2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Sim-to-Real Learning for Humanoid Box Loco-Manipulation

Jeremy Dao, Helei Duan, Alan Fern

Abstract— **In this work we propose a learning-based approach to box loco-manipulation for a humanoid robot. This is a particularly challenging problem due to the need for whole-body coordination in order to lift boxes of varying weight, position, and orientation while maintaining balance. To address this challenge, we present a sim-to-real reinforcement learning approach for training general box pickup and carrying skills for the bipedal robot Digit. Our reward functions are designed to produce the desired interactions with the box while also valuing balance and gait quality. We combine the learned skills into a full system for box loco-manipulation to achieve the task of moving boxes from one table to another with a variety of sizes, weights, and initial configurations. In addition to quantitative simulation results, we demonstrate successful sim-to-real transfer on the humanoid robot Digit. To our knowledge this is the first demonstration of a learned controller for such a task on real world hardware.**

I. Introduction

Most reinforcement learning (RL) research for legged robots focuses on locomotion skills such as walking, running, and navigating various terrains [1–5]. On the other hand, most RL research for object manipulation has focused on non-legged robots with fixed or stable bases (e.g. grounded robotic arms) [6–8]. RL has been much less explored for loco-manipulation with legged robots, which involves both movement with and manipulation of objects. In particular, we are unaware of any work that has demonstrated successful sim-to-real RL for loco-manipulation with a bipedal humanoid robot. This is a particularly challenging problem due to the need for full-body coordination to manipulate and move objects while also maintaining balance.

The main goal of this paper is to design and evaluate a sim-to-real RL approach for loco-manipulation with a real humanoid robot. In particular, our target task is to walk up to a box, stop, pick up the box, carry it to another table, and then set it down. Using data-driven RL for this purpose has the potential to facilitate generalizion to boxes of different sizes, masses, poses, and locations. Designing a sim-to-real RL solution, however, raises a question. There is ambiguity about what is the "best" or most desirable way to pickup and move a box. How should the hands move, and how should the overall motion of the body coordinate with the hands to produce smooth, non-agressive motion that is still robust? Furthermore, what kind of dense reward signal can we provide to faciliate learning such a motion?

*This work is supported by the NSF Grant No. IIS-1849343 and DARPA Contract N66001-19-2-4035.

All authors are with Collaborative Robotics and Intelligent Systems Institute, Oregon State University, Corvallis, Oregon, 97331, USA.
Email: {daoje, duanh, afern}@oregonstate.edu.

Fig. 1: We learn box loco-manipulation policies in simulation and transfer directly to real hardware. We break the task down into 5 separate policies: Walk, Stand, PickUp, WalkWithBox, and StandWithBox.

This paper addresses this challenge by making the following contribtions:

- We construct a reward function for learning a "box pickup" skill and the associated locomotion skills. By designing these functions with box interaction in mind, specifying when the hands should move to contact the box, and when to apply forces to move the box, along with regulating the motion of the hands, we can learn smooth, gentle motions that still achieve the task.
- Compared to previous works on locomotion [2–5], we find that for box pickup using a "relative" action space, where the policy outputs are PD setpoint commands to be added to the current joint position, is more suited for learning box pickup. This change in action space results in faster learning and less forceful interaction with the box.
- We present a full box loco-manipulation system where each skill, standing and walking with and without a box, and box pickup, is individually learned. We show successful sim-to-real on the humanoid robot Digit, which to our knowledge is the first demonstration of a learning controller achieving a loco-manipulation task on real world hardware.

II. Related Works

A. Loco-Manipulation for Humanoids

Much of prior research on loco-manipulation for humanoids has been largely model-based and focuses on mostly static manipulation tasks. Model-based approaches have been explored

[9, 10] for lifting up boxes, planning a whole-body motion and compensating for the extra force applied by the box. However, these methods produce very conservative motion to ensure absolute stability at all times, resulting in pickup motions that take over 20 seconds to be completed. Planning based approaches [11–13] have been explored for producing long loco-manipulation trajectories, with [11] even showing hardware results. Work focused on locomotion [14, 15] has explored how to modify a humanoid's gait when interacting with objects. These strategies often augment some baseline gait controller to account for and modify the motion of the manipulated object.

