Sim-to-Real Learning of Compliant Bipedal Locomotion on Torque Sensor-Less Gear-Driven Humanoid

Shimpei Masuda, Kuniyuki Takahashi

Abstract—Sim-to-real is a mainstream method to cope with the large number of trials needed by typical deep reinforcement learning. However, transferring a policy trained in simulation to actual hardware remains challenging due to the reality gap. In particular, the characteristics of actuators in legged robots have a considerable influence on sim-to-real transfer. High reduction ratio gears are widely used in actuators, and the reality gap issue becomes especially pronounced when even the utilization of backdrivability is considered to control joints compliantly. We propose a new simulation model of gears to address this gap. Additionally, the difficulty in achieving stable bipedal locomotion causes typical methods to fail to tune physical parameters in simulation with the behavior of transferred policy. Thus, we propose a method for system identification that can utilize failed attempts. The method’s effectiveness is verified using a biped robot, the ROBOTIS-OP3, and the sim-to-real transferred policy can stabilize the robot under severe disturbances and walk on uneven surfaces without force and torque sensors.

I. INTRODUCTION

If robots can achieve stable bipedal walking, they can traverse various environments, such as slopes, uneven roads, and stairs. To achieve this mission, robots need to be robust against disturbances and changes in environments. As approaches to improve the robustness of legged robots, research based on torque control and the use of compliance has been popular in recent years [1]–[3]. On the other hand, robots that require high-power joints, such as full-scale humanoids, are often equipped with gears with high reduction ratios to provide output torque while downsizing the actuators. In addition, since installing a torque sensor increases the cost of the actuator, a configuration that uses a high reduction ratio gear without a joint torque sensor is still a popular option [4]–[6]. Such actuators have elements that are difficult to model, primarily represented by gear friction. Due to this friction, the ideal torque output may not always be achieved if using the joint-level feed-forward compensation. Generally, it is not easy to manually design a controller that can work with these considerations.

Deep reinforcement learning (DRL) is being established as a method for developing high-performance robot controllers [7]–[9]. Since DRL requires many trials, the sim-to-real approach, in which policies trained using simulations are transferred to a real-world robot, has been attracting attention [10]–[12]. In addition, simulation training is considered more promising because it can add various disturbances and try unusual cases without the risk of breaking robots. However, there are differences in behavior between the simulation and the real environment, commonly referred to as reality gap, which prevents the policies trained on the simulation from working in the real environment. Especially for legged robots, it has been reported that the model of the actuators has a considerable influence on the reality gap [12]–[14]. To address this challenge, methods to construct a realistic simulation by identifying the simulation parameters of dynamics reproducing actual robot behavior are studied. However, the instability of the bipedal locomotion causes the sim-to-real transfer to fail catastrophically. It makes system identification with the behavior of transferred policy challenging.

In this study, we propose 1) an actuator model featuring the high reduction ratio gears for physics simulation and 2) a new system identification method that utilizes the failure experience when a sim-to-real transfer fails and realizes robust bipedal locomotion, including balancing and walking, using DRL.

II. RELATED WORK

A. Joint control in RL for legged robots

In recent years, many studies on reinforcement learning for legged robots have been reported, and many of them have achieved high performance on real robots [10], [14], [15]. These studies often employed a position controller in each joint, and RL policy commands target angles. One of the reasons is that position control improves learning efficiency and task performance compared to torque or velocity control [16]. On the other hand, some studies have been conducted to improve robustness to disturbances and environmental recognition errors by exploiting passivity (in other words, utilizing the joint backdrivability) through...
compliant control [1]–[3]. Currently, these studies mainly focus on the robot with torque-controlled joints. So, utilizing backdrivability by sim-to-real RL is not a well-explored area, especially in robots with gear-driven and torque sensor-less joints. In this study, we focus on it and address some difficulties described following subsections.

