dis}} \sum_i \frac{s_t^i - s_t^{\min}}{s_t^{\max} - s_t^{\min}} + r_t^{\text{safe}}, \tag{10}$$

where we use $\frac{s_t^i - s_t^{\min}}{s_t^{\max} - s_t^{\min}}$ to normalize the values of the current state vector to eliminate the effect of different scales. Thus, s_t^i represents the i-th element of the state vector, and $s_t^{\min}$ and $s_t^{\max}$ represent the minimum and maximum values of the current state vector, respectively. Moreover, in order to consider the safety concerns, we also define the second term r_t^{safe},

$$r_t^{\text{safe}} = \begin{cases} -5 \text{ if} & \text{(unsafe)} \\ 0 \text{ if} & \text{(otherwise).} \end{cases} \tag{11}$$

Here the unsafe situation means that the current state exceeds the permitted values, i.e., the motor current of the cutter head $I_c > 955$A, the motor current of the underwater pump $I_p > 180$A, the degree of suction vacuum $D_v < -70$bar, and the flow velocity $V_f < 5$m/s. The permitted values for each CSD are different, and in this work those values are provided by the experienced human operator that manipulates the CSD for data acquisition. In the learning approach, a big negative reward -5 will be sent to the agent so as to avoid such a situation. The basic idea of the reward function is that the agent is expected to pursue the goal state associated with high slurry density; otherwise, the penalty will be enlarged along with the distance to the goal state. At the same time, the agent should also pay attention to the limit of the permitted values with regard to the safety concerns.

4.4. Sample Strategy

In RL, the agent has to incrementally update its control policy by collecting a stream of experience, while it interacts with the environment. The experience replay mechanism allows the mixture of recent transitions. For example, DQN (Mnih et al., 2015) uses a sliding window to sample the transitions uniformly. In this work, we borrow the idea of prioritizing some transitions than others (Schaul et al., 2015), with the aim of utilizing the replay memory in a more effective manner.

At the beginning of exploration, the agent has to take considerable steps until it reaches to the goal state. As a consequent, many transitions stored in the replay memory are redundant, and the most useful transitions will be hidden by those redundant ones. In order to prioritize the most useful transitions, we have to measure the importance of each transition. This value cannot be obtained directly, but we can use temporal difference (TD) error to approximately indicate the magnitude.

In order to consider variations in the distribution, we assess the sampling weights to update the network. Thus, we use $P(i)$ to denote the probability of selecting a transition i, and it will be calculated based on its priority p_i

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}. \tag{12}$$

Here, the higher the probability $P(i)$ of a transition i is, the more likely the transition i will be sampled in the next episode. In addition, α is chosen as 0.3 representing the sampling weight of the agent's initial exploration. We use the following formula to determine the priority of a transition,

$$p_i = |\delta_i| + \lambda \left| \nabla_a Q\left(s_i, a_i \mid \theta^Q\right) \right| + \epsilon, \tag{13}$$

where the first term δ_i is the TD error: $\delta_i = r_{i+1} + \lambda Q(s_{i+1}) - Q(s_i)$, the second term is the actor's loss: $Q(s_t, a_t|\theta^Q) - y_t$ (see Equation 5), and the third term ϵ is used to ensure that all the transitions have a change to be sampled. Here we use λ to balance the contributions between the TD error and the actor's loss.

4.5. Learning with Demonstrations

As mentioned before, it is expensive for the agent to learn the optimal policy from scratch in a real industrial environment, so we will apply human demonstrations to regulate the exploration policy from the beginning of actual interaction. Before the agent performs exploration, the state transitions from human experts can be stored in the replay buffer. The pseudo-code of learning with demonstrations is sketched in Algorithm 1. Before the agent interacts with the interactive environment, the actor network and critic network are initialized with their respective parameters, and the replay buffer is initialized with human demonstrations in the form of (s_t, a_t, r_t, s_{t+1}) transitions. To learn the optimal policy, the initial state for exploration will be set randomly in each episode, and the total learning steps and episodes are set to T and M, respectively. In each time-step, the agent chooses an action to execute according to the current policy and noise, and then gets the corresponding reward and arrives at the next state. Of course, the replay buffer can also be updated along with the agent's exploration, and the oldest transitions

Algorithm 1 Swing process learning with demonstrations.

