Perceptive Position-Conditioned Local Navigation for Agile Quadrupedal Object Catching

Abstract— **Legged robots require robust agility to perceive and interact with complex and dynamic environments within a constrained time. However, most existing quadruped locomotion works rely on velocity-based policies, which struggle to reach precise targets within strict temporal constraints. Moreover, perception and locomotion are frequently coupled, sometimes resulting in instability during dynamic object interception. To concretely study and benchmark such agility in dynamic settings, we introduce a challenging ball-catching task for legged robots. This paper proposes a decoupled framework that combines a vision module for landing point and time prediction with a position-conditioned RL locomotion policy, effectively handling the tight coupling between perception and locomotion in the dynamic object interception task. We conducted extensive ball-catching experiments for the legged robot. Through comparative experiments against a velocity-based locomotion baseline, our position-based approach achieves a higher success rate in catching balls with predicted landing spots within 2 meters and flight times between 0.8 and 1.2 seconds. This shows that the robot has successfully finished the agile high-speed ball-catching task. Furthermore, our policy exhibits a smaller performance gap after deployment, proving its superior sim-to-real adaptability.**

I. Introduction

Agility is the ability to rapidly perceive, plan, and execute coordinated whole-body motions under tight time constraints. This is reflected in the robot's extremely reactive performance to specific tasks, such as catching a ball within a short time. These tasks are difficult for quadruped robots, but easy for animals. For example, dogs have high-fidelity perception, fast prediction of ballistic trajectories, and coordinated whole-body dynamics to intercept thrown objects with apparent ease. For legged robots, reproducing this behavior is not merely a question of locomotion speed: it requires proper whole-body coordination and pushing the limits of sensing latency, state estimation, prediction accuracy, and dynamic control simultaneously. Small errors in perception or timing can render an otherwise agile gait useless for interception, while overly aggressive control can induce instability, falls, or hardware stress. [6], [18], use perception-based methods to enhance the real-time performance and motion accuracy.[28], [20], [4], [9], [23], [14] use reinforcement learning (RL) methods to produce agile locomotion performance and obstacle avoidance strategies in challenging terrains. [29], [13], [7] combine multi-stage pipelines through multi-skill learning, enhancing task coverage and generalization capabilities. However, few studies explicitly tackle the perceptive position-conditioned, time-aware interception problem. The current methods usually optimize either velocity tracking, orientation, or energy consumption separately or their linear combination, lacking an end-to-end strategy design that can simultaneously balance these objectives holistically under strict time constraints. The time-constrained positionally goal-conditioned training fashion provides a more holistic and more general end-to-end approach, opening up a larger space of possible solutions that may lead to the discovery of efficient motion patterns[20].

In this work, we focus on the position-conditioned, time-aware regime and take the perceptive dynamic ball catching task as the case of extreme agility. We identify and address two practical challenges that are often underestimated in prior RL locomotion work: (1) *temporal misalignment* where policies that minimize spatial error do not necessarily arrive at the correct time for interception and (2) degenerate locomotion which means naive emphasis on rapid arrival can produce pathological gaits, such as base sinking or excessive pitching, or poor omnidirectional responsiveness. To overcome these issues, we integrate a locally coordinated perception pipeline that provides both predicted impact positions and explicit time-to-impact estimates, and we design an RL objective that prioritizes spatial accuracy and arrival-time synchronization while preserving stability and energy efficiency.

The main contributions are summarized as follows:

1) Perceptive position-based local navigation for extreme agility: We train an agile control policy with RL for the quadruped robot, which enables it to receive position-conditioned commands and scheduled time and reach this position within the constrained time.
2) Dynamic object catching with real-time perceptive locomotion system: We proposed a decoupled framework that contains a position-based RL locomotion policy, integrated perception and prediction modules to address the tight coupling between perception and locomotion system in dynamic object catching.
3) We conduct extensive experiments to verify the effectiveness of our proposed method and the superiority over the velocity-based baseline.

II. Related Work

A. Legged locomotion for extreme agility

Legged RL has recently pushed agility, adaptability, and skill composition to new heights. Works like [28], [20], [4] demonstrate that policies can navigate risky and irregular terrains, perform obstacle-traversal, and achieve strong locomotion robustness. Safety and contact constraints have also been incorporated in high-speed locomotion in [9], while curriculum and multi-skill learning methods, like [13], [7],

[29], enable flexible switching among behaviors and more generalization. Additionally, recent work on jumping behaviors via RL with impedance matching shows that legged robots can perform large jumps in real hardware, such as 55 cm distance or 38 cm height, while still maintaining stable walking across multiple directions. [8]

On the catching or interception side, a number of works combine perception, prediction, and locomotion. For example, [6] uses an event camera for very low latency catching; [21] enables a quadruped robot's front legs to catch thrown objects using vision and trajectory prediction; [29] trains a mobile manipulator with a dexterous hand to track and then catch in flight. Many of these focus on minimizing perception delay or improving spatial observability, but seldom incorporate explicit time-limited conditions into the policy or shape rewards to enforce synchronized interception. Our work addresses precisely that gap: by conditioning policies on time-to-impact estimates and designing timing-aware reward signals, we aim to align arrival time with landing events under realistic sensing and actuation constraints.

