CONFIDENTIAL. Limited circulation. For review only.

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

Yuanhang Zhang[1†], Tianhai Liang[2†], Zhenyang Chen[4], Yanjie Ze[5] and Huazhe Xu[1,2,3]

***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. Our project page is available at https://mobile-dex-catch.github.io.**

I. 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 [4], [7], [17], [23], restricting the workspace to catch distant objects, while others lack dexterous hands [1], [8], limiting interaction with diverse objects. In contrast, we develop a mobile manipulator with a dexterous hand, expanding the workspace and enabling adaptability 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.

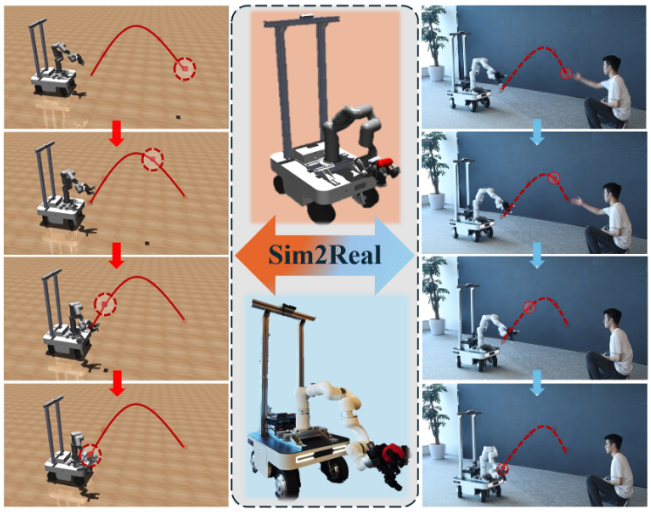


Fig. 1: **Sim2Real Illustration of Catching Motions**.

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 in a zero-shot manner.

† Equal Contribution: Yuanhang Zhang led the project, implemented the training in simulation, and handled real robot controller and deployment; Tianhai Liang conducted the majority of the real robot experiments and contributed to the sim2real and hardware debugging.

[1]Shanghai Qi Zhi Institute. [2]Tsinghua University, [3]Shanghai AI Lab. [4]Georgia Institute of Technology, [5]Stanford University.

Contact: yuanhanz@andrew.cmu.edu, huazhe_xu@mail.tsinghua.edu.cn.

Manuscript 1540 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 15, 2024.