B. Sim-to-real RL for Robotics

It is widely known that controllers trained by RL in simulations do not work well in actual robots due to the reality gap, and various approaches have been developed to overcome this challenge. Domain randomization [17] is a frequently used approach for the reality gap. The method trains RL policy in many simulations that randomized dynamics or sensory appearances, such as joint friction or textures of the object. Domain randomization can increase the robustness of some domain gaps, and it tends to successful sim-to-real transfer. However, specifying appropriate randomization distribution is a difficult problem. Too large randomization often makes RL policy too conservative, and performance becomes low. Another approach to address the reality gap is precisely reproducing the real behavior on the simulation [11], [18]. Construct a realistic simulation, then train RL on that. Moreover, it often combines small domain randomization. Assuming that simulation is characterized by some dynamics model parameters, such as the size of joint friction, reproducing the real behavior corresponds to finding parameter values or distributions. In this paper, we call this system identification. Researches along this approach, we can roughly categorize them into 1) manual system identification [10], [14], 2) system identification based on collected task irrelevant data [12], [13], 3) system identification based on rolled out RL policy behavior [11], [18]. Since focusing on frictional gear-driven joint and utilization of backdrivability, we combine 2) and 3). On the focus, the behavior of interaction with the environment is important, so 1) or 2) approach may require a large amount of data. Moreover, tasks with instability, such as bipedal walking, where a slight difference in action can easily make a big difference in the future compared to tasks with arms fixed to a base. In such case, precise identification is challenging by 3) with trajectory comparison [18] or parameter prediction [11]. In this paper, we use both 2) and 3) complementary approaches. It allows relatively small data collection and system identification using the failing behavior of the rolled-out RL policy.

C. Modeling Actuators in Sim-to-Real Learning

The reality gap is especially critical for actuators in legged robots. Several kinds of research have been conducted to improve the actuator models in the simulation, broadly classified into two categories to address the challenge.

1) Use a neural network for the actuator model and train with largely collected data [12], [13] (Network training corresponds to system identification).

2) Design a detailed model of the actuator and implement it in the simulation [10], [19]

The first method has the potential to build highly accurate models but requires large amounts of data. In this study, we extend category 2) to model actuators with high reduction ratio gears with low-dimensional parameters. It is expected that parameters can be obtained with relatively small amounts of data. In addition, since high reduction ratio gears are a major characteristic, the model can be applied to many actuators.

III. PRELIMINARY OF RL

Let \((S, A, T, R, \gamma, s_0)\) be a Markov decision process with state \(s \in S\), action \(a \in A\), transition probability \(T(s_{t+1}|s, a)\), reward function \(R(s, a) \in \mathbb{R}\), discount factor \(\gamma \in [0, 1]\), and initial state distribution \(s_0\). RL typically learns a policy \(\pi : S \rightarrow A\) to maximize the expected discounted accumulated reward \(J = \mathbb{E}[R_t]\) from the initial state, where \(R_t = \sum_{i=t}^{T} \gamma^{(i-t)} R(s_i, a_i)\). This time, the state transition is a transition from control timing \(t\) to \(t + 1\) in a physical simulation or the real world. \(S\) corresponds to the sensor data and \(A\) to the commands to the joint actuators. Although various RL algorithms have been proposed, in this study, we use Soft-Actor-Critic [20], which supports continuous action spaces and has been reported to have high performance.

IV. Method

The proposed sim-to-real learning method consists of two main components, the first is improved robot simulation featuring an actuator model, and the second is a system identification method to specify the simulation parameters for RL training. The method consists of the following three phases. Phase 1: Perform first system identification with exciting motion, such as squatting motion. Phase 2: Using the identified parameters, train the RL policy using simulation, then run the trained policy on the actual robot. Phase 3: If the robot does not work well, re-identify the simulation parameters using the failure experience. Phase 2 is then executed again.

We first describe the proposed actuator model in section IV-A. Then, the first system identification is described in section IV-B.1 followed by the system re-identification in section IV-B.2.

