Multi Task Reinforcement Learning for Non-Prehensile Manipulation

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—A key challenge in deep reinforcement learning for multi-task settings is performing long-horizon tasks consecutively and reaching the final goal through a single trained model. The objective of this research is to define multiple consecutive tasks for non-prehensile manipulation and achieve the final goal, which is hovering a thrown ball at a specific position. Separate but similar rewards are designed for each task to facilitate a smooth transition from one task to the next. The agents are trained in parallel with dynamic entropy levels, and the tasks are defined as catching, stabilizing, hovering, carrying to a target point, and hovering again. The results indicated that the agents can perform multiple tasks through one network for all the tasks and complete them consecutively to reach the final target point. This study provides new insights into multi-task training for non-prehensile manipulation and reward design to transition smoothly from one state to the next.**

***Index Terms*—Reinforcement Learning, Non-Prehensile Manipulation, Multi-Tasks Learning**

I. Introduction

Nowadays, there are numerous classical, data-driven, and hybrid control approaches for various robotic applications. With advancements in hardware such as new GPUs and TPUs for sppeding up calculations and improvements in sensor accuracy, new methods are emerging to tackle increasingly complex tasks. The complexity of these tasks heavily depends on both the capabilities of robotic systems and the approaches employed to accomplish them. Consequently, accurately defining the reward function is essential, particularly in multi-task learning scenarios.

There is extensive research on robotic manipulation tasks, most of which focuses on prehensile manipulation, where the robotic manipulator is more constrained than in non-prehensile tasks. For example, when displacing objects using grasp-based approaches, it generally suffices for the manipulator to grasp the object, with the actual displacement then handled through motion planning. However, for non-prehensile object displacement, the manipulator must address static and dynamic friction between the object and the end effector, ensuring either that no unwanted sliding occurs or that it can compensate for such frictional effects. Furthermore, in consecutive multi-task training of a robotic manipulator using a single network, a carefully designed reward function is crucial to encourage the system to progress from one task to the next and eventually achieve the overall goal. Long-horizon consecutive tasks present additional challenges, as the agent may succeed at one task but struggle to perform subsequent tasks effectively, resulting in poor transitions. Consequently, an accurate reward function should be defined for each individual task, and the exploration strategy should ensure that the agent acquires sufficient knowledge of its environment at each stage.

To address these limitations, this paper proposes: (1) an exploration strategy that prevents training instability while avoiding early-stage determinism, (2) a method for designing reward functions in consecutive multi-task scenarios to ensure smooth transitions between tasks, and (3) a discussion of existing challenges in training high-speed manipulation tasks.

Our research incorporates high randomization during the training process and introduces a dynamic entropy coefficient based on the mean standard deviation of the actor–critic network. This approach encourages greater exploration of the environment while allowing the training process to become more deterministic over time. Furthermore, we briefly present the reward functions for each task and propose a straightforward method for defining subsequent tasks, thereby enhancing the agent's motivation to reach the final goal.

The primary contributions of this work are:

- Introducing an entagled entropy coefficient to mean standard deviation of the actor–critic network
- Detailed rewards for consecutive multi tasks are provided
- the trained model is deployed on real manipulator to evaluate the performance

A. Contributions

II. Related Works

There are various non-prehensile manipulation primitives [1], each of which has been thoroughly investigated using classical control algorithms. Examples include non-prehensile object transportation via model predictive nonsliding manipulation control [2], tossing and catching pizza dough [3], non-prehensile ball-catching [4], [5], and non-prehensile rolling manipulation [6], [7]. The integration of reinforcement learning (RL) approaches for dynamic object tracking and manipulation tasks has gained significant attention in recent research.

Identify applicable funding agency here. If none, delete this.

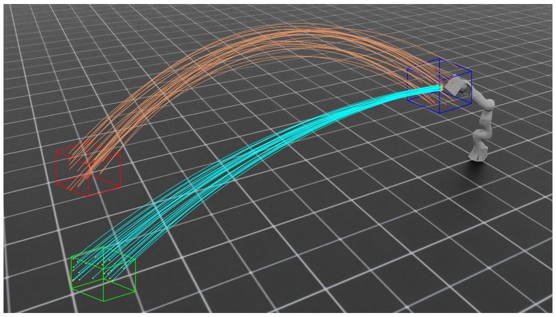

Fig. 1. Throwing Ball and Giving it Back Simulation

A variety of RL algorithms, including both policy-based and value-based methods, have been explored to improve the performance and robustness of robotic systems in dynamic and cluttered environments.

Simulation environments underpin much of modern robotics research, enabling rapid prototyping and policy iteration in safe, controllable conditions. RoboCasa [8] presents a large-scale simulation framework featuring diverse kitchen scenes and high-quality 3D objects. By harnessing generative AI tools (text-to-3D, text-to-image models) and large language models (LLMs) for task design, RoboCasa enables training on a wide range of tasks, from elementary skills to complex multi-skill sequences. In the domain of surgical robotics, ORBIT-Surgical [9] offers an open simulation framework targeting augmented dexterity for surgical tasks. By integrating both reinforcement learning (RL) and imitation learning (IL) algorithms, this environment supports research on fine-grained and safety-critical medical applications.

