frontiers | Frontiers in Neurorobotics

TYPE Original Research
PUBLISHED 14 September 2023
DOI 10.3389/fnbot.2023.1271607

OPEN ACCESS

EDITED BY
Alois C. Knoll,
Technical University of Munich, Germany

REVIEWED BY
Chen Chen,
Harbin University of Science and Technology, China
Weida Li,
Soochow University, China

*CORRESPONDENCE
Fusheng Zha
✉ zhafusheng@hit.edu.cn
Pengfei Wang
✉ wangpengfei1007@163.com

RECEIVED 02 August 2023
ACCEPTED 31 August 2023
PUBLISHED 14 September 2023

CITATION
Wang S, Sun L, Zha F, Guo W and Wang P (2023) Learning adaptive reaching and pushing skills using contact information. *Front. Neurorobot.* 17:1271607. doi: 10.3389/fnbot.2023.1271607

Learning adaptive reaching and pushing skills using contact information

Shuaijun Wang, Lining Sun, Fusheng Zha*, Wei Guo and Pengfei Wang*

State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin, China

In this paper, we propose a deep reinforcement learning-based framework that enables adaptive and continuous control of a robot to push unseen objects from random positions to the target position. Our approach takes into account contact information in the design of the reward function, resulting in improved success rates, generalization for unseen objects, and task efficiency compared to policies that do not consider contact information. Through reinforcement learning using only one object in simulation, we obtain a learned policy for manipulating a single object, which demonstrates good generalization when applied to the task of pushing unseen objects. Finally, we validate the effectiveness of our approach in real-world scenarios.

KEYWORDS

reinforcement learning, pushing, contact information, adaptivity, task efficiency

1. Introduction

Robotic Pushing is a fundamental manipulation skill in the field of robotics, finding utility in various scenarios such as scene rearrangement (Zeng et al., 2018), object manipulation, and environmental interaction. Efficient and adaptive object pushing skills are crucial for autonomous robotic systems to interact with their surroundings effectively (Bauza et al., 2018). To achieve successful object pushing, two main approaches have been explored in past research (Stüber et al., 2020): analytical-based methods and data-driven methods. While analytical methods rely on explicit knowledge of the forward and inverse kinematics and dynamics models of manipulation, data-driven approaches leverage machine learning techniques to acquire pushing skills through experience.

Analytical-based methods (Yoshikawa and Kurisu, 1991; Howe and Cutkosky, 1996; Dogar and Srinivasa, 2010; Zhu et al., 2017) require a clear understanding of the object's physical properties, including mass, center of mass, friction coefficients, and other contact parameters. Zhou et al. (2019) demonstrates the differentially flat nature of quasi-static pushing with sticking contact and an ellipsoid approximation of the limit surface. The pusher-slider system is effectively reducible to the Dubins car problem with bounded curvature, enabling easy trajectory planning and time-optimality. The paper proposes closed-loop control with dynamic feedback linearization or open-loop control using mechanical feedback from two contact points for trajectory stabilization. Lynch (1993)proposes a method for a robot to plan object manipulation through pushing. The approach involves estimating the geometry and friction properties of the object by conducting experimental pushes and observing the resulting motion. Additionally, the paper explores object recognition based on their distinctive friction parameters. The focus is on developing a practical framework for the robot to understand and interact with objects in its environment by learning their friction-related characteristics. Lynch and Mason (1996)

addresses the problem of planning stable pushing paths for robots to position and orient objects in the plane, especially when grasping and lifting are not feasible due to the objects' size or weight. The study focuses on stable pushing directions and the controllability issues arising from non-holonomic velocity constraints, presenting a planner for finding such paths amidst obstacles, with practical demonstrations on various manipulation tasks. In Lloyd and Lepora (2021) researchers propose a reactive and adaptive method for robotic pushing using high-resolution optical tactile feedback instead of analytical or data-driven models. The method demonstrates accurate and robust performance on planar surfaces, and highlights the need to explore explicit models and test on non-planar surfaces for improved generalization. By explicitly modeling these characteristics, analytical methods can achieve precise control and manipulation of specific objects (Hogan et al., 2018). However, these methods often lack the adaptability to handle novel objects whose properties are not known a priori. Consequently, they struggle to generalize their pushing skills to diverse and previously unseen objects.

On the other hand, data-driven approaches (Bauza and Rodriguez, 2017; Eitel et al., 2020; Song and Boularias, 2020; Yu et al., 2023) have achieved significant attention in recent years, primarily due to their ability to adapt to different objects without prior knowledge of their physical properties. These approaches utilize machine learning algorithms, such as supervised learning (Li et al., 2018) and reinforcement learning (Raffin et al., 2021; Cong et al., 2022), to acquire pushing skills from experience. Supervised learning techniques can estimate unknown object properties (Mavrakis et al., 2020), such as the center of mass, while reinforcement learning enables the learning of continuous control policies for pushing actions (Xu et al., 2021). A novel neural network architecture for learning accurate pushing dynamics models in tabletop object manipulation tasks is proposed in Kim et al. (2023). The proposed model possesses the desirable SE(2)-equivariance property, ensuring that the predicted object motions remain invariant under planar rigid-body transformations of object poses and pushing actions. Extensive empirical validations demonstrate that this new approach outperforms existing data-driven methods, leading to significantly improved learning performances. In Kumar et al. (2023), Inverse Reinforcement Learning (IRL) is explored to learn the reward instead of designing it by humans, achieving better performance in the pushing tasks. Chai et al. (2022) introduces a novel large planar pushing dataset encompassing various simulated objects and a new representation for pushing primitives, facilitating data-driven prediction models. Additionally, it proposes an efficient planning method with a heuristic approach to minimize sliding contact between the pusher and the object, addressing challenges in reasoning due to complex contact conditions and unknown object properties. Paus et al. (2020) proposes an approach for parameterizing pushing actions in robotic tasks using internal prediction models. The method involves representing scenes as object-centric graphs, training the internal model on synthetic data, and evaluating it on real robot data to achieve high prediction accuracy and generalization to scenes with varying numbers of objects. Data-driven approaches have shown promising results in handling a wide range of objects, exhibiting better generalization and adaptability compared to analytical methods.

Reinforcement learning (RL) methods have recently gained popularity in low-level robot control tasks (Singh et al., 2022; Elguea-Aguinaco et al., 2023), as they enable the learning of task controllers using low-dimensional data as inputs, which are often easier to learn policies from compared to high-dimensional data, such as images. These pieces of information can be incorporated into RL as observations and can also be utilized in the design of RL reward functions. In this paper, we present a novel approach for object pushing based on deep reinforcement learning. Our method aims to efficiently push unknown objects from random initial poses to predetermined target positions using the contact information in the design of the reward function. The robot experiment system is shown in Figure 1.

We leverage contact force information between the pusher and the object to design an effective reward function that guides the learning process. Contact force information plays a critical role in pushing tasks, as it provides valuable cues about the interaction dynamics and the effectiveness of the applied forces. Surprisingly, contact force information has been underutilized in previous reinforcement learning-based pushing tasks (Dengler et al., 2022), especially in the reward design phase.