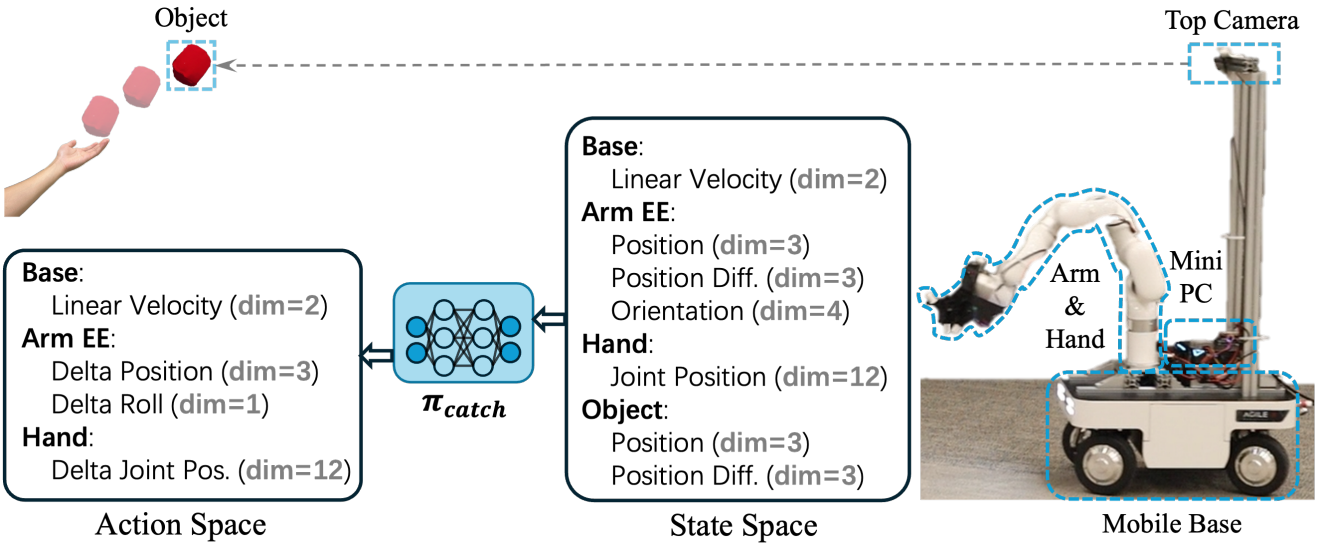


Fig. 2: **System Overview**. Our system comprises a mobile base, a 6-DoF arm, and a 16-DoF hand, whose goal is to catch objects thrown randomly by humans. The state difference (Diff.) is calculated between two consecutive timesteps. We employ a RGB-D camera to track the object in real-time, and a Mini-PC equipped with a GPU to handle onboard computation.

II. Related Work

A. Whole-Body Control for Mobile Manipulation

There has been a large corpus of prior work addressing the whole-body control for mobile manipulation, developing both classical optimization method [3], [5], [6], [11], [22], [24] and learning-based method [9], [12], [18], [19], [28], [31], [34]. In the classical methods [5], [11], a reactive on-the-move architecture is proposed to break down the task (i.e., pick-and-place) into multiple phases. It expresses the coordinately base-and-arm control problem as a quadratic program (QP). Previous works [6], [24] also consider motion safety in mobile manipulation and incorporate the collision-free constraints in the Model Predictive Control (MPC).

While classical methods have achieved some progress in mobile manipulation, they primarily focus on tasks such as navigate-pick-and-place [3], [6], [24] and door-opening [22]. For more complex dynamic tasks, these methods struggle due to intricate optimization modeling. By contrast, learning-based approaches [5], [9], [12], [19], [31], [33] offer promising solutions for addressing complex tasks in mobile manipulation. Recent approaches [9], [33] leverage RL or Imitation Learning (IL) to enable high-DoF mobile manipulators to perform long-horizon tasks in the real-world scenarios. However, they do not explore the implicit relationships between actions and optimization, which could potentially enhance training efficiency. To address this, Wu *et al* [31] introduce "spatial action maps" for fast learning through pixel-wise action generation, while other work [12] proposes *Causal MoMa*, which uncovers causal dependencies between actions and reward function components to improve training efficiency. Inspired by the existing literature [5], [12], this paper enhances efficiency by specifically decomposing the object catching task into two subtasks, employing RL with tailored reward designs to train control policies for the key robot components in each subtasks.

B. Catching Dynamic Objects with Manipulation

Catching objects in flight is a key challenge within Dynamic Object Manipulation (DOM). Previous works have explored *slow* DOM using fixed-base manipulators [2], [13], [14], [21], [26], [32]. A prior study [21] develops a learned grasp planner that evaluates potential grasps online for objects moved by humans, selecting from the collision-free and feasible ones. In [13], [32], the DOM problem is framed as a two-agent RL task: the robot aims to "catch" the object, while the adversarial mover (the object) attempts to evade. This approach enables the trained catching policy to generalize to various object slow-motion patterns.

To learn to catch objects in flight requires greater agility than *slow* DOM, some researchers develop both classical and learning-based methods with an arm, a dexterous hand, and a fixed base [4], [7], [17], [23]. Previous study [4] formulates the ball-catching problem as a unified nonlinear optimization problem to solve. Another line of work [23] approximates the parameters of linear parameter varying system, which generates a reach-and-follow motion to catch flying objects. Kim *et al* [17] learns the object falling dynamics and catching policy from throwing demonstrations. Recent work [15] proposes a bimanual system that is designed for the throw-and-catch tasks using multi-agent RL.