Inputs: D^{replay} the replay buffer initialized with demonstrations; θ^Q parameters for initial critic network $Q(s,a|\theta^Q)$; θ^μ parameters for initial actor network $\mu(s|\theta^\mu)$. $\theta^{Q'}$ parameters for initial critic target network $Q(s,a|\theta^{Q'})$; $\theta^{\mu'}$ parameters for initial actor target network $\mu(s|\theta^{\mu'})$; k is the number of human demonstrations.
for *episode* $e = 1$ to M **do**
  Initialize environmental state s_o
  for *step* $t = 1$ to T **do**:
    Sample noise from Gaussian $n_t = \mathcal{N}(0, \xi)$
    Select an action $a_t = \mu(s|\theta^\mu) + n_t$ to perform
    Get the next step s_{t+1} and reward r_t
    Store the transition (s_t,a_t,r_t,s_{t+1}) to D^{replay}
    Overwrite oldest transitions if the capacity is full
    Sample a minibatch from D^{replay} with prioritization.
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s|\theta^{\mu'}|\theta^{Q'}))$
    Update the critic by minimizing the loss $L(\theta^Q)$
    Update the actor $\nabla_{\theta^\pi} J(\theta^\pi)$
    Update the target networks: $\theta^Q \rightarrow \theta^{Q'}$;$\theta^\mu \rightarrow \theta^{\mu'}$
  end for
end for

can be overwritten if the capacity is full. Thus, at the beginning of exploration, the sampled mini-batch of transitions already contains human demonstrations, and the sample strategy can ensure that the most useful transitions can be prioritized. Then the actor collects samples according to the priority sampling mechanism, and the critic network and the actor target network are updated by computing the loss function and the policy gradient, respectively. When enough exploration has been performed, the replay buffer can be incrementally replaced by the transitions of the agent's self-exploration. In such a way, the agent's behavior can be regulated by demonstrations in the beginning, but it still possesses the possibility to exceed the performance of human experts through self-exploration.

5. Evaluation and Results

To evaluate the proposed approach, we will first validate the accuracy of predicting the state transitions of the swing process. Thus, an experienced human operator is required to manipulate a real CSD for 6 hours to collect the operational data. We utilize the collected data to train the DLSTM network, as well as to initialize the replay buffer with state transitions generated by the human expert. Specifically, the DLSTM network needs the training data to learn the dynamics or state transitions of the swing process, and, afterwards, it can output the predicted states for the RL agent. The historical data operated by the human expert will also be considered as demonstrations to regulate the exploration of the RL agent.

5.1. State Transition Prediction by DLSTM

As mentioned above, the raw dataset for the state transition prediction contains 21600 observations (6 hours), in which 19440 observations (90% of the dataset) have been used for training the network, and the remaining 2126 observations (10% of the dataset) are used for evaluating the performance. Figure 7 depicts the performance of the DLSTM network to predict the state transitions of the swing process. In this work, the DLSTM network is responsible for providing the feedback on what the next state will be when the agent performs an action. As mentioned previously, the states includes the slurry density, the motor current of the cutter head, the motor current of the pump, the suction vacuum and the flow velocity, whereas the action is the swing speed.

In comparison with the actual collected data, we can conclude that, in general, the predicted state transitions satisfy the dynamic changes of the environment in response to the actions. Thus, it can serve as a state transition function to output the coming state. As mentioned before, the measured slurry density has a time lag with respect to the control action. As shown in Figure 7, the predicted slurry density curve closely matches the tendency of the actual measured curve. Thus, we can claim that the proposed DLSTM network can take account of the time delay when predicting state transitions.

With regard to the prediction accuracy, we consider the root mean square error (RMSE), the root mean square percentage error (RMSPE), the relative error and the coefficient of determination (R^2). The best results of the DLSTM network are achieved using three LSTM layers after 2000 episodes, in which RMSE and RMSPE are 0.22 and 2.98 respectively, and $R^2 = 0.99$. In addition, we can see the relative errors in Figure 8, where the suction vacuum is the worst case but the relative error is only 3.66%. Thus, we can say that the DLSTM network introduced in this work is acceptable to predict the state transitions for the following RL agent.