To address this limitation, we introduce appropriate observations and continuous action outputs that enable the agent to perform continuous and adaptive pushing control (Shahid et al., 2020). Our reward function design encourages the alignment between the direction of the contact force and the line connecting the object and the target. Furthermore, we incentivize minimizing the distance between the object's center of mass and the contact point where the force is applied. By doing so, we effectively encourage pushing actions that generate zero rotational torque, resulting in pure translation.

In our evaluation, we conducted extensive experiments in a simulated environment. Remarkably, our approach achieved higher task success rates, improved task completion efficiency, and demonstrated superior generalization capabilities compared to previous methods that did not consider contact force information. Notably, our training process utilized only one object, yet the learned policy successfully generalized to other unseen objects without prior knowledge of their physical characteristics. These findings demonstrate the effectiveness and potential of our method for real-world applications involving diverse objects and environments.

By leveraging contact force information and employing deep reinforcement learning, our proposed method contributes to improving adaptive reaching and pushing skills in robotics. The integration of contact force information in the reward function design provides valuable insights into the physical interactions involved in object pushing, leading to more precision and adaptive pushing policy.

In summary, our work highlights the importance of contact force information in object pushing tasks and presents a novel approach that effectively utilizes this information in the reward design. By combining deep reinforcement learning (DRL) and contact force guidance, we achieve improved task success rates, task completion efficiency, and generalization capabilities.

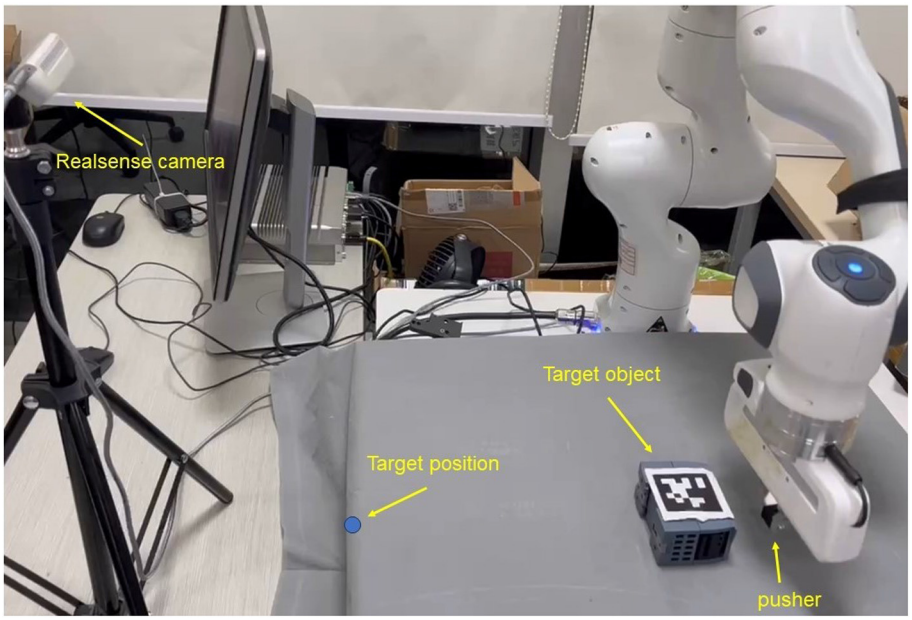

FIGURE 1
The robot pushing system: the object pose is detected by Apriltag techniques, and the pusher is the end effector of the franka robot. The objective of the task is for the pusher to move the object to the target position through pushing.

Our research contributes to the ongoing efforts in enhancing robotic manipulation skills and lays the foundation for future advancements in autonomous systems interacting with diverse objects and environments.

The contributions of this paper are listed as follows:

- We introduce a RL-based reaching and pushing framework that uses only one object during training, allowing the trained policy to adapt effectively to unseen objects;
- We consider contact information in the design of the reward function to enhance task performance;
- We demonstrate that the learned policy, developed with our proposed approach, achieves higher success rates, task efficiency, and adaptability compared to policies that do not consider contact information.

2. Preliminaries

We model the object pushing task as a Markov decision process (MDP), which is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where:

$\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T}$ is the transition function that specifies the probability of moving to a new state s' given the current state s and action a taken, i.e.,

$$\mathcal{T}(s, a, s') = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a) \tag{1}$$

$\mathcal{R}$ is the reward function that maps a state-action pair (s, a) to a scalar reward $r \in \mathbb{R}$, and γ is the discount factor that determines the importance of future rewards. The goal of our learning system is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward:

$$\sum_{t=0}^{\infty} \gamma^t R_{t+1} \tag{2}$$

where R_{t+1} is the reward obtained at time $t+1$, and γ^t is the discount factor applied to future rewards at time t.

In this work, the SAC reinforcement learning method is applied. Soft Actor-Critic (SAC) Haarnoja et al. (2018) is a reinforcement learning algorithm that can be used to solve Markov decision processes (MDPs), which are mathematical models for decision-making problems. The goal of an agent in an MDP is to maximize its expected cumulative reward by taking actions in an environment.

SAC is based on the actor-critic architecture, which consists of an actor network that selects actions and a critic network that estimates the value of a state or state-action pair. However, SAC introduces several improvements over the traditional actor-critic algorithm, such as the use of a soft value function and entropy regularization.

The soft value function is defined as the expected sum of rewards plus a temperature-scaled entropy term. The entropy term encourages exploration and prevents premature convergence by penalizing deterministic policies. The temperature parameter can be learned during training or set manually.

The entropy regularization term is added to the actor objective to encourage exploration and to prevent the policy from becoming too confident. The overall objective of SAC can be written as:

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r_t + \alpha \mathcal{H}(\pi_\theta(\cdot|s_t)) \right] - \mathbb{E}_{s_t \sim \mathcal{D}} \left[V_\phi(s_t) \right] \tag{3}$$

where θ and ϕ are the parameters of the actor and critic networks, respectively, τ is a trajectory sampled from the policy π_θ, r_t is the reward at time step t, α is the temperature parameter, $\mathcal{H}$ is the entropy function, and $\mathcal{D}$ is the replay buffer containing past experiences.

The first term in the objective is the expected cumulative reward with entropy regularization, and the second term is the value function approximation error. The parameters are updated using gradient descent on the objective.

3. Methods

3.1. Proposed method framework

In this work, we propose a manipulation framework namely "pusher", which is an RL-based method to achieve objects reaching and pushing manipulation skills. The framework can be seen in Figure 2.

Our approach involves training a deep neural network policy using the Soft Actor-Critic algorithm (SAC) to learn continous pushing behaviors from sensory input. Specifically, we consider the contact information in the procedures of interaction with the environment in the reward design part.