A. Actuator Modeling

This section describes our actuator model. The actuator consists mainly of a DC motor (Section IV-A.1) and gears (Section IV-A.2), as illustrated in Fig. 2. The actuator is controlled by variable gain PD, described in Section IV-A.3. In this study, we implement this actuator model on the physics engine Mujoco [21], and some descriptions assume that, but note that it can be implemented in other physics engines.
\textbf{1) DC Motor Model:} For the DC motor model, we use the commonly used model \cite{10}, which is represented by the following equation:
\begin{equation}
\tau = K_i I
\end{equation}
\begin{equation}
I = \frac{V_{pwm} - V_{back-emf}}{R}
\end{equation}
\begin{equation}
V_{back-emf} = K_i \dot{q}
\end{equation}
where \( \tau \) is the exerted torque by the motor, \( I \) is the current, \( K_i \) is the torque constant of the motor, \( R \) is the terminal resistance, \( V_{pwm} \) is the voltage applied to the motor, \( V_{back-emf} \) is the back electromotive force (EMF) voltage, and \( \dot{q} \) is the angular velocity of the shaft.

A current controller controls voltage to drive the DC motor with the target current. The applied voltage \( V_{pwm} \) at each simulation time step can be formulated by the following equation:
\begin{equation}
\begin{align}
V_{pwm}^* &= RI_{target} + V_{back-emf} \\
V_{pwm} &= \min(\max(V_{pwm}^*, -V_{battery}), V_{battery})
\end{align}
\end{equation}
where \( I_{target} \) is the target current and \( V_{battery} \) is battery voltage.

\textbf{2) Gear Model with Directional Transmission Efficiency:} The gear model considers the transmission efficiency and load-independent friction loss. The former depends on the force, while the latter mainly depends on the angular velocity.

For high reduction ratio gears, there is an asymmetry in transmission efficiency depending on the drive direction \cite{22, 23}. This work introduces this directional transmission efficiency (DTE) and proposes an implementation method for existing physics engines.

The method treats the force loss due to transmission efficiency as an additional brake torque applied to the joint in each simulation time step. The brake torque is calculated from the generated torque by the motor, load torque, and the forward and backward transmission efficiencies. Assuming that the simulation time step is small enough for the target dynamics, the brake torque is calculated by the following equation:
\begin{equation}
\tau_{brake}(\tau_m, \tau_a, \eta_{fw}, \eta_{bw}) =
\begin{cases}
-L_{fw} \text{ if } \text{sign}(\eta_{fw}\tau_m + \tau_a) = \text{sign}(\tau_m) \\
-L_{bw} \text{ if } \text{sign}(\tau_m + \eta_{bw}\tau_m) = \text{sign}(\tau_m) \\
-(\tau_m + \tau_a) \text{ else }
\end{cases}
\end{equation}
where, \( L_{fw} = (1 - \eta_{fw})\tau_m \), \( L_{bw} = (1 - \eta_{bw})\tau_a \)
where \( \tau_m \) is the generated torque from the motor, \( \tau_a \) is the load torque on the joint, and \( \eta_{fw} \) and \( \eta_{bw} \) are the forward and backward transmission efficiencies, respectively.

\( \tau_m = r_{gear} \tau \) where \( r_{gear} \) is gear reduction ratio. When \( \tau_m \) is clearly greater than \( \tau_a \), or the directions of \( \tau_m \) and \( \tau_a \) are the same, it is regarded as a forward drive state, and the brake torque based on the \( \eta_{fw} \) is generated. If the directions of \( \tau_m \) and \( \tau_a \) are opposite and \( \tau_a \) is larger even after considering the \( \eta_{bw} \), it is considered to be in the backward drive state, and the brake torque based on the \( \eta_{bw} \) is generated. A state that does not meet either conditions is an antagonistic state with no apparent difference between \( \tau_m \) and \( \tau_a \), and thus generates a brake torque, such that \( \tau_m + \tau_a + \tau_{brake} = 0 \). Since \( \tau_a \) is approximated using the value of the current simulation state, there is a concern that the simulation may become unstable. However, it is expected to be stable when used in conjunction with the load-independent friction as below. The load-independent friction loss generates torque that cancels joint torque in the absolute upper bound. In particular, MuJoCo will handle it as a constraint and allow the constant value. So we calculate the friction loss value at each time step to generate approximated static and viscous friction. For the calculation, Stribeck function \( \tau_{fric} \) is used:
\begin{equation}
s = \frac{\exp(|\dot{q}|)}{\eta_{static}} \\
\tau_{fric} = f_c + s(f_s - f_c) + k_v|\dot{q}|
\end{equation}
where each parameter \( f_s, f_c, k_v, \) and \( \dot{q}_{static} \) are the static friction force, the Coulomb friction coefficient, the viscous friction coefficient, and the angular velocity value regarded as non-static, respectively.