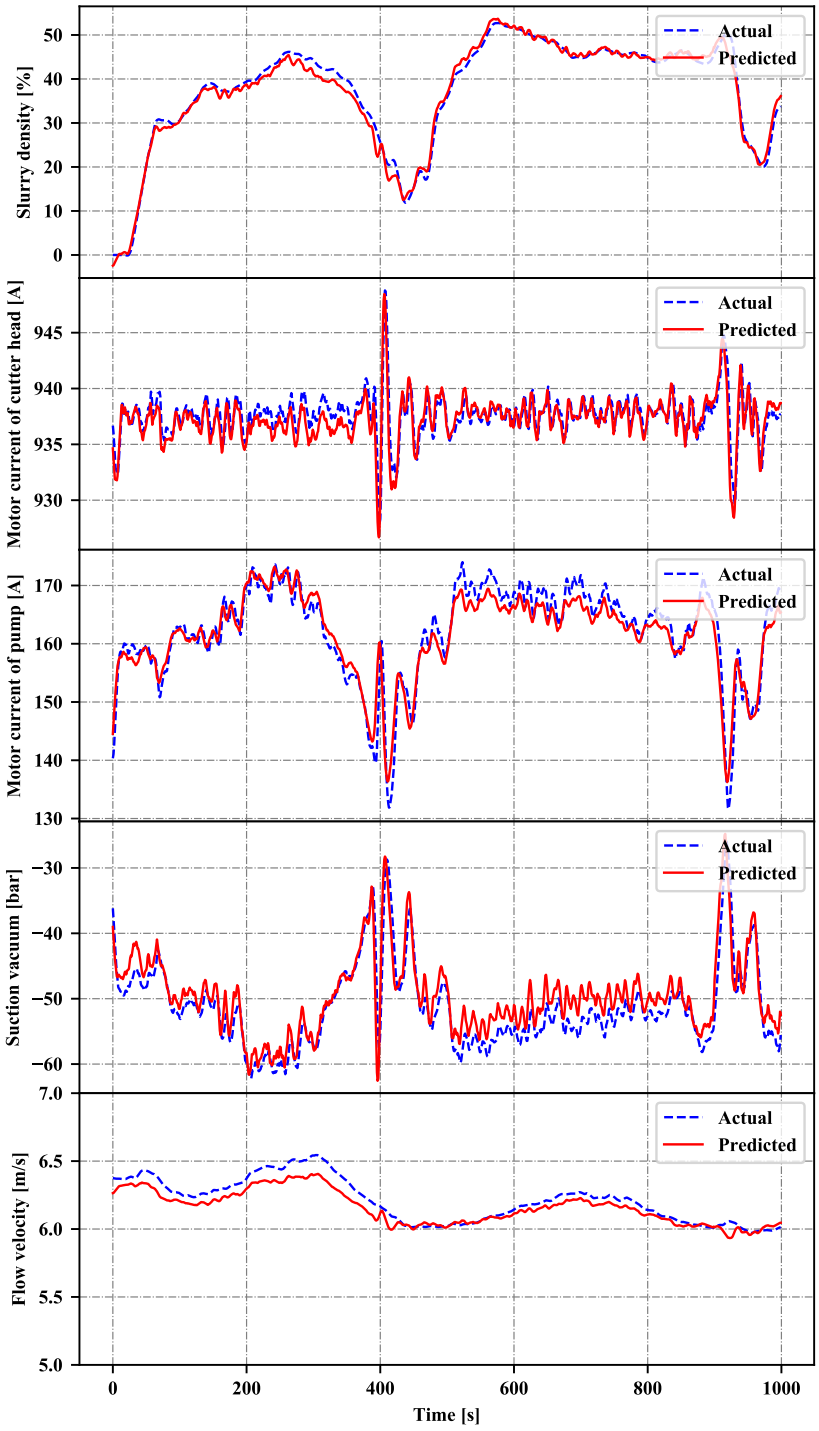

Figure 7: The state transition prediction by the DLSTM network.