Our framework involves three parts: actor-network, robot simulation environment and critic-network. In the training phase, we use a physics simulator to generate a dataset of pushing movements for a single object. We then use data collected in the progress to train a neural network policy (actor-network). During training, the policy learns to map raw sensory input to pushing actions that achieve the desired goal. The actor-network and critic network parameters are updated with the interaction with the environment.

The control framework can be seen in Figure 3. The robot state and object state includes the end-effector position, object position, distance from object to target, and distance from end-effector to object. They are treated as input of the learned policy which is represented by the actor-network. The output of the policy is the action of the SAC algorithm. Robot pose is updated in frequency of 20 Hz. Inverse kinematics is applied to update the joint positions which are sent to the real robot in the real world or simulation.

3.2. State and action space

The observation space includes the position of the target object, the position and orientation of the end effector of the robot arm, as well as the distance between the target object and the end effector, and the distance between the target object and the target location. The action space is defined by the increments in the position and orientation of the end effector. Specifically, in this paper it is constrained to the planar space, specifically in the x and y directions, as well as the angle of rotation around the z-axis.

The observation space and action space of pushing can be formulated as follows:

$$\mathcal{O}push = [d_1, d_2, \theta, p_x, p_y, e_x, e_y] \tag{4}$$

where d_1 represents the distance of target object and end effector of the manipulator, d_2 represents the distance of target object and target position. θ is rotation degree in z-aisx, p_x, p_y is the position of target object, and e_x, e_y is the position of the end effector of the manipulator.

The action space of pushing can be formulated as follows, which is an incremental action:

$$\mathcal{A} = [\Delta x, \Delta y, \Delta\theta] \tag{5}$$

where Δx, Δy represents the position increment of the end effector, and $\Delta\theta$ represents the orientation increment of the end effector. Once we obtain the incremental output from our policy, we add them to current robot position and rotation to update the control signals of the end effector in cartesian space.

3.3. Reward design

In reinforcement learning-based frameworks, the reward function plays a crucial role as it guides the learning policy toward desired states. In the context of manipulation tasks in this paper, which involves extensive interaction with the environment, considering contact information in the reward function becomes essential. Surprisingly, this aspect has received limited attention in previous reinforcement learning-based approaches.

Firstly, let us introduce the contact model employed in our work. The contact model for pushing is represented in a straightforward manner, as depicted in Figure 4. It comprises the Center of Mass (CoM) of the target objects, their radius, and the orientation of the contact force. Notably, in this paper, we do not explicitly consider the magnitude of the contact force; Only the force direction is utilized.

In contrast to merely observing the state of the center of mass before and after the robot's interaction, incorporating contact geometric information at the time of contact can provide more precise and valuable insights. These geometric parameters offer advantages beyond determining the proximity of the target object to the desired position. They also contribute to a more accurate prediction of the potential closeness or distance of the target object to the desired state.

The mathematical representation of the reward functions for pushing in our work can be expressed as:

$$\text{Reward} = \begin{cases} -\theta - \lambda \times d3, & \text{if contact} \\ e^{-d3} - 2, & \text{otherwise} \end{cases} \tag{6}$$

In the formula, θ represents the angle between the contact force direction and the line connecting the object's center of mass to the target position. $d3$ refers to the distance from the object's center of mass to the contact force direction, which can be considered as the lever arm. λ is set to 30 in this work. As evident from this design, when there is no contact, we encourage the pusher and the object to make contact. Conversely, when there is contact, we incentivize the contact force to act through the object's center of mass, aligned with the line connecting the object to the target position, effectively encouraging the generation of zero rotational

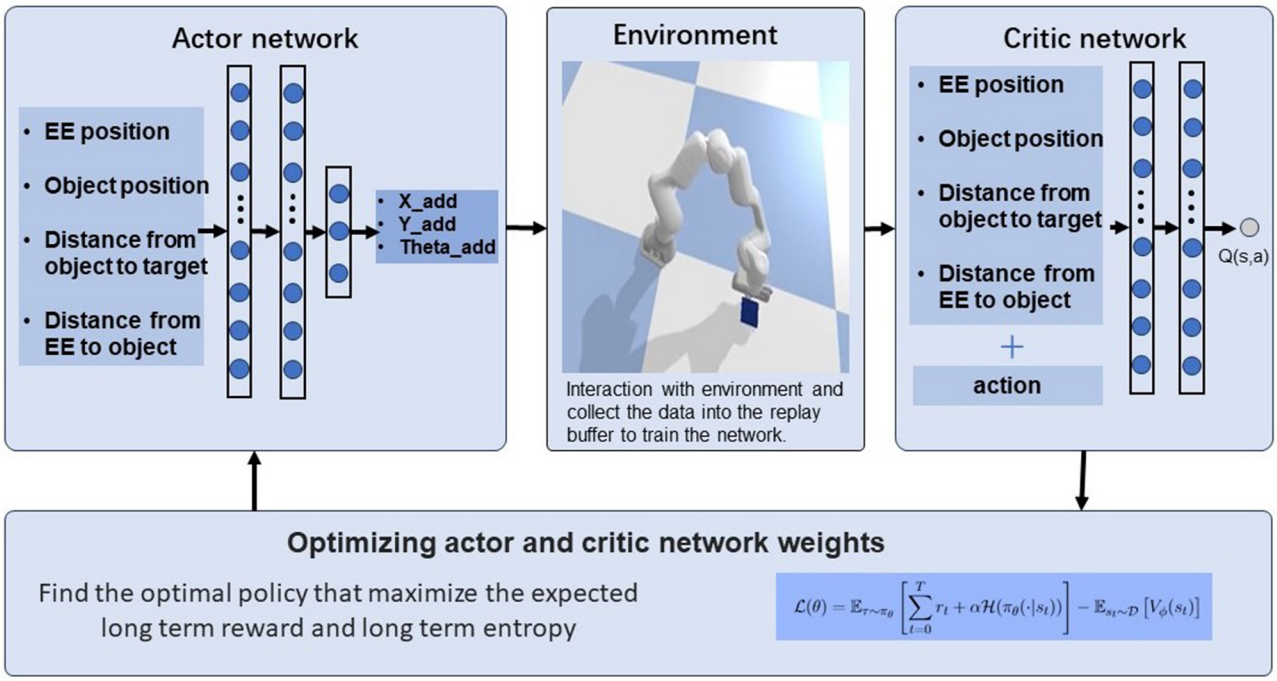

FIGURE 2
Framework of deep reinforcement learning for adaptive pushing using contact information.