\textbf{3) Variable Gain PD:} The actuator model described so far is controlled by the PD controller. We adopted this because previous research has reported that placing the position control at the lower level of the RL policy improves learning efficiency and performance. Generally, the PD controller is used with fixed gains, but in this study, the gains are also commanded with the target position. It allows a strategy to increase backdrivability while reducing target position tracking performance by specifying a smaller gain, depending on the task situation.

\textbf{B. System Identification for Sim-to-real}

This section describes the system identification method of the simulation parameters, such as the motor torque constant or frictions of the actuator model shown in Section \ref{V-A}.

\textbf{1) First System Identification:} Computing simulation parameters with black-box optimization \cite{12, 24} to match the observed results on the actual robot with the observed results on the simulation in similar motions is suitable for sim-to-real learning due to the following advantages: 1) The simulation can be improved from the results of the actual robot alone without detaching the actuator from the robot, and 2) Since optimizing the various parameters simultaneously, the behavior of the combination of each parameter can be matched to the actual robot.

First, a suitable exciting motion for system identification is designed as time series data \( \theta_{target} \) of the target angle of each joint. The exciting motion used in this study will be described in section \ref{V-B}. A simple position control feedback system is provided to track \( \theta_{target} \) in the actual robot. We then collect a set of sensor information \( O^{real} \) for the robot in motion. This sensor information includes the angle of each joint, angular velocity, current, and the tilt of the upper body. Similarly, on the simulation built based on the parameters \( \phi \) that includes parameters of body inertial and actuator model (actually used in this study are shown in Table \ref{table:parameters}), the same control strategy feedback system is used to perform the motion to track \( \theta_{target} \) and obtain \( O^{sim} \). The parameter
φ that minimizes the difference between \(O^{\text{real}}\) and \(O^{\text{sim}}\) is calculated by sampling-based black-box optimization as follows:

\[
\min_{\phi} L_{\text{exc}}(O^{\text{sim}}(\phi, \theta^{\text{targ}}), O^{\text{real}}(\theta^{\text{targ}}))
\]

where \(L_{\text{exc}}\) is an error function calculates distance between \(O^{\text{sim}}\) and \(O^{\text{real}}\).

2) System Re-Identification from Failure Experience:

Even if we use the simulation improved by the identification described in section [V-B.1] for RL training, the transferred policy sometimes does not work. For example, the actual robot may not even be able to stand on flat ground if a policy has been trained to balance against disturbances in a simulation. In such cases, task-relevant data is lacking in first system identification, and a possible solution is using data while transferred policy performing the task. The goal here is to find simulation parameters that reproduce the behavior of an actual robot not only for the excitation motion but also for the results of the policy rollout. However, in such a failure case, control commands by the policy become oscillatory, and the sensor information, such as joint angle trajectory, widely diverges. So, it is challenging to calculate the distance between simulation and actual policy behavior (policy gap) by comparison of sensor information for each time series in the same way as equation (5). Therefore, we introduce multi-objective optimization with the minimization of the excitation motion gap to achieve accurate identification while allowing for a rough calculation of the policy gap.

\[
\min_{\phi} L_{\text{exc}}(O^{\text{sim}}(\phi, \theta^{\text{targ}}), O^{\text{real}}(\theta^{\text{targ}}))
\]

\[
L_{\text{policy}}(O^{\text{policy}}(\phi, \pi), O^{\text{real}}(\theta^{\text{targ}}))
\]