Prior work that has used learning for humanoid or bi-manual manipulation tasks have been mostly trajectory based or rely on demonstrations as targets for imitation learning. For example, Seo et al. (2023) uses teleoperation to obtain expert trajectories to imitate, while Liu et al. (2023) uses motion tracked human demonstrations for a similar manipulation tracking task. This requirement of prior expert data can introduce another failure point in the learning system, as motion tracking error or alignment error between the human and robot data can cause componding issues down the line.

B. RL for Loco-Manipulation

The use of learning for loco manipulation has been largely centered around mobile base or quadraped robots with an attached manipulator arm. Due to the inherent stability of such a system a common strategy is to separate out the locomotion and manipulation control in to two separate modules. Ma et al. (2022) utilizes RL for only locomotion while the manipulator arm is controlled by a MPC planner, while Sun et al. (2022) learns separate grasping and navigation policies. We utilize a similar separation of locomotion and manipulation skills, in that locomotion and box pickup are separate control policies, but unlike Ma et al. (2022) our learned policies control all the joints, so that coordinated full body motion is always possible. Fu et al. (2022) and Arcari et al. (2022) both learn a more unified control approach that commands both the legs and manipulator arm from the same controller. Fu et al. (2022) learns a policy that executes base velocity and end effector position commands from a higher level planner, while Arcari et al. (2022) learns from a MPC trajectories.

Perhaps the most relevant prior work to our research is Xie et al. (2023), which targets the same humanoid box pickup and locomotion task that we do. We follow a similar strategy of breaking down the task into different skills with a control policy for each. The main differences that we try to improve upon is to remove the reliance on trajectory tracking and focus more on hardware execution instead of an animation setting.

III. System Overview

We break down the full box loco-manipulation task into 5 distinct behaviors: 1) Walk, for basic walking, 2) Stand, for transitioning to a standing position after walking, 3) PickUp, for picking up the box, 4) WalkWithBox, for walking while holding a box, and 5) StandWithBox for transitioning to standing while holding a box. For each behavior we learn a distinct control policy, which has behavior specific command paramters. Box loco-manipulation involves only a subset of the possible transitions between these policies. For example, we will never go directly from walking to box pickup, and instead will always use standing as an inbetween behavior. Fig. 2 shows the allowed behavior transitions that we target for the box loco-manipulation task.

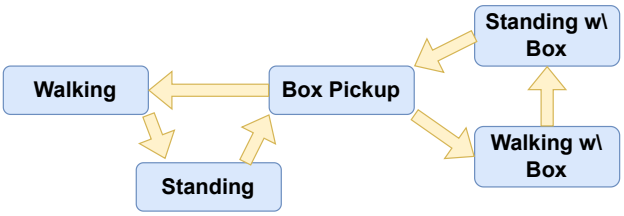

Fig. 2: Allowed transitions between the 5 different policies.

All of our policies follow the same general learning framework, with changes mainly to the reward function to learn different skills. Following prior work on bipedal locomotion [23] we use an LSTM neural network with two 128-dimensional recurrent hidden layers to represent our policies, with the inputs and outputs adapted for the Digit humanoid robot, which has 20 DoFs and 10 unactuated joints. The input to each policy is a pair (s, c), where s is the 67-dimensional robot-state vector and is the same for all policies, and c contains a policy specific command. The policy runs at 50Hz and outputs actions corresponding to PD setpoint targets for each actuated joint. These are passed to a PD controller running at 2kHz with fixed gains.

We train policies in simulation using the MuJoCo physics engine [24] and use the actor-critic PPO algorithm [25] with a clipped objective and gradient clipping. To facilitate sim-to-real transfer, we use dynamics randomization [26] during training on parameters such as body mass, joint damping, body center of mass location, and friction. Ranges for these randomizations can be found in table Table I. Rather than use a state estimator during training and testing, we simply use the true simulation state and apply random noise on top of it. We train using 80 cores on a dual Intel Xeon Platinum 8280 server.

Each of our reward functions is defined in terms of a distinct set $\{(r_i, w_i)\}$ of paired reward terms and weights. The corresponding reward function is then defined by:

$$R = \sum_i w_i \exp(-r_i)$$

In most cases, each of the reward terms will represent some cost to be avoided, so that the negative exponential will be smaller for larger costs. The state variables used below to define the reward functions are defined in Table II.

