IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Learning Adaptive Policies for Autonomous Excavation Under Various Soil Conditions by Adversarial Domain Sampling

Takayuki Osa[1,2], Naoto Osajima[3], Masanori Aizawa[4], Tatsuya Harada[1,2]

***Index Terms*—Robotics and Automation in Construction, Reinforcement Learning, Deep Learning Methods**

***Abstract*—Excavation is a frequent task in construction. In this context, automation is expected to reduce hazard risks and labor-intensive work. To this end, recent studies have investigated using reinforcement learning (RL) to automate construction machines. One of the challenges in applying RL to excavation tasks concerns obtaining skills adaptable to various conditions. When the conditions of soils differ, the optimal plans for efficiently excavating the target area will significantly differ. In existing meta-learning methods, the domain parameters are often uniformly sampled; this implicitly assumes that the difficulty of the task does not change significantly for different domain parameters.**

In this study, we empirically show that uniformly sampling the domain parameters is insufficient when the task difficulty varies according to the task parameters. Correspondingly, we develop a framework for learning a policy that can be generalized to various domain parameters in excavation tasks. We propose two techniques for improving the performance of an RL method in our problem setting: adversarial domain sampling and domain parameter estimation with a sensitivity-aware importance weight. In the proposed adversarial domain sampling technique, the domain parameters leading to low expected Q-values are actively sampled during the training phase. In addition, we propose a technique for training a domain parameter estimator based on the sensitivity of the Q-function to the domain parameter. The proposed techniques improve the performance of the RL method for our excavation task. We empirically show that our approach outperforms existing meta-learning and domain adaptation methods for excavation tasks.

I. INTRODUCTION

EXCAVATION is one of the most frequent tasks in construction, and automating such excavation is expected to reduce hazard risks and labor-intensive work. To this end, the automation of excavation has been studied for decades [1]–[4]. Recent studies have employed reinforcement learning (RL) to obtain policies for controlling an excavator [5]–[7]. In RL, the optimal policy that maximizes the expected return is obtained through autonomous trials and errors [8]. Based on recent success of RL in robotics, RL is considered as a promising approach for automating excavation.

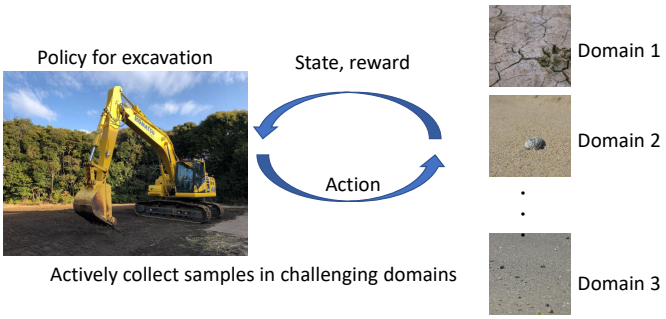


Fig. 1. To obtain an adaptive policy that can be generalized to various condition, a policy is trained more frequently on more challenging domains.

When the conditions of soil are different, the optimal plans for efficiently excavating the target area will be significantly different. When considering the problem of learning a policy that can work across a set of various domains, the task difficulties can vary among a given domain distribution. The problem of learning a policy applicable to various domains is often formulated as multitask RL [9] or meta RL [10], which help obtain a policy that works in or rapidly adapts to various domains. However, in existing meta-RL and multitask-RL methods, the domain parameters are often uniformly sampled during the training phase, thereby implicitly assuming that the difficulty of the task does not change significantly for different domain parameters. However, we have observed variance in the task difficulty within the range of domain parameters for excavation tasks, and policies trained by existing methods have not performed well for domains where a task was more difficult than in other domains.

To address this issue, we develop a framework for learning a policy that can be generalized to various domain parameters in excavation tasks even if there is variance in the task difficulty. We propose two techniques for improving the performance of an RL method in our problem setting: adversarial domain sampling, and domain parameter estimation with a sensitivity-aware importance weight. In the proposed domain sampling method, the domain parameters leading to low expected Q-values are sampled during the training phase. This approach can be viewed as a type of adversarial training where the agent attempts to maximize the expected return and the domain sampler attempts to minimize the expected return. In addition, we also propose a method for training a model to estimate the domain parameters. In our excavation task, it is necessary to estimate the domain parameters to adjust the policy behavior. To this end, we propose an importance weight incorporating the sensitivity of the Q-values to the domain parameters. Our experimental results show that the proposed method enables us to obtain a policy that works in various domains (including challenging ones). Moreover, our approach outperforms

*This work was done as a joint research by the University of Tokyo and Komatsu Ltd.

Manuscript received:February 7, 2023; Revised May, 10, 2023; Accepted June 18, 2023.

This paper was recommended for publication by Editor Hyungpil Moon upon evaluation of the Associate Editor and Reviewers' comments.

[1]T. Osa and T. Harada are with the University of Tokyo, Tokyo, 113-8656, Japan. {osa, harada}@mi.t.u-tokyo.ac.jp

[2]T. Osa and T. Harada are with RIKEN Center for Advanced Intelligence Project, Chuo-ku, Tokyo, 103-0027, Japan

[3]N. Osajima is with Kyushu Institute of Technology, Kitakyushu, Fukuoka, 808-0135, Japan

[4]M. Aizawa is with Komatsu Ltd., kanagawa, 254-0913, Japan

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

existing methods on an excavation task.

The remainder of the paper is structured as follows. Section III describes the related work. Section II introduces the RL formulation and related approach. Section IV describes the proposed method.. Section V explains the experimental results, and Section VI describes the conclusions.