\(L_{\text{policy}}\) is the policy gap function that is described later, \(O^{\text{policy}}\) and \(O^{\text{real}}\) are the sensor information observed in rollouts with the policy on the sim and real. We use the difference between cumulative rewards acquired in policy rollouts for the policy gap function

\[
L_{\text{policy}} = \frac{1}{T} \sum_{t=1}^{T} \sum_{k} R_p(O^{\text{policy}}(\theta^k_t), k, t) - \frac{1}{T} \sum_{t=1}^{T} \sum_{k} R_p(O^{\text{real}}(\theta^k_t), k, t),
\]

where \(T\) is the number of rollouts, \(t\) is the time index and \(R_p\) is the reward function for re-identification. The maximum length is set for \(t\) (we use 4 sec), and subsequent observations are not considered to prevent the variance of cumulative reward becomes too large. In this study, we use a proprioceptive reward function for \(R_p\), which can be calculated using only the robot’s sensor information and is extracted from the task’s reward function. \(R_p\) includes characteristics of the behavior, such as the time until fall based on survival rewards or joint angular velocity penalties, so we can assume that cumulative reward can be one feature.

Then, the parameters of the simulation in which the policy fails can be obtained while keeping the error in the exciting motion as small as possible. Note that if the problem is ideal, \(L_{\text{exc}}\) and \(L_{\text{policy}}\) can be optimal simultaneously, and multi-objective optimization is not needed. However, in the real problem, \(L_{\text{exc}}\) and \(L_{\text{policy}}\) are in a bit trade-off relationship, so multi-objective is suited for this method. Multi-objective optimization returns the Pareto-front solutions as a result. To decide the best from Pareto-fronts, we use simple heuristics that select the solution (simulation parameters) with a minimum \(L_{\text{exc}}\) value while \(L_{\text{policy}}\) is under the threshold based on the average of \(L_{\text{policy}}\) values in whole searched solutions.

V. EXPERIMENTAL SETUP

A. Robot Setup: ROBOTIS-OP3

In this study, we use ROBOTIS-OP3 [25]. ROBOTIS-OP3 has a length of 51 cm, a weight of about 3.5 Kg, 6-DOFs in each leg, and 20-DOFs in total. The control is executed at 125 Hz.

All the joints of the robot are composed of Dynamixel servo motors XM430-W350. The stall torque is 4.1 Nm at 12.0 V and 2.3 A. The gears consist of metal spur gears and have a reduction ratio of 353.5. XM430-W350 equips a mode that allows the command of the target motor current directly, and this mode is used.

B. System Identification

In this study, simulation parameters for system identification are shown in Table [I]. Note that the motor armature is the additional link inertia caused by the rotor inertia of the DC motor and was amplified with the square of the gear reduction ratio from the rotor inertia itself. The simulation was implemented using Mujoco [21]. The size of the simulation time step was set to 1 ms.

For the equation (5) of the black-box optimization for first system identification, we use the Tree-structured Parzen Estimator algorithm implemented in Optuna [26] since it gives better results than CMAES in our experimental setup. For the multi-objective optimization equation (6), we used the NSGA-II algorithm with also Optuna. We set the number of trials for each optimization in the experiment to 2000.

For system identification, a simple squatting motion is used for the exciting motion \(\theta^{\text{targ}}\). While keeping a forward-leaning posture, execute a flexion/extension movement such that the knee joint angle flexes from 1.47 rad to 0.6 rad at a speed of approximately 0.5 Hz. To evaluate the error \(L_{\text{exc}}\) from the squatting motion, equation (7) is used.

\[
L_{\text{exc}} = \frac{1}{T} \sum_{t=1}^{T} \sum_{i} (||r_i^{\text{sim}} - r_i^{\text{real}}||^2 + ||\dot{r}_i^{\text{sim}} - \dot{r}_i^{\text{real}}||^2) + \frac{1}{NT} \sum_{t=1}^{T} \sum_{i=1}^{N} (||\theta_i^{\text{sim}} - \theta_i^{\text{real}}||^2 + ||\dot{\theta}_i^{\text{sim}} - \dot{\theta}_i^{\text{real}}||^2 + ||I_i^{\text{sim}} - I_i^{\text{real}}||^2)
\]

Where \(r\) is upper body orientation, \(\dot{r}\) is body angular velocity, \(\theta\) is the joint angle, and \(I\) is joint current. \(N\) is the number of focusing joints.
C. Task for Evaluation