A key challenge in multi-task manipulation is learning policies that generalize across diverse objectives and environments. RoboCLIP [10] demonstrates how pretrained visual language models (VLMs) can generate reward functions for RL agents. By providing a video demonstration or textual description, the agent compares its episodic interactions to the reference, effectively translating high-level semantics into training signals. Similarly, RVT-2 [11] addresses multi-task manipulation with minimal demonstrations—fewer than 10 per task—achieving millimeter-level precision through a framework tailored for high-accuracy tasks. To enhance sample efficiency, researchers have looked at explicit task-representation and contextual metadata. For example, Multi-Task Reinforcement Learning with Context-based Representations [12] encodes state-space information via interpretable encoders, improving performance across multiple tasks. Transformer-based methods have recently shown promise. Perceiver-Actor [13] grounds language in an action-centric framework, using Transformers for perception, action, and goal specification, allowing a single agent to handle multiple simulated and real-world tasks. Another line of work investigates modular or prioritized auxiliary tasks. Multi-Task Reinforcement Learning with Soft Modularization [14] introduces a “soft modularization” strategy to handle task diversity, and Adaptive Auxiliary Task Weighting for Reinforcement Learning [15] learns to weight each auxiliary task based on their contribution to efficient transfer. Large-scale real-world data collection has also been explored. MT-Opt [16] acquires extensive robotic interaction data directly from real-world scenarios, training a single network on many short-horizon tasks. In a complementary direction, Augmenting Reinforcement Learning with Behavior Primitives [17] decomposes manipulation into heterogeneous primitives with relevant parameters. A task policy then selects the most appropriate primitive and parameter set to tackle each scenario. RoboAgent [8] demonstrates how semantic augmentation and action chunking (MT-ACT) can boost low-data regimes, showing that data multiplication and structured action spaces drive significant gains in generalization and efficiency.

Non-prehensile manipulation (e.g., pushing, rolling, pivoting) often involves complex contact dynamics. End-to-End Nonprehensile Rearrangement [18] employs a DQN-based approach that directly maps raw image pixels to manipulator joint actions, focusing on object displacement through pushing. Learning Synergies between Pushing and Grasping [19] combines pushing and grasping to improve object arrangement in cluttered scenes, demonstrating that non-prehensile primitives can complement prehensile skills. Methods such as HAC-Man [20] use point cloud data to plan and execute 6D pushing actions. Meanwhile, CORN: Contact-Based Object Representation [21] leverages a contact-informed representation and patch-based Transformer architecture to learn a diverse set of non-prehensile skills (pushing, toppling, pivoting, rolling) in a data- and time-efficient manner. Multi-stage decomposition is another strategy. Multi-Stage Reinforcement Learning for Non-Prehensile Manipulation [22] partitions tasks by object pose and contact points, albeit sometimes using a single reward function that may limit fine-grained stage learning. Expanding on exploration techniques, Nonprehensile Planar Manipulation through RL with Multimodal Categorical Exploration [23] discretizes action spaces into bins and uses a categorical distribution to define exploration probabilities. Before advanced randomization methods emerged, policies trained in simulation for non-prehensile tasks showed promise despite limited transfer. Reinforcement Learning for Non-Prehensile Manipulation [24] demonstrated the feasibility of moving from simulation to reality using ensemble-trained policies for dynamic pushing.

Dynamic manipulation tasks like throwing and catching pose a distinct set of challenges. TossingBot [25] shows that residual physics helps improve throwing parameter estimation. By training grasping and throwing jointly, the system picks optimal grasp locations for more precise throws. Catching is similarly complex, involving real-time target tracking and interception. Learning to Catch Reactive Objects [26] uses a single-task RL approach where a mobile manipulator intercepts moving targets. For dexterous catching, Catch It [27] employs a two-stage RL pipeline on a mobile manipulator with a dexterous hand, but highlights issues such as camera imprecision and object elasticity. Multiple studies address the intricacies of ball catching. Kinematically Optimal Catching

[28] leverages Extended Kalman Filtering (EKF) for trajectory prediction and experiments with various impedance control modes. A Goal Oriented Just-In-Time Visual Servoing [29] focuses on vision-based control that circumvents explicit camera or manipulator calibration. Further, Catching Objects in Flight [30] and A Dynamical System Approach for Softly Catching [31] stress robust trajectory prediction and seamless velocity matching, respectively. The interplay of Model Predictive Control (MPC) and RL for agile catching is explored in [32], while high-speed motions via inverse dynamics learning for mobile manipulators are discussed in [33].

Bridging the simulation-to-reality gap is critical when policies are trained in virtual environments. Sim-to-Real Transfer with Dynamics Randomization [34] samples a wide range of possible physical parameters during training so that the learned policy becomes robust to real-world uncertainties. For deformable objects, Sim-to-Real for Deformable Object Manipulation [35] uses 20 demonstrations to initialize training and domain randomization to strengthen transfer. andomized-to-Canonical Adaptation Networks (RCAN) [36] translates simulation images from randomized setups to a canonical form, and subsequently applies the same translation to real-world images, thus aligning visual domains for robust grasping skills. Auto-Tuned Sim-to-Real Transfer [37] examines adjusting simulation parameters by comparing real-world RGB data and actions to simulated outcomes via a binary classifier. Meanwhile, Reconciling Reality through Simulation [38] addresses a real-to-sim-to-real approach that refines policy robustness but focuses on imitation learning for single-path solutions.

Reward design is often a bottleneck in RL. Active Reward Learning from Online Preferences [39] proposes a system that queries humans with pairwise action preferences in critical states, refining the agent's policy online. This effectively scales human feedback for reward shaping, tailoring policies to specific operational conditions.

Object tracking in dynamic scenarios remains an active research topic. Active Object Tracking of Free-Floating Space Manipulators [40] leverages a PPO algorithm in CoppeliaSim, with a camera on the manipulator's end-effector for vision-based tracking. A survey on object tracking in RL, The Use of Reinforcement Learning Algorithms in Object Tracking [41], highlights the benefits of integrating policy-based and value-based methods for robust tracking.

Several studies address complementary challenges in manipulation. Learning to Grasp the Ungraspable [42] investigates extrinsic dexterity, particularly in occluded grasps, through RL. Points2Plans [43] and Skill-Based Model-Based RL [44] explore long-horizon tasks by combining point cloud representations, relational dynamics, and skill abstractions to reduce planning complexity and improve sample efficiency.