IV. Learning Loco-Manipulation Skills

A. Box Pickup Policy

We want to train a control policy that is capable of picking up boxes. It should be able to handle boxes of varying sizes and masses, along with the box starting in a range of positions

Body Mass	$[-0.2, 0.2]$
Joint Damping	$[-0.5, 2.5]$
Ground Friction	$[-0.3, 0.2]$
Center of Mass Position	$[-0.05, 0.05]$

TABLE I: Dynamics randomization ranges. All ranges are in percentage values, so a range of $[-0.2, 0.2]$ corresponds to a range of $\pm 20\%$ of the original value.

$p_{\text{x,t}}$	position of x at time t	$p^*_{\text{x,t}}$	target position of x at time t
$F_{\text{x, y}}$	contact force between x and y	$c_{\text{x, y}}$	contact indicator between x and y (1 if in contact, 0 otherwise)
Φ_{x}	roll angle of x	Θ_{x}	pitch angle of x
v_{x}	velocity of x	a_{x}	acceleration of x
q^*_{x}	quaternion orientation of x	$p^i_{x/y/z}$	x/y/z position of i
u_t	action at time t	τ	torque

TABLE II: Definition of state variables used in reward functions.

in front the robot. In addition, we aim to learn pickup behavior that has smooth, natural motions and timings (e.g. not too fast or too slow). To define the pickup behavior we break it down into two phases: 1) a contact phase, where the hands aim to make contact with the box, and 2) the lift phase, where the hands aim to move the box to a target position. The phases are dictated by hand selected times, relative to the start of the behavior, which is time $t = 0$. In particular, the contact phase ends at time $t_{\text{contact}} = 100$ and then lift phase ends at $t_{\text{lift}} = 175$.

The command input c to the pickup policy consists of: 1) box dimensions, mass, and starting pose, 2) the target position to move the box, and 3) two phase indicators p_{contact} and p_{lift}, which each go linearly from 0 to 1 from the start to end of each phase. For ease of hardware testing, we assume that the box pose is only explicitly known at the start of the behavior. Once the box is picked up and has moved, we approximate the pose with the average pose of the hands, which avoids the need for real-time estimation of box pose. Given that the hands are in contact with the box, it is reasonable to assume that the box centroid position is going to be roughly between the hands.

The action space of box pickup policy differs from the locomotion policies. In particular, the locomotion policies add the policy's action output to a static "neutral" offset. Rather, for box pickup we found that adding the action output to the current motor positions resulted in faster learning (see Section V). This is likely due to the much larger variance in motion profile of the pickup behavior from the neutral position compared to locomotion behavior.

Scenario Distribution. At the beginning of each training episode, we spawn a box in a random location in front of the robot. The task is then to move the hands to the box, pick it up, and move it to a randomized target location above the table. The starting box location may be randomized within 35 to 50cm in front of the robot, $\pm$30cm to the side of the robot, and be on the ground or up to 1.3m in the air. The starting box yaw orientation may be $\pm 22.5°$. The target location is randomized within the same range, except the z position of the target will only be above the starting location. The box length, width, and height may each be between 20 to 45cm, and the mass may be between 1 to 10kg.

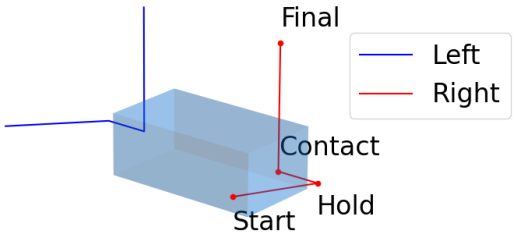

Fig. 3: Example hand position trajectory for a box pickup. The hands start from the initial robot pose, move to the side of the box (shown in blue), make contact with it, and then bring it to the target location. In this example the target location is directly above the box.

Reward Function. To train box pickup policies, we take inspiration from previous work on learning locomotion policies [5, 23]. Those works described walking through stance and swing phases, in other words specifying how and when the feet should contact with the ground. This general principal of perscribing contact can be applied to almost any interaction task, and is the main idea we utilize for our reward function. In contrast to locomotion, box pickup is a non-periodic behavior and we use a non-periodic clock signal with phase reference points t_{contact} and t_{lift} to define the reward.

In particular, the reward function has the form

$$R = R_{\text{traj}} + R_{\text{box}} + R_{\text{stand}} + R_{\text{reg}}$$

where R_{traj} rewards hand motion consistent with a box pickup, R_{box} is designed to encourage proper interaction with the box, R_{stand} aims to ensure the robot is stably standing, and R_{reg} aims to regulate forces to ensure smoothness and help sim-to-real transfer. Below we describe the set of reward terms used to define each of these components noting that the weights for each term are provided in Table III.