1) Balancing task: The objective of the balancing task is to maintain balance during standing upright on a board with a dynamically changing tilt. The action of the policy in this RL is the target angle and P gain of PD control for each joint. The target joints are five for each leg except for the hip-yaw axis, for 10 joints for both legs, and the action space has 20 dimensions. The observation of RL policy in this task includes the following elements:

- The position and velocity of the six key points placed on the three corners of each leg soles with the base link as the origin (36 dims)
- Command in the previous step (20 dims)
- Body orientation in the Euler angle (3 dims)
- Body angular velocity (3 dims)

To be aware of the time series of the state, we concatenate the above observation obtained in the previous step with the observation obtained in the latest step, then feed it to the policy. The reward function is calculated by subtracting scores from the survival reward by each penalty term, including the following elements: the difference from the reference posture, the magnitude of the joint angular velocity, the distance of the base link position from the reference point, the tilt of the base link, the angular velocity of the base link, and the magnitude of the joint current.

For evaluation on the actual robot, we use a board that tilts max of about 6° for the roll axis. Disturbances are given by 1Kg weight falling on the board’s edge from two fixed heights.

2) Walking task: The bipedal walking task is an advanced task that requires more active motion than the balancing task. The objective is to walk forward on a flat surface at a constant speed. The action and observation spaces are the same as in the balancing task, but the periodic signals of the walking phase encoded in two dimensions are added to observations. The reward function is designed similarly to the prior study by Siekmann et al. [15], so refer to their paper for details. The penalty terms of the reward included the difference from the target velocity, the floor reaction force in the swing leg and the toe velocity in the supporting leg, the base link posture difference from the target posture, the acceleration and angular velocity of the base link, the magnitude of the joint current, and the distance to the action of the previous step. The target walking cycle was set at 1.0 sec, and the ratio of the support leg phase to the swing leg phase was set at 0.7 : 0.3.

VI. Results

The following three points are evaluated to verify the effectiveness of the proposed method.

- Validation of the proposed system identification method for sim-to-real (section VI-A)
- Verification of the effectiveness of actuator model for sim-to-real (section VI-B)
- How well does the learned policy perform on the actual robot (section V-C.1 & section VI-C.2)

### TABLE II: Methods and success rates in balancing task on actual robot

| Parameter used for policy training | Standing on flat board | Balancing on tilting board |
|-----------------------------------|------------------------|---------------------------|
| Excitation only                   | 0/3                    | 0/3                       |
| Re-Identified (ours)              | 3/3                    | 3/3                       |
| Excitation only (w/o DTE)         | 0/3                    | 0/3                       |
| Re-Identified (w/o DTE)           | 2/3                    | 0/3                       |
| Domain randomization              | 0/3                    | 0/3                       |

### TABLE III: Methods and speed performances on walking task

| Parameter used for policy training | Attempt 1 | Attempt 2 |
|-----------------------------------|-----------|-----------|
| | robot (sim) | robot (sim) |
| Excitation only                   | 2.1 (5.3) | 1.2 (5.7) |
| Re-Identified (ours)              | 0 (2.3)   | 8.3 (10.1) |
| Domain randomization              | 0.5       | 0         |

A. Evaluation of Sim-to-real Learning Method

For the balancing task, we performed the sim-to-real learning process described in section IV. The whole learning process was conducted three times, with different seed values, and the results are shown in Table II. The left column lists the names of the set of simulation parameters $\phi$ to train the policy, and the right column shows the number of successful attempts. “Excitation only” corresponds to the identified parameter values with the first system identification at phase 1. As shown in Table II, the policy failed to work on the actual robot. Then re-identification was processed, and the result simulation parameters are labeled as “Re-Identified” on Table II. Policies that trained with “Re-Identified”, succeeded in balancing on the tilting board in all attempts. Despite the small amount of actual robot operation time for data collection (only 5 seconds of squat motion and five failure rollouts), sim-to-real was achieved without domain randomization.

In addition, we trained a balancing policy with naive domain randomization as a baseline. During training, each simulation parameter is uniformly sampled within a range shown in Table I. The transferred policy could not succeed in the balancing task in any attempt.

