Catch It! Learning to Catch in Flight with Mobile Dexterous Hands

Anonymous Author(s)
Affiliation
Address
email

Abstract: Catching objects in flight (i.e., thrown objects) is a common daily skill for humans, yet it presents a significant challenge for robots. This task requires a robot with agile and accurate motion, a large spatial workspace, and the ability to interact with diverse objects. In this paper, we build a mobile manipulator composed of a mobile base, a 6-DoF arm, and a 12-DoF dexterous hand to tackle such a challenging task. We propose a two-stage reinforcement learning framework to efficiently train a whole-body-control catching policy for this high-DoF system in simulation. The objects' throwing configurations, shapes, and sizes are randomized during training to enhance policy adaptivity to various trajectories and object characteristics in flight. The results show that our trained policy catches diverse objects with randomly thrown trajectories, at a high success rate of about 80% in simulation, with a significant improvement over the baselines. The policy trained in simulation can be directly deployed in the real world with onboard sensing and computation, which achieves catching sandbags in various shapes, randomly thrown by humans.

Keywords: Mobile Manipulation, Reinforcement Learning, Catching Objects in Flight

1 Introduction

Humans possess an innate ability to catch thrown objects, a skill that is crucial not only in everyday activities but also in specialized contexts such as athletic sports. The incorporation of similar capabilities in robotic systems has the potential to revolutionize human-robot interaction, particularly in scenarios that involve dynamic handovers. By enabling robots to adeptly perform agile and long-distance catching maneuvers, we can significantly enhance operational efficiency in various applications. Such advancements allow robots to facilitate object transfers between distant locations, thereby completing tasks within the short airborne duration of the objects.

However, existing research on such agile manipulation has notable limitations. Some studies omit mobile platforms [1, 2, 3, 4], restricting the workspace to catch distant objects, while others lack dexterous hands [5, 6], limiting interaction with diverse objects. In contrast, we develop a mobile manipulator with a dexterous hand, expanding the workspace and adapting to diverse objects.

There are several challenges to enable a mobile manipulator with a dexterous hand to catch objects in flight: (i) *accurate and agile whole-body control*: the mobile base and the arm must coordinate to make the arm's end-effector move to the object in flight precisely while the dexterous hand needs to grasp just in time. It also requires agile and real-time movement because the overall execution period only lasts for about 2s, which is the object's flying time in the air. (ii) *high-dimensional action space*: the system, comprising three components, presents a large action space, which complicates the optimization of the control policy. (iii) *randomly thrown and diverse objects*: objects are thrown from random positions with random velocities and vary in shapes, which demands a highly adaptive control policy.

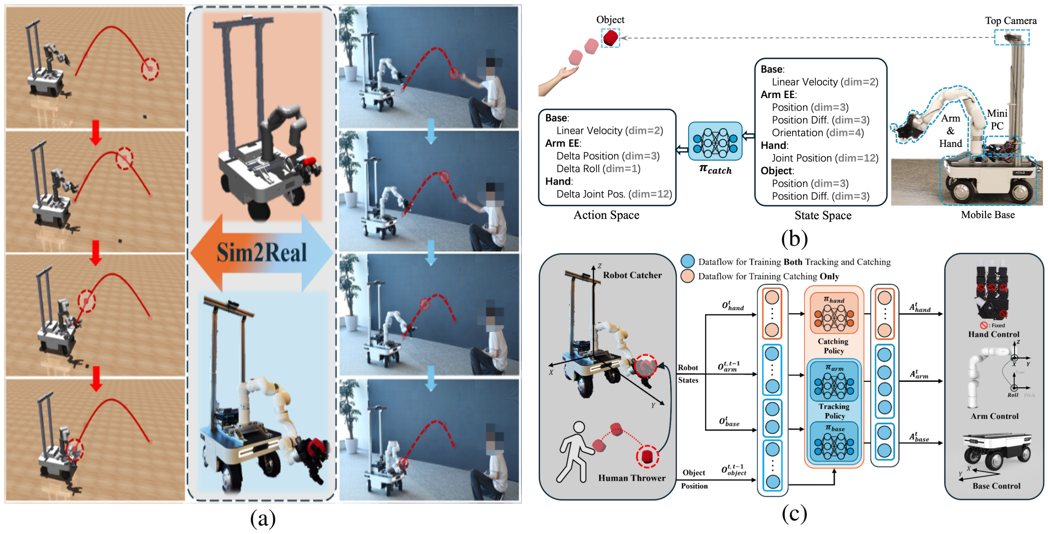

Figure 1: (a) **Sim2Real Catching Motions**. (b) **System Overview**: Our system comprises a mobile base, a 6-DoF arm, and a 16-DoF hand, whose goal is to catch objects thrown randomly by humans. (c) **Two-Stage RL Framework**: We use two consecutive proprioception $O^{t,t-1}$ as the input.