II. Preliminary

A. Reinforcement Learning

We consider a Markov decision process defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, d)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ is the transition probability density, $r(\boldsymbol{s}, \boldsymbol{a})$ is the reward function, γ is the discount factor, and $d(\boldsymbol{s}_0)$ is the probability density of the initial state. A policy $\pi(\boldsymbol{a}|\boldsymbol{s}) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is defined as the conditional probability density over the actions given the states. A return is defined as the sum of the discounted rewards over time $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$. The goal of RL is to obtain a policy that maximizes the expected return $\mathbb{E}[R|\pi]$.

B. Meta-Learning

We consider a policy $\pi_{\boldsymbol{\theta}}$ parameterized with a vector $\boldsymbol{\theta}$. The objective function for meta-learning in general can be expressed as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})} \left[\mathbb{E}[R|\pi_{\boldsymbol{\theta}}, \mathcal{T}] \right], \tag{1}$$

where $\mathcal{T}$ represents a task, and $p(\mathcal{T})$ is the task distribution to which the meta-policy must adapt. In meta-RL, a meta-policy is obtained by maximizing the expected return across tasks with respect to a given task distribution. In meta-RL methods, such as PEARL, a policy conditioned on the latent variable is typically trained in the training phase [11]–[13]. In the test phase, the model was adapted to a task sampled from the task distribution; however, the task descriptions were unknown. Thus, latent variables or task descriptions are typically estimated during the testing phase. However, in risk-aware RL [14], maximizing the expectation does not necessarily improve the worst-case performance. As shown in our experiment, a policy trained with an existing meta-RL method may not work for some of the tasks even if the average performance across the tasks is acceptable.

C. Problem Setting in This Study

In our problem setting, we observe a depth image of the landscape as a state $\boldsymbol{s}$. An action $\boldsymbol{a}$ represents a single trajectory for excavation. We assume that the domain parameter $\boldsymbol{z}$defines the characteristic of the soil in the target excavation area and that the transition probability $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{z})$ varies according the domain parameter $\boldsymbol{z}$. For example, the soil can be hard and crumbly in certain domains while soft and sticky in other domains. Thus, when the domain parameter $\boldsymbol{z}$ differs, even if we take the same action under the same state, the next state $\boldsymbol{s}'$ can be significantly different. Our goal is to obtain a policy for planning a trajectory to efficiently excavate the target area under various values of the domain parameter. To this end, we train a model conditioned on the domain parameters that provides different excavation strategies for different values of the domain parameters. We assume that the values of the domain parameters are fixed during an episode.

In this study, we assume that the domain parameter $\boldsymbol{z}$ is known during the training phase. However, in the test phase, the domain parameter $\boldsymbol{z}$ is unknown and therefore needs to be estimated. Considering the case where a policy is trained in a simulation and tested in the real world, we think that our problem setting is reasonable. Our problem setting is an intermediate problem between multitask RL and meta-RL. In multitask RL, the task ID or task description is provided, and multiple tasks are solved simultaneously. In meta-RL, unlike multitask RL, the task ID/description is not provided during both the training and test phases. Meta-RL often learns a policy conditioned on the latent variable [11]; thus, the latent variable corresponding to the test task needs to be estimated online in the test phase.

D. QT-Opt

We briefly introduce QT-Opt [15], a variant of Deep Q-learning for tasks with a continuous action space. In QT-Opt, the optimal Q-function is approximated by applying the optimal Bellman operator to the approximated Q-function. Given an approximated Q-function $Q_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{a})$ parameterized with a vector $\boldsymbol{w}$, the Q-function is updated by minimizing an objective function as follows:

$$\boldsymbol{w} = \sum_{(\boldsymbol{s}_i, \boldsymbol{a}_i) \in \mathcal{B}} \left\| y_i - Q_{\boldsymbol{w}}(\boldsymbol{s}_i, \boldsymbol{a}_i) \right\|^2 \tag{2}$$

where $\mathcal{B}$ is a batch of samples selected from the replay buffer. The target value y_i is computed as follows:

$$y_i = r_i + \gamma \max_{\boldsymbol{a}'} Q_{\boldsymbol{w}'}(\boldsymbol{s}', \boldsymbol{a}'). \tag{3}$$

In QT-Opt, the maximum Q-values are approximated using the cross-entropy method (CEM) [16]. Unlike actor-critic methods [17], [18], a policy is not explicitly modeled in QT-Opt. In this study, we extend QT-Opt to a meta-learning setting.

III. Related Work

Autonomous excavation approaches can be classified into three categories: 1) optimization-based methods [19], 2)imitation-learning-based (IM-based) methods [1], [4], and 3) RL-based methods [5]–[7]. Applying an optimization-based approach, such as that reported in [19], allows us to realize a robust controller.. However, the method presented in [19] is applied to two-dimensional trajectories, and extending the optimization-based approach to a three-dimensional space is not trivial. Using RL for autonomous excavation can be advantageous as it requires less expert knowledge; furthermore, it allows a policy for planning an excavation trajectory to be established even if the dynamics of the simulator are a black box [5]. In IM-based methods, data from human operators are acquired and a controller is trained in a supervised learning manner. However, compared with RL, IM-based methods involve time-consuming and costly data acquisition processes.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Recently, Pascal et al. applied RL to a 2D autonomous excavation and demonstrated that the obtained policy is applicable to actual excavator [7]. Lu et al. proposed to a framework for excavating a pile of solid objects [20]. In their method, a convolutional neural network predicted the excavation success for a given image. Although the authors showed how their method could be effectively applied to a real-world system, their focus was on excavating solid objects and did not address how to generalize to various soil conditions. Our study is built upon a previous study in [5], which investigated an RL method for learning to plan a trajectory for a depth image. However, the study in [5] did not address how to deal with various soil conditions.