B. RL for Position-Conditioned Legged Locomotion

Position-conditioned policies, where the low-level controller receives target poses or positions as commands, have become a standard design choice for modular quadruped control, as they simplify high-level planning while reusing learned primitives across tasks. End-to-end and hierarchical RL approaches have demonstrated strong agility and generalization. For example, [28] presented a generalist locomotion policy fine-tuned for risky terrains such as stepping stones and narrow beams, achieving real-world traversal at speeds above 2.5 m/s. Similarly, [20] introduced an end-to-end framework combining locomotion and navigation via dense pose/velocity shaping rewards. Hierarchical skill-selection methods, such as [10], extend locomotion policies with jump, crouch, and climb skills to achieve parkour-like agility. Other studies leverage priors, residual learning, or skill adaptation to broaden flat-terrain policies to complex terrain [19], [11], while whole-body loco-manipulation combines locomotion with manipulation by conditioning on task-space poses [15].

Despite these advances, two issues remain underexplored: (1) *temporal misalignment*, where position-conditioned policies lack explicit objectives to synchronize arrival with event timing, and (2) *degenerate locomotion*, where naive position tracking can cause undesirable behaviors such as base sinking or limited omnidirectional responsiveness. Our work explicitly addresses these limitations by augmenting the observation space with time-to-impact information and by designing fused rewards that preserve locomotion stability while ensuring interception timing.

C. Object Detection and Tracking

Object detection is a foundational step for tracking and subsequent trajectory prediction. A common approach is to mark the target object with a specific color and segment it within the frame by selecting a predefined range in the HSV (hue-saturation-value) color space. Alternatively, neural networks such as YOLOv8n[25] can be used to detect and localize the target via bounding boxes. Once 2D image coordinates are obtained, an Intel RealSense RGB-D camera can provide depth information, enabling the computation of the object's 3D position relative to the camera using intrinsic and extrinsic calibration parameters. In this work, we use YOLOv13n[12] as a cutting-edge object detector, along with an RGB-D camera, to achieve robust detection and localization.

Another widely used method involves motion capture systems such as VICON[5], which offer highly accurate 3D tracking through marker-based detection. However, such systems require extensive multi-camera setups and physical markers attached to objects, limiting their flexibility in unstructured environments.

Frame-based cameras are frequently used to follow flying objects in systems such as the Hamlet badminton robot[21] and the *"Catch It!"* mobile manipulator[29]. These systems typically rely on onboard cameras, which are susceptible to motion blur and occlusion during rapid robot movements. To mitigate these issues, some works propose using event cameras [6], which offer high temporal resolution and reduced latency, enabling tracking at speeds up to 15 m/s. However, many vision-based methods impose specific operational constraints. In contrast, our work adopts a globally coordinated perception system that uses two global cameras for both ball tracking and robot localization. This design decouples perception from robot motion, minimizing issues such as blur and occlusion, and providing reliable state estimates for robot tracking tasks.

D. Dynamic Trajectory Prediction

Accurately predicting the trajectory of dynamic objects remains a core challenge in robotic interception. Traditional model-based techniques, such as Kalman filters and extended Kalman filters (EKF), form a common foundation by integrating simplified physical dynamics, including gravity, for trajectory propagation. Regression-based methods, notably including [21], are also widely employed. These techniques are computationally efficient and perform reliably under predictable motion models.

With advances in machine learning, data-driven methods have gained prominence. For instance, recurrent neural networks (RNNs) such as the Neural Acceleration Estimator (NAE) [27] have been used to learn complex dynamics and improve prediction accuracy. Nevertheless, such approaches typically require large datasets and extensive training.

Some methods seek to increase reaction time by observing the thrower's motion before object release [6], while others leverage event-based sensing to overcome conventional latency-bandwidth trade-offs. Despite these innovations, many prediction frameworks remain vulnerable to perceptual delays and environmental disturbances.

In this work, we use a Kalman filter for trajectory prediction due to its simplicity and adequacy for our task.