In this work, we propose *Catch It!*, a learning-based method that leverages reinforcement learning (RL) to learn a whole-body control policy to catch objects in flight in simulation, which can also be used to perform sim-to-real (sim2real) transfer on a real robot. The key technical contributions of *Catch It!* are summarized as follows:

1. **Whole-Body Control for Mobile Dexterous Catch**: We train a unified control policy for the base, arm, and hand to be controlled simultaneously. It enables them to work together seamlessly for coordinated, agile, and accurate objects catching skills.
2. **Two-Stage RL Framework**: To deal with the high-dimensional action space, we introduce a model-free RL framework that divides the object-catching task into two subtasks, enhancing training efficiency by focusing on different components in each subtask.
3. **Sim2Real for Mobile Dexterous Catch**: We trained the control policy in simulation with careful design to ensure physical and kinematic alignment with the real-world robot. Using sim2real techniques, we successfully deployed our catching policy on the real robot.

2 System Setup

2.1 Task Description

Our goal is to train a mobile manipulator to catch various objects thrown randomly by humans. Catching objects in flight involves approaching the object with the palm and grasping it stably, which can be divided into two subtasks: (i) The hand needs to track and reach the object. During this phase, only the base and arm are controlled; the hand remains in its initial position. We name this subtask “tracking task”. (ii) When the object is about to be reached (i.e., near the palm), the hand needs to grasp the object. Meanwhile, the base and arm are fine-tuned to achieve optimal grasping position. We name this subtask “catching task”. In the tracking task, if the palm touches the objects in flight, we consider it as a tracking success. In the catching task, if the object keeps being held in hand until the episode’s maximum time, which is set to 2.5s, we consider it a catching success.

2.2 State and Action Space

The state space and the action space are shown in Fig. 1b and the details are explained in A.3. Note that we fix 4 hand joints to reduce the space complexity, improving training efficiency while ensuring graspability. During the tracking task, hand states and actions are excluded.

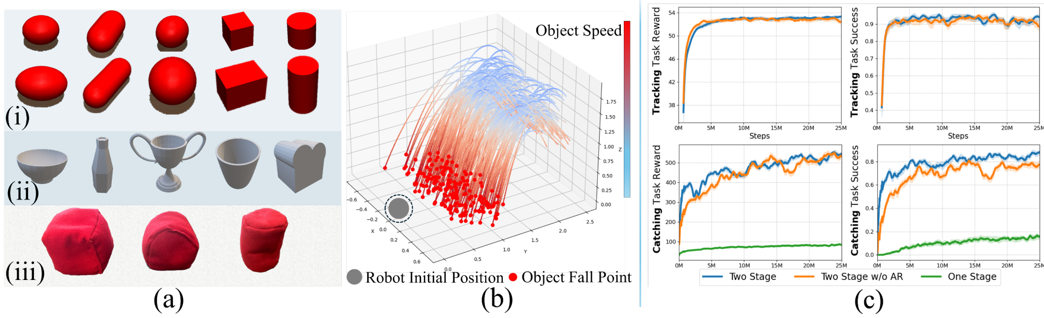

Figure 2: (a): **Object Set Overview**. (i) Objects in training; (ii) Objects in evaluation; (iii) Objects in the real world; (b) **Random Throwing Trajectory Visualization**; (c): **Training Curves**. The blue, orange, and green curves represent the two-stage (T.S.), two-stage without arm's roll (T.S. w/o AR), and one-stage (O.S.) methods. The first row corresponds to the episode rewards and success rates for the tracking task, while the second row shows the same metrics for the catching task.

2.3 System Setup

We construct a mobile manipulator system, as depicted in Fig 1 (b), which is similar to [7] except for a dexterous hand. The details of our real robot system are in A.1 (b). In simulation, we choose Mujoco [8] as our simulation environment and use sw2urdf[1] to build a URDF/MJCF model that mirrors the real robot. For each component of the robot, we develop the PID controller and realize its kinematics respectively. For the arm, we implement its inverse kinematics (ik) to control the joint positions from its end-effector's expected pose.