For vision-based perturbations, ViSaRL [45] introduces a saliency predictor, image encoder, and policy network to maintain performance under random color changes and video backgrounds. Finally, Attention-Privileged Reinforcement Learning [46] uses dual actor-critic networks for better sim-to-real generalization, tackling the common challenge of overfitting to simulated observations.

III. System Setup

A. Task Description

There has been some research on catching a thrown object through grasping manipulation, but these studies are limited to prehensile tasks, which are more deterministic than non-prehensile manipulation. In this paper, we define a series of consecutive non-prehensile manipulation tasks to demonstrate the ability of a trained manipulator to perform more challenging tasks. However, it should be noted that we must always consider the limitations of a robot's capabilities, such as its maximum speed and maximum torque. We define the tasks as follows:

1) **Catching a thrown ball**: The first task is to catch a thrown ball from a specific range of distances, ensuring the ball passes through the manipulator's workspace. Therefore, the initial positions and velocities must be selected within a certain range so that the ball indeed travels through that workspace.
2) **Stabilizing the ball**: While the ball rests on the plate, certain conditions ensure that the relative distance between the ball and the plate does not exceed $2cm$. This requirement encourages the agent to consider the alignment between the relative velocity of the end effector and that of the plate. Furthermore, both the relative velocities and the ball's velocity must lie within a defined range, which constitutes the stabilized condition.
3) **Hovering the ball**: Once stabilized under the above conditions, the ball should be hovered at its current position for $250\Delta t$. This position can be anywhere within the manipulator's workspace.
4) **Moving to a target location**: After $250\Delta t$ of hovering, the manipulator should move the ball to a predefined position in its workspace. This position might be the same location where the ball was stabilized, or it could be different, depending on the initial conditions of the thrown ball.
5) **Hovering the ball a target location**: The manipulator should hover the ball at the target location for $60\Delta t$ to signify the successful completion of all the previous tasks.
6) **Throwing back the ball to the initial position**: Finally, the manipulator should throw back the ball to the initial position of the ball. For this task, some specific reward functions are defined to smoothly reach to the orientation and velocity of which the ball can fly to the inital position. This position is out of the manipulator's workspace and unlike the other works on throwing ball, this task is done with non predhensile manipulation. So, the manipulator should hold the ball on the end effector to reach the thrown position with the physics based calculated velocity.

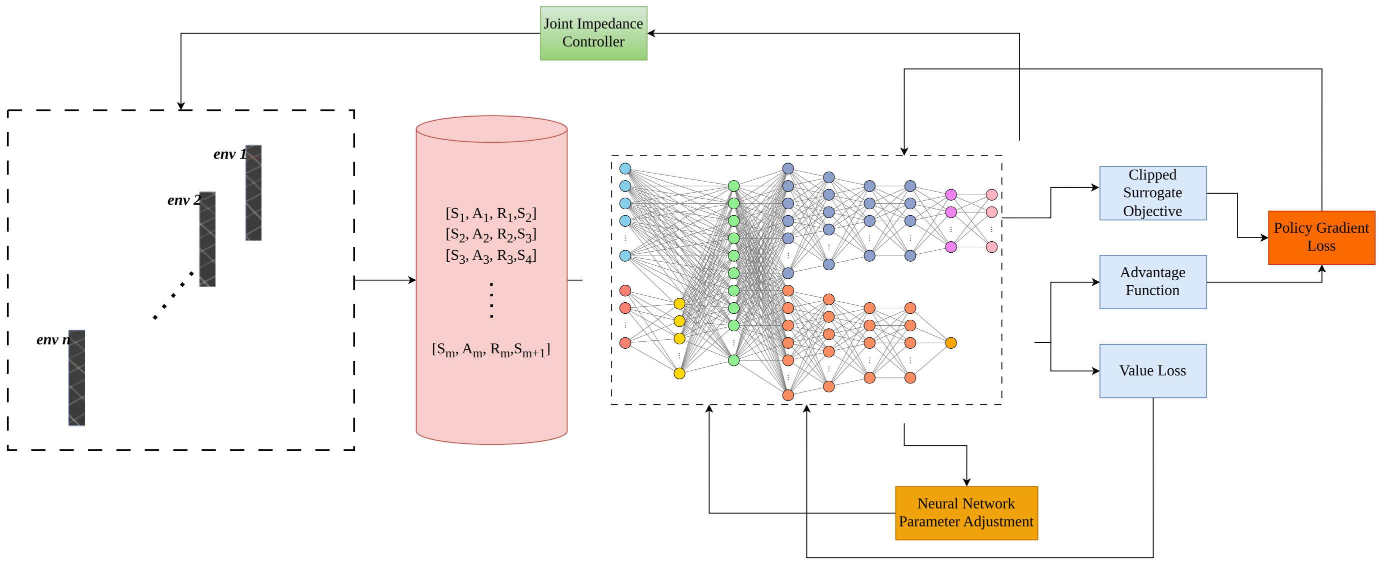

Fig. 2. Methodology

These tasks are defined consecutively. If the manipulator attempts to revert to a previous task, it receives negative rewards, which ultimately promotes forward task completion.

B. Observation and Action Space

The observation space consists of 52 values, while the action space consists of 7 values corresponding to the joint position commands. The observation parameters and their dimensions are provided in Table I.

TABLE I
SUMMARY OF OBSERVATION SPACE

Component	Dimension	Scaling Factor
Previous Actions	7	1.0
Scaled Joint Positions	7	1.0
Joint Velocities	7	0.31
Ball Position	3	0.25
End Effector Position	3	1.0
Ball Linear Velocity	3	0.15
End Effector Linear Velocity	3	0.15
End Effector Orientation	4	1.0
Relative Position to Target Hovering Position	3	1.0
Relative Ball Distance to Desired Throwing Velocity	3	0.5
Relative Ball Velocity to Desired Throwing Velocity	3	0.15
Task	6	1.0

The *Task* observation is given by a row of a one-hot encoder corresponding to the current task at each time step. Furthermore, for the first three tasks, the Distance to Final Target is set to zero for all elements, and only the last two tasks use actual values, in order to enhance training performance.