R_{traj} rewards being close to the target trajectory over time and having a hand orientation of zero using the following terms:

$$r_{\text{hand\_pos}} = \|p^{\text{left}}_{\text{hand,t}} - p^{*\text{left}}_{\text{hand,t}}\| + \|p^{\text{right}}_{\text{hand,t}} - p^{*\text{right}}_{\text{hand,t}}\|$$
$$r_{\text{hand\_roll}} = |\Phi^{\text{left}}_{\text{hand}}| + |\Phi^{\text{right}}_{\text{hand}}|$$

To describe the desired hand motions, we construct a hand position trajectory by choosing three goal hand positions, along with a timestamp for each, and linearly interpolate between them as shown in Figure 3. Each of these goal positions is based off of the box's starting position, size, and the target location. So the generated hand trajectory adapts to each pickup scenario. For the contact phase, the hands should first be out along each side of the box, and then eventually actually touching the box along it's transverse faces. Since t_{contact} is 100 timesteps (2 seconds), we design the hands to be 10cm besides the box (goal position 1) at 1.5 seconds and

then at 2 seconds the hands should be on the side faces of the box (goal position 2). The last goal postion 3 occurs at t_{lift} (3.5 seconds) with the hands on either side of the target location, spaced evenly with a box's width distance between them. We were also able to train policies without this hand position trajectory. However, we found that these policies trained significantly slower (see section VII) and had less success on hardware.

R_{box} rewards contact with the box at the correct time (not too early), lifting it off the table, and moving the box to the target location with a flat orientation using the following terms:

$$r_{\text{contact}} = \ln\left(0.05 \cdot c^{\text{left}}_{\text{hand,box}} + 0.05 \cdot c^{\text{right}}_{\text{hand,box}}\right)$$
$$r_{\text{box\_pos}} = \|p_{\text{box}} - p^{*}_{\text{box}}\|$$
$$r_{\text{box\_orient}} = |\Phi_{\text{box}}| + |\Theta_{\text{box}}|$$
$$r_{\text{table}} = \mathbf{F}_{\text{table,box}}$$

Note that yaw is not included in $r_{\text{box\_orient}}$ since some target locations will be to the side of the robot base. When considering objects that may be inside the box, rotation about the gravity axis (yaw rotation) is allowed, but other rotation axes would disturb the objects. It is important to highlight the weighting for r_{contact}, weight $w_{\text{contact}} = I[t \geq t_{\text{contact}}]\cdot]$, where $I[\cdot]$ is the indicator function. This causes the contact reward term to only be active at the end of the first phase, which helps prevent early contact with the box.

R_{stand} rewards the robot for standing stably while picking up the box. This is done by keeping the center of pressure (CoP) in the middle of the support polygon (or the average foot location), the torso orientation upright and the feet orientation facing forward, motor velocity small (try to move as little as possible), and the feet velocities at zero:

$$r_{\text{CoP}} = \|p_{\text{CoP}} - p_{\text{avg foot pos}}\|$$
$$r_{\text{base\_orient}} = 1 - \langle q_{\text{torso}}, (1,0,0,0)\rangle^2$$
$$r_{\text{foot\_orient}} = (1 - \langle q^{\text{left}}_{\text{foot}}, (1,0,0,0)\rangle^2) + (1 - \left\langle q^{\text{right}}_{\text{foot}}, (1,0,0,0)\right\rangle^2)$$
$$r_{\text{motor\_vel}} = \|v_{\text{motor}}\|$$
$$r_{\text{feet\_vel}} = \|v^{\text{left}}_{\text{foot}}\| + \|v^{\text{right}}_{\text{foot}}\|$$

Finally, R_{reg} encourages smooth motions of both the robot and box itself. It also rewards gentle hand interaction with the box:

$$r_{\text{action}} = \|u_t - u_{t-1}\|$$
$$r_{\text{torque}} = \|\tau\|$$
$$r_{\text{hand\_force}} = \|F^{\text{left}}_{\text{hand,box}}\| + \|F^{\text{right}}_{\text{hand,box}}\|$$
$$r_{\text{box\_acc}} = \|a_{\text{box}}\|$$

The total reward R also has two sparse penalties. There is a -0.1 penalty if there is a self-collision or if the velocity of either hand exceeds 1.0 m/s. The weightings of each dense reward term used is in Table III. To encourage useful exploration, we also use the following termination conditions, which encourages learning critical aspects of the behavior before more detailed aspects.