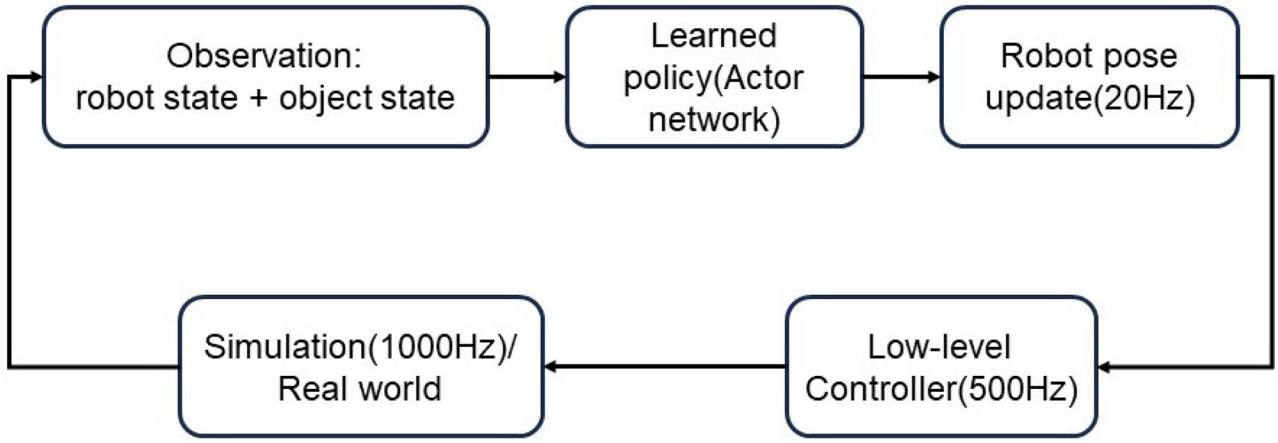

FIGURE 3
The control framework in the simulation, real-world, and joint-level control.

torque. This preference leads to the most direct path to push the object toward the target point.

3.4. Domain randomization and training procedures

In our study, we employed domain randomization to enhance the generalization of the trained reinforcement learning (RL) models. Specifically, we randomized the object's mass, friction coefficient, and initial position and orientation.

The reinforcement learning training process was conducted using the Soft Actor-Critic (SAC) algorithm. The hyperparameters used in the training were set as follows: a batch size of 100, a learning rate of 0.001, replay buffer is 1e6, discount factor is 0.99, interpolation factor in polyak averaging for target networks is 0.995, entropy regularization coefficient is 0.2. The PyBullet physics engine (Coumans and Bai, 2016) as the simulator, PyTorch as the learning framework, and Python as the programming language.

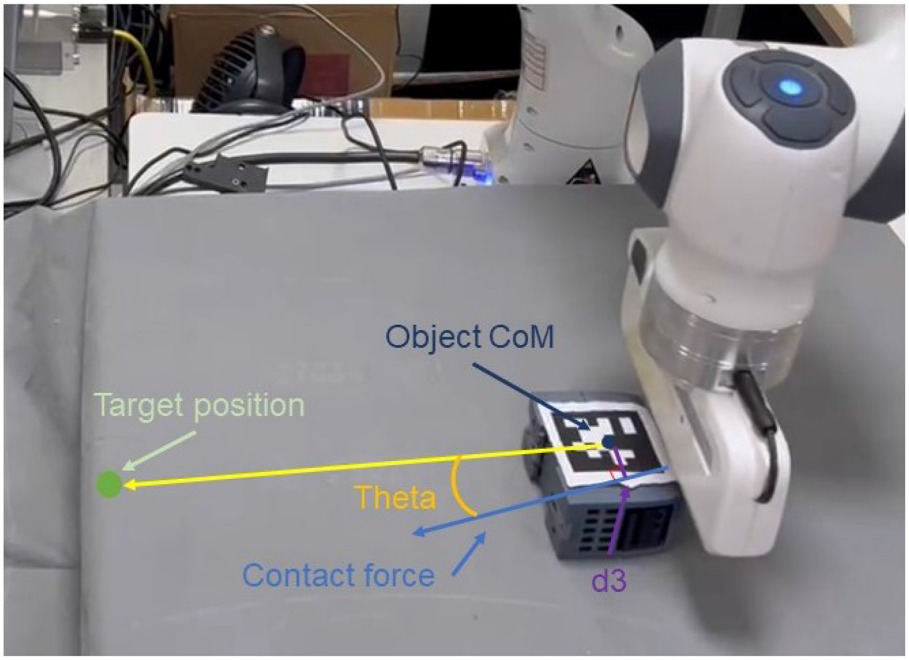

FIGURE 4
Contact model of interaction between object and pusher.

TABLE 1 Domain randomization parameters for pushing tasks.

Task	Parameter	Range
Pushing	Object mass	(0.2, 0.5) kg
	Object friction	(0.5, 1.0)
	Object position	[(0.1, 0.2), (0.45, 0.55)] m
	Object orientation	(−0.15, 0.15) rad

The Table 1 shows the range of values used for domain randomization.

We adopt the soft actor-critic (SAC) algorithm to learn the pushing tasks under the domain randomization framework. The training process involves the following steps as shown in the pseudocode implementation of the training procedure in Algorithm 1.

Input: environment E, policy π, replay buffer D
Output: optimized policy π^*

while *not converged* **do**
  sample mini-batch of transitions B from D
  for $i \leftarrow 1$ **to** n **do**
    sample action $a_i \sim \pi(\cdot|s_i)$
    calculate target $y_i = r_i + \gamma Q_{\theta^-}(s_{i+1}, \pi_{\phi^-}(s_{i+1}))$
    update critic parameters θ with respect to mean-squared Bellman error:
    $\mathcal{L} = \frac{1}{|B|}\sum_{i=1}^{|B|}(y_i - Q_\theta(s_i, a_i))^2$
    update actor parameters ϕ using the sampled policy gradient:
    $\nabla_\phi J \approx \frac{1}{|B|}\sum_{i=1}^{|B|}\nabla_a Q_\theta(s_i, a)|_{a=\pi_\phi(s_i)}\nabla_\phi \pi_\phi(s_i)$
    update target network parameters:
    $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$
    $\phi^- \leftarrow \tau\phi + (1-\tau)\phi^-$
  end
end

Algorithm 1. Soft actor-critic (SAC) training algorithm.

4. Experimental results and analysis

4.1. Experiments setup

The simulation and real experiments are conducted in this part to demonstrate the proposed method's performance. For simulation, we randomized several parameters of the objects to create a diverse training environment. Specifically, we randomized the mass, friction coefficient, and initial position of the objects. The mass and friction coefficient were randomly sampled from predefined ranges, and the initial position of the objects was randomly set within a specified region. This randomization process helps to introduce variability in the objects' properties, making the training environment more challenging and realistic. The use of randomized parameters allows the policy to learn to adapt to different object configurations, which can improve the policy's generalization performance.

As can be seen in Figure 1, in real experiments, an external camera is applied to obtain the pose of the target object. This is completed by the hand-eye calibration method (Tsai et al., 1989), transferring the target object pose in the camera coordination to the world coordinate. We use the Apriltag (Olson, 2011) attached to the target object to detect the pose of objects. For real experiments, we

selected three representative objects to illustrate the effectiveness of our approach, as depicted in Figure 5.