C. Simulation Setup

The simulation is set up at the frequency of $500Hz$ in with decimation of 2 which means the action commands are derived at $250Hz$ and are sent to the simulation environment two times. The ball and the manipulator are considered as rigid bodies, but some parameters of the maniulator, ball, and environment are randomized which are discussed in section IV-C. Moreover, a Gaussian Noise is considered to be applied to actions and the observation space to enhance the robustness of training procedure. To be consistent with the real world scenarios, Stiffness values of $[400, 400, 400, 400, 400, 100, 100]$ and Damping values of $[28, 28, 28, 28, 28, 14, 14]$ are provided for the Joint Position Control with fixed impedance with almost critical damping system.

D. Real-world Setup

E. Reinforcement Learning For Robot

In this paper, a framework is developed to train a manipulator to carry out consecutive non prehensile manipulation tasks. These tasks are considered to be Catching a thrown ball, Stabilizing the ball on the plate, and Carrying the ball to a predefined target position. To do that, it should be considered that the observation state should have MDP charactristics. in other words, the decisions taken from each state should not be dependant on the previous states and actions. The observation state are shown in II.

IV. LEARNING NON-PREHENSILE MANIPULATION TASK

A. Six Consecutive Stage Reinforcement Learning

In Fig.3, the states and transitions are shown. In this figure, the abbreviations C, S, H, MTP, HTP, TB, Term., and Suc. represent the Catching, Stabilizing, Hovering, Moving

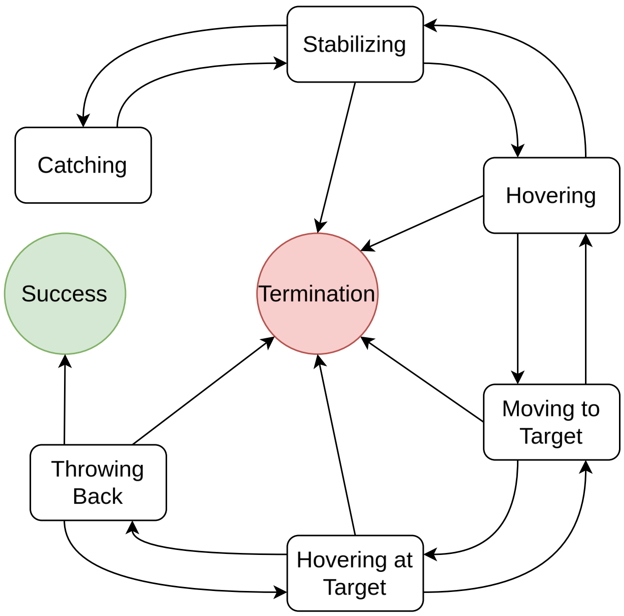

Fig. 3. Sequential Multi-Task Reinforcement Learning.

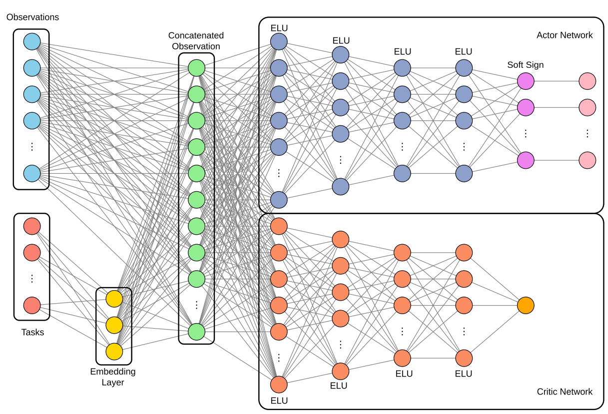

Fig. 4. Fully Connected Neural Networks with Embedding Layer.

to a Target Point, Hovering at the Target Point, Throwing Back, Termination, and Successful states, respectively. As is evident, at any given state, the agent can encounter one of the terminations discussed in Section IV-H. Additionally, the agent can revert to the previous state if the conditions of the current state are not satisfied, incurring a penalty each time this occurs.

B. Network's Structure

1) Embedding Layer: Since, there are six different tasks, and each task has it's own observation space, one of the best approaches to identify the contribution of each observation at specific task can be generated by an embedding layer [47].

2) Output Layer: The input and hidden layers of the Actor and Critic networks are the same; the only difference lies in their output layers. In the Actor Network (Policy Network), there are seven output units, corresponding to the seven joints of the manipulator. Furthermore, an activation function is applied to these outputs to prevent erratic movements in real-world applications. Specifically, we use the *SoftSign* activation function, which squashes the outputs into the range $[-1, +1]$. On the other hand, the output of the Critic Network (Value Network) consists of a single unit without any activation function.

TABLE II
NETWORK STRUCTURE OF THE PPO ALGORITHM

Layer Type	Number of Units	Activation Function
Input Layer	52	-
Embedding Layer	9	-
Fully Connected (fc1)	256	ELU
Fully Connected (fc2)	128	ELU
Fully Connected (fc3)	64	ELU
Fully Connected (fc3)	64	ELU
Policy Network Output	7	Softsign
Value Network Output	1	Linear

TABLE III
PPO PARAMETERS

Parameter	Value	Parameter	Value
Value Loss Coef	7	Learning Rate Scheduler	
Clip Param	7	Gamma	
Entropy Coef	3	Lambda	
Num of Learning Epochs	3	Desired KL	
Num of Mini Batches	5	Max Grad Norm	
learning Rate	3	Actor & Critic Layers	
activation			

C. Randomization Strategy

To enhance the robustness of the trained model in simulation, randomization was applied to certain parameters, as reported in IV.