We processed the sim-to-real learning in two attempts with different seed values for the walking task. To evaluate the performance of each policy rollout, we use the average velocity (distance forward in the target direction divided by time). Table III shows the results, the unit of value is cm/s, and zero is put when the policy could not go forward. The values in the brackets show the policy performance in the simulation used for training. Attempt 2 succeeded in sim-to-real. The attempt 1, the walking policy trained with “Excitation only” seems conservative (relatively stable, but slow), which would have made it difficult to evaluate the policy gap with $L_{policy}$ in equation (6) and it tends to miss identification. Especially the challenging RL task, such as walking, the simulation parameters affect the learning performance. “Re-Identified” on attempt 1 that miss identified might make RL difficult.

In addition, we trained a walking policy with naive domain randomization same as the balancing task with two seeds. Both policies failed to walk on the actual robot.

We also found that the re-identified parameters with the balancing task can generalize to the walking task. We trained a walking policy with that parameters, the policy performs better than the re-identified with walking tasks.
B. Evaluation of Actuator Modeling

First, we describe the effectiveness of the DTE model in the sim-to-real process of the balancing task. In the Table II, parameter “Excitation only (w/o DTE)” is the result of the first system identification without DTE model, also “Re-Identified (w/o DTE)” is the result of the re-identification without DTE. As shown in the table, it failed to sim-to-real without the DTE model. From this result, we can say that DTE is one of the critical components of the actuator model.

Next, we show the effect of the DTE model in the first system identification with the squatting motion. The error scores of the first system identification (Equation (5) and (7)), the best of three attempts was 0.037 for “Excitation only” and 0.070 for “Excitation only (w/o DTE)”. Fig. 3 shows the graphs of the sensor information of the actual and the simulated robot. With DTE model reproduced sensor information well. In particular, the DTE’s ability to represent changes in a frictional loss in static and driving conditions is thought to have contributed to the reproducibility of the overall behavior.

C. Evaluation of Policy Performance

To verify the performance of the policy acquired with the proposed method, we demonstrate and evaluate the tasks on an actual robot. Each motion of the policy can be viewed in the supplementary video.

1) Balancing Task: Fig. 4 shows the snapshot of balance task experiment. It was confirmed that the policy could maintain balance under impulsive disturbances, and the robot could maintain an upright posture.

2) Walking Task: To realize the robust policy utilizing the backdrivability that is the aiming of this research, we trained a walking policy on the simulation with random uneven terrains. For training, the re-identified simulation parameters with the balancing task were used. The trained policy passed three uneven surfaces for testing; the surface and snapshot of walking are shown in Fig. 5. We used some toy brick parts for unevenness. Each brick has a 5 mm height on top. Fig. 6 shows the command for the left ankle roll joint by the policy while walking. The P gain is reduced as the leg transitions into the support phase, indicating a compliant landing.

In addition, we tested an existing walking module of ROBOTIS-OP3 with the default settings as a baseline using stiff position-controlled joints. The walking module failed on these uneven surfaces. The fact that the policy was able to traverse uneven surfaces, which is difficult with a general walking controller, without force or contact sensors indicates that the policy can take advantage of backdrivability.

VII. Conclusion

In this paper, we propose a method to acquire a robust control policy of the actual robot by RL, using a model of high reduction ratio gears and a system identification method that utilizes the experience of failed sim-to-real transfers. The method was verified on the actual robot, ROBOTIS-OP3, and achieved balancing control under disturbance and walking on uneven surfaces without any force/torque sensor.

Acknowledgement

The authors would like to thank Yasuhiro Fujita and Avinash Ummadisingu for their support and discussions on this research.

References

[1] S. Hyon and G. Cheng, “Passivity-based full-body force control for humanoids and application to dynamic balancing and locomotion,” in
2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 4915—4922.

[2] G. Mesesan, J. Englsberger, G. Garofalo, C. Ott, and A. Albu-Schäffer, “Dynamic walking on compliant and uneven terrain using dcm and passivity-based whole-body control,” in 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), 2019, pp. 25—32.