However, when thrown objects land outside the arm's reach, a mobile platform becomes essential to extend the workspace [1], [8]. For example, prior work [1] uses a one-dimensional actuator base with MPC and RL strategies to catch thrown objects, while other work [8] employs an omni-base with a bi-level catching motion planning algorithm and a learning-based controller for tracking. Nevertheless, previous works have not integrated a dexterous hand with mobile manipulators, limiting their generalization to diverse object shapes. In this work, we develop an omni-mobile manipulator with a dexterous hand and use model-free RL to train a catching policy for various randomly thrown objects.

Manuscript 1540 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 15, 2024.

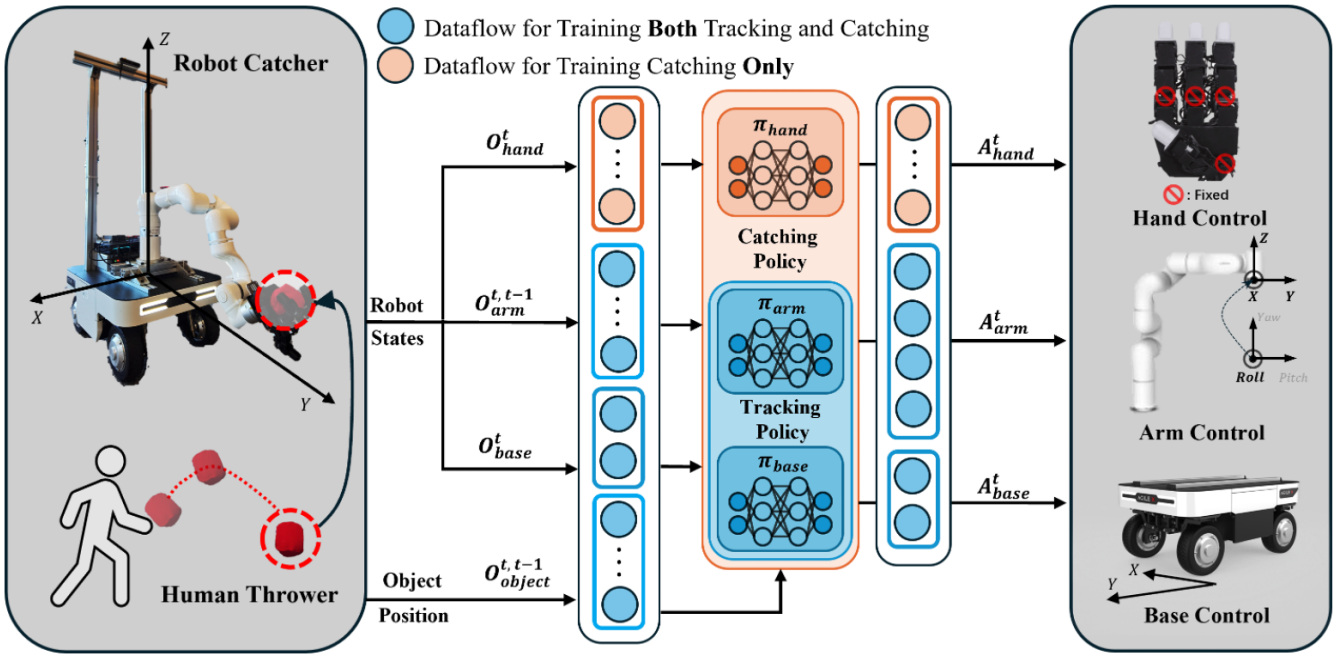


Fig. 3: **Two-Stage RL Framework**. Note that we use two consecutive prioproception $O^{t,t-1}$ as the policy input.

III. System Setup

A. 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".

B. State and Action Space

The state space consists of two consecutive 3D positions of the object and the arm's end-effector, both 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. 2). 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. V-B.2).

C. Real-world Setup

We construct a mobile manipulator system, as depicted in Fig 2. 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.

D. Simulation Setup

We choose Mujoco [30] as our simulation environment. We 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:

1) Mobile Base: The motion patterns include *Double-Ackerman* [16] with forward and yaw velocity as input, and *Parallel* with forward and horizontal velocity as input. Here, we only apply *Parallel* motion pattern, because the base control policy only outputs 2D planar velocity.