4.2. Baseline

To highlight our proposed contact-aware based method, we compare it with a previous state-based method, which serves as the baseline approach. The state-based method and our proposed contact-aware-based method share the same observation but have different reward design manners. In the state-based method, it does not consider any contact interaction information in the reward design, in other words, the baseline method only takes state changement before and after the action is executed. Specifically, the baseline reward is designed as Equation (7):

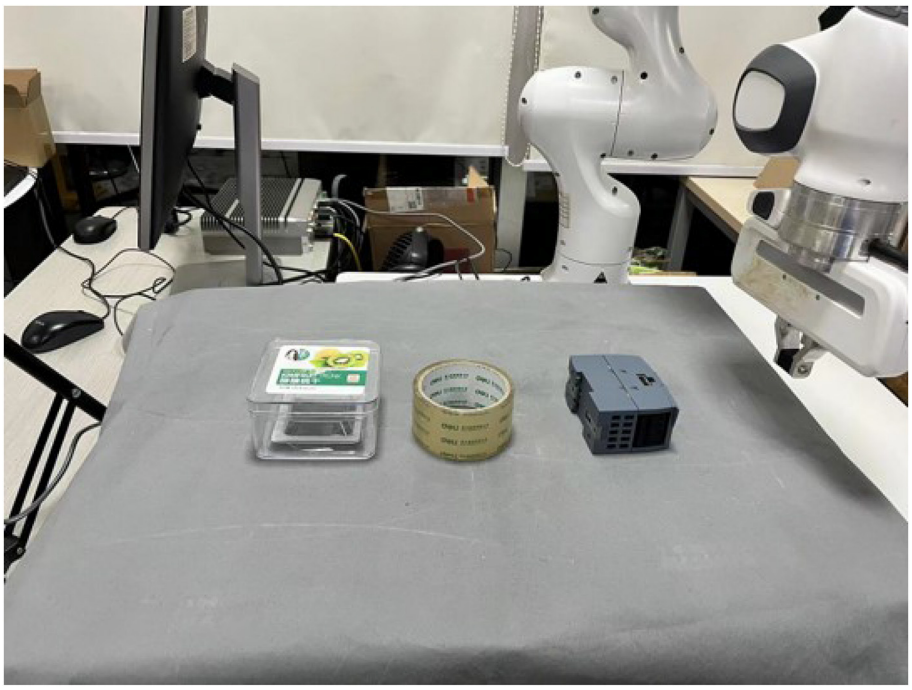

FIGURE 5
Real-world experiments setup and test objects with different mass, shape, and coefficient of friction.

FIGURE 6
Task success rate completing 50 times tasks. The result is the average of three runs with different random seeds.

$$\text{Reward} = \begin{cases} 1, & \text{if } d_{\text{current}} - d_{\text{last}} < 0 \\ e^{-d3} - 2, & \text{otherwise} \end{cases} \quad (7)$$

In Equation (7), $d_{current}$ and d_{last} refer to the distance before and after action execution. $d3$ refers to the distance from the object's center of mass to the contact force direction as the same in Equation (6).

4.3. Performance analysis and discussion

4.3.1. Success rate

We conducted experiments in the Pybullet simulation to showcase the performance of our proposed contact-aware method in achieving a higher task success rate by considering the contact information during interactions with the environment. We trained the model for 150 epochs, with 20,000 steps per epoch for both

FIGURE 7
Reward curve in the training progress. The result is the average of three runs with different random seeds.

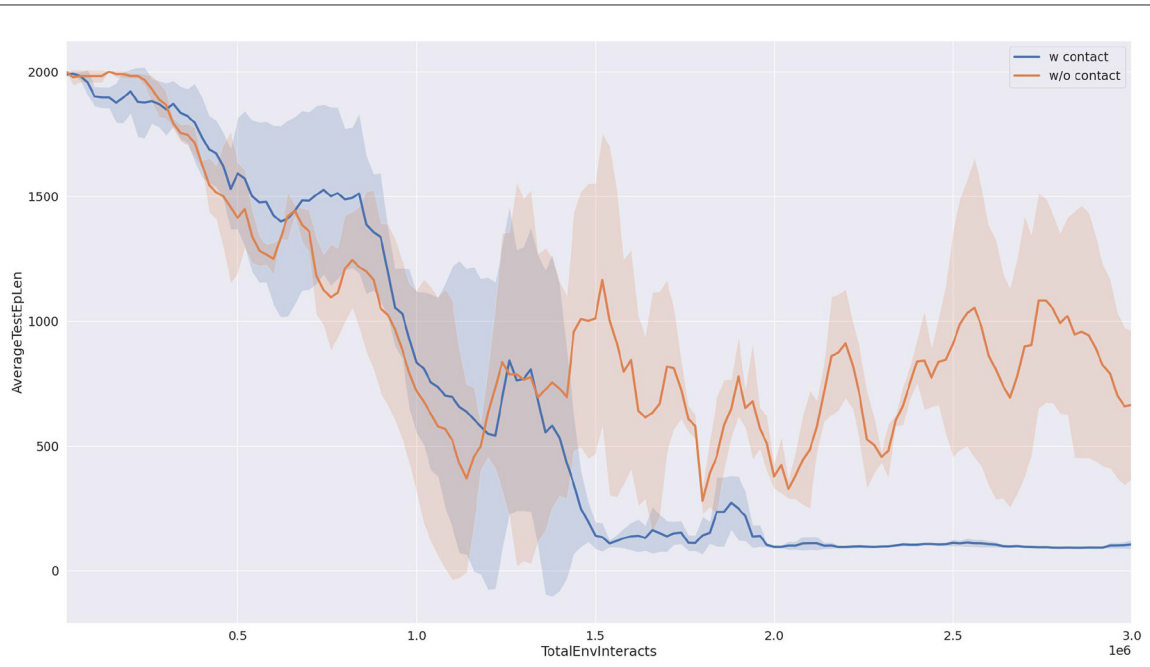

FIGURE 8
Average test length of the trajectory. The result is the average of three runs with different random seeds.