3 Learning Mobile Dexterous Catching Policies

3.1 Two-Stage Reinforcement Learning

Training the whole-body control policy from scratch to catch objects in flight is inefficient due to the complex dynamics and high-dimensional action space. Thus, our method *Catch It!* leverages a two-stage reinforcement learning (RL) framework to train the catching policy more efficiently. As described in Sec. 2.1, we first train the control policies for the base and arm in the tracking task. Then in the subsequent catching task, we train the hand's control policy while fine-tuning the base and arm's policy from the tracking task, to achieve a better grasping position. In this way, the control policy of the base and arm is pre-trained in the tracking task before starting the catching task, which gives them an initial ability to track and reach the object. Additionally, since the high-dimensional 12-DoF hand movements are unnecessary when the object is distant, fixing the hand in a neutral position during the tracking task training enhances training efficiency. The two-stage RL process is shown in Fig. 1 (c), with Proximal Policy Optimization (PPO) [9] used to train the neural network.

3.2 Reward Design

Careful reward design in RL is the key to train a robust policy successfully. In both tasks, we reward the policy approaching the object and the orientation alignment between the palm and object. We also give a high reward for the the palm touching the object. Finally, we discourage excessive motion via penalizing policy output, and joint limit violation. The reward details are shown in A.2.

4 Experiments

4.1 Thrown Object Settings

We use diverse objects during training, evaluation and real-world deployment, as depicted in Fig. 2 (a). As shown in Fig 2 (b), we randomize the initial positions and velocities of the objects in each

[1]https://github.com/ros/solidworks_urdf_exporter

Track S.R. (%)	Bowls	Bottles	Win-Cups	Cups	Breads
T.S. w/o A.R.	88±4	92±3	90±5	92±5	91±4
T.S. (ours)	**92±3**	**90±4**	**88±3**	**94±5**	**95±4**
Catch S.R. (%)	**Bowls**	**Bottles**	**Win-Cups**	**Cups**	**Breads**
O.S.	22±2	10±3	6±2	13±2	15±3
T.S. (ours)	**84±5**	**78±6**	**65±3**	**80±4**	**80±3**

(a)

Track S.R. (%)	Cube	Sphere	Cylinder	Irregular
T.S. w/o LPF	10	10	5	15
T.S. (ours)	**70**	**65**	**70**	**75**
Catch S.R. (%)	**Cube**	**Sphere**	**Cylinder**	**Irregular**
T.S. w/o LPF	0	0	0	5
T.S. (ours)	**25**	**25**	**15**	**20**

(b)

Table 1: (a) **Evaluation of Unseen Objects in Simulation**. It shows the average Success Rate (S.R.) in the tracking and catching tasks for 200×64 trials; (b) **Evaluation in Real Robot Deployment**. It shows the average Success Rate (S.R.) in the tracking and catching tasks for 40×4 trials.

episode to collect diverse thrown trajectories. Note that the farthest landing point is about 1.5m from the robot's start, which is beyond the arm's reach (about 0.8m), necessitating the mobile base.

4.2 Baselines

We compare our two-stage reinforcement learning framework with the following two baselines:

- **One-Stage without Tracking Task**: In the one-stage baseline, we skip the tracking task and directly train the catching task from scratch. The base and arm's control policy would not be pre-trained from the tracking task.
- **Two-Stage without Arm's Roll**: According to Sec. 2.2, we remove the rolling action of the arm but still train in a two-stage manner.

4.3 Simulation Results

We first compare our two-stage training method with the two baselines on their training performance in simulation, using 64 parallel environments, each running 200 trials. Then, we evaluate their success rate in simulation with the 8 unseen objects. As shown in Table 1 (a) and Fig. 2 (c), ours outperforms both training efficiency and success rates compared to the baselines. In addition, the trained catching policy achieves high catching success rate with unseen and diverse objects, which indicates the effectiveness and adaptability of our method, making it suitable for deployment on real-world robots with unseen object geometries.

4.4 Sim2Real Transfer

There remains a large sim2real gap due to the complexity of our mobile manipulator system. To bridge the sim2real gap, we leverage some techniques explained in A.4 in detail.

4.5 Real-world Deployment

4.5.1 Multi-processing Controller

We develop a multi-processing control system to manage the synchronization among various components of the mobile manipulator, which is depicted in A.5.