2) Arm: We use inverse kinematics (ik) to control the joint positions of the 6-DOF arm, from its end-effector's expected pose. We implement it with two numerical methods: Levenberg-Marquardt (LM) and Quadratic Programme (QP) Method, both of which can find the joint positions corresponding to predefined end-effector position. Besides, the null-space motion is considered to satisfy the angle limits for different joints. For further implementation details, we refer readers to the previous work [10].

3) Hand: For the LEAP hand, we use a position actuator to control each joint's position.

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

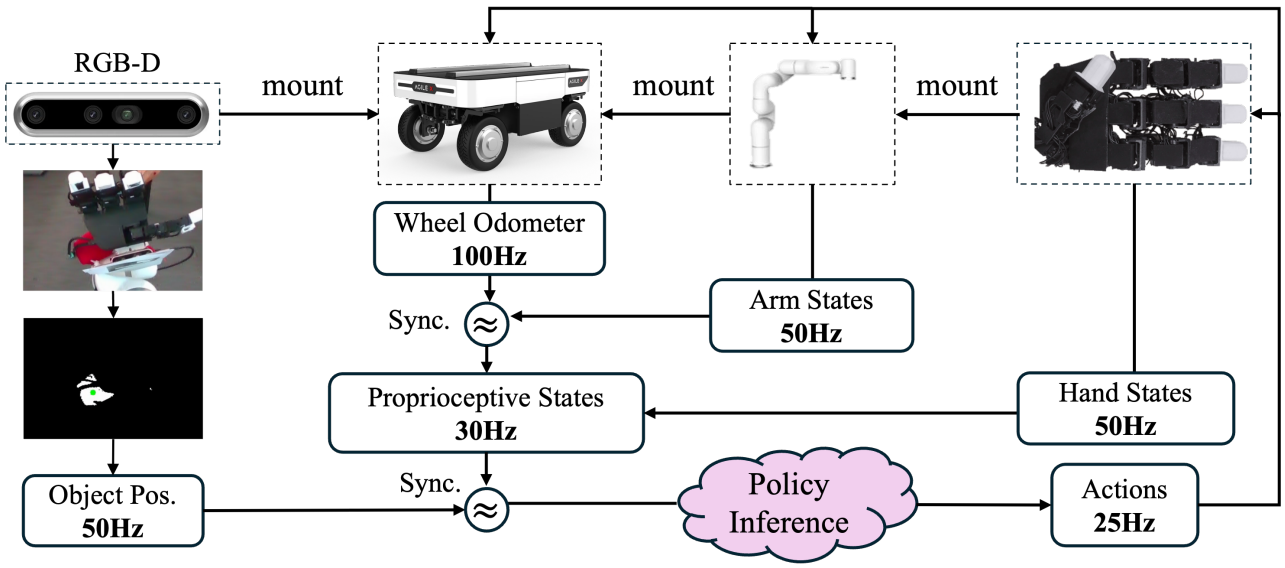


Fig. 4: **Multi-Process Controller**. A ROS-based multi-process controller synchronizes proprioceptive states and object position data for policy inference in real-time control of the mobile manipulator.

IV. Learning Mobile Dexterous Catching Policies

A. Two-Stage Reinforcement Learning

Training the whole-body control policy from scratch to catch objects in flight is inefficient in the face of such complex dynamic task with a high-dimensional action space. Thus, our method *Catch It!* leverages a two-stage reinforcement learning framework to train the catching policy more efficiently. As described in Sec. III-A, 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 of training enhances training efficiency. This two-stage reinforcement training framework is illustrated in Fig. 3.