TABLE IV
RANDOMIZATION PARAMETERS

Parameter	Range	Randomization Method
Static Friction of EE	$[0.7, 1.0]$	Reset
Dynamic Friction of EE	$[0.5, 0.8]$	Reset
Restitution of End Effector	$[0.2, 0.4]$	Reset
Static Friction of Ball	$[0.7, 1.0]$	Reset
Dynamic Friction of Ball	$[0.5, 0.8]$	Reset
Restitution of Ball	$[0.2, 0.4]$	Reset
Mass of Ball	$[0.8, 1.2]$	Reset
Gravity	$[-0.05, 0.05]$	Reset
Initial Joint Positions	$[-0.05, 0.05]$	Reset
Initial Ball Position **X**	$[-0.1, 0.1]$	Reset
Initial Ball Position **Y**	$[-0.05, 0.05]$	Reset
Initial Ball Position **Y**	$[-0.05, 0.05]$	Reset
Initial Ball Velocity **X**	$[-0.1, 0.1]$	Reset
Initial Ball Velocity **Y**	$[-0.1, 0.1]$	Reset
Initial Ball Velocity **Z**	$[-0.1, 0.1]$	Reset
Initial Joint Positions	$[-0.02, 0.02]$	at Reset and Scaled

In throwing back the ball, an initial fixed position of $[0.2m, -0.35m, 0.9m]$ is used, and the final throwing-back position is considered the initial throwing position. Since all parameters are known, the agent should carry the ball to that initial throwing-back position with a velocity calculated from the ballistic projectile equation, ensuring that the velocity in the z direction at the moment of throwing back is zero. Consequently, due to the randomization of initial throwing positions, the throwing-back velocities are also randomized.

D. Reward Design

The reward function is defined to smoothly increase as the agent progresses from one task to the next. This approach prevents the agent from reverting to previous tasks and encourages forward movement to reach the final goal. The details of the rewards are presented in the table IV-E.

E. Reward Function

Given at time step t, the end effector's 3D position $\mathbf{p}_t^{ee} \in \mathbb{R}^3$, end effector's linear velocity $\mathbf{v}_t^{ee} \in \mathbb{R}^3$, ball's position $\mathbf{p}_t^b \in \mathbb{R}^3$, ball's linear velocity $\mathbf{v}_t^b \in \mathbb{R}^3$, normalized end effector z-direction $\hat{\mathbf{z}}_t^{ee} \in \mathbb{S}^2$, termination condition T, world Z-direction $\hat{\mathbf{Z}} \in \mathbb{S}^2$, and target position $\mathbf{p}^{\text{Target}} \in \mathbb{R}^3$, the reward function comprises the following components:

- **Relative Distance Reward**: Encourages the agent to minimize the distance between the end effector and the ball smoothly. It is defined as
$$r_{\text{distance}} = \frac{1.0}{1.0 + \|\mathbf{p}_t^{ee} - \mathbf{p}_t^b\|^2}.$$
- **End Effector Orientation Reward**: Incentivizes the agent to align the end effector's orientation opposite to the ball's velocity direction. It is formulated as
$$r_{\text{orientation}} = -\left(\hat{\mathbf{z}}_t^{ee} \cdot \hat{\mathbf{v}}_t^b\right),$$
where $\hat{\mathbf{v}}_t^b = \frac{\mathbf{v}_t^b}{\|\mathbf{v}_t^b\|}$ is the unit vector of the ball's velocity.
- **Relative Velocity Reward**: Encourages the agent to match the end effector's velocity with that of the ball, especially as the ball approaches the end effector, thereby preventing collision. It is defined as
$$r_{\text{relative velocity}} = e^{-\|\mathbf{p}_t^{ee} - \mathbf{p}_t^b\|^2} \cdot \frac{1.0}{1.0 + \|\mathbf{v}_t^{ee} - \mathbf{v}_t^b\|^2}.$$
- **Stability Reward**: Promotes stability by penalizing high velocities of the ball. It is given by
$$r_{\text{stability}} = \frac{1.0}{1.0 + \|\mathbf{v}_t^b\|}.$$
- **Gravity Compensation Reward**: Encourages the manipulator to orient the end effector's normal vector opposite to the gravity direction. It is expressed as
$$r_{\text{gravity}} = \hat{\mathbf{z}}_t^{ee} \cdot \hat{\mathbf{Z}}.$$
- **Distance to Target Position Reward**: Applicable to the final two tasks—moving to the target location and hovering there—the reward is defined as
$$r_{\text{target distance}} = \frac{1.0}{1.0 + 5.0 \times \|\mathbf{p}^{\text{Target}} - \mathbf{p}_t^b\|^2}.$$

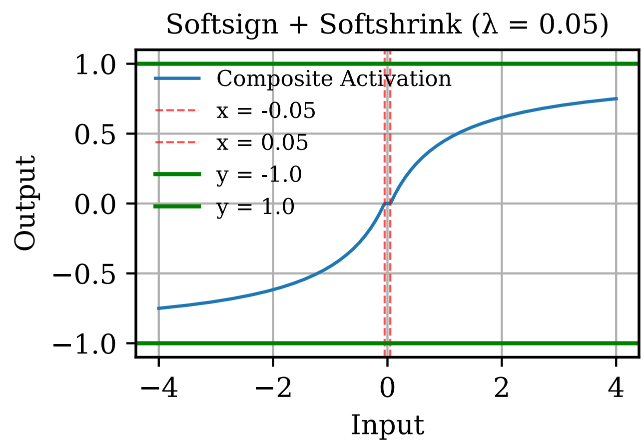

Fig. 5. SoftShrink function with $\lambda = 0.05$.

