A constrained RL control policy for contact-rich non-prehensile mobile manipulation

Ioannis Dadiotis[1,2], Mayank Mittal[3,4], Nikos Tsagarakis[1], Marco Hutter[3]

***Abstract*— We develop a learning-based controller for a mobile manipulator to move an unknown object to a desired position and yaw orientation through a sequence of contact-rich pushing actions. The proposed controller for the robotic arm and the mobile base motion is trained using a constrained Reinforcement Learning (RL) formulation. We demonstrate its capability in experiments with a quadrupedal robot equipped with an arm. The learned policy achieves a success rate of 91.35% in simulation and at least 80% on hardware in challenging scenarios. Through our extensive hardware experiments, we show that the approach demonstrates high robustness against unknown objects of different masses, materials, sizes, and shapes. It reactively discovers the pushing location and direction, thus achieving contact-rich behavior while observing only the pose of the object. Additionally, we demonstrate the adaptive behavior of the learned policy towards preventing the object from toppling. Further details for this work can be found in the full paper [1].**

I. Introduction

Moving and reorienting heavy or bulky objects along large and complex real-world pathways requires combining mobility and manipulation. This task is achievable through non-prehensile pushing actions without requiring a dedicated gripper or the need to grasp a handle on the object. Additionally, non-prehensile pushing interaction may yield relative motion between the robot and object at the contact point, *e.g.* contact sliding or relative rotation. This motion necessitates a controller capable of reactively adapting the pushing location and direction by dynamically breaking and making contact at new locations with the object. We refer to this contact-rich behavior as contact switching.

Achieving online contact switching behavior during pushing is challenging with model-based techniques [2], [3] or controllers that rely solely on force/tactile feedback [4], [5]. As a result, recent works leverage Reinforcement Learning (RL) to address contact switching [6] and demonstrate notable robustness against unknown objects [7]. Despite these achievements, the method in [6] is limited to fixed-base manipulators pushing a lightweight, small object on a table. Jeon *et al.* [7] achieve object pushing with the base of a mobile robot, without an arm. In both cases, the policies generate only 2D planar motion commands, which are insufficient for manipulating objects that are prone to toppling (*e.g.* objects with a thin base, large CoM height, or high friction coefficient flooring). To address this limitation, we focus on interacting with objects by pushing them to different 3D locations on their surface, enabling more versatile and stable manipulation.

This project has received funding from the European Union's Horizon Europe Framework Programme under grant agreement No 101070596 (euROBIN), the SNSF NCCR dfab, and an ETH RobotX research grant funded through the ETH Zurich Foundation. Contact: ioannis.dadiotis@iit.it

[1]HHCM lab, IIT, Genoa 16163, Italy
[2]DIBRIS, University of Genoa, Genoa 16145, Italy.
[3]RSL, ETH Zürich, Zürich 8092, Switzerland. [4]NVIDIA.

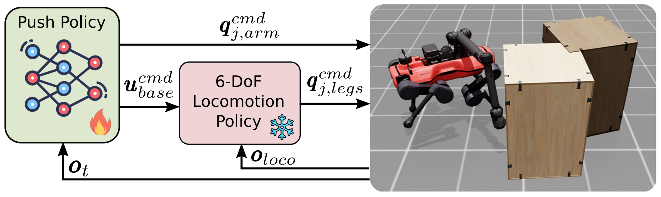


Fig. 1. The control pipeline used for moving and reorienting an object to a planar goal (darker object). Push policy is the proposed mobile manipulation controller. The motions are included in the supplementary video (link).

In this work, we present a learning-based controller for a mobile manipulator to dynamically move and reorient unknown objects using non-prehensile pushing actions. We tackle the task complexity by using a state-of-the-art constrained RL algorithm [8] that minimizes reward engineering efforts and can satisfy the various constraints of the task, *e.g.* arm actuator limits, self-collisions. The control policy is validated across diverse scenarios in simulation and real-world experiments with a quadrupedal manipulator. The policy's action space consists of cartesian commands for the base and joint-space commands for the mounted articulated arm; thus, we directly control the arm in joint space. Even though the proposed control policy only observes the object's 6D pose, it achieves robustness against unknown objects with differing physical properties and learns online contact switching to push the object to various planar goal poses. The resulting behavior demonstrates the adaptability of the pushing location, which is crucial for avoiding object toppling and keeping the object balanced.