The tracking and the catching task can be formulated as a Partially-Observable Markov Decision Process (POMDP), defined by the tuple (S, O, A, R, P) where S is the state space partially observed by O, the observation space, A is the action space, $R : S \times A \mapsto \mathbb{R}$ is the reward function and $P : S \times A \mapsto S$ is the dynamics function. The optimization objective is to learn a parameterized policy $\pi_\theta : O \mapsto A$ that maximizes the expected total episode return, $J(\theta) = E_{\pi=(s_0,a_0,\ldots,s_T)} \sum_{t=0}^{T} r(s_t, \pi_\theta(o_t))$. The state space consists of the object positions, which are approximated from the raw observations generated by the real-time object tracking system described in Sec. III-C, thereby justifying the POMDP categorization. In this work, we use Proximal Policy Optimization (PPO) [27] to train our neural network.

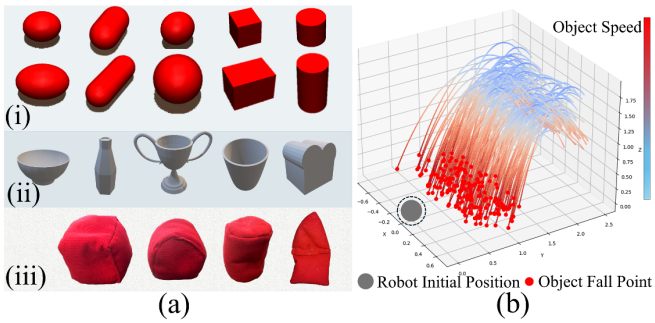
B. Reward Design

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.

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 $\hat{\boldsymbol{u}}_{\boldsymbol{t}}$, 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 [20]: $r_t^{pre} = \exp(-50 \cdot \|\boldsymbol{d_t}\|_2^2)$.
- **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: $r_t^{orient} = clamp(\boldsymbol{v_t} \cdot \hat{\boldsymbol{u}}_{\boldsymbol{t}}, -1, 1)$.
- **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): To avoid excessive motion, we 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) (b)

Fig. 5: **Object Set Overview**. (a): (i) Objects in training; (ii) Objects in evaluation; (iii) Objects in real-world deployment. (b) Random Throwing Trajectory Visualization.

V. Experiments

A. Experiment Settings

1) Training Settings: We create 64 parallel environments to collect large amounts of trajectories of various objects to train our tracking and catching policy. The training process is performed on 128 Intel i7 CPUs and a single NVIDIA GeForce RTX 3090 Ti GPU.

2) Frequency Settings: In the simulation, the basic step frequency is set to 500 Hz. The policy inference is made every 20 simulation steps, corresponding to a frequency of 25 Hz. During each set of 20 simulation steps, the robot consistently executes the same action as determined by the latest inference. When deploying the control policy on the real robot, we maintain a control frequency of 25 Hz, ensuring consistency with the simulation.

3) Thrown Object Settings: In the training process, We use objects in 5 different shapes: a box, a sphere, an ellipsoid, a cylinder and a capsule, as shown in Fig. 5 (a) (i). Among these objects, the cylinder and capsule may require the robot to utilize all of its fingers [15]. We randomize the object's shapes, sizes, mass, and damping in each environment initialization only. To validate the policy's generalization in simulation, we test with 5 unseen object shapes, as shown in Fig. 5 (a) (ii). In the real-world experiments, we employ sandbags in 4 different shapes, as shown in Fig. 5 (a) (iii): a ball, a cube, a cylinder, and an irregular combination formed by stitching together a triangular sandbag and a flat sandbag.

To collect diverse object trajectories during training, we randomize the initial positions and velocities of the obejcts in each episode. As shown in Fig 5 (b), the farthest landing point is approximately 1.5m from the robot's starting position, which is beyond the arm's reach (about 0.8m), necessitating the use of a mobile base.

4) 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. III-B, we remove the rolling action of the arm but still train in a two-stage manner.