In the field of RL, methods have been investigated for obtaining a policy applicable to various conditions. One prevalent approach is domain randomization, in which a policy is trained under various conditions [21]. Another approach is meta-learning [10], [11]. Given a task distribution, the goal of meta-learning is to obtain a policy that can rapidly adapt to a given test domain [10]. The fundamental difference between domain randomization and meta-learning is that although domain randomization obtains a single robust policy for various domains, in meta-learning, the policy adapted to the test domain will be different for each domain in a few-shot adaptation. Our problem setting is close to those of [12], [13] in the sense that a model for estimating the domain parameters can be trained in a supervised-learning manner. Nevertheless, the focus of these studies was on how to generalize the locomotion policies for the various terrains; they did not investigate the effects of the variance of task difficulties among a set of target tasks. In previous studies, learning methods were often evaluated in problem settings where the task difficulty did not significantly change among the given task distribution.

In this study, we show that it is necessary to employ a method that explicitly incorporates the task difficulty in our problem setting. Our approach can be seen as a type of adversarial training, i.e., the max-min game, where the agent is trying to maximize the expected return and the domain sampler is trying to minimize the expected return. The concept of our approach is similar to those of previous studies on min-max games in adversarial training [22], [23].

IV. Proposed Method

A. Meta-QT-Opt

In this study, our aim is to obtain a policy that maximizes the expected return across various domains. Thus, our problem can be formulated as follows:

$$\max_{\pi} \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \mathbb{E}_{\pi, p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a};\boldsymbol{z})} \left[Q^{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z}) \right], \tag{4}$$

where $\boldsymbol{z}$ is the domain parameter defining the transition probability. $p(\boldsymbol{z})$ is the prior distribution of $\boldsymbol{z}$ and defines the domain distribution. The objective function in (4) can be seen as a variant of the meta-learning objective in (1).

We extend the concept of QT-Opt [15], a variant of Q-learning [24], [25]. In a previous study [5], QT-Opt outperformed TD3 [18] and SAC [17] on the excavation task. In our framework, we approximate the Q-function conditioned on the domain parameter $\boldsymbol{z}$ with a function $Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})$ parameterized with a vector $\boldsymbol{w}$. We update $Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})$ by minimizing an objective function expressed as follows:

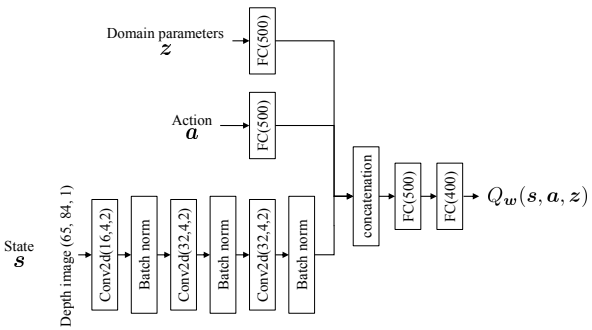


Fig. 2. Neural network architecture for modeling $Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})$.

$$\mathcal{L}(\boldsymbol{w}) = \sum_{(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{s}'_i,\boldsymbol{z}_i)\in\mathcal{B}} \left\| y_i - Q_{\boldsymbol{w}}(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{z}_i) \right\|^2 \tag{5}$$

where $\mathcal{B}$ represents a batch of samples selected from the replay buffer. The target value y_i is computed as follows:

$$y_i = r_i + \gamma \max_{\boldsymbol{a}'} \min_{j=1,2} Q_{\boldsymbol{w}'_j}(\boldsymbol{s}'_i,\boldsymbol{a}',\boldsymbol{z}_i). \tag{6}$$

Here, the value of $\max_{\boldsymbol{a}'} Q_{\boldsymbol{w}'_j}(\boldsymbol{s}',\boldsymbol{a}',\boldsymbol{z}_i)$ is approximated using the CEM. When selecting an action $\boldsymbol{a}$ for a given state $\boldsymbol{s}$, the action is determined as follows:

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} \min_{j=1,2} Q_{\boldsymbol{w}_j}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z}). \tag{7}$$

In the standard CEM, N samples are randomly generated using the sampling distribution at each iteration and the sampling distribution is fit to the best M samples. As in [5], we use the sample with the highest score as the output of the CEM instead of the mean of the best M samples. The neural network architecture in our implementation is shown in Fig. 2.

B. Adversarial Domain Sampling

In our excavation task, there is a variance in the task difficulty in a given task distribution. Thus, a uniform sampling of the domain parameter does not result in satisfactory performance from the policy. To address this issue, we sample a domain parameter that minimizes the expected Q-value during the training phase. In the proposed method, the domain parameter $\boldsymbol{z}$ is determined as follows:

$$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}} \sum_{\boldsymbol{s}\in\mathcal{B}} \max_{\boldsymbol{a}} \min_{j=1,2} Q_{\boldsymbol{w}_j}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z}), \tag{8}$$

where $\mathcal{B}$ is a batch of samples selected from the replay buffer. This technique enables us to perform trials and errors under domain parameters for which the current policy does not work well. Therefore, this technique allows us to improve the worst case performance. Considering a case where a policy π generates an action by following (7), our approach can be seen as a way of approximately solving a problem expressed as follows:

$$\max_{\pi} \min_{\tilde{p}(\boldsymbol{z})} \mathbb{E}_{\boldsymbol{z}\sim\tilde{p}(\boldsymbol{z})} \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a})\sim\pi, p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a};\boldsymbol{z})} \left[Q^{\pi}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z}) \right]. \tag{9}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

This formulation implies that our approach can be viewed as a type of adversarial training with the max-min game [23], i.e., the agent attempts to maximize the expected return and the domain sampler attempts to minimize the expected return. The proposed approach can also be regarded as generating an automatic curriculum [26] based on the current estimate of the Q-values.