II. Method

We train a push policy for mobile manipulators to repose objects on a plane. The policy provides cartesian commands for the mobile base and joint position commands for the first five joints of the arm. We freeze the 6th joint since it is only useful when using a gripper. The velocity commands are sent to a pre-trained locomotion controller to convert them into joint position commands for the legs. Both the push policy and locomotion policy are inferred at the same frequency of 50 Hz. The overall architecture is shown in Fig. 1.

A. Locomotion control

The approach is validated using a quadrupedal mobile platform, ANYmal, with a six DoF robotic arm mounted on it. The locomotion controller is a student policy similar to the one in [9]. It accepts the base command $\boldsymbol{u}_{base}^{cmd} = (v_x,\ v_y,\ \omega_z,\ \zeta,\ \theta,\ h) \in \mathbb{R}^6$ and outputs leg joint position targets. The six components of the command $\boldsymbol{u}_{base}^{cmd}$ consist of linear velocity in x and y directions, angular yaw velocity, roll and pitch angle, and height position, respectively. While training the proposed controller (push policy), we freeze the pre-trained locomotion controller.

B. RL goal pushing environment

We implement the task of moving and reorienting an object using NVIDIA Isaac Lab [10] for training the RL policy with 4096 parallel simulated robots for 20000 iterations. We modified the Proximal Policy Optimization (PPO) implementation from [11] with the changes in [8] to derive the constrained PPO formulation.

Notation: In the following we use $\boldsymbol{p} \in \mathbb{R}^3$, $\boldsymbol{R} \in \mathbb{SO}(3)$, $\boldsymbol{v} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ to denote position, rotation matrix, linear and angular velocity of a body's frame, respectively. The body frame name is denoted as a right subscript and the reference frame as a left superscript (omitted when the reference frame and body frame are the same). We use the letters w, b, o, g, and e to refer to the world, robot base, object, object goal, and arm end-effector frames, respectively. For the relative position between two body frames, two letters are used at the right subscript, *e.g.* ${}^{b}\boldsymbol{p}_{oe}$ is the relative position of the end-effector w.r.t. the object frame expressed in the robot base frame. We use $\hat{\cdot}$ and $\|\cdot\|$ to express a given vector's unit vector and length, respectively.

Environment & commands: The RL training environment consists of multiple object-centered environments (parallelly simulated), each reset when there is an episode timeout (20 sec after the last reset) or when there is an unrecoverable object or robot fall. Fig. 2A shows a single environment after a reset when the robot, object, and object goal positions are resampled in polar coordinates as described in the caption of Fig. 2A. We consider success when the distance between the object's frame and the goal is less or equal to $d_{success} = 10$ cm and the angle between their orientation is less or equal to $\theta_{success} = 10$ deg.

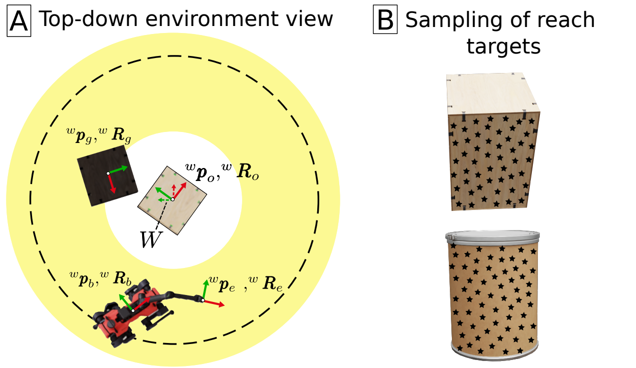
During exploration in simulation, we want to encourage interactions of the robot with the whole surface of the object so that the RL agent discovers which part of the object is better to interact with. To that end, a reaching target position ${}^{w}\boldsymbol{p}_r$ is randomly sampled on the object's vertical surfaces, as shown in Fig. 2B, at each environment reset. This target is used in the reward function to guide the robot EE towards interacting with the object, as explained in Section II-D.