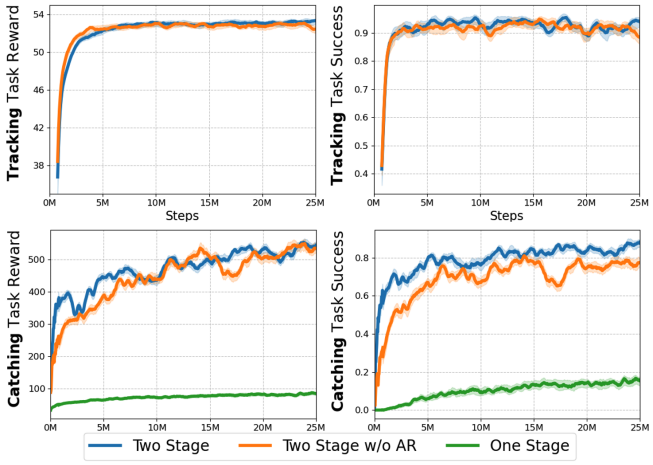


Fig. 6: **Training Curves**. The blue, orange, and green curves represent the two-stage (T.S.), two-stage without arm's roll (T.S. w/o AR), and one-stage (O.S.) methods, respectively. The first row corresponds to the episode rewards and success rates for the tracking task over the training steps, while the second row shows the same metrics for the catching task.

B. Simulation Results

We first compare our two-stage training method with the two baselines on their training performance in simulation. Then, we evaluate their success rate in simulation with the 8 unseen objects. 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.

1) Two-Stage versus One-Stage: Fig. 6 shows that the two-stage method (ours) significantly outperforms the one-stage in both training efficiency and final success rate, highlighting both the superiority and necessity of our method for achieving great catching performance discussed in Sec. IV-A.

2) Two-Stage versus Two-Stage w/o AR: Fig. 6 shows that while ours and two-stage w/o arm rolling both achieve efficient training and high success rates in the tracking task, our method stands out in terms of success rates in the catching task. This advantage is due to the roll action, which benefits the orientation reward and optimizes the palm's orientation for catching.

3) Evaluation: As shown in Table I, our method outperforms the other two baselines in the catching success rate for unseen objects. Moreover, our trained policy demonstrates robust generalization to unseen objects, achieving high catching success rates across various shapes not encountered during training. This indicates the effectiveness and adaptability of our method, making it suitable for deployment on real-world robots with unseen object geometries.

C. 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:

- **Low-Pass Filter**: For the Ranger Mini V2 mobile base, we observe that, unlike in simulation where the base can

Manuscript 1540 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 15, 2024.

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. w/o A.R.	80±6	73±4	61±7	77±5	75±3
T.S. (ours)	**84±5**	**78±6**	**65±3**	**80±4**	**80±3**

TABLE I: **Evaluation of Unseen Objects in Simulation**. It shows the average Success Rate (S.R.) in the tracking and catching tasks for 200×64 trials.

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

TABLE II: **Evaluation in Real Robot Deployment**. It shows the average Success Rate (S.R.) in the tracking and catching tasks for 40×4 trials.

steer immediately while maintaining forward motion, significant adjustments to the steering angle during forward motion are not feasible in the real world. This limitation causes the base to brake with frequent, large steering changes in velocity commands. To mitigate this, we applied a Low-Pass Filter (LPF) [25] to smooth the velocity commands, as shown in Fig. 7, ensuring they are executable by the mobile base in the real world.

- **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 process.
- **Domain Randomization**: Domain Randomization has been proven to be effective in the sim2real transfer [29]. In addition to the randomization of thrown objects discussed in Sec. V-A.3, 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.

D. Real Robot Deployment

1) Controller: We develop a multi-processing control system based on ROS to manage the various components of the mobile manipulator, as depicted in Fig. 4. Proprioceptive states from the base, arm, and hand are collected at different frequencies and synchronized to 30 Hz. The object's position is obtained from an RGB-D camera operating at 50 Hz. Both the proprioceptive data and the object's positional information are synchronized and used as inputs for inferring our whole-body control policy, which runs at 25 Hz and matches the control frequency in simulation.

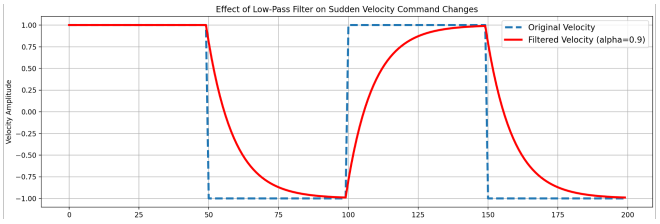


Fig. 7: **Low-Pass Filter Curve**. The comparison between the original and filtered velocity commands, using a recursive coefficient of 0.9, consistent with the real robot deployment.