In practice, the domain parameter $\boldsymbol{z}$ is either continuous or discrete, and analytically computing $\min_{\boldsymbol{z}} \sum_{\boldsymbol{s}\in\mathcal{B}} \max_{\boldsymbol{a}} Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})$ is not tractable. Thus, we uniformly generate L samples of the domain parameter $\boldsymbol{z}$ and select the value leading to the minimum value of $\sum_{\boldsymbol{s}\in\mathcal{B}} \max_{\boldsymbol{a}} Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})$. To maintain the balance between exploration and exploitation during the training phase, we use this strategy in an ϵ-greedy-like fashion; we sample the domain parameter $\boldsymbol{z}$ uniformly with the probability ϵ; otherwise, the domain parameter $\boldsymbol{z}$ is determined based on (8). In our implementation, the domain parameter is sampled in the beginning of an episode and fixed until the end of the episode.

C. Training of Domain Parameter Estimator with Sensitivity-Aware Importance Weight

In our problem setting, the value of the domain parameter $\boldsymbol{z}$ is known during the training phase, but unknown in the test phase. Thus, we need to estimate the domain parameter $\boldsymbol{z}$ from the observed state-action pairs in the test phase. To this end, it is necessary to obtain a model for estimating the domain parameter. By denoting the model for estimating the domain parameter by $\boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{s}'_i)$ as parameterized by a vector $\boldsymbol{\phi}$, the model can be trained by minimizing the mean squared error as follows:

$$\mathcal{L}(\boldsymbol{\phi}) = \sum_{(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{s}'_i,\boldsymbol{z}_i)\in\mathcal{B}'} \left\|\boldsymbol{z}_i - \boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{s}'_i)\right\|_2^2, \tag{10}$$

where $\mathcal{B}'$ is a batch of samples selected from the latest N samples in the replay buffer. As the policy π changes over the training phase, the distribution over the transitions $(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ in the replay buffer differs from that induced by the current policy. To mitigate the covariate shift between the training and test data distributions, we select a batch of samples from the latest M samples instead of the entire replay buffer when minimizing the loss in (10). In our implementation, we set $M = 10,000$.

In our preliminary experiment, we found that the estimation of the domain parameter was the bottleneck for achieving satisfactory performance. Thus, we propose training the domain parameter estimator $\boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{s}'_i)$ by using an importance weight incorporating the sensitivity of the Q-value to the domain parameter. In our method, the domain parameter estimator is trained by minimizing the objective function as follows:

$$\mathcal{L}(\boldsymbol{\phi}) = \sum_{(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{s}_i,\boldsymbol{z}_i)\in\mathcal{B}'} w_i \left\|\boldsymbol{z}_i - \boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{s}_i,\boldsymbol{a}_i,\boldsymbol{s}'_i)\right\|_2^2, \tag{11}$$

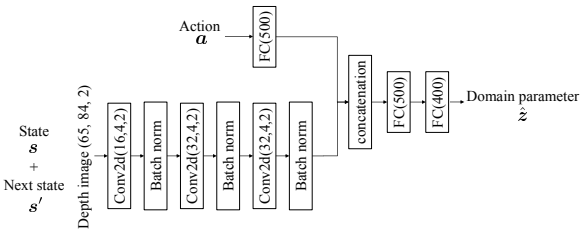


Fig. 3. Neural network architecture for $f_{\boldsymbol{\phi}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$.

The corresponding calculation is as follows:

$$w_i = \min(w_{\max}, \max(\bar{w}_i, w_{\min}), \tag{12}$$

$$\bar{w}_i = \frac{\left\|\nabla_{\boldsymbol{z}} Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})|_{\boldsymbol{s}=\boldsymbol{s}_i,\boldsymbol{a}=\boldsymbol{a}_i,\boldsymbol{z}=\boldsymbol{z}_i}\right\|_2}{\sum_{(\boldsymbol{s}_j,\boldsymbol{a}_j,\boldsymbol{z}_j)\in\mathcal{B}} \left\|\nabla_{\boldsymbol{z}} Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})|_{\boldsymbol{s}=\boldsymbol{s}_j,\boldsymbol{a}=\boldsymbol{a}_j,\boldsymbol{z}=\boldsymbol{z}_j}\right\|_2}. \tag{13}$$

Herein, $w_{\max}$ and $w_{\min}$ are the upper and lower bound for clipping the importance weight, respectively. In our method, we use the L2-norm of the gradient of the Q-value with respect to the domain parameter $\|\nabla_{\boldsymbol{z}} Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})\|_2$ as a metric for quantifying the sensitivity of the Q-value to the domain parameter. When the norm of the gradient $\nabla_{\boldsymbol{z}} Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})$ is large at a point, the Q-value is considered to be sensitive to the value of the domain parameter z at that point and accurately estimating the value of $\boldsymbol{z}$ is essential. We can incorporate this observation using the importance weight in (13). As in other techniques with importance weights [27], we clip the importance weight. In our implementation, we use $w_{\min} = 0.5$ and $w_{\max} = 3.0$.

D. Proposed Algorithm

The proposed training procedure is summarized in Algorithm 1. The domain parameter $\boldsymbol{z}$ is sampled in the beginning of the episode and is known to the RL agent in the training phase. In the beginning of the training, the domain parameters are uniformly sampled. The adversarial domain sampling strategy described in Section IV-B is used after N_{adapt} time steps. The model to estimate the domain parameter is trained in a supervised manner using the importance weight proposed in (13).