5.2. Learn the Control Policy with Demonstrations

To evaluate the performance of the proposed approach with demonstrations, the well-known deep deterministic policy gradients (DDPG) algorithm is implemented as the baseline for comparison. Since our proposed approach combines deep deterministic policy gradients with human demonstrations, DDPGwD is used as abbreviation throughout this work. In the experiment we also investigate how the fragments of the demonstration data in the replay buffer will influence the learning performance. Thus, the proposed approach is distinguished by three groups: (i) DDPGwD-beginning, (ii) DDPGwD-interleaved and (iii) DDPGwD-first-half. In the first group, all the demonstration data is stored into the replay buffer before the self-exploration of the DRL agent. In the second group, the demonstration data is divided into 10 fragments, and each fragment is inserted into the replay

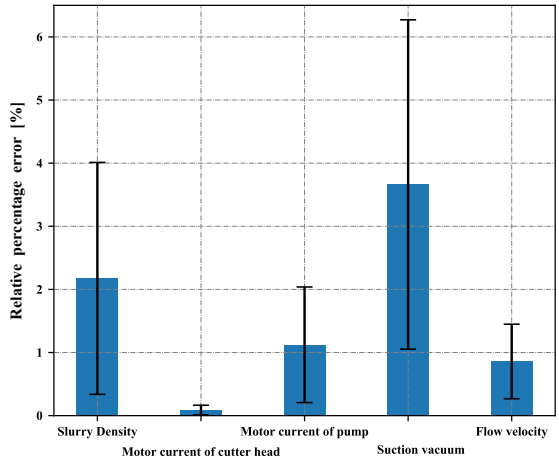

Figure 8: Relative error between the real and predicted transitions.

buffer every 100 learning episodes. In the third group, the demonstration data is divided into two parts, and the first half is stored into the replay buffer before the self-exploration, while the second half is inserted into the replay buffer after 500 learning episodes. In order to reduce variance and filter out random effects, each algorithm has been run for 20 times in the experiments. The maximum learning episodes is 1000, and the maximum step for each episode is 500.

As shown in Figure 9, we can clearly see that the proposed DDPGwD approach can outperform the baseline. According to Equation 10, the cumulative reward of each episode can reveal whether the agent moves towards the desired goal state or not, as well as whether safety concerns have been triggered during each learning episode. Thus, if the cumulative reward is negative, it means the agent has swayed back and forth towards the desired goal state, or the trajectory of the agent exceeds the limit of the safety range.

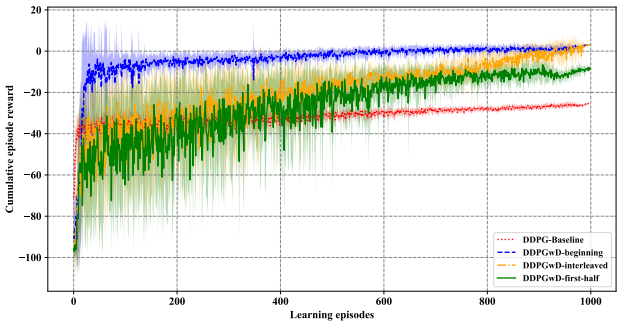

Figure 9: The cumulative reward for each learning episode.

With regard to the performance of the four algorithms, we can also conclude that the way of storing the demonstration data indeed influences the learning performance. The most promising case is the DDPGwD-beginning, where all the demonstration data has been stored into the replay buffer before self-exploration. More demonstrations can facilitate the regulation of exploration behaviour at the beginning of learning. In contrast, if demonstrations are gradually stored into the replay buffer, as in DDPGwD-interleaved and the first-half case, they can be overwhelmed by exploration data, thereby reducing the probability of being sampled in the mini-batch. Thus, we can claim that the proposed approach can find the optimal control policy through the least number of learning steps. Once the optimal control policy is converged, it can directly output the optimal control action for each state.