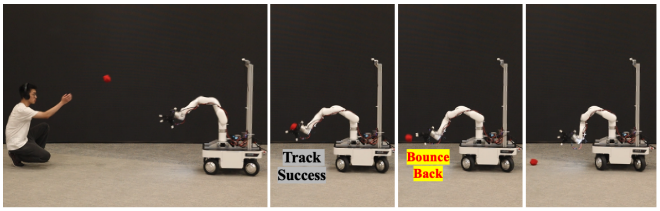


Fig. 8: **Failure Case**. The object bounces off the palm.

2) Deployment Result: We deployed the trained catching policy on the real robot across 160 trials, with 40 trials per object shape. Each set of 40 trials included 20 with the LPF and 20 without. As shown in Table II, 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%, primarily due to the elasticity of the objects (as shown in Fig 8), 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.

VI. Conclusion and Limitation

In this work, We present *Catch It!*, addressing the challenge of catching flying objects with a mobile manipulator and dexterous hand. We propose a two-stage reinforcement learning framework which efficiently trains a whole-body control policy to catch randomly thrown objects in various shapes. Results show high success rates in simulation and successful real-world deployment using sim2real techniques.

However, when deploying the trained policy in the real world, our system may struggle with elastic objects, which tend to rebound off contact surfaces, particularly when both of the object and robot move at high speeds. Therefore, in the future, we plan to refine the hand's grasping strategy with tactile sensors to better manage rebounding dynamics. We are committed to releasing the simulation training code and providing references for real robot deployment.

Manuscript 1540 submitted to 2025 IEEE International Conference on Robotics and Automation (ICRA). Received September 15, 2024.

REFERENCES

[1] Saminda Abeyruwan, Alex Bewley, Nicholas Matthew Boffi, Krzysztof Marcin Choromanski, David B D'Ambrosio, Deepali Jain, Pannag R Sanketi, Anish Shankar, Vikas Sindhwani, Sumeet 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.

[2] Artemij Amiranashvili, Alexey Dosovitskiy, Vladlen Koltun, and Thomas Brox. Motion perception in reinforcement learning with dynamic objects. In *Conference on Robot Learning*, pages 156–168. PMLR, 2018.

[3] Max Bajracharya, James Borders, Richard Cheng, Daniel M. Helmick, Lukas Kaul, Dan Kruse, John Leichty, Jeremy Ma, Carolyn Matl, Frank Michel, Chavdar Papazov, Josh Petersen, Krishna Shankar, and Mark Tjersland. Demonstrating mobile manipulation in the wild: A metrics-driven approach. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.

[4] Berthold Bauml, 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.

[5] Ben Burgess-Limerick, Chris Lehnert, Jürgen Leitner, and Peter Corke. An architecture for reactive mobile manipulation on-the-move. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1623–1629. IEEE, 2023.

[6] Jia-Ruei Chiu, Jean-Pierre Sleiman, Mayank Mittal, Farbod Farshidian, and Marco Hutter. A collision-free mpc for whole-body dynamic locomotion and manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4686–4693, 2022.

[7] 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. IEEE, 2008.

[8] Ke Dong, Karime Pereida, Florian Shkurti, and Angela 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.

[9] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *arXiv*, 2024.

[10] Jesse Haviland and Peter Corke. Manipulator differential kinematics: Part i: Kinematics, velocity, and applications. *IEEE Robotics and Automation Magazine*, pages 2–11, 2023.

[11] Jesse Haviland, Niko Sünderhauf, and Peter Corke. A holistic approach to reactive mobile manipulation. *IEEE Robotics and Automation Letters*, 7(2):3122–3129, 2022.

[12] Jiaheng Hu, Peter Stone, and Roberto Martín-Martín. Causal policy gradient for whole-body mobile manipulation. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.