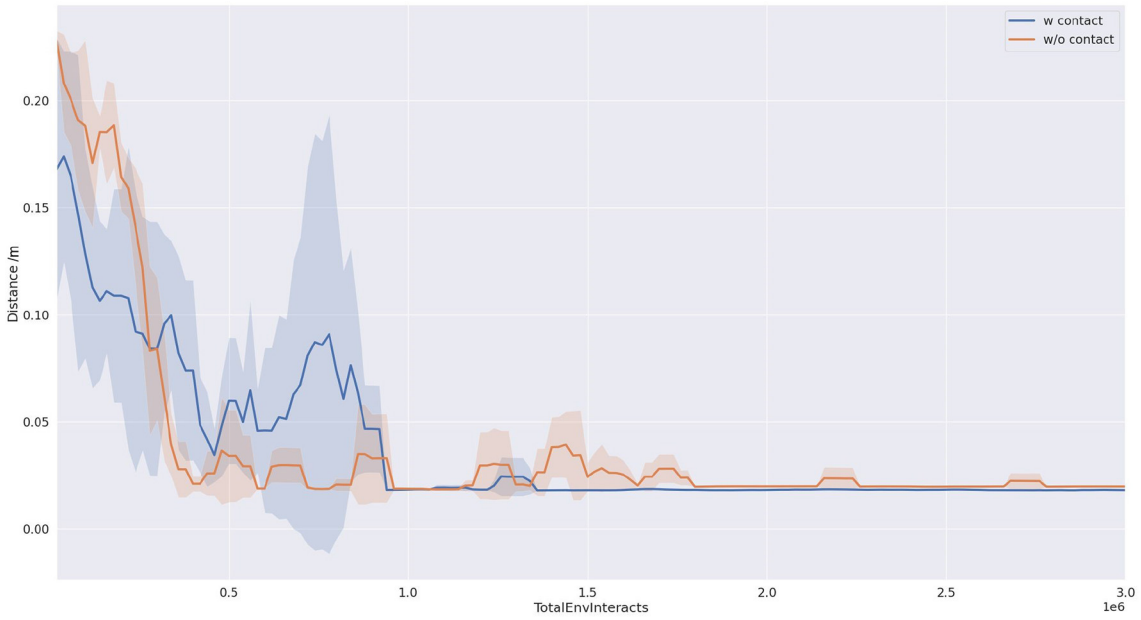

FIGURE 9
The distance from the target pose. The result is the average of three runs with different random seeds.

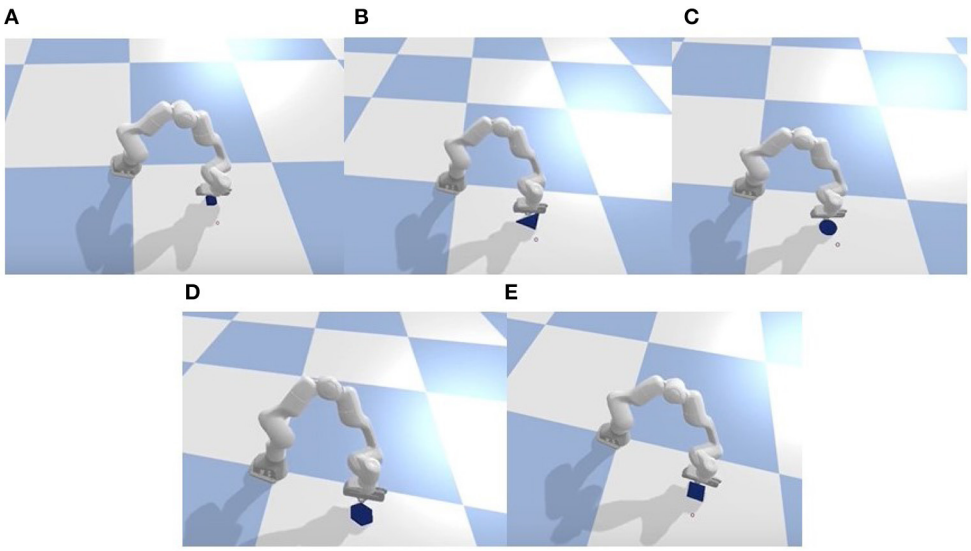

FIGURE 10
Generalization test in the simulation of four unseen objects: triangle, circle, hexagon, square. Label **(A)** is the training object and labels **(B–E)** are unseen objects.

tasks. The environment is terminated once the task goal is satisfied, the object is out of the workspace, or the maximum number of steps per episode is reached. To evaluate the performance of our method, we used the task success rate, which is calculated by evaluating the learned policy 50 times at the end of each epoch.

In this section, we employed the task success rate as one of the evaluation metrics. The task success rate is defined as the percentage of episodes in which the robot successfully completed the task, based on the predefined criteria: a successful completion is recorded when the object is positioned within 2 cm of the target location. By measuring the task success rate, we were able to assess the effectiveness of our contact-aware method in improving the performance of the pushing task.

As shown in Figure 6, we compared the success rates of our proposed method and the baseline approach when pushing a block object. It can be observed from the figure that our proposed method (depicted by the blue curve) achieves a success rate approaching 100%. Furthermore, after interacting with the environment for one

and a half million steps, the success rate of our proposed method stabilizes with minimal fluctuations. In contrast, the baseline method (represented by the yellow curve) exhibits success rates fluctuating around 80%, with a large variance, indicating poor strategy stability. This point also can be seen in the reward curve from Figure 7.

4.3.2. Task efficiency and precision

Our proposed method not only demonstrates advantages over the baseline approach in terms of task success rate but also exhibits superiority in task efficiency and precision.

The efficiency of task execution in this study is quantified by measuring the total number of steps taken during the implementation of the learned strategy. The number of steps required to complete the pushing task is recorded during the testing phases, and the statistical results are presented in Figure 8.

As shown in Figure 9, our proposed method requires significantly fewer steps to accomplish the task compared to the baseline approach, indicating higher task efficiency. This can be attributed to the reward function designed based on contact information, which encourages the strategy to interact with the object's center of mass and push it directly toward the target direction, resulting in fewer required steps. Additionally, from the graph, it can be observed that this design also contributes to an improvement in precision.

TABLE 2 Comparison of success rates in 50 times experiments for different objects.

Object	Ours	Baseline
Triangle	1.00	0.30
Circle	1.00	0.30
Hexagon	1.00	0.00
Square	1.00	0.40

4.3.3. Adaptivity performance

Although we only trained our method with a single object in the simulation, it demonstrates adaptability to multiple unseen objects. We specifically compared this aspect with the baseline method.

For the adaptability experiment, we utilized four objects other than those used during training, as shown in Figure 10. Each object took 50 task trials, and the success rate was recorded. The results are presented in the table.

From the Table 2, it is evident that our proposed method exhibits excellent adaptability to previously unseen objects, showcasing robust strategy generalization. In comparison, the baseline method demonstrates relatively poorer adaptability or strategy generalization. In terms of the success rate metric, our method consistently outperforms or performs on par with the baseline method.

4.3.4. Real experiments

We conducted real robot task validation for our proposed method, as illustrated in Figure 11. Our objective was to push three representative objects from random positions to the target location, which was set at the edge of the platform. This particular setup was chosen to demonstrate the role of pushing, as in our laboratory, the selected objects' dimensions exceeded the gripper's maximum