5.3. Comparison with Human Expert Operation

In order to demonstrate the effectiveness of the proposed learning approach, we take the performance of a human expert as the baseline. Figure 10 depicts the control curves of how our learning approach in comparison with a human expert operates the swing process of a CSD. The dashed curves represent a fragment of the operation data by a human expert, in which the CSD has started excavating the soil from a standstill state. Thus, the standstill state is also considered as the initial state to evaluate the optimal control policy obtained by our learning approach. In each step, we add random noise to take account of the environmental disturbances during the swing process. The solid curves represent how the obtained control policy operates the CSD from the standstill state. In particular, we can see that the slurry density is maintained at a high level, and all the other safety concerns (i.e., the motor current of the cutter head, the motor current of the pump, the suction vacuum, and the flow velocity) are kept within the safety range.

In comparison, the human expert cannot stabilize the slurry density at a high value because of the safety concerns. For example, as shown in Figure 10, the motor current of the cutter head begins to increase at 270s and reaches a peak at 290s where its value exceeds the maximum allowable value (955A). In this case, the human expert immediately reduces the swing speed so as to avoid causing potential accidents. As a result, the slurry density has dropped below 20%, which cannot lead to a good performance with regard to the production rate for a CSD. Thus, we can draw obvious conclusion that the proposed learning approach can outperform the human expert by make quicker responses to the dynamics of the swing process. Moreover, all the safety concerns have been taken into consideration during the stabilization of the slurry density. The proposed approach can provide a competitive solution to the intelligent control of the swing process of a CSD.

With regard to the actual engineering application of the proposed approach, the intelligent control mechanisms can be summarized in three aspects. Firstly, the effect of control actions will always satisfy the safety constraints, as defined in Equation 10, where all the key variables should not exceed the permitted values. Otherwise, negative penalties will be sent to the learning agent. Secondly, the optimal control policy will always try to minimize the difference between the current state and the desired goal state (see Figure 9), taking account of the safety constraints as well. Of course, the desired goal state is provided by the end user, and, in particular, the goal state should explicitly indicate the desired slurry density. This means that the real-time slurry density