[13] Zhe Hu, Yu Zheng, and Jia Pan. Grasping living objects with adversarial behaviors using inverse reinforcement learning. *IEEE Transactions on Robotics*, 39(2):1151–1163, 2023.

[14] Baichuan Huang, Jingjin Yu, and Siddarth Jain. Earl: Eye-on-hand reinforcement learner for dynamic grasping with active pose estimation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2963–2970. IEEE, 2023.

[15] Binghao Huang, Yuanpei Chen, Tianyu Wang, Yuzhe Qin, Yaodong Yang, Nikolay Atanasov, and Xiaolong Wang. Dynamic handover: Throw and catch with bimanual hands. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pages 1887–1902. PMLR, 2023.

[16] Lionel Hulttinen and Jouni Mattila. Flow-limited path-following control of a double ackermann steered hydraulic mobile manipulator. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM 2020, Boston, MA, USA, July 6-9, 2020*, pages 625–630. IEEE, 2020.

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

[18] Julien Kindle, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Roland Siegwart, and Juan Nieto. Whole-body control of a mobile manipulator using end-to-end reinforcement learning. *arXiv preprint arXiv:2003.02637*, 2020.

[19] Thomas Lew, Sumeet Singh, Mario Prats, Jeffrey Bingham, Jonathan Weisz, Benjie Holson, Xiaohan Zhang, Vikas Sindhwani, Yao Lu, Fei Xia, Peng Xu, Tingnan Zhang, Jie Tan, and Montserrat Gonzalez. Robotic table wiping via reinforcement learning and whole-body trajectory optimization. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7184–7190, 2023.

[20] A. Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James 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.

[21] Naresh Marturi, Marek Sewer Kopicki, Alireza Rastegarpanah, Vijaykumar Rajasekaran, Maxime Adjigble, Rustam Stolkin, Ales Leonardis, and Yasemin Bekiroglu. Dynamic grasp and trajectory planning for moving objects. *Auton. Robots*, 43(5):1241–1256, 2019.

[22] Maria Vittoria Minniti, Farbod Farshidian, Ruben Grandia, and Marco Hutter. Whole-body mpc for a dynamically stable mobile manipulator. *IEEE Robotics and Automation Letters*, 4(4):3687–3694, 2019.

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

[24] Johannes Pankert and Marco Hutter. Perceptive model predictive control for continuous mobile manipulation. *IEEE Robotics and Automation Letters*, 5(4):6177–6184, 2020.

[25] M.T. 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.

[26] Seyed Sina Mirrazavi Salehian, Nadia Figueroa, and Aude Billard. Dynamical system-based motion planning for multi-arm systems: Reaching for moving objects. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4914–4918. ijcai.org, 2017.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[28] Charles Sun, Jedrzej Orbik, Coline Manon Devin, Brian H Yang, Abhishek Gupta, Glen Berseth, and Sergey Levine. Fully autonomous real-world reinforcement learning with applications to mobile manipulation. In *Conference on Robot Learning*, pages 308–319. PMLR, 2022.

[29] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[30] Emanuel Todorov, Tom Erez, and Yuval 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.

[31] Jimmy Wu, Xingyuan Sun, Andy Zeng, Shuran Song, Johnny Lee, Szymon Rusinkiewicz, and Thomas A. Funkhouser. Spatial action maps for mobile manipulation. In Marc Toussaint, Antonio Bicchi, and Tucker Hermans, editors, *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*, 2020.

[32] Tianhao Wu, Fangwei Zhong, Yiran Geng, Hongchen Wang, Yongjian Zhu, Yizhou Wang, and Hao Dong. Grasparl: Dynamic grasping via adversarial reinforcement learning. *arXiv preprint arXiv:2203.02119*, 2022.

[33] Haoyu Xiong, Russell Mendonca, Kenneth Shaw, and Deepak Pathak. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024.

[34] Ruihan Yang, Yejin Kim, Aniruddha Kembhavi, Xiaolong Wang, and Kiana Ehsani. Harmonic mobile manipulation. *arXiv preprint arXiv:2312.06639*, 2023.