C. Observation & action space

This work uses an asymmetric actor-critic approach [12], [13] where the critic can access privileged information available only in simulation and noiseless. All observed quantities are described in Table I. It is worth noting that the only information regarding the object included in the actor's observation vector $\boldsymbol{o}_t \in \mathbb{R}^{54}$ is the object 6D pose and, thus, the deployed policy has no knowledge about the object size and dynamics. The action vector $\boldsymbol{a}_t = (\Delta\boldsymbol{u}_{base}^{cmd},\ \Delta\boldsymbol{q}_j^{cmd}) \in \mathbb{R}^{11}$ consists of base commands $\Delta\boldsymbol{u}_{base}^{cmd}$ for the locomotion policy described in Section II-A and arm joint position commands $\Delta\boldsymbol{q}_j^{cmd} \in \mathbb{R}^5$. The actions generated by the policy refer to deviations from a default base state (zero velocities, zero orientation, and default height of 0.5 m) and a default arm configuration, respectively. Thus, they are transformed into absolute values before being passed on to the locomotion policy and low-level joint impedance controllers. In Table I, the quantities observed by the critic $\boldsymbol{o}_t^{critic} = (\boldsymbol{o}_t,\ \boldsymbol{o}_t^{pr}) \in \mathbb{R}^{73}$, including the privileged information $\boldsymbol{o}_t^{pr}$.

D. Rewards & constraints

In this section, we describe the reward and constraint terms included in the training. We tune them manually to achieve



A
Top-down environment view
B
Sampling of reach targets
${}^{w}\boldsymbol{p}_g, {}^{w}\boldsymbol{R}_g$
${}^{w}\boldsymbol{p}_o, {}^{w}\boldsymbol{R}_o$
W
${}^{w}\boldsymbol{p}_b, {}^{w}\boldsymbol{R}_b$
${}^{w}\boldsymbol{p}_e, {}^{w}\boldsymbol{R}_e$

Fig. 2. A) The object's position is set to the environment origin (W), the robot base position is randomly sampled within an origin-centered annulus (yellow-shaded area), and the object goal (dark rectangle) within a circular area (dashed line). The robot, object, and goal are spawned with a random yaw orientation. B) Sampling on the object surface encourages interaction with different parts of the object during training.

TABLE I
OBSERVATIONS FOR THE ACTOR ($\boldsymbol{o}_t$) AND CRITIC ($\boldsymbol{o}_t, \boldsymbol{o}_t^{pr}$). UNLIKE THE ACTOR, THE CRITIC RECEIVES NOISELESS OBSERVATIONS.

		Description	Dim	Noise
$\boldsymbol{o}_t$	${}^{b}\boldsymbol{p}_{oe}$	EE-object relative position w.r.t. base	3	$\mathcal{U}(\pm 0.02)$
	${}^{b}\boldsymbol{R}_o$	object rotation matrix w.r.t. base	9	$\mathcal{U}(\pm 0.01)$
	$\Delta\boldsymbol{q}_j$	arm joint position readings w.r.t. default configuration	5	$\mathcal{U}(\pm 0.01)$
	$\boldsymbol{v}_b$	robot base linear velocity	3	$\mathcal{U}(\pm 0.01)$
	$\boldsymbol{\omega}_b$	robot base angular velocity	3	$\mathcal{U}(\pm 0.20)$
	$\dot{\boldsymbol{q}}_j$	arm joint velocity readings	5	$\mathcal{U}(\pm 0.50)$
	${}^{b}\boldsymbol{g}_z$	projected gravity unit vector	3	$\mathcal{U}(\pm 0.05)$
	${}^{b}\boldsymbol{p}_{og}$	object-goal relative position w.r.t. base	3	$\mathcal{U}(\pm 0.02)$
	${}^{o}\boldsymbol{R}_g$	goal orientation w.r.t. object	9	$\mathcal{U}(\pm 0.01)$
	$\boldsymbol{a}_{t-1}$	previous actions	11	-
$\boldsymbol{o}_t^{pr}$	λ_e	EE-object contact state	1	-
	${}^{b}\boldsymbol{p}_{com}$	object CoM position w.r.t. robot base	3	
	m	object mass	1	
	$\boldsymbol{d}$	object dimensions	3	
	$\boldsymbol{I}_o$	object's principal moments of inertia	3	
	${}^{b}\boldsymbol{v}_o$	object linear velocity w.r.t. robot	3	
	${}^{b}\boldsymbol{\omega}_o$	object angular velocity w.r.t. robot	3	
	$\boldsymbol{\kappa}_{sh}$	one-hot vector for object shape	2	