- If the height of the robot torso is less than 40cm.
- If the pitch angle of the robot torso is greater than $35°$.
- If either foot loses contact with the ground.
- If the robot makes contact with the table.
- If the box is on the ground.
- If it has been 0.5 seconds after the contact countdown and the hands are not in contact with the box.
- If it has been 0.5 seconds after the pickup countdown and the box is still in contact with the table.

Domain Randomization. In addition to the standard dynamics randomization we use for all policies, we also add randomization to the policy's estimate of the box mass ($\pm$0.5kg), dimensions ($\pm$5cm), and starting location ($\pm$5cm).

B. Locomotion Policies

To learn locomotion skills we use the same learning setup as described in [5, 23] and we refer readers to those works for more details on the reward function. The only changes made for this work was an additional reward term on the hand positions, to keep them in front of the torso and equidistant from each other.

The box locomotion policies follow largely the same learning setup as the regular locomotion policies, with a few tweaks to accomodate the box. Similar to the box pickup policy, the command input c for the box locomotion policy contains the box dimensions, mass, and a height command as input. This height command specifies at what height the box should be held at while walking.

Scenario Distribution. At the start of each training episode the robot is initialized to a random starting pose (see Section IV-D for more details), at a random phase in the walking cycle, and with a random command. The commanded x velocity may be randomized between $[-0.5, 1.0]$m/s, y velocity between $[-0.3, 0.3]$m/s, turning rate between $[-\pi/8, \pi/8]$rad/s, and box height between $[1.0, 1.3]$m. At a random time during the trajectory the commands with be randomized again.

Reward Function. The reward function for learning box locomotion policies is the same as the regular locomotion policies [5, 23] with the addition of a few terms similar to the box pickup rewards. These terms form the box reward we denote as R_{box}:

$$r_{\text{box height}} = \|p_{\text{box}} - [0.4, 0, h_{\text{cmd}}]\|$$
$$r_{\text{box orient}} = |\Phi_{\text{box}}| + |\Theta_{\text{box}}|$$
$$r_{\text{box force}} = \|F^{\text{left}}_{\text{hand,box}}\| + \|F^{\text{right}}_{\text{hand,box}}\|$$
$$r_{\text{hand roll}} = |\Phi^{\text{left}}_{\text{hand}}| + |\Phi^{\text{right}}_{\text{hand}}|$$

There is also an additional termination condition. The box locomotion policy will always be initialized with a box already in the hands, and it is not allowed to lose contact with the box else the training episode will terminate.

Domain Randomization. The regular locomotion policy uses only the base dynamics randomization described in Section III. For the box locomotion policy, similar to the box pickup policies, there is randomization to the policy's box mass ($\pm$0.5kg) and dimension ($\pm$5cm) input.

C. Standing Policies

We want to train standing policies to transition from walking to static standing states. Since the standing policy only has the goal of standing still, there is no c input to the standing policy and it only receives the robot state s.

The standing box policy has the same goal and learning setup as the normal standing policy, with only minor modifications to the reward. Since in this case there is a box that we would like to control, the command input c for the standing box policy includes the box dimensions, mass, and a box height command.

Scenario Distribution. At the start of each training episode the robot is initialized to a random walking state (see Section IV-D for more details). For the box standing policy the box height command is also randomized.

Reward Function. The reward function consists of the following terms (in addition to R_{stand} and the usual regulator reward R_{reg}):

$$
\begin{aligned}
r_{\text{base vel}} &= \|v_{\text{base}}\| \\
r_{\text{height}} &= |h_{\text{base}} - 0.9| \\
r_{\text{arm}} &= \|p_{\text{arm}}^{\text{left}} - [0.15, 0.3, -0.1]\| + \\
&\qquad \|p_{\text{arm}}^{\text{right}} - [0.15, -0.3, -0.1]\| \\
r_{\text{stance width}} &= |(py_{\text{foot}}^{\text{left}} - py_{\text{foot}}^{\text{right}}) - 0.385| \\
r_{\text{stance x}} &= |px_{\text{foot}}^{\text{left}} - px_{\text{foot}}^{\text{right}}|
\end{aligned}
$$

Note that here $p_{\text{arm}}^{\text{left}\backslash\text{right}}$ denotes the position of the arms in the torso's frame. There is also a sparse reward penalty of -0.1 is there is a self collision. The training episode terminates if the robot has fallen over or if either foot is off the ground after 100 policy timesteps.