The procedure for deploying the policy trained by Algorithm 1 is summarized in Algorithm 2. The domain parameter $\boldsymbol{z}$ is sampled in the beginning of the episode; however, its true value is unknown to the RL agent. The domain parameter is estimated from an observed transition $(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ using the domain parameter estimator $\boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$. Subsequently, the action that maximizes the Q-value $Q_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{z})$ is estimated using the CEM.

V. Experiments

A. Simulation setup

We used a 3D excavation simulator developed by Komatsu Ltd., as described in [5]. In the simulation, each action represented a single trajectory of the excavator bucket. We assumed that a controller for the excavator arm was provided.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

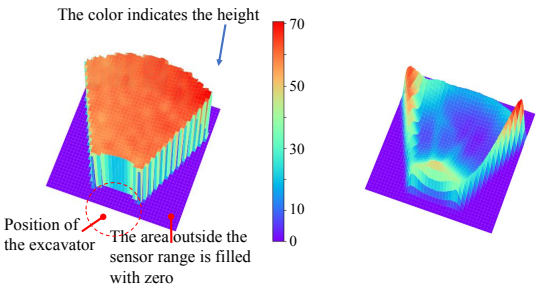
Algorithm 1 Meta-QT-Opt with Adversarial Domain Sampling and Sensitivity-Aware Importance Weight

Input: ϵ_a for the ϵ-greedy action exploration, ϵ_d for the ϵ-greedy exploration of the domain parameter, time steps to activate the adversarial domain sampling N_{adapt}
Initialize the experience replay buffer $\mathcal{D}$, the parameters for the domain parameter estimator $\boldsymbol{\phi}$, and the parameters for critics and target critics $\boldsymbol{w}_i$, $\boldsymbol{w}'_i$ for $i = 1, 2$.
Set the number of samples in $\mathcal{D}$ as $N = 0$
for each episode **do**
  Generate random value $x \in [0, 1]$
  if $x < \epsilon_d$ and $N > N_{\text{adapt}}$ **then**
    Sample the domain parameter randomly
  else
    Select the domain parameter using the strategy in (8)
  end if
  for $t = 0$ **to** T **do**
    generate random value $x_a \in [0, 1]$
    if $x_a < \epsilon_a$ **then**
      Select action randomly
    else
      Select action that maximizes $Q_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z})$
    end if
    Observe reward r and new state $\boldsymbol{s}'$
    Store tuple $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}', r)$ in $\mathcal{D}$
    $N \leftarrow N + 1$
    Sample mini-batch $\mathcal{B}$ from $\mathcal{D}$
    Update the critics by minimizing $\mathcal{L}_Q(\boldsymbol{w}_i)$ in (5)
    Update the target critics by $\boldsymbol{w}'_i \leftarrow (1 - \tau)\boldsymbol{w}'_i + \tau \boldsymbol{w}_i$
    Sample mini-batch $\mathcal{B}'$ from the latest N samples
    Update $\boldsymbol{\phi}$ by minimizing $\mathcal{L}(\boldsymbol{\phi})$ in (11)
  end for
end for

Algorithm 2 Meta-QT-Opt Meta-Testing

Input: Domain distribution for testing $p_{\text{test}}(\boldsymbol{z})$
for each episode **do**
  Sample domain parameter $\boldsymbol{z} \sim p_{\text{test}}(\boldsymbol{z})$
  for $t = 0$ **to** T **do**
    if $t = 0$ **then**
      Set $\hat{\boldsymbol{z}} = 0$
    else
      Estimate the domain parameter $\hat{\boldsymbol{z}} = \boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$
    end if
    Select action: $\boldsymbol{a}^* = \arg\max Q_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{a}, \hat{\boldsymbol{z}})$
    Observe reward r and new state $\boldsymbol{s}'$
  end for
end for

This policy generated a trajectory for a specified controller. The bucket trajectory was approximated as an arc, and the action corresponded to the bucket trajectory parameter. The arc trajectory was parameterized with a three-dimensional vector, which was used as the action. The state was provided by a depth image capturing the landscape in front of the excavator. The number of observations was 65 $\times$ 84, and we used only the depth information. An example of a depth image is shown in Fig. 4. The range of the sensor was presented in sector form, and the shape of the state was arranged in a rectangle by filling the blank region with zeros. The reward was based on the amount of soil removed by the previous action. For example, if the amount of soil removed by the previous action is 0.5 m^3, then the reward is 0.5. The episode was terminated when the amount of soil removed by the previous action was less than a predefined threshold, i.e., when the bucket of the excavator was almost empty. The dimensionality of the domain parameters was nine. In our simulation, to represent the stochastic behavior of soil, the form of the remaining soil was collapsed stochastically. Parameters that define the degree and probability of soil collapse were included in the domain parameters.



(a) Example of the initial state (b) Example of the goal state

Fig. 4. Initial and goal states of the excavation task. An initial state is randomly generated in the beginning of the episode. The state is represented as a depth map of the landscape.

TABLE I
CHARACTERISTICS OF SOILS IN TEST DOMAINS

Domain name	Hardness	Crumbliness
Test Domain 1	Hard	Solid
Test Domain 2	Hard	Crumbly
Test Domain 3	Soft	Solid
Test Domain 4	Soft	Crumbly

To evaluate the performance of the trained policy, we used four sets of typical domain parameters within their respective ranges. The characteristics of the four test domains are summarized in Table I. In general, when the soil is hard, an excavator cannot dig deeply. When the soils are crumbly, the remaining soil in the target area can easily collapse and tends to be relatively smooth. When the soil is hard, the excavator bucket cannot dig deeply. Therefore, the amount of soil that can be removed by a single action is less when the soil is harder. Consequently, when the soil is harder, more processes are required to completely remove the target amount of soil. When more processes are required to complete the task, the estimation of the Q-function becomes more challenging because more time steps must be considered to accurately predict the return. Additionally, the soil collapses stochastically. If the probability of soil collapse is high, then the prediction of return becomes uncertain, which can result in more challenging tasks. In addition to these four test domains, we evaluated the average return over various domains by uniformly sampling the domain parameters.