TABLE II
REWARDS AND CONSTRAINTS USED FOR TRAINING ALONG WITH THE INITIAL AND FINAL VALUES OF THE CONSTRAINT HYPERPARAMETER p_i^{max} FROM CAT [8] AND THEIR RESPECTIVE CURRICULUM SCHEDULE OVER THE LEARNING ITERATIONS.

Rewards	**Constraints**				
	Description	**Formulation**	**Dim**	$\boldsymbol{p_i^{\max}}$	**Iterations**
$r_{1,t} = \exp\left(-\frac{\|{}^w\boldsymbol{K}_g - {}^w\boldsymbol{K}_o\|}{\sigma_3^2}\right)$	base command limits	$\boldsymbol{c}_a^{base} = \max(\boldsymbol{u}_{base}^{cmd} - \boldsymbol{u}_{base}^{upper}, \boldsymbol{u}_{base}^{low} - \boldsymbol{u}_{base}^{cmd})$	6	$0.01 \rightarrow 0.2$	$0 \rightarrow 12 \cdot 10^3$
$r_{2,t} = \exp\left(-\frac{\|{}^w\boldsymbol{p}_{er}\|}{\sigma_1^2}\right)$	arm command limits	$\boldsymbol{c}_a^{arm} = \max(\boldsymbol{q}_j^{cmd} - \boldsymbol{q}_j^{upper}, \boldsymbol{q}_j^{lower} - \boldsymbol{q}_j^{cmd})$	5	$0.05 \rightarrow 0.9$	
	arm action rate limits	$\boldsymbol{c}_{\dot{a}}^{arm} = \frac{\|\Delta\boldsymbol{q}_{j,t}^{cmd} - \Delta\boldsymbol{q}_{j,t-1}^{cmd}\|}{dt} - \dot{\boldsymbol{q}}^{lim}$	5	$0 \rightarrow 0.05$	
$r_{3,t} = \exp\left(\frac{{}^w\widehat{\boldsymbol{v}}_o \cdot {}^w\widehat{\boldsymbol{p}}_{og}}{\sigma_2^2} - 1\right)$	arm joint position limits	$\boldsymbol{c}_{q_j} = \max(\boldsymbol{q}_j - \boldsymbol{q}_j^{upper}, \boldsymbol{q}_j^{low} - \boldsymbol{q}_j)$	5	$0.05 \rightarrow 0.9$	
	arm joint velocity limits	$\boldsymbol{c}_{\dot{q}_j} = \|\dot{\boldsymbol{q}}_j\| - \dot{\boldsymbol{q}}^{lim}$	5	$0.05 \rightarrow 0.9$	
	arm joint torque limits	$\boldsymbol{c}_{\tau_j} = \|\boldsymbol{\tau}_j\| - \boldsymbol{\tau}_j^{lim}$	5	$0 \rightarrow 0.015$	
$r_{4,t} = \exp\left(-\frac{\|\Delta\boldsymbol{u}_{base,t}^{cmd} - \Delta\boldsymbol{u}_{base,t-1}^{cmd}\|}{\sigma_{4,b}^2}\right) + \exp\left(-\frac{\|\Delta\boldsymbol{q}_{j,t}^{cmd} - \Delta\boldsymbol{q}_{j,t-1}^{cmd}\|}{\sigma_{4,a}^2}\right)$	leg joint torque limits	$\boldsymbol{c}_{\tau_{j,leg}} = \|\boldsymbol{\tau}_{j,leg}\| - \boldsymbol{\tau}_{j,leg}^{lim}$	12	$0 \rightarrow 0.01$	
	undesired robot-object & self-collisions	$c_{coll} = \begin{cases} 1 & \text{, if a collision occurs,} \\ 0 & \text{, otherwise.} \end{cases}$	18	1.0	No curriculum
	object balance	$c_{\theta_{obj}} = \begin{cases} \|\theta\| - \theta^{lim} & \text{, if } \|{}^b\boldsymbol{v}_b\| > 0, \\ 0 & \text{, otherwise.} \end{cases}$	1	0.25	