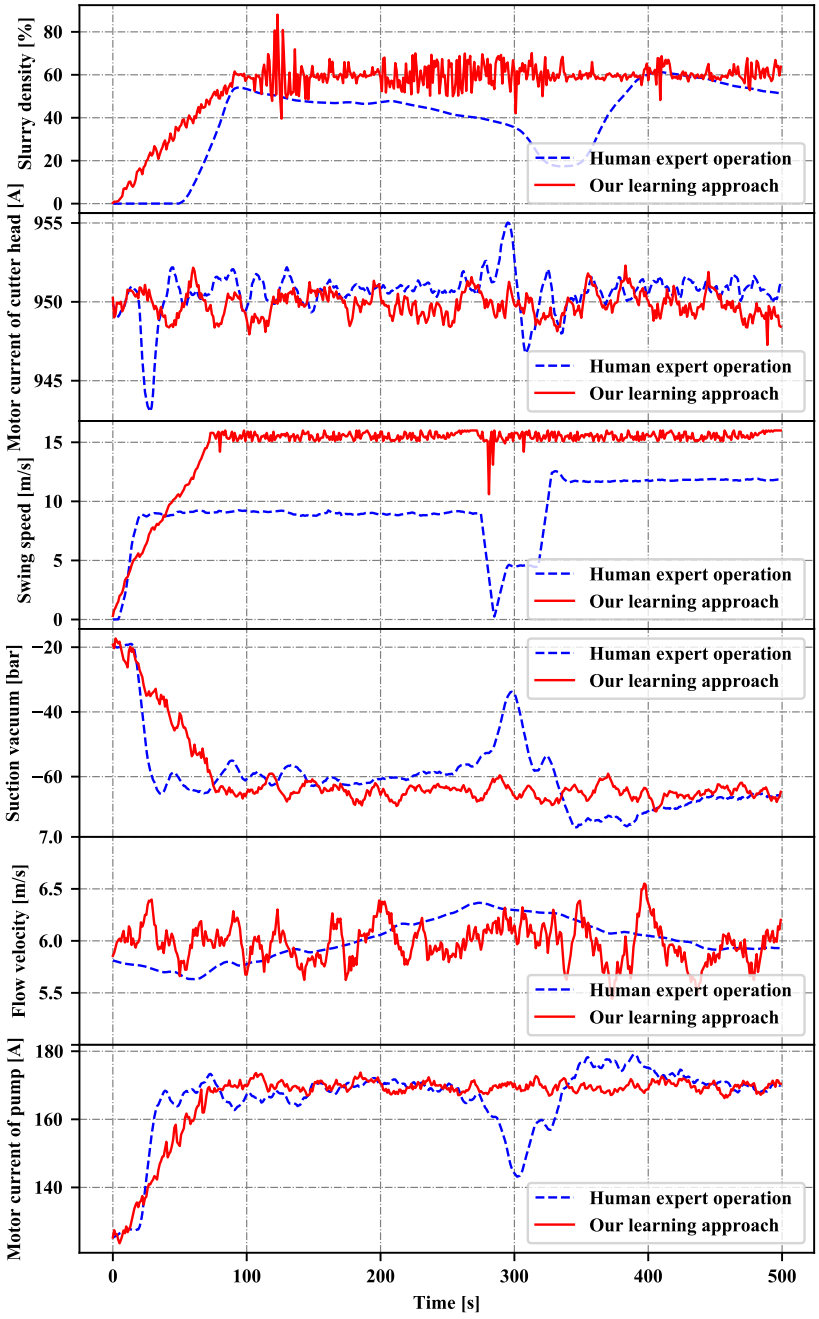

Figure 10: Performance comparison between human expert operation and our learning approach.

will track the desired value so as to avoid potential pipeline blockage. Thirdly, since it is inefficient and dangerous for an agent to learn from scratch in a real CSD, the proposed approach can allow the agent to obtain acceptable manipulation skills based on historical data. Afterwards, the pre-trained control policy can be applied to a real CSD so as to continuously improve the policy network by real-time generated data.

6. Conclusions and Future Work

The manipulation of the swing process of a CSD involves various uncertainties, so it is hard to design a precise model to handle the dynamics of the environment. In this paper, we focus on a reinforcement learning approach that can learn the optimal control policy with human demonstrations for the swing process control. Specifically, a DLSTM network is introduced to serve as the state transition function to provide one-step transition prediction, and it is trained by the collected demonstration data. Moreover, the DRL agent can

make use of the demonstration data to regulate its exploration behaviour at the start of learning, and then continue improving its behaviour by self-generated data. The experiment results show that the proposed approach can provide a competitive solution to the swing process control problem. The work also opens up the possibility of apply the DRL framework to many practical applications, where accurate simulators do not exists and we cannot allow the agent to learn from scratch in real environments.

Of course, fully automated and intelligent manipulation control of a CSD involves many other control variables, such as the rotation speed of the cutter head, the inclination angle of the ladder, and the swing width of the ladder. In this work, we only focus on the swing speed control because it is the most tedious and recurring work for the operators. According to on-site investigations, the rotation speed of the cutter head does not need to be adjusted frequently, as it is mainly determined by the soil type, and the swing width of the ladder is often regulated by the construction process. The inclination angle of the ladder decides the cutting depth, which means that when one layer of soil is excavated, the cutting depth needs to be adjusted intermittently. Moreover, when a swing movement is completed, the CSD may need to be pushed forward a small step through the cooperation of the main spud pole and the auxiliary spud pole. Therefore, in future work, we will also look at the decision making problem of finding the optimal control policies for the cutting depth and the step length of the spud poles. In this case, multiple learning agents will be required to find their own optimal control policies in a cooperative manner.

References

Bai, S., Li, M., Kong, R., Han, S., Li, H., Qin, L., 2019. Data mining approach to construction productivity prediction for cutter suction dredgers. Automation in Construction 105, 102833.

Brys, T., Harutyunyan, A., Suay, H.B., Chernova, S., Taylor, M.E., Nowé, A., 2015. Reinforcement learning from demonstration through shaping, in: Twenty-Fourth International Joint Conference on Artificial Intelligence.

Chan, L., Naghdy, F., Stirling, D., 2014. Application of adaptive controllers in teleoperation systems: A survey. IEEE Transactions on Human-Machine Systems 44, 337–352.

Cui, R., Yang, C., Li, Y., Sharma, S., 2017. Adaptive neural network control of auvs with control input nonlinearities using reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics: Systems 47, 1019–1029.