[3] H. Suzuki, Y. Nagamatsu, T. Shirai, S. Nozawa, Y. Kakizuchi, K. Okada, and M. Inaba, “Torque based stabilization control for torque sensorless humanoid robots,” in 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), 2017, pp. 425—431.

[4] K. Kaneko, H. Kaminaga, T. Sakaguchi, S. Kajita, M. Morisawa, I. Kumatagi, and F. Kanehiro, “Humanoid robot hrp-5p: An electrically actuated humanoid robot with high-power and wide-range joints,” IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1431—1438, 2019.

[5] K. Kojima, T. Karasawa, T. Kozuki, E. Kurioka, S. Yukizaki, S. Iwaishi, T. Ishikawa, R. Koyama, S. Noda, F. Sugai, S. Nozawa, Y. Kakizuchi, K. Okada, and M. Inaba, “Development of life-sized high-power humanoid robot jaxon for real-world use,” in 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), 2015, pp. 838—843.

[6] ROBOTIS CO., LTD, “THORMANG3 INTRODUCTION,” https://emanual.robotis.com/docs/en/platform/thormang3/introduction (accessed on 27 Feb 2022).

[7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” Science Robotics, vol. 5, no. 47, Oct 2020. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abc5986

[8] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, “Blind bipedal stair traversal via sim-to-real reinforcement learning,” https://arxiv.org/abs/2105.08328, 2021.

[9] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, “Learning agile robotic locomotion skills by imitating animals,” in Robotics: Science and Systems, 07 2020.

[10] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, “Sim-to-real: Learning agile locomotion for quadruped robots,” 2018. [Online]. Available: https://arxiv.org/abs/1804.10332

[11] Y. Du, O. Watkins, T. Darrell, P. Abbeel, and D. Pathak, “Auto-tuned sim-to-real transfer,” in 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 1290—1296.

[12] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, “Sim-to-real transfer for biped locomotion,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 3503—3510.

[13] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsoinis, V. Koltun, and M. Hutter, “Learning agile and dynamic motor skills for legged robots,” Science Robotics, vol. 4, no. 26, Jan 2019. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.aau5872

[14] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, “Learning locomotion skills for cassie: Iterative design and sim-to-real,” in Proceedings of the Conference on Robot Learning, ser. Proceedings of Machine Learning Research, vol. 100, 2020, pp. 317—329.

[15] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, “Sim-to-real learning of all common bipedal gaits via periodic reward composition,” in 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 7309—7315.

[16] X. B. Peng and M. van de Panne, “Learning locomotion skills using deeprl: Does the choice of action space matter?” in Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation, ser. SCA ’17, 2017, pp. 12:1—12:13.

[17] 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,” https://arxiv.org/abs/1703.06907, 2017.

[18] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Isaac, N. Ratliff, and D. Fox, “Closing the sim-to-real loop: Adapting simulation randomization with real world experience,” in 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 8973—8979.

[19] M. Taylor, S. Bashkirov, J. F. Rico, I. Toriyama, N. Miyada, H. Yanagisawa, and K. Ishizuka, “Learning bipedal robot locomotion from human movement,” in 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 2797—2803.

[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” https://arxiv.org/abs/1801.01290, 2018.

[21] E. Todorov, T. Erez, and Y. Tassa, “Mujoce: A physics engine for model-based control,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 5026—5033.

[22] A. Wang and S. Kim, “Directional efficiency in geared transmissions: Characterization of backdrivability towards improved proprioceptive control,” in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 1055—1062.

[23] H. Matsuki, K. Nagano, and Y. Fujimoto, “Bilateral drive gear—a highly backdrivable reduction gearbox for robotic actuators,” IEEE/ASME Transactions on Mechatronics, vol. 24, no. 6, pp. 2661—2673, 2019.

[24] J. Tan, Z. Xie, B. Boots, and C. K. Liu, “Simulation-based design of dynamic controllers for humanoid balancing,” in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 2729—2736.

[25] ROBOTIS CO., LTD, “ROBOTIS-OP3 INTRODUCTION,” https://emanual.robotis.com/docs/en/platform/op3/introduction (accessed on 27 Feb 2022).

[26] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.