To demonstrate the differences in the task difficulty, Fig. 5

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

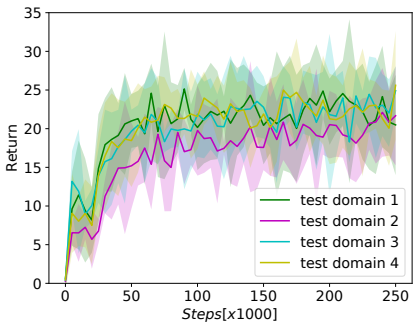


Fig. 5. The learning curve of QT-Opt when a policy is directly trained on each test domain. Results on four test domains are shown, respectively. There is a variance of the task difficulty among these four domains.

depicts the learning curves of QT-Opt on each test domain. In Fig. 5, a policy is trained on a single domain, unlike in meta-RL and multitask RL. As shown, learning a policy on test domain 2 requires more iterations of trial-and-error than in the other test domains. This result demonstrates the variance in the task difficulty in a given task distribution. Although the variance of task difficulty was insignificant, it significantly affected our task, as will be discussed later.

As a baseline, we used the variant of QT-Opt proposed in [5], which employed conservative adversarial training to reduce the overestimation of the Q-values. We evaluated a variant of QT-Opt in [5] with domain randomization. Below, we refer to this baseline method as QT-Opt with domain randomization. In addition, we evaluated a simplified version of PEARL proposed in [11]. In PEARL, the Q-function conditioned on the latent variable $Q_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z})$ is estimated, and a latent-conditioned policy $\pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ is learned using SAC as a base RL algorithm. Although the original PEARL learns the latent variable in an unsupervised manner, we considered with the domain parameters as the latent variable in PEARL, and the posterior distribution was trained in a supervised-learning manner. Thus, we refer to this variant of PEARL as "simple-PEARL."

B. Results

Fig. 6 shows the learning curves for the baseline methods and proposed method. We report the average return over five random seeds with 30 test episodes. In Fig. 6, the shaded area represents the 90 % confidence region. Notably, policies trained with the baseline methods do not work in test domain 2 even if the average performance across the tasks is acceptable. In QT-Opt with domain randomization, simple-PEARL, and meta-QT-Opt, the performance on test domain 2 evidently does not improve appropriately during the training. This result shows that uniform domain sampling, as often used in previous studies, is not effective when there is significant variance in the task difficulty among a given task distribution. In contrast, when using the proposed adversarial domain sampling technique, meta-QT-Opt significantly outperformed baseline methods. Our proposed technique successfully addresses the above issue by actively sampling the domains where the expected return is low. Interestingly, our adversarial domain sampling improves the overall performance across various domains as well as the performance on the most challenging domain. In addition, the performance was further improved using a sensitivity-aware importance weight to train the posterior distribution. As shown in Fig. 6(f), the mean squared errors of the domain parameter estimated by the proposed method with and without importance sampling did not differ significantly. The proposed importance-sampling method aims to assign a higher weight to the estimation accuracy at the point where the Q-function is more sensitive to domain parameters. Thus, using importance weighting does not significantly affect the overall estimation error. However, as shown in Table II, the proposed importance sampling technique improved the policy performance.

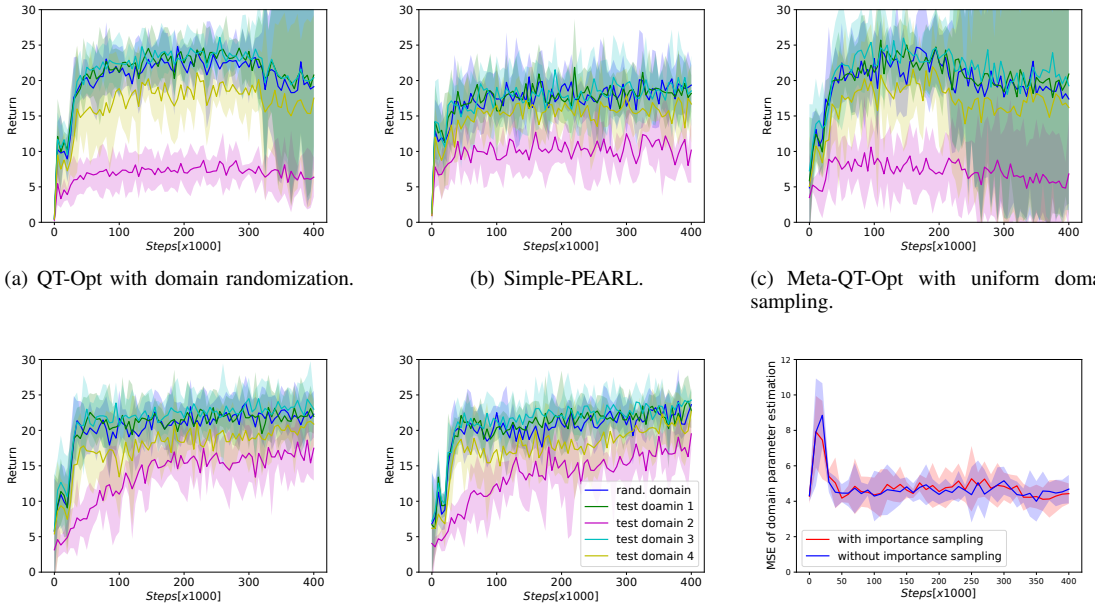
The performances of the baseline methods deteriorated as the training progressed. A recent study [28] showed that Q-learning-based algorithms typically result in performance deterioration, which is attributable to unsatisfactory feature learning in Q-learning based on bootstrapping. Based on the results, the proposed algorithm can mitigate the abovementioned issue; however, further investigation is required. Table II summarizes the performances of the baseline and proposed methods after training. The proposed method significantly outperforms the baseline methods across the test domains. The difference between the baseline methods and proposed method is especially evident on test domain 2. Although the policies trained by the baseline methods do not work well on test domain 2, the proposed method successfully trains a policy that works on test domain 2 as well as on the other domains.