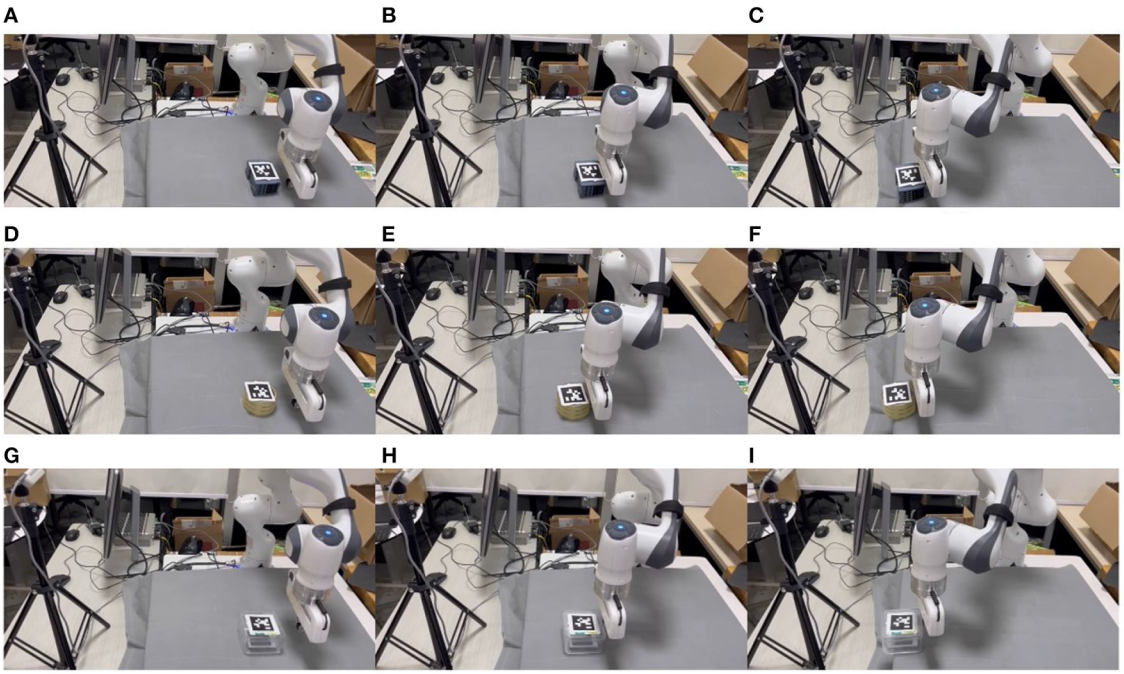

FIGURE 11
Snaps of pushing three representative objects with different mass, shape, and coefficient of friction in the real world. Labels **(A–C)** is for pushing an irregular-shaped electrical component, labels **(D–F)** is for pushing a circular tape, and labels **(G–I)** is for pushing a plastic square box.

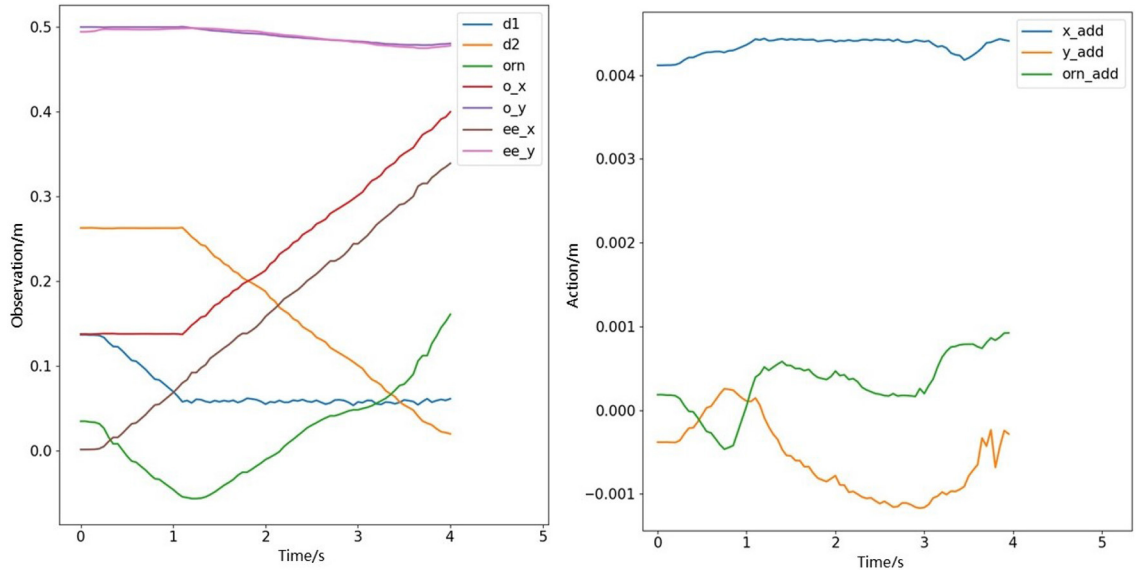

FIGURE 12
Observation and action data of pushing an object in the real world.

width. Thus, pushing the objects to the platform's edge allowed for grasping from the thinner side.

As seen in Figure 11, our reinforcement learning strategy successfully transferred to the real-world setting, even with unseen different objects from the simulation. Nevertheless, our method accomplished the task, demonstrating its effectiveness in real-world scenarios.

We further showcase the real robot's data, as depicted in Figure 12. The observation space's d2 represents the distance of the object from the target position, gradually decreasing from an initial distant position to reaching the target position. The action space outputs increments, and it can be observed that the position increment is adjusted around zero in the y-direction, while in the x-direction, the position increment maintains a constant value, allowing the robot to push the object to the target position.

5. Conclusion and future work

In this study, we introduced a novel Reinforcement Learning (RL)-based approach to address the task of object pushing in both simulated and real-world environments. Specifically, we utilized the Soft Actor-Critic (SAC) algorithm for training the RL policy while incorporating essential contact information between the robot and the environment through a specialized reward function. As a result, the learned policy demonstrated significantly higher task success rates in comparison to the baseline method, showcasing enhanced stability, generalization, and adaptability of the strategy. Furthermore, the successful transfer of the learned policy from simulation to real-world settings validated the efficacy of our proposed method.

In future endeavors, we aim to further enhance the policy's perceptual and decision-making capabilities by integrating additional sensory inputs, such as vision or tactile feedback.

To conclude, our study presents a promising and pragmatic approach for addressing object pushing tasks using RL, incorporating domain randomization and the SAC algorithm. The proposed method exhibits the potential for generalizing to novel objects and sets the stage for advancing research in RL-based manipulation tasks.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary material, further inquiries can be directed to the corresponding authors.