The box standing policy follows the same reward function with the addition of the box reward R_{box} described above. The termination conditions are the same as well, with an additional termination if either hand loses contact with the box.

Domain Randomization. Like the locomotion policies, the regular standing policy use only the base dynamics randomization while the box standing policy has randomization to the box mass (± 0.5kg) and dimension (± 5cm) input.

Name	Weight	Name	Weight
Hand Position	0.15	Hand Roll	0.05
Box Position	0.15	Box Orientation	0.05
Table force	0.05	CoP	0.1
Base Orientation	0.05	Foot Orientation	0.1
Motor Velocity	0.05	Foot Velocity	0.05
Hand Force	0.05	Box Acceleration	0.05
Action Change	0.05	Torque	0.05

TABLE III: Reward weightings for the box pickup reward.

D. Skill Transitions

Once we have all 5 necessary box loco-manipulation skills, we need to connect them together in order to achieve the full task. Through out execution, we will transtion from one skill to another, and the policies need to be able to handle such a transition. Fig. 2 shows the allowed transitions between different policies. Note that we do not care about every possible transition between all of the skills, just the ones that are needed for the box loco-manipulation task we target. For example we will never go directly from walking to box pickup, and instead will always use standing as an inbetween state. Using this we can create a workflow in which one policy is used to generate an initial state distribution with which we can use to train the next skill.

More specifically, we first train a regular walking policy. We then use this to generate many low-speed states from which we want to transition to standing from. These form the initial state distribution for training the standing policy. We can then use the standing policy to generate intial states for the box pickup policy and so on.

To ensure smooth transitions between policies, we also linearly interpolate between the two policies' actions over 10 timesteps. This action "warmup" is applied during training as well, and so during initial state distribution generation we save the last applied action along with the robot state.

V. Simulation Results

We first conduct quantitative evaluations for the Digit model in the MuJoCo simulator. Each evaluation averages over 10K episodes using the same episode generation approach as described for training. For example, for the evaluation of the PickUp policy we randomly sample box pose, mass, size, and target location.

Overall Policy Performance. For each of the 5 learned policies we consider an episode a success if: 1) the robot does not fall over, and 2) the box remains in the robot's hands at the end of the episode for behaviors that involve the box. Further, for behaviors that involve the box, we also measure the error for successful episodes, which is the distance between the final position of the box and the commanded target position of the box. Table IV records the success rate and error for each of the policies. Overall we see that the policies all achieve a high success rate and also produce relatively small errors. These results show that the policies are all quite robust to the randomization of the scenario distribution and the transition starting states that they will typically be initiated in.

Limits of PickUp Policy. We also conducted additional simulation experiments for our most complex full-body behavior, PickUp. First, we evaluated its performance outside of the training distribution of box parameters. To do this we performed an incrementally increasing search for the maximum box mass, size, and starting pose that the policy can successfully pickup. In each test, only one parameter type is varied at a time with all other box parameters kept at the mean value of the training distribution. In particualar, we turned domain randomization off and continually incremented the particular parameter under test until a pickup failure occured.

	PickUp	Stand	StandBox	Walk	WalkBox
Success %	96.15%	100%	95.25%	96.1%	100%
Error (cm)	7.93	N/A	1.64	N/A	2.39

TABLE IV: Success rate and accuaracy for the 5 behavior policies as evaluated over 10K random episodes.

	Maximum	Training Range
Mass (kg)	22.9	[1, 10]
Size (cm)	64	[20, 45]
X displacement (cm)	56	[35, 50]
Y displacement (cm)	41	[-30, 30]
Z displacement (cm)	130	[0,130]
Rotation	45°	$[-22.5, 22.5]°$

TABLE V: Extrapolation perforamnce of the PickUp policy. Each row shows the maximum successful value for the policy for that box parameter along with the range used for training.

Table V shows that the learned PickUp policy generalizes quite well outside of the training distribution and is able to pickup boxes much larger and heavier than those it saw during training.