Fig. 7 visualizes the Q-function with different values of the domain parameter. For the visualization, we show $\max_{a_3} Q_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z})$ as the action is three-dimensional in our excavation task. We show a depth image used as state $\boldsymbol{s}$ in Fig. 7 (a). The plots of $\max_{a_3} Q_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z})$ with the values of the domain parameter that correspond to the domain 1, domain 2, domain 3, and domain 4 are shown in Fig. 7(b)-(e), respectively. Different scales are used in Fig. 7(b)-(e) to visualize the landscape of the Q-function. As shown, the outputs of the trained model of $Q_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z})$ change according to the value of the domain parameter. The dense red area corresponds to the distribution of near optimal actions in Fig. 7 and the location of the dense red area is significantly different between test domaion 2 and the other test domains. This result implies that different strategies were obtained for different domains using the proposed method.

Fig. 8 shows the values of the domain parameter sampled during the training with the proposed adaptive sampling technique. For the visualization, the values of the domain parameters are normalized to the range $[-1, 1]$. Among the domain parameters, z_4, z_5, z_6, and z_7 are discrete variables. In this study, the domain parameter was sampled uniformly in the beginning of the training and the adversarial domain sampling was activated after 50,000 time steps, i.e., after approximately 800 episodes in Fig. 8. In the top row of Fig. 8, the values near $z_0 = -1$ and $z_2 = 1$ are intensively sampled after the adversarial sampling strategy is activated,implying that these parameters significantly affect the task difficulty. In test domain 2, which we found was the most challenging

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



(a) QT-Opt with domain randomization. (b) Simple-PEARL. (c) Meta-QT-Opt with uniform domain sampling.

(d) Meta-QT-Opt with adversarial domain sampling. (e) Meta-QT-Opt with adversarial domain sampling + importance weight (IW). (f) Accuracy of domain parameter estimation.

Fig. 6. Learning curves of the proposed and baseline methods. (a)-(e) show the average returns of each method during training. The proposed method, Meta-QT-Opt + adaptive sampling + IW, achieved the best performance in all of the test domains. (f) shows the accuracy of domain parameter estimation.

TABLE II
PERFORMANCE ON TEST DOMAINS AFTER 400,000 TRAINING STEPS.

Domain name	QT-Opt with Domain rand.	Simple-PEARL	Meta-QT-Opt	Meta-QT-Opt + adv. sampling	Meta-QT-Opt + adv. sampling + IW
Test domain 1	20.8±5.1	18.2±1.3	20.9± 10.3	22.0±1.2	**22.3±1.6**
Test domain 2	6.4±1.4	10.2±2.3	6.8±3.9	17.5±2.3	**19.5±1.0**
Test domain 3	20.4±5.7	18.5±2.5	19.3±8.9	23.0±1.0	**23.5±1.8**
Test domain 4	17.5±6.0	16.5±2.8	16.2±7.0	**20.9±1.6**	**22.7±1.4**
Random Domain	19.1±5.9	19.4± 1.4	17.4± 7.5	**22.3±1.7**	**23.1±1.2**

domain, the values of these domain parameters are $z_0 = -1$ and $z_2 = 1$. This result indicates that the proposed method actively samples the domain parameters leading to difficult tasks. Therefore, we conclude that our approach successfully improves the worst case performance.

VI. CONCLUSIONS

In this study, we investigated a learning framework for planning a trajectory for automating an excavator in various domains. Although prior works on meta-RL and multitask RL often do not pay attention to the variance in the task difficulty among a given task distribution, we empirically showed that it is important to consider the variance in the task difficulty. We proposed two techniques for improving the performance of an RL method in our problem setting: adversarial domain sampling and domain parameter estimation with a sensitivity-aware importance weight. Our adversarial domain sampling technique can be regarded as a type of the adversarial training that can help improve the worst-case performance. We empirically showed that the proposed method significantly outperformed baseline methods on the excavation task. To use the proposed framework in practical applications, one must address the remaining challenges. For example, the robustness of the policy against observational noise must be guaranteed. In reality, depth images obtained from depth sensors typically contain non-negligible noise. We will address these remaining issues in our future studies.

ACKNOWLEDGMENT

This work was partially supported by JST AIP Acceleration Research JPMJCR20U3, Moonshot R&D Grant Number JPMJPS2011, CREST Grant Number JPMJCR2015, JSPS KAKENHI Grant Number JP19H01115, JP19K20370.