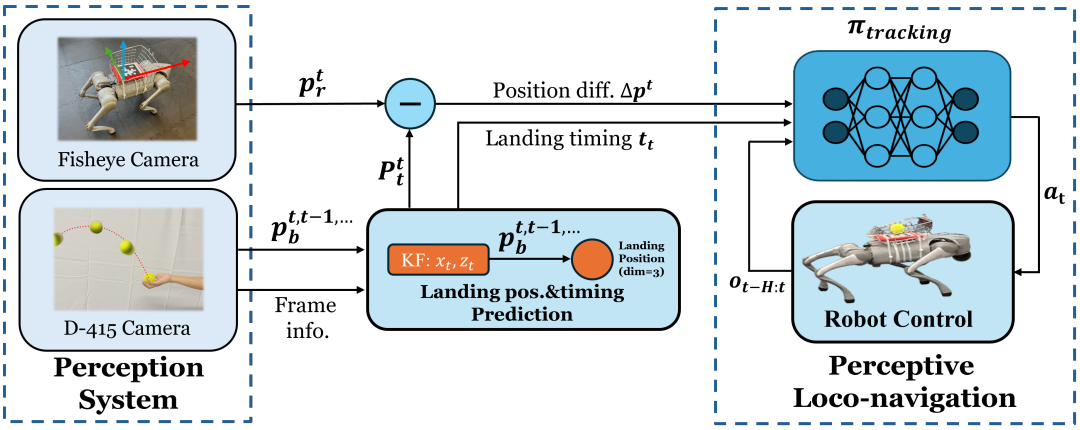

Fig. 1: **System Overview**. Our system comprises a visual part and an RL part. The visual part gets the RGB input of the ball and the April-tag on the back of the robot, and outputs the position of the robot and the estimated ball landing position and time. Then the RL part gets that information as policy input to send the action to the robot.

III. Method

Our goal is to develop a framework for quadruped robots that couples perception and prediction with RL locomotion training. This framework is capable of capturing a moving ball at high speed and calculating the landing position and timing before the ball hits the ground, allowing the robot to catch it successfully. In this section, we will elaborate on how we utilize the vision-based localization and trajectory prediction, time-aware RL, and teacher-student distillation framework-based local navigation system to accomplish the task of high agility. In this section, we elaborate on our proposed time-aware and position-conditioned RL locomotion training framework and how we integrate vision-based localization and object trajectory prediction for the dynamic object catching, and finally enable the agile locomotion.

A. Visual Perception and Prediction

Agility in catching tasks is not only limited by the controller but is strongly constrained by the perception latency and robustness. To ensure that our reinforcement learning (RL) policy receives precise target states under rapid motions, we design a globally coordinated, multi-stage perception pipeline. Our pipeline leverages a global coordination setup to overcome the limitations of frame-based[1] and event-based approaches[6].

1) Globally Coordinated Multi-Camera Framework: We deploy an Intel RealSense D415 depth camera near the release point of the tennis ball and a fisheye camera to track the quadruped robot via AprilTags. The ground AprilTag defines the world origin, while the onboard AprilTag provides robot position states $\boldsymbol{p}_r^t$. The transformation from camera to world frame is defined as

$$\boldsymbol{p}_{\text{world}} = T_{\text{cam}}^{\text{world}} \cdot \boldsymbol{p}_{\text{cam}} \tag{1}$$

To deal with the jitter of AprilTag between frames and raise the agility of our project, we average the basis vectors and origin position across a sliding window before orthonormalization. This would yield a temporally stable transformation matrix. Compared to an onboard binocular depth camera, this design provides centimeter-level accuracy without reduction under high-speed throwing or robot-induced motion blur.

2) Lightweight Object Detection and Throw Event Triggering: Dynamic objects such as tennis balls are detected using a fine-tuned YOLOv13n[12] model trained on thousands of images under various lighting and motion conditions. To further improve robustness, a simple HSV threshold-based detector runs in parallel as a fallback. This hybrid design ensures reliable detections across cluttered scenes while maintaining inference latency below 10 ms per frame. Once detected, a finite-state machine (FSM) monitors velocity and displacement signals to identify the actual throwing event. A transition from State 0 `WAITING` to State 1 `THROWN` occurs only when

$$C_{0\to1} = C_{\text{Zvel}} \wedge C_{\text{height}} \wedge C_{\text{data-valid}} \tag{2}$$

where $C_{\text{Zvel}} : v_z > v_{\text{thresh}}$, $C_{\text{height}} : z > h_{\text{thresh}}$ and $C_{\text{data-valid}} : \Delta t < \Delta t_{\text{thresh}}$. This prevents false positives caused by small jitters, unlike event-based systems that often fail at low speeds.

3) Trajectory and Landing Position and Timing Prediction: After the object thrown is confirmed, a Kalman Filter estimates the object's 3D position $\boldsymbol{p}_b^{t,t-1,\cdots}$ and velocity under a parabolic motion model:

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{w}_t, \quad \boldsymbol{z}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{v}_t \tag{3}$$

where $\boldsymbol{w}_t$ and $\boldsymbol{v}_t$ is the process and measurement noise, $\boldsymbol{x}_t = [\boldsymbol{p}, \boldsymbol{v}]^T$ the state vector with position and velocity, and $\boldsymbol{z}_t = [\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]^T$ the measurement vector. Even when YOLO temporarily misses detections, the filter maintains stable state estimates.