PickUp Ablation. We also perform some ablation studies on the learning setup with a focus on the PickUp behavior. To test if a simpler reward setup will still learn, we remove the hand trajectory tracking reward component. The target location for each hand p^* is instead just the center of the left and right face of the box in the box target position, and does not change throughout the training episode. We found that we can still learn a successful box pickup policy, however, the sample efficiency suffers greatly. As can be seen in Fig. 4, learning without the hand trajectory ("No Hand Trajectory"), approximately doubles the number of samples needed to reach the same reward as learning with the trajectory information ("Baseline").

Our PickUp policy uses a different action space than the locomotion and standing policies. Rather than adding the policy output to a fixed "neutral" offset robot pose, the PickUp policy instead adds the policy output to the current joint motor positions. To test the effect of this change, we train policies with both action spaces. The reward curves for each are presented in Fig. 4. As can be seen, using the fixed offset action space learns ("Absolute Action Space") significantly slower, about as slow as not using the hand trajectory. We did not observe this same loss in sample efficiency for the standing and walking policies, leading us to believe that when the desired motion deviates greatly from the fixed position offset, it is beneficial to use a "relative" action space instead.

VI. Hardware Results

We show successful sim-to-real transfer for all trained policies, including transitions between each skill. We are able to perform box pickups for multiple boxes with different masses (from 1kg to 8kg) and sizes. The pickup will also work for boxes on the ground and those starting on a table. We refer readers to the attached video for hardware experiments.

To get the pose of the box during hardware experiments we utilize Digit's on board RGBD camera and an ArUco marker placed on the box. The box size and mass policy input is hardcoded beforehand. During hardware execution, skill transitions, navigation, and walking commands are handled by a human operator.

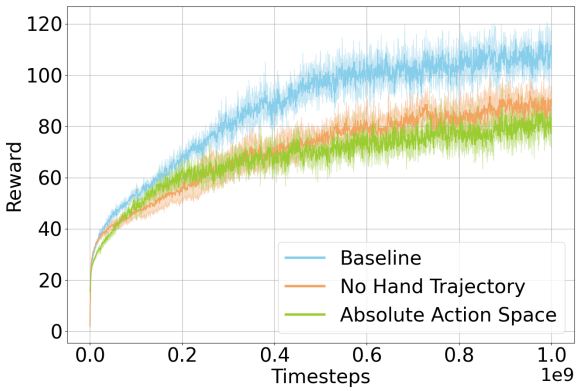

Fig. 4: Reward curve comparison between different learning setups.

Despite the success, we do observe sim-to-real issues on hardware. In particular, we observe policies leaning towards one direction during hardware execution. We suspect this is due to imu/orientation estimate differences on hardware. To counteract this, we offset the orientation input in the same direction as the observed leaning. While this "center-of-mass trim" only needed turning once, it was policy specific. We also found that the selected trim values did not work for the heaviest boxes (8kg), where we observed the policy leaning forward, requiring an increase in the trim. We hypthosize that this error might be caused by force or contact inconsistencies between simulation and hardware.

VII. Conclusion

We presented a learned system for performing box loco-manipulation on the Digit humanoid robot. We broke box loco-manipulation down into 5 different behaviors and learn separate policies for each. We were able to learn policies that achieve high performance in simulation for boxes of varying masses, sizes, and starting locations. We also demonstrated sucessful sim-to-real transfer of those polcies for full loco-manipulaiton episodes, involving walking to a box, pick it up, walk to a target location, and setting it down. To our knowledge this is the first such sim-to-real demonstration using fully learned policies. Future work will focus on making our system more autonomous by incorporating planning, navigation, and vision systems to remove the reliance on a human operator and ArUco markers. There is also significant manual design in our reward setup. Many parts of the pickup are hand chosen, like contact timing and location. Working to remove this would help learn a more general skill and could maybe expand to picking up other objects. There is also room to improve our sim-to-real transfer to hardware. For this we will consider both enhancing the domain randomization using in simulation based training, and integrating real-world hardware data from successful and failed trials into the learning process.

REFERENCES

[1] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5136–5143.

[2] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, pp. 3699–3706, 4 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9028188/

[3] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots." IEEE, 5 2021, pp. 2811–2817. [Online]. Available: https://ieeexplore.ieee.org/document/9560769/

[4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, p. eabc5986, 10 2020. [Online]. Available: http://robotics.sciencemag.org/content/5/47/eabc5986.abstract

[5] 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.

[6] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3651–3657.

[7] C. Wang, Q. Zhang, Q. Tian, S. Li, X. Wang, D. Lane, Y. Petillot, and S. Wang, "Learning mobile manipulation through deep reinforcement learning," *Sensors*, vol. 20, p. 939, 2 2020.