4.5.2 Deployment Result

We deployed the trained policy on the real robot across 160 trials (40 per object shape, 20 with and 20 without LPF). As shown in Table 1 (b), the success rates for both tracking and catching were low without LPF. In contrast, with LPF, we achieved a high tracking success rate of approximately 70%, demonstrating the effectiveness of LPF and the robustness of the whole-body control policy trained in simulation. The policy also successfully caught objects of all shapes, highlighting its adativeness in real-world scenarios with varied object geometries. However, the catching success rate did not exceed 25%, the reason for this is further discussed in A.6.

References

[1] K. Deguchi, H. Sakurai, and S. 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. IEEE, 2008.

[2] B. Bauml, T. Wimböck, and G. 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. doi:10.1109/IROS.2010.5651175.

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

[4] S. Kim, A. Shukla, and A. Billard. Catching objects in flight. *IEEE Trans. Robotics*, 30 (5):1049–1065, 2014. doi:10.1109/TRO.2014.2316022. URL `https://doi.org/10.1109/TRO.2014.2316022`.

[5] K. Dong, K. Pereida, F. Shkurti, and A. P. Schoellig. Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6718–6725, 2020. doi: 10.1109/IROS45743.2020.9341134.

[6] S. Abeyruwan, A. Bewley, N. M. Boffi, K. M. Choromanski, D. B. D'Ambrosio, D. Jain, P. R. Sanketi, A. Shankar, V. Sindhwani, S. Singh, et al. Agile catching with whole-body mpc and blackbox policy learning. In *Learning for Dynamics and Control Conference*, pages 851–863. PMLR, 2023.

[7] H. Xiong, R. Mendonca, K. Shaw, and D. Pathak. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024.

[8] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[9] S. John, W. Filip, D. Prafulla, R. Alec, and K. Oleg. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[10] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 561–591. PMLR, 2018. URL `http://proceedings.mlr.press/v87/mahmood18a.html`.

[11] M. Pham, M. Gautier, and P. Poignet. Identification of joint stiffness with bandpass filtering. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 3, pages 2867–2872 vol.3, 2001. doi:10.1109/ROBOT.2001.933056.

A APPENDIX

A.1 Real Robot Setup

The system consists of a Ranger Mini V2 omni-mobile base, a 6-DoF XArm, and a 12-DoF LEAP Hand. To capture the object's real-time 3D positions in the real world, we use an overhead-mounted RealSense D455 camera to extract the object's pixel coordinates and apply a perspective transformation for 3D position estimation relative to the camera. We utilize eye-on-base calibration algorithm to transform this 3D position to the arm's base frame. For the onboard computation, we use a Thunderobot MIX Mini-PC with an i7-13620H CPU and an RTX 4060 GPU. All the components of our robot are powered by the extensive 48V power interface from the Ranger Mini V2.

A.2 Reward Design

Given the times t, the object's 3D position $\boldsymbol{p_t}$ and velocity $\boldsymbol{v_t}$ (estimated as the difference between consecutive positions), the end-effector's position $\boldsymbol{e_t}$, the z-axis vector $\boldsymbol{\hat{u}_t}$, the previous closest hand-to-object distance $\boldsymbol{d_{t-1}}$, and the policy output $\boldsymbol{a_t}$, the detailed reward definitions are:

- **Object Position Reward** (track/catch): The difference of hand-to-object distance in two consecutive time steps during the episode: $r_t^{pos} = \|\boldsymbol{d_{t-1}}\|_2 - \|\boldsymbol{e_t} - \boldsymbol{p_t}\|_2$.
- **Object Precision Reward** (track/catch): This reward scales the d_t with an exponential function, which facilitates learning the policy to approach the target with a higher precision [10]: $r_t^{pre} = \exp(-50 \cdot \|\boldsymbol{d_t}\|_2^2)$.
- **Object Orientation Reward** (track/catch): This reward is computed as the dot product of the delta position vector of the object and the z-axis of the palm, clamped between -1 to 1: $r_t^{orient} = clamp(\boldsymbol{v_t} \cdot \boldsymbol{\hat{u}_t}, -1, 1)$.
- **Object Touch Reward** (track): A binary reward is given if the palm touches the object: $r_t^{touch} = 1 \; or \; 0$.
- **Object Stability Reward** (catch): This reward is computed according to the time length when the object is held by the hand: $r_t^{stab} = \Delta t_{grasp}$.
- **Control Penalty** (track/catch): Penalize the policy output: $r_t^{ctrl} = \|\boldsymbol{a_t}\|_2^2$.
- **Constraint Penalty** (track/catch): A binary penalty is provided if the robot joints exceed their joint limits: $r_t^{cstr} = -1 \; or \; 0$.