- **Awake Reward**: Encourages the agent to remain active during training and as it progresses through tasks. It is defined based on the current task:
$$r_{\text{awake}} = \begin{cases} 0.2 & \text{if the task is Catching Ball,} \\ 0.4 & \text{if the task is Stabilizing Ball,} \\ 0.8 & \text{if the task is Hovering Ball,} \\ 1.2 & \text{if the task is Moving to Target,} \\ 1.6 & \text{if the task is Hovering at Target,} \\ 1.8 & \text{if the task is Throwing Ball.} \end{cases}$$
- **Termination Penalty**: If any termination condition as described in IV-H is met, the agent incurs a penalty of 10. Additionally, the significance of this penalty increases as the agent progresses in completing tasks.
- **Task Completion Reward**: Upon completing each task, the agent receives a corresponding reward:
$$r_{\text{TC}} = \begin{cases} 5.0 & \text{if the ball is caught,} \\ 15.0 & \text{if the ball is stabilized and hovering is completed,} \\ 25.0 & \text{if the target position is reached,} \\ 35.0 & \text{if hovering at the target is completed,} \\ 30.0 & \text{if the ball is thrown successfully.} \end{cases}$$
- **Joint Positions at Limits Penalty**: If any joint reaches 97% of its maximum range, the agent incurs a penalty.
- **Joint Velocities at Limits Penalty**: If any joint velocity reaches 97% of its maximum velocity, the agent incurs a penalty.
- **Action Sign Change Penalty**: Discourages frequent changes in the sign of actions, promoting smoother joint trajectories. It is defined as
$$r_{\text{AS}} = \text{sign}\left(\text{SoftShrink}(a_t) - \text{SoftShrink}(a_{t-1})\right),$$
where a_t denotes the action at time t. The SoftShrink function is illustrated in 5.

The SoftShrink function mitigates minor action changes, particularly when the agent attempts to maintain a rest state for extended periods.

- **Joint Acceleration Penalty**: This second-order penalty discourages large accelerations, penalizing them more heavily than smaller ones:
$$p_{\text{acc}} = \left\| \sum_{i=1}^{7} a_i^2 \right\|^{\frac{1}{2}},$$
where a_i represents the acceleration of the i-th joint.
- **Joint Jerk Penalty**: Similar to the acceleration penalty, this term discourages abrupt jerky motions by penalizing large jerks:
$$p_{\text{jerk}} = \left\| \sum_{i=1}^{7} j_i^2 \right\|^{\frac{1}{2}},$$
where j_i denotes the jerk of the i-th joint.
- **Reversion to Previous Task Penalty**: Prevents the manipulator from reverting to previously completed tasks, thereby motivating continual task progression. The penalty value is set to 0.4.
- **Reach Throwing Velocity Reward**: Encourages the manipulator to achieve the required end effector velocity necessary for the ball to reach its initial position based on ballistic motion equations:
$$r_{\text{throw velocity}} = \frac{1.0}{1.0 + \|\mathbf{v}_t^{ee} - \mathbf{v}^{\text{required}}\|^2},$$
where $\mathbf{v}^{\text{required}}$ is the velocity derived from ballistic calculations.
- **Reach Throwing Position Reward**: Assuming a fixed throwing position for all manipulators, this reward focuses on achieving the necessary end effector position:
$$r_{\text{throw position}} = \frac{1.0}{1.0 + \|\mathbf{p}_t^{ee} - \mathbf{p}^{\text{Throw Position}}\|^2}.$$
- **Reach Throwing Orientation Reward**: Beyond position and velocity, the direction of the ball's velocity is crucial for the throwing task. This reward ensures that the end effector's orientation facilitates the desired velocity vector:
$$r_{\text{throw orientation}} = \cos(\theta),$$
where θ is the angle between the desired throwing velocity $\mathbf{v}^{\text{desired}}$ and the actual ball velocity $\mathbf{v}_t^b$, defined as
$$\theta = \cos^{-1}\left(\frac{\mathbf{v}^{\text{desired}} \cdot \mathbf{v}_t^b}{\|\mathbf{v}^{\text{desired}}\|\|\mathbf{v}_t^b\|}\right).$$

F. Action Space

1) Target Joint Position Calculation:

$$q_{\text{targets}} = q_{\text{pos}} + \Delta t \cdot a \cdot k_a \tag{1}$$

where:

- q_{targets} is the target joint position.
- q_{pos} is the current joint position.
- Δt is the time step.
- a is the control action (output from the policy/network).
- $k_a = 18$ is a scaling factor.

2) Clamping the Joint Targets:

$$q_{\text{targets}} \in [0.98 \cdot q_{\text{lower}}, 0.98 \cdot q_{\text{upper}}] \tag{2}$$

where q_{lower} and q_{upper} are the lower and upper joint limits, respectively.

3) Position and Velocity Errors:

$$e_{\text{pos}} = q_{\text{targets}} - q_{\text{pos}} \tag{3}$$
$$e_{\text{vel}} = -\dot{q}_{\text{vel}} \tag{4}$$

where:

- e_{pos} is the position error.
- e_{vel} is the velocity error (desired velocity is zero).
- $\dot{q}_{\text{vel}}$ is the current joint velocity.

4) Desired Acceleration:

$$\ddot{q}_{\text{des}} = k_s \cdot e_{\text{pos}} + k_d \cdot e_{\text{vel}} \tag{5}$$

where:

- $\ddot{q}_{\text{des}}$ is the desired acceleration.
- k_s is the stiffness coefficient.
- k_d is the damping coefficient.

5) Torque Calculation:

$$\boldsymbol{\tau} = \mathbf{M} \cdot \ddot{q}_{\text{des}} + g \tag{6}$$

where:

- $\boldsymbol{\tau}$ is the computed torque.
- $\mathbf{M}$ is the joint-space mass matrix.
- g is the generalized gravity forces vector.

6) Clamping Torque to Effort Limits:

$$\boldsymbol{\tau}_{\text{target}} = \text{clip}(\boldsymbol{\tau}, -\boldsymbol{\tau}_{\text{max}}, \boldsymbol{\tau}_{\text{max}}) \tag{7}$$

where $\boldsymbol{\tau}_{\text{max}}$ is the maximum allowable torque.