[8] C. Sun, J. Orbik, C. M. Devin, B. H. Yang, A. Gupta, G. Berseth, and S. Levine, "Fully autonomous real-world reinforcement learning with applications to mobile manipulation," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 308–319. [Online]. Available: https://proceedings.mlr.press/v164/sun22a.html

[9] H. Arisumi, J.-R. Chardonnet, A. Kheddar, and K. Yokoi, "Dynamic lifting motion of humanoid robots." IEEE, 4 2007, pp. 2661–2667.

[10] K. Harada, S. Kajita, H. Saito, M. Morisawa, F. Kanehiro, K. Fujiwara, K. Kaneko, and H. Hirukawa, "A humanoid robot carrying a heavy object." IEEE, 2007, pp. 1712–1717.

[11] A. Settimi, D. Caporale, P. Kryczka, M. Ferrati, and L. Pallottino, "Motion primitive based random planning for loco-manipulation tasks," *IEEE-RAS International Conference on Humanoid Robots*, pp. 1059–1066, 2016.

[12] P. Ferrari, M. Cognetti, and G. Oriolo, "Humanoid whole-body planning for loco-manipulation tasks," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017, pp. 4741–4746.

[13] S. Karumanchi, K. Edelberg, I. Baldwin, J. Nash, J. Reid, C. Bergh, J. Leichty, K. Carpenter, M. Shekels, M. Gildner, D. Newill-Smith, J. Carlton, J. Koehler, T. Dobreva, M. Frost, P. Hebert, J. Borders, J. Ma, B. Douillard, P. Backes, B. Kennedy, B. Satzinger, C. Lau, K. Byl, K. Shankar, and J. Burdick, "Team robosimian: Semi-autonomous mobile manipulation at the 2015 darpa robotics challenge finals," *Journal of Field Robotics*, vol. 34, pp. 305–332, 3 2017.

[14] K. Harada, S. Kajita, F. Kanehiro, K. Fujiwara, K. Kaneko, K. Yokoi, and H. Hirukawa, "Real-time planning of humanoid robot's gait for force-controlled manipulation," *IEEE/ASME Transactions on Mechatronics*, vol. 12, pp. 53–62, 2 2007.

[15] S. Sato, Y. Kojio, K. Kojima, F. Sugai, Y. Kakiuchi, K. Okada, and M. Inaba, "Drop prevention control for humanoid robots carrying stacked boxes." IEEE, 9 2021, pp. 4118–4125.

[16] M. Seo, S. Han, K. Sim, S. H. Bang, C. Gonzalez, L. Sentis, and Y. Zhu, "Deep imitation learning for humanoid loco-manipulation through human teleoperation," 2023.

[17] J. Liu, H. Sim, C. Li, and F. Chen, "Birp: Learning robot generalized bimanual coordination using relative parameterization method on human demonstration," 2023.

[18] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter, "Combining learning-based locomotion policy with model-based manipulation for legged mobile manipulators," *IEEE Robotics and Automation Letters*, vol. 7, pp. 2377–2384, 4 2022.

[19] C. Sun, J. Orbik, C. M. Devin, B. H. Yang, A. Gupta, G. Berseth, and S. Levine, "Fully autonomous real-world reinforcement learning with applications to mobile manipulation," A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 9 2022, pp. 308–319. [Online]. Available: https://proceedings.mlr.press/v164/sun22a.html

[20] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: Learning a unified policy for manipulation and locomotion," 2022.

[21] E. Arcari, M. V. Minniti, A. Scampicchio, A. Carron, F. Farshidian, M. Hutter, and M. N. Zeilinger, "Bayesian multi-task learning mpc for robotic mobile manipulation," 11 2022.

[22] Z. Xie, J. Tseng, S. Starke, M. van de Panne, and C. K. Liu, "Hierarchical planning and control for box loco-manipulation," 6 2023. [Online]. Available: http://arxiv.org/abs/2306.09532

[23] J. Dao, K. Green, H. Duan, A. Fern, and J. Hurst, "Sim-to-real learning for bipedal locomotion under unsensed dynamic loads." IEEE, 5 2022, pp. 10 449–10 455.

[24] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," jul 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[26] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2018, pp. 3803–3810. [Online]. Available: https://ieeexplore.ieee.org/document/8460528/