convergence in simulation and then transfer the policy to the hardware zero-shot without further adjustments.

Rewards: The total reward $r_t^{tot} = \sum_{n=1}^{4} w_i r_{i,t}$ consists of the sum of the reward terms shown in Table II. The term $r_{1,t}$ is the main task reward, which encourages minimizing the distance between the eight keypoints of the object and the keypoints of the goal, where these are defined as the vertices of the oriented bounding box of the object (similarly to [7], [14]). We denote the position of the keypoints of the object and its goal as ${}^w\boldsymbol{K}_o \in \mathbb{R}^{24}$ and ${}^w\boldsymbol{K}_g \in \mathbb{R}^{24}$, respectively. The reward term $r_{2,t}$ encourages the agent to minimize the distance between the arm EE and the reach target (${}^w\boldsymbol{p}_r$) sampled on the object's surface at each episode. The weight w_2 of this term is downscaled by a factor of 4 after 1500 learning iterations. We do this to encourage the robot EE to approach and interact with different parts of the object at the beginning, and we do not care about accurately reaching the sampled position. The term $r_{3,t}$ rewards object linear velocity with direction towards the object goal. We do not include the magnitude of the velocity in this term to avoid the robot pushing the object aggressively. Finally, the term $r_{4,t}$ comprises action rate regularization. If the task is successful, we increase the task reward to $r_{1,t} = 2$, which is two times the maximum possible value of this term, so that the robot learns to achieve the specified tolerance instead of staying close to it.

Constraints: As proposed in [8], we apply curriculum learning over most of the constraints by increasing the maximum probability for reward termination along training (linearly increasing $p_i^{\max}$ during training). In practice, at the beginning of the training, we encourage exploration by limiting the effect of constraint violation on the reward termination probability. We emphasize achieving strict constraint satisfaction for undesired collisions, arm joint position, and velocity limits. The undesired collisions include robot self-collisions and collisions of the object with the robot base, legs, and the arm's shoulder, upper arm, elbow, and forearm links. We also include a constraint for the base command $\Delta\boldsymbol{u}_{base}^{cmd}$ using as limits the ranges used during the training of the locomotion controller. Finally, the object balance constraint requires that the object's inclination angle be less than a specified threshold $\theta^{lim} = 10°$. All the constraints used for training, their dimension, and the curriculum applied are shown in Table II.

E. Domain randomization & deployment

To render the learned policy robust for deployment on the hardware, we randomize several factors in the simulated environment, including the object-floor friction, the object's mass, center-of-mass (CoM), dimensions, and shape (cuboids and cylinders). The actor's observations are subject to additive uniform noise, as specified in Table I, and the robot's mass and initial joint positions are, as well, randomized around nominal values.

During deployment, we rely on an external motion capture system to get the object and robot base 6D pose information needed to derive the observation $\boldsymbol{o}_t$.

III. RESULTS & EVALUATION

A. Robustness against unknown objects & contact switching

The trained policy achieves a success rate of 91.35%, evaluated in simulation for 4096 runs. We extensively test the controller on the hardware to move and reorient different objects. The tests were conducted on the protective floor mats of our testing area, which have high friction and can even exhibit small gaps along the seams of the mat. This renders the task more challenging. In the next paragraph, we present the success rates for these tests.