G. Exploration Strategy

In this paper, the exploration is considered through two approaches, action noise through policy's standard deviation and entropy bonus added to the loss function of PPO. The coefficients of the entropy are pre-sceduled based on following conditions, and these values are defined based on trial and error.

$$\text{Entropy}(it, std, Ent_{\text{min}}, Ent_{\text{max}}) = \begin{cases} y_{\text{min}} + \frac{(y_{\text{max}} - y_{\text{min}})}{1+\exp\left(10\cdot(std-0.75)\right)}, & \text{if } it \leq 3500, \\ 0, & \text{otherwise.} \end{cases}$$

where

- it: is the iteration number.
- std: is the mean standard deviation of the actor-critic network.
- Ent_{max}: is the upper bound of entropy at that iteration.
- Ent_{min} is the lower bound of entropy at that iteration.

TABLE V
REWARD FUNCTIONS

Reward & Penalties	Catching Ball	Stabilizing Ball	Hovering Ball	Moving to Target Point	Havering at Target Point	Throwing Ball
Relative Distance	✓($\times$1.0)	✓($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)		
End-Effector Orientation	✓ ($\times$1.0)	constant(1.0)	constant(1.0)	constant(1.0)		
Relative Velocity	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)		
Stability	✗					
Gravity Compensation	✗					
Distance to Target	✗					
Joint Pose Limit	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)		
Joint Velocity Limit	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)		
Termination Penalty	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)		
Joint Acceleration	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)		
Joint Jerk	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)		
Revert to Previous Task	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)	✓ ($\times$1.0)		
Throwing Back Velocity	✗					
Throwing Back Position	✗					
Task Done	✗					

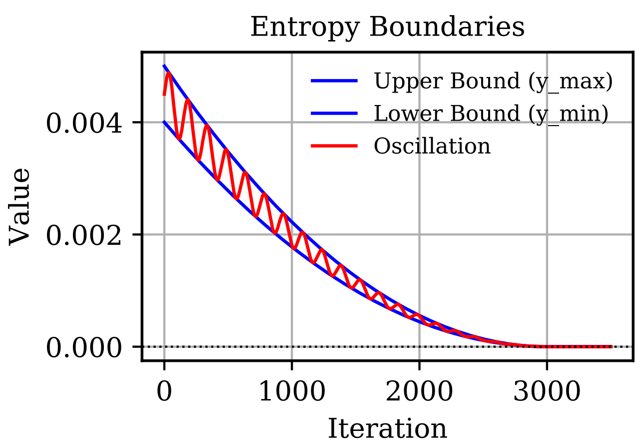

Fig. 6. Entropy Boundaries

Therefore, the higher the standard deviation, the closer the standard deviation is to the lower bound and vice versa. In this way, the agents are controlled to not be exploited during training, and it makes a trade-off between the exploration strategies.

H. Termination Conditions

The following termination conditions are defined to prevent the agents from reaching unsafe situations and to enhance the training process.

- If one joint of the manipulator reaches to 0.98% of its maximum angle limit
- If one joint of the manipulator reaches to 0.98% of its maximum velocity limit
- If the height of the end effector position along Z direction be lower than $30cm$
- if the ball position along the normal vector of the plate be negative
- If the difference between the height of the ball and the end effector be more than $10cm$. this encourage the agents to be beneath the ball most of the times.

V. Experiments

A. Simulation Results

B. Experiment Settings

The stiffness and damping coefficint for joint impedance controller of real LBR iiwa 7 are secelcted as $[400, 400, 400, 400, 400, 100, 100]$ and $[28, 28, 28, 28, 28, 14, 14]$ respectvely. Also, the end effector and ball parameters are reported in VI. Moreover, the mass and radius of the ball are $100g$ and $0.12cm$ respectively.

TABLE VI
PPO PARAMETERS

Parameter	Tray	Ball
Static Friction		
Dynamic Friction		
Restitution		

C. Sim2Real Transfer

The model is completely trained in simulation, and due to wide range of randomization, the trained model is robust enough to be implemented directly on real robot. The interface for communication between the manipulator and the model for sending and receiving commands is provided through the Fast Robot Interface (FRI).

D. Real Robot Deployment

E. Conclusion and Discussion

Acknowledgment

References

[1] Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. Nonprehensile dynamic manipulation: A survey. *IEEE Robotics and Automation Letters*, 3(3):1711–1718, 2018.

[2] Mario Selvaggio, Akash Garg, Fabio Ruggiero, Giuseppe Oriolo, and Bruno Siciliano. Non-prehensile object transportation via model predictive non-sliding manipulation control. *IEEE Transactions on Control Systems Technology*, 31(5):2231–2244, 2023.

[3] Aykut C. Satici, Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. A coordinate-free framework for robotic pizza tossing and catching. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3932–3939, 2016.

[4] Georg Bätz, Arhan Yaqub, Haiyan Wu, Kolja Kühnlenz, Dirk Wollherr, and Martin Buss. Dynamic manipulation: Nonprehensile ball catching. In *18th Mediterranean Conference on Control and Automation, MED'10*, pages 365–370, 2010.

[5] Markus M. Schill, Felix Gruber, and Martin Buss. Quasi-direct nonprehensile catching with uncertain object states. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2468–2474, 2015.

[6] Vincenzo Lippiello, Fabio Ruggiero, and Bruno Siciliano. The effect of shapes in input-state linearization for stabilization of nonprehensile planar rolling dynamic manipulation. *IEEE Robotics and Automation Letters*, 1(1):492–499, 2016.

[7] K.-K.Lee. Basketball robot: Ball on plate with pure haptic information. *Proc. IEEE Int. Conf. Robot. Autom.*, pages 2410–2415.

[8] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking, 2023.