Gu, S., Holly, E., Lillicrap, T., Levine, S., 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in: 2017 IEEE international conference on robotics and automation (ICRA), IEEE. pp. 3389–3396.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al., 2018. Deep q-learning from demonstrations, in: Thirty-Second AAAI Conference on Artificial Intelligence.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.

Lewis, F.L., Vrabie, D., Vamvoudakis, K.G., 2012. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. IEEE Control Systems Magazine 32, 76–105.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2016. Continuous control with deep reinforcement learning. International Conference on Learning Representations .

Lin, C.C., Deng, D.J., Chih, Y.L., Chiu, H.T., 2019. Smart manufacturing scheduling with edge computing using multi-class deep q network. IEEE Transactions on Industrial Informatics .

Liu, Y.J., Li, J., Tong, S., Chen, C.P., 2016. Neural network control-based adaptive learning design for nonlinear systems with full-state constraints. IEEE Transactions on Neural Networks and Learning Systems 27, 1562–1571.

Liu, Y.J., Tang, L., Tong, S., Chen, C.P., Li, D.J., 2014. Reinforcement learning design-based adaptive tracking control with less learning parameters for nonlinear discrete-time mimo systems. IEEE Transactions on Neural Networks and Learning Systems 26, 165–176.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518, 529.

Oh, J., Guo, X., Lee, H., Lewis, R.L., Singh, S., 2015. Action-conditional video prediction using deep networks in atari games, in: Advances in neural information processing systems, pp. 2863–2871.

Pan, Y., Yu, H., 2015. Composite learning from adaptive dynamic surface control. IEEE Transactions on Automatic Control 61, 2603–2609.

Quillen, D., Jang, E., Nachum, O., Finn, C., Ibarz, J., Levine, S., 2018. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 6284–6291.

Ross, S., Gordon, G., Bagnell, D., 2011. A reduction of imitation learning and structured prediction to no-regret online learning, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, pp. 627–635.

Sagheer, A., Kotb, M., 2019. Time series forecasting of petroleum production using deep lstm recurrent networks. Neurocomputing 323, 203–213.

Schaul, T., Quan, J., Antonoglou, I., Silver, D., 2015. Prioritized experience replay. arXiv preprint arXiv:1511.05952 .

Shao, K., Zhu, Y., Zhao, D., 2018. Starcraft micromanagement with reinforcement learning and curriculum transfer learning. IEEE Transactions on Emerging Topics in Computational Intelligence 3, 73–84.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al., 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 362, 1140–1144.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al., 2017. Mastering the game of go without human knowledge. Nature 550, 354.

Sun, W., Venkatraman, A., Gordon, G.J., Boots, B., Bagnell, J.A., 2017. Deeply aggrevated: Differentiable imitation learning for sequential prediction, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org. pp. 3309–3318.

Tang, J., Wang, Q., Zhong, T., 2009. Automatic monitoring and control of cutter suction dredger. Automation in Construction 18, 194–203.

Tang, J.Z., Wang, Q.F., Bi, Z.Y., 2008. Expert system for operation optimization and control of cutter suction dredger. Expert Systems with Applications 34, 2180–2192.

Wang, H., Huang, T., Liao, X., Abu-Rub, H., Chen, G., 2016. Reinforcement learning in energy trading game among smart microgrids. IEEE Transactions on Industrial Electronics 63, 5109–5119.

Wei, C., Ni, F., Chen, X., 2019. Obtaining human experience for intelligent dredger control: A reinforcement learning approach. Applied Sciences 9, 1769.

Xu, J., Hou, Z., Wang, W., Xu, B., Zhang, K., Chen, K., 2018. Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks. IEEE Transactions on Industrial Informatics 15, 1658–1667.

Yahya, A., Li, A., Kalakrishnan, M., Chebotar, Y., Levine, S., 2017. Collective robot reinforcement learning with distributed asynchronous guided policy search, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 79–86.

Zhang, B., Liu, C.H., Tang, J., Xu, Z., Ma, J., Wang, W., 2017. Learning-based energy-efficient data collection by unmanned vehicles in smart cities. IEEE Transactions on Industrial Informatics 14, 1666–1676.