We command the robot to sequentially move and reorient the object between two fixed goal poses in the space, as shown in Fig. 3. We do not manually move the robot before sending a new object goal; the policy successfully moves the robot to the appropriate side of the object to push in the correct direction. We tested the learned controller with objects of varying mass, size, shape, and material (Table III), with yaw angle differences $\Delta\theta_z$ of 0°, 90°, or 180° between

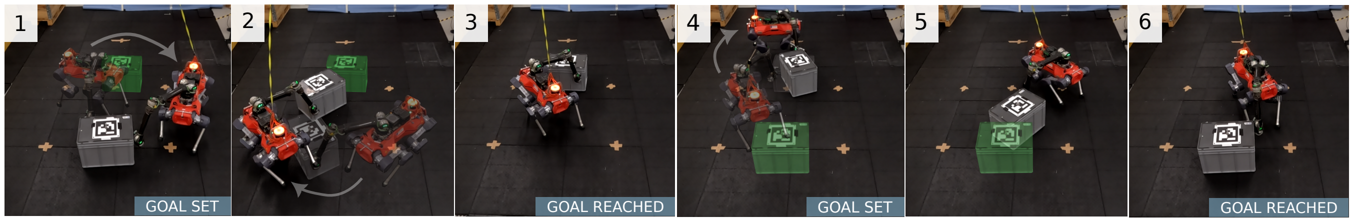


Fig. 3. Experimental validation of the proposed controller for sequentially moving and re-orienting an object between two goal poses. The robot pushes a plastic box of 6.4 kg from one goal to another. The goal poses are shown as green boxes. Snapshots of previous times are shown with lower opacity. The robot successfully goes around the object to push from the correct side towards the goal (1-2, 5).

TABLE III

SUCCESS RATE DURING HARDWARE EXPERIMENTS WITH OBJECTS OF DIFFERENT MATERIAL: PLASTIC (P), CARDBOARD (C), WOOD (W) AND DIFFERENT SHAPE: CUBOID (CU), CYLINDER (CY)

Object	**Mass [kg]**	**Size [cm^3]**	**$\Delta\theta_z$ [deg]**	**# of face switches / goal**	**Success rate [%]**
P-CU	6.43	60x34x40	180	0.90	91.6
C-CU	5.30	50x50x53	0	0.23	92.9
C-CU	8.32	50x50x53	90	0.75	83.3
C-CU	4.5	100x50x53	0	0.14	80.0
W-CU	6.30	40x40x60	180	1.00	91.6
C-CU	13.30	50x50x60	0	4.80	83.3
C-CY	2.45	Φ30x40	0	-	83.3

the two goal poses. As goals are sent sequentially, the yaw angle difference between the object and the new goal matches $\Delta\theta_z$ ($\pm$ the success tolerance). The policy achieves a success rate of at least 80%. For the cuboids, we report in Table III the average contact face switches per goal. A face switch is considered when the robot switches contact to a different face of the cuboid. It can be seen that for higher object yaw orientation changes $\Delta\theta_z$, the controller makes more contact and face switches to properly align the object with the goal, while for $\Delta\theta_z = 0$ the robot can most of the time achieve the task without face switch. For cylindrical shapes, a face switch cannot be defined; contact switching is still observed in any case. The contact-rich behavior of the controller for all objects can be seen in the accompanying video.

B. *Adaptability to object size*

As mentioned in Section II-E, the dimensions of the objects during training were randomized. We investigate the adaptability of the policy with respect to the object's xy-footprint since thin objects can be prone to toppling on high-friction surfaces. To that end, we select six object xy-footprint sizes equally distributed across the training range and simulate the policy for 1000 successful episodes per size. We fix the object height, mass, center of mass, and friction values to constant and disable the additive observation noise to evaluate the effect of the object's base. In Fig. 4, we report the height distribution of the robot EE expressed in the world frame while in contact with the object. The policy learns to push lower for objects that have smaller bases. It is of particular interest that the policy does not observe any explicit information regarding the object size or dynamics. This implies that the robot's adaptive behavior is based on the object 6D pose observation. In practice, the robot adapts