[9] Qinxi Yu, Masoud Moghani, Karthik Dharmarajan, Vincent Schorp, William Chung-Ho Panitch, Jingzhou Liu, Kush Hari, Huang Huang, Mayank Mittal, Ken Goldberg, and Animesh Garg. Orbit-surgical: An open-simulation framework for learning surgical augmented dexterity, 2024.

[10] Sumedh Anand Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Biyik, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[11] Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt-2: Learning precise manipulation from few demonstrations. *ArXiv*, abs/2406.08545, 2024.

[12] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9767–9779. PMLR, 18–24 Jul 2021.

[13] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *6th Annual Conference on Robot Learning*, 2022.

[14] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization, 2020.

[15] Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[16] Dmitry Kalashnikov, Jake Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Scaling up multi-task robotic reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 557–575. PMLR, 08–11 Nov 2022.

[17] Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks, 2022.

[18] Weihao Yuan, Kaiyu Hang, Danica Kragic, Michael Y. Wang, and Johannes A. Stork. End-to-end nonprehensile rearrangement with deep reinforcement learning and simulation-to-reality transfer. *Robotics and Autonomous Systems*, 119:119–134, 2019.

[19] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas A. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *CoRR*, abs/1803.09956, 2018.

[20] Wenxuan Zhou, Bowen Jiang, Fan Yang, Chris Paxton, and David Held. Hacman: Learning hybrid actor-critic maps for 6d non-prehensile manipulation, 2024.

[21] Yoonyoung Cho, Junhyek Han, Yoontae Cho, and Beomjoon Kim. Corn: Contact-based object representation for nonprehensile manipulation of general unseen objects, 2024.

[22] Dexin Wang, Faliang Chang, and Chunsheng Liu. Multi-stage reinforcement learning for non-prehensile manipulation, 2023.

[23] Juan Del Aguila Ferrandis, João Moura, and Sethu Vijayakumar. Nonprehensile planar manipulation through reinforcement learning with multimodal categorical exploration. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5606–5613, 2023.

[24] Kendall Lowrey, Svetoslav Kolev, Jeremy Dao, Aravind Rajeswaran, and Emanuel Todorov. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system. *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 35–42, 2018.

[25] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. 2019.

[26] Kai Lu, Jia-Xing Zhong, Bo Yang, Bing Wang, and Andrew Markham. Learning to catch reactive objects with a behavior predictor. 05 2024.

[27] Yuanhang Zhang, Tianhai Liang, Zhenyang Chen, Yanjie Ze, and Huazhe Xu. Catch it! learning to catch in flight with mobile dexterous hands. *arXiv preprint arXiv:2409.10319*, 2024.

[28] Berthold Bäuml, Thomas Wimböck, and Gerd Hirzinger. Kinematically optimal catching a flying ball with a hand-arm-system. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2592–2599, 2010.

[29] Koichiro Deguchi, Hironari Sakurai, and Shun Ushida. A goal oriented just-in-time visual servoing for ball catching robot arm. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3034–3039, 2008.

[30] Seungsu Kim, Ashwini Shukla, and Aude Billard. Catching objects in flight. *IEEE Transactions on Robotics*, 30(5):1049–1065, 2014.

[31] Seyed Sina Mirrazavi Salehian, Mahdi Khoramshahi, and Aude Billard. A dynamical system approach for softly catching a flying object: Theory and experiment. *IEEE Transactions on Robotics*, 32(2):462–471, 2016.

[32] Saminda Abeyruwan, Alex Bewley, Nicholas M. Boffi, Krzysztof Choromanski, David D'Ambrosio, Deepali Jain, Pannag Sanketi, Anish Shankar, Vikas Sindhwani, Sumeet Singh, Jean-Jacques Slotine, and Stephen Tu. Agile catching with whole-body mpc and blackbox policy learning, 2023.

[33] Kelvin Sheng Pei Dong, Karime Pereida, Florian Shkurti, and Angela P. Schoellig. Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6718–6725, 2020.

[34] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, page 1–8. IEEE Press, 2018.

[35] Jan Matas, Stephen James, and Andrew J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. *ArXiv*, abs/1806.07851, 2018.

[36] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12619–12629, 2018.

[37] Yuqing Du, Olivia Watkins, Trevor Darrell, P. Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1290–1296, 2021.

[38] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation, 2024.

[39] Vivek Myers, Erdem Biyik, and Dorsa Sadigh. Active reward learning from online preferences. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7511–7518, 2023.

[40] Wenxiao Lei, Hao Fu, and Guanghui Sun. Active object tracking of free floating space manipulators based on deep reinforcement learning. *Advances in Space Research*, 70(11):3506–3519, 2022.

[41] David J. Barrientos R., Marie Chantelle C. Medina, Bruno J. T. Fernandes, and Pablo V. A. Barros. The use of reinforcement learning

algorithms in object tracking: A systematic literature review. *Neurocomputing*, 596:127954, 2024.
[42] Wenxuan Zhou and David Held. Learning to grasp the ungraspable with emergent extrinsic dexterity, 2022.
[43] Yixuan Huang, Christopher Agia, Jimmy Wu, Tucker Hermans, and Jeannette Bohg. Points2plans: From point clouds to long-horizon plans with composable relational dynamics. *arXiv preprint arXiv:2408.14769*, 2024.
[44] Lucy Xiaoyang Shi, Joseph J. Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Conference on Robot Learning*, 2022.
[45] Anthony Liang, Jesse Thomason, and Erdem Bıyık. Visarl: Visual reinforcement learning guided by human saliency. *IROS*, 2024.
[46] Sasha Salter, Dushyant Rao, Markus Wulfmeier, Raia Hadsell, and Ingmar Posner. Attention-privileged reinforcement learning, 2021.
[47] Myungsik Cho, Whiyoung Jung, and Youngchul Sung. Multi-task reinforcement learning with task representation method. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022.