Author contributions

SW: Methodology, Conceptualization, Data curation, Formal analysis, Investigation, Project administration, Software, Supervision, Validation, Visualization, Writing—original draft, Writing—review and editing. LS: Funding acquisition, Investigation, Resources, Supervision, Writing—review and editing. FZ: Funding acquisition, Investigation, Resources, Supervision, Visualization, Writing—review and editing. WG: Funding acquisition, Investigation, Resources, Visualization, Writing—review and editing. PW: Funding acquisition, Investigation, Resources, Visualization, Writing—review and editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This work was supported by the National Natural Science Foundation of

China (U2013602 and 52075115), National Key R&D Program of China (2020YFB13134 and 2022YFB46018), Self-Planned Task (SKLRS202001B, SKLRS202110B, and SKLRS202301A12) of State Key Laboratory of Robotics and System (HIT), Shenzhen Science and Technology Research and Development Foundation (JCYJ20190813171009236), Basic Research on Free Exploration of Shenzhen Virtual University Park (2021Szvup085), and Basic Scientific Research of Technology (JCKY2020603C009).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot.2023.1271607/full#supplementary-material

References

Bauza, M., and Rodriguez, A. (2017). "A probabilistic data-driven model for planar pushing," in *2017 IEEE International Conference on Robotics and Automation (ICRA)* (New York, NY: IEEE), 3008–3015. Available online at: https://www.ieee.org/conferences/publishing/index.html

Bauza, M., Hogan, F. R., and Rodriguez, A. (2018). "A data-efficient approach to precise and controlled pushing," in *Conference on Robot Learning* (PMLR), 336–345. Available online at: https://proceedings.mlr.press/

Chai, C.-Y., Peng, W.-H., and Tsao, S.-L. (2022). Object rearrangement through planar pushing: a theoretical analysis and validation. *IEEE Transact. Robot.* 38, 2703–2719. doi: 10.1109/TRO.2022.3153785

Cong, L., Liang, H., Ruppel, P., Shi, Y., Görner, M., Hendrich, N., et al. (2022). Reinforcement learning with vision-proprioception model for robot planar pushing. *Front. Neurorobot.* 16, 829437. doi: 10.3389/fnbot.2022.829437

Coumans, E., and Bai, Y. (2016). Pybullet, a python module for physics simulation for games, robotics and machine learning. *J. Robotics Mach. Learn.* Available online at: http://pybullet.org

Dengler, N., Großklaus, D., and Bennewitz, M. (2022). "Learning goal-oriented non-prehensile pushing in cluttered scenes," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (New York, NY: IEEE), 1116–1122.

Dogar, M. R., and Srinivasa, S. S. (2010). "Push-grasping with dexterous hands: mechanics and a method," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (New York, NY: IEEE), 2123–2130.

Eitel, A., Hauff, N., and Burgard, W. (2020). "Learning to singulate objects using a push proposal network," in *Robotics Research: The 18th International Symposium ISRR* (Springer), 405–419.

Elguea-Aguinaco, Í., Serrano-Muñoz, A., Chrysostomou, D., Inziarte-Hidalgo, I., Bøgh, S., and Arana-Arexolaleiba, N. (2023). A review on reinforcement learning for contact-rich robotic manipulation tasks. *Robot. Comp. Integr. Manuf.* 81, 102517. doi: 10.1016/j.rcim.2022.102517

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning* (PMLR), 1861–1870. Available online at: https://proceedings.mlr.press/

Hogan, F. R., Grau, E. R., and Rodriguez, A. (2018). "Reactive planar manipulation with convex hybrid mpc," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (New York, NY: IEEE), 247–253.

Howe, R. D., and Cutkosky, M. R. (1996). Practical force-motion models for sliding manipulation. *Int. J. Robot. Res.* 15, 557–572. doi: 10.1177/027836499601500603

Kim, S., Lim, B., Lee, Y., and Park, F. C. (2023). "Se (2)-equivariant pushing dynamics models for tabletop object manipulations," in *Conference on Robot Learning* (PMLR), 427–436. Available online at: https://proceedings.mlr.press/

Kumar, S., Zamora, J., Hansen, N., Jangir, R., and Wang, X. (2023). "Graph inverse reinforcement learning from diverse videos," in *Conference on Robot Learning* (PMLR), 55–66. Available online at: https://proceedings.mlr.press/

Li, J. K., Lee, W. S., and Hsu, D. (2018). Push-net: deep planar pushing for objects with unknown physical properties. *Robot. Sci. Syst.* 14, 1–9. doi: 10.15607/RSS.2018.XIV.024

Lloyd, J., and Lepora, N. F. (2021). Goal-driven robotic pushing using tactile and proprioceptive feedback. *IEEE Transact. Robot.* 38, 1201–1212. doi: 10.1109/TRO.2021.3104471

Lynch, K. M. (1993). "Estimating the friction parameters of pushed objects," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93), Vol. 1* (New York, NY: IEEE), 186–193.

Lynch, K. M., and Mason, M. T. (1996). Stable pushing: mechanics, controllability, and planning. *Int. J. Robot. Res.* 15, 533–556. doi: 10.1177/027836499601500602

Mavrakis, N., Stolkin, R., and Stolkin, R. (2020). "Estimating an object's inertial parameters by robotic pushing: A data-driven approach," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (New York, NY: IEEE), 9537–9544.

Olson, E. (2011). "Apriltag: a robust and flexible visual fiducial system," in *2011 IEEE International Conference on Robotics and Automation* (New York, NY: IEEE), 3400–3407.

Paus, F., Huang, T., and Asfour, T. (2020). "Predicting pushing action effects on spatial object relations by learning internal prediction models," in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (New York, NY: IEEE), 10584–10590.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.* 22, 12348–12355.

Shahid, A. A., Roveda, L., Piga, D., and Braghin, F. (2020). "Learning continuous control actions for robotic grasping with reinforcement learning," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (New York, NY: IEEE), 4066–4072.

Singh, B., Kumar, R., and Singh, V. P. (2022). Reinforcement learning in robotic applications: a comprehensive survey. *Artif. Intell. Rev.* 55, 945–990. doi: 10.1007/s10462-021-09997-9

Song, C., and Boularias, A. (2020). "Identifying mechanical models of unknown objects with differentiable physics simulations," in *Learning for Dynamics and Control* (PMLR), 749–760. Available online at: https://proceedings.mlr.press/

Stüber, J., Zito, C., and Stolkin, R. (2020). Let's push things forward: a survey on robot pushing. *Front. Robot. AI* 7, 8. doi: 10.3389/frobt.2020.00008

Tsai, R. Y., and Lenz, R. K. (1989). A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration. *IEEE Transact. Robot. Automat.* 5, 345–358. doi: 10.1109/70.34770

Xu, Z., Yu, W., Herzog, A., Lu, W., Fu, C., Tomizuka, M., et al. (2021). "Cocoi: contact-aware online context inference for generalizable non-planar pushing," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (New York, NY: IEEE), 176–182.

Yoshikawa, T., and Kurisu, M. (1991). "Identification of the center of friction from pushing an object by a mobile robot," in *Proceedings IROS'91: IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91* (New York, NY: IEEE), 449–454.

Yu, S., Zhai, D.-H., and Xia, Y. (2023). A novel robotic pushing and grasping method based on vision transformer and convolution. *IEEE Transact. Neural Netw. Learn. Syst.* 1–14. doi: 10.1109/TNNLS.2023.3244186

Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., and Funkhouser, T. (2018). "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (New York, NY: IEEE), 4238–4245.

Zhou, J., Hou, Y., and Mason, M. T. (2019). Pushing revisited: differential flatness, trajectory planning, and stabilization. *Int. J. Robot. Res.* 38, 1477–1489. doi: 10.1177/0278364919872532

Zhu, S., Kimmel, A., Bekris, K. E., and Boularias, A. (2017). Fast model identification via physics engines for data-efficient policy search. *arXiv*. doi: 10.24963/ijcai.2018/451