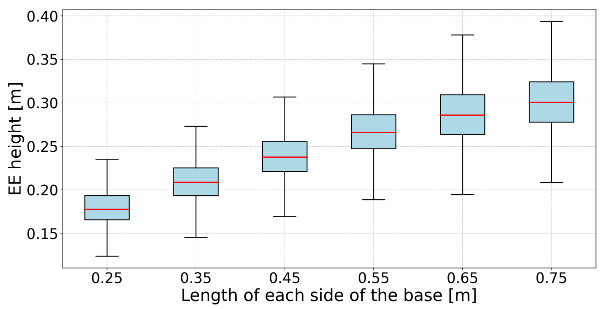


Fig. 4. Boxplot of the arm EE height for objects with rectangular bases of different sizes. The policy pushes lower for objects with smaller base.

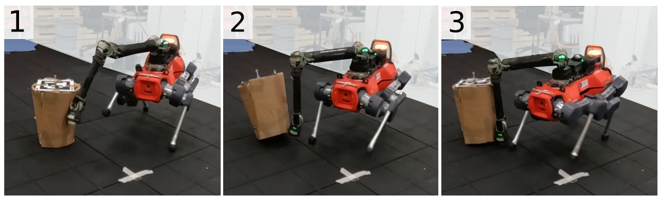


Fig. 5. Arm EE height, base height, and orientation while pushing a thin cylinder. The shaded region consists of the time when the cylinder is tilted due to an initial push. The base height and orientation (pitch down, roll) contribute towards pushing power immediately after the object tilts.

the pushing location to lower when the object is inclined. We observed such behavior while testing on the real hardware. As shown in Fig. 5, the policy can approach and push a thin cylinder on a flooring of high-friction mats. When the cylinder starts tilting, the robot reactively changes the pushing location to lower and avoids object toppling.

IV. CONCLUSION

We proposed a constrained RL-based controller for dynamically moving and reorienting objects with a mobile manipulator, which is a relatively long mobile manipulation task. The generated motion behaviors are characterized by online contact switching and robustness concerning unknown objects of different mass, size, and shape on a high friction floor. By only relying on object pose information, the controller changes the object pushing location to lower for thin objects on high-friction surfaces.

ACKNOWLEDGMENTS

We gratefully acknowledge the advice of Yuntao Ma for policy deployment, Takahiro Miki for the locomotion policy, and Elliot Chane-Sane for the constrained PPO [8].

References

[1] I. Dadiotis, M. Mittal, N. Tsagarakis, and M. Hutter, "Dynamic object goal pushing with mobile manipulators through model-free constrained reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2502.01546

[2] F. R. Hogan and A. Rodriguez, "Reactive planar non-prehensile manipulation with hybrid model predictive control," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 755–773, 2020.

[3] J. Moura, T. Stouraitis, and S. Vijayakumar, "Non-prehensile planar manipulation via trajectory optimization with complementarity constraints," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 970–976.

[4] A. Heins and A. P. Schoellig, "Force push: Robust single-point pushing with force feedback," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 6856–6863, 2024.

[5] I. Ozdamar, D. Sirintuna, R. Arbaud, and A. Ajoudani, "Pushing in the dark: A reactive pushing strategy for mobile robots using tactile feedback," *IEEE Robotics and Automation Letters*, 2024.

[6] J. D. A. Ferrandis, J. Moura, and S. Vijayakumar, "Nonprehensile planar manipulation through reinforcement learning with multimodal categorical exploration," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 5606–5613.

[7] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo, "Learning whole-body manipulation for quadrupedal robot," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 699–706, 2024.

[8] E. Chane-Sane, P.-A. Leziart, T. Flayols, O. Stasse, P. Souères, and N. Mansard, "Cat: Constraints as terminations for legged locomotion reinforcement learning," *arXiv preprint arXiv:2403.18765*, 2024.

[9] T. Miki, J. Lee, L. Wellhausen, and M. Hutter, "Learning to walk in confined spaces using 3d representation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 8649–8656.

[10] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar *et al.*, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[11] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[12] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *arXiv preprint arXiv:1710.06542*, 2017.

[13] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 149–12 155.

[14] A. Allshire, M. Mittal, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg, "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 11 802–11 809.