The final reward at t, is computed as the weighted sum of the previously mentioned reward terms, each multiplied by a respective scaling coefficient l_k: $r_t^{track/catch} = \sum l_k \cdot r_t^k$.

A.3 State and Action Space

The state space consists of two consecutive 3D positions of the object, 3D position and the orientation of the arm's end-effector, all relative to the arm base fixed on the mobile base, along with the robot's proprioception, including the base's 2D planar velocity in its body frame and the hand joint positions (Fig. 1b). We fix 4 hand joints to reduce the space complexity, improving training efficiency while ensuring graspability. During the tracking task, hand states and actions are excluded.

The action space includes the 2D planar velocity of the base, the 12 delta joint positions of the hand, and the 3D delta position as well as the delta roll rotation of the arm's end-effector. We find that controlling the yaw and pitch axes of the arm's end-effector can destabilize the catch policy training, as these movements often lead to unfavorable hand orientations for successful object catching. In contrast, the roll rotation remains beneficial (Sec. 4.3).

A.4 Sim2Real Transfer

There remains a large sim2real gap due to the complexity of our mobile manipulator system. To bridge the sim2real gap as much as possible, we leverage the following techniques:

A.4.1 Low-Pass Filter

We applied a Low-Pass Filter (LPF) [11] to smooth the velocity commands, ensuring they are executable by the mobile base in the real world.

A.4.2 System Identification

We employ system identification to align the behavior of the PID controllers for the base, arm, and hand between the simulation and the real world. This process serves as a preliminary estimation of the PID parameters within the simulation, which subsequently facilitates the domain randomization.

A.4.3 Domain Randomization

In addition to the randomization of thrown objects discussed in Sec. 4.1, we apply Domain Randomization to the PID parameters, the gravity, the timing of throwing objects, the observation noise, and the action noise. It is important to note that randomizing the throw timing is crucial, as in real-world scenarios, humans typically throw objects at unpredictable moments.

A.5 Multi-process Controller

As depicted in Fig. 3, object's position and proprioceptive states from the base, arm, and hand, collected at different frequencies, are synchronized as inputs for inferring our whole-body control policy, which runs at 25 Hz and matches the control frequency in simulation.

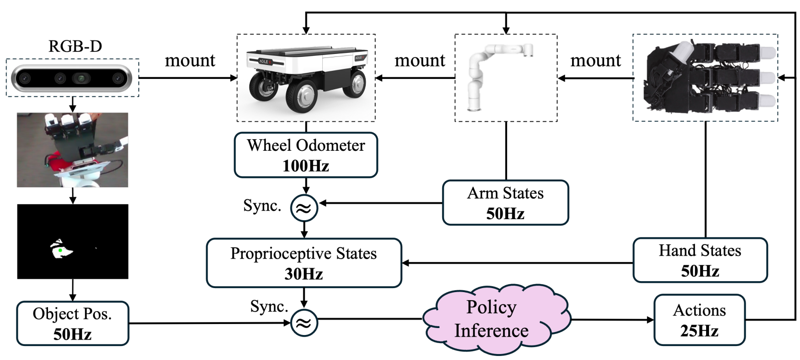

Figure 3: **Multi-Process Controller**. A ROS-based controller synchronizing proprioceptive states and object position data for policy inference in real-time control of the mobile manipulator.

A.6 Relatively Low Catching Success Rate in the Real World

The relatively low catching success rate is primarily caused by the elasticity of the objects (Fig 4), which introduced challenges not present in the simulation. Additionally, the RGB-D camera used for position tracking generated errors when the object moved quickly or was occluded by the hand. We believe integrating a global localization system, such as VICON, could improve catching performance by providing more accurate and robust object tracking.

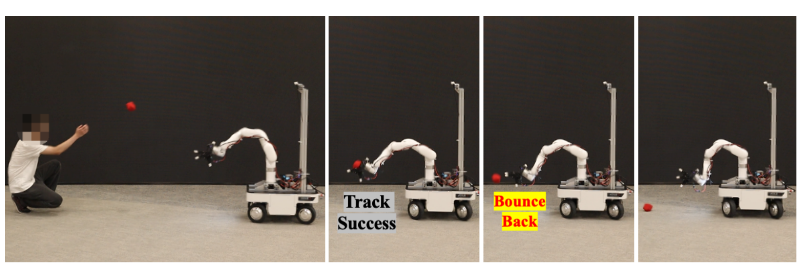

Figure 4: **Failure Case**. The object bounces off the palm.