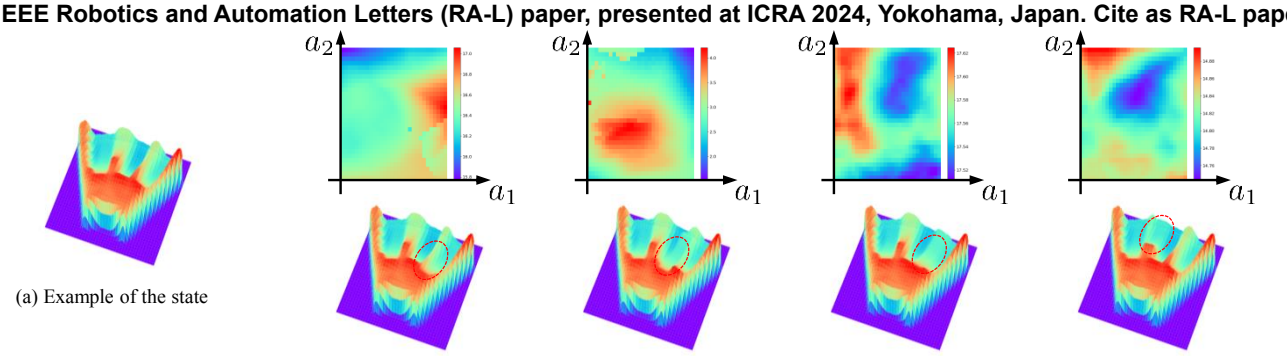
REFERENCES

[1] X. Shi, P. Lever, and F.-Y. Wang, "Experimental robotic excavation with fuzzy logic and neural networks," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, 1996, pp. 957–962 vol.1.

[2] A. Stentz, J. Bares, S. Singh, and P. Rowe, "Robotic excavator forautonomous truck loading," *Autonomous robots*, vol. 7, pp. 175–186, 1999.

[3] G. J. Maeda, D. C. Rye, and S. P. N. Singh, "Iterative autonomous excavation," *Field and Service Robotics*, pp. 369–382, 2014.

[4] R. Fukui, T. Niho, M. Nakao, and M. Uetake, "Imitation-based control of automated ore excavator: improvement of autonomous excavation database quality using clustering andassociation analysis processes," *Advanced Robotics*, vol. 31, no. 11, pp. 595–606, 2017.



(a) Example of the state

(b) Q-function (top) and the resulting action (bottom) for test domain 1

(c) Q-function (top) and the resulting action (bottom) for test domain 2

(d) Q-function (top) and the resulting action (bottom) for test domain 3

(e) Q-function (top) and the resulting action (bottom) for test domain 4

Fig. 7. Visualization of Q-function and action corresponding to state in (a) with different domain parameters.

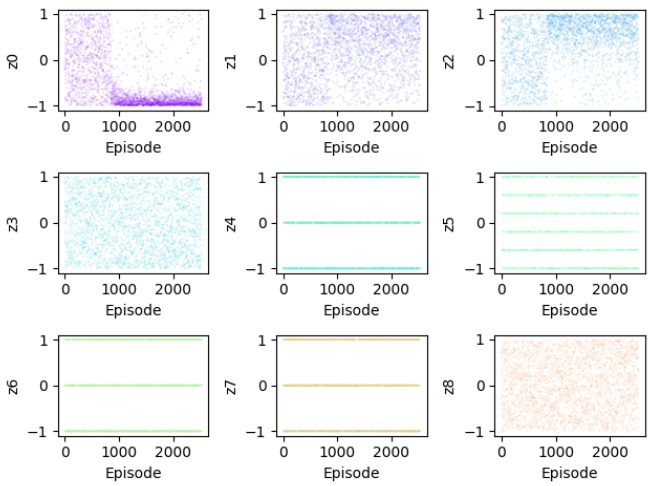


Fig. 8. Visualization of domain parameters sampled using the proposed adversarial domain sampling.

[5] T. Osa and M. Aizawa, "Deep reinforcement learning with adversarial training for automated excavation using depth images," *IEEE Access*, vol. 10, pp. 4523–4535, 2022.

[6] B. J. Hodel, "Learning to operate an excavator via policy optimization," *Procedia Computer Science*, vol. 140, pp. 376–382, 2018.

[7] P. Egli, D. Gaschen, S. Kerscher, D. Jud, and M. Hutter, "Soil-adaptive excavation using reinforcement learning," *IEEE Robotics and Automation Letters*, pp. 1–8, 2022.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[9] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Scaling up multi-task robotic reinforcement learning," in *Proceedings of the 5th Conference on Robot Learning*, ser. 164, 2022, pp. 557–575.

[10] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 70, 2017, pp. 1126–1135.

[11] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 97, 2019, pp. 5331–5340.

[12] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," in *Procedings of Robotics: Science and Systems*, 2021.

[13] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.

[14] C. Bodnar, A. Li, K. Hausman, P. Pastor, and M. Kalakrishnan, "Quantile qt-opt for risk-awarevision-based robotic grasping," in *Robotics and Science and Systems*, 2020.

[15] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *Proceedings of the Conference on Robot Learning*, 2018.

[16] P. T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, pp. 19–67, 2005.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[18] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 1587–1596. [Online]. Available: http://proceedings.mlr.press/v80/fujimoto18a.html

[19] Y. Yang, P. Long, X. Song, J. Pan, and L. Zhang, "Optimization-based framework for excavation trajectory generation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1479–1486, 2021.

[20] Q. Lu and L. Zhang, "Excavation learning for rigid objects in clutter," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7373–7380, 2021.

[21] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[22] J. Bose, G. Gidel, H. Berard, A. Cianflone, P. Vincent, S. Lacoste-Julien, and W. Hamilton, "Adversarial example games," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[23] J. Wang, T. Zhang, S. Liu, P.-Y. Chen, J. Xu, M. Fardad, and B. Li, "Adversarial attack generation empowered by min-max optimization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, and A. K. Fidjeland, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[25] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[26] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv*, 2017, arXiv:1707.06347.

[28] A. Kumar, R. Agarwal, D. Ghosh, and S. Levine, "Implicit under-parameterization inhibits data-efficient deep reinforcement learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.