4) Integration with RL Policy: The perception pipeline, as shown in Fig. 1, outputs three key variables: the robot position in the world frame $\boldsymbol{p}_r^t$, the predicted landing coordinates $\boldsymbol{p}_t^t$, and the estimated landing time t_t. We would get $\Delta \boldsymbol{p}^t$ from the difference between the robot position and the landing point coordinates. $\Delta \boldsymbol{p}^t$ and t_t are transmitted via LCM to the Jetson Orin Nano onboard and form part of the observation space for the RL policy. By conditioning

the policy on both spatial and temporal predictions, the quadruped can align its arrival time with the object's landing while preserving stable locomotion.

B. Reinforcement Learning for Position-conditioned Locomotion

We employ the teacher-student distillation framework following[26] and use Proximal Policy Optimization (PPO)[22] for RL policy training, as illustrated in Fig. 2. And we construct the parallel simulation environment for the RL training with the IsaacGym simulator[16]. We denote the

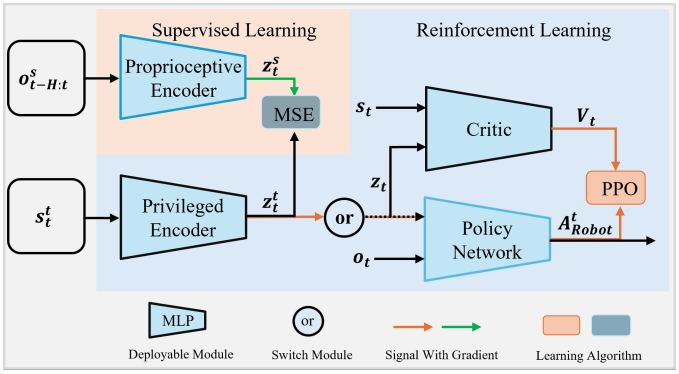

Fig. 2: Overview of the proposed position-conditioned local navigation policy training framework. We first train a teacher policy with full observability under PPO, and a student policy is trained in a supervised manner with only the proprioceptive observation to enable the sim-to-real transfer.

$\boldsymbol{o}_t \in \mathbb{R}^{46}$ in Fig. 2, as the observation of the agent consisting of the angular velocity components of the robot base in the world coordinate system, $\boldsymbol{\omega}_t \in \mathbb{R}^3$; the projection of the gravitational vector in the robot coordinate system, $\mathbf{g}_{proj} \in \mathbb{R}^3$; objective commands, $\mathbf{c} \in \mathbb{R}^3$; normalized remaining time, $t_t \in \mathbb{R}^1$; the deviation between the current joint position of each degree of freedom and the default position, $\Delta\mathbf{q} \in \mathbb{R}^{12}$; joint velocities of each degree of freedom, $\dot{\mathbf{q}} \in \mathbb{R}^{12}$; and the action of the previous time $\mathbf{a}_{t-1} \in \mathbb{R}^{12}$. The commands include the target position error $(\Delta x, \Delta y)$ and orientation error $(\Delta\theta)$, which are calculated by the position difference between the robot position and predicted ball landing point, $\boldsymbol{p}^t_t$, as shown in Fig. 1. This command can guide the robot to move to the target. In addition, the previous action is the proprioception and $\boldsymbol{o}_{t-H:t}$: as its history, where H represents the history length. To avoid signal delay caused by an overly large input historical dimension, we set $H = 5$. The privileged observation, $\boldsymbol{s}_t \in \mathbb{R}^{148}$, consists of $\boldsymbol{o}_t$, joint torques and accelerations, body linear velocity, feet contact forces, PD control gains, friction coefficients, motor strength, leg mass, and base mass. The policy network also receives the compressed latent representation of the full robot states, $\boldsymbol{z}^s_t \in \mathbb{R}^{32}$, which is the reconstructed latent vector of the normalized latent representation $\boldsymbol{z}_t$. $\boldsymbol{V}_t$ refers to the value estimation by the critic network. With the observation input, the policy will output the action, $\boldsymbol{a}_t$ in Fig. 1, to the robot to complete the time-constrained position-reaching task.

TABLE I: Reward Function Elements

Reward	Equation(r_i)	Weight(w_i)
Lin. velocity tracking	$\exp\left(-10\|\boldsymbol{c}_{xy} - \boldsymbol{v}_{xy}\|^2\right)$	1.0
Ang. velocity tracking	$\exp\left(-10(c_z - \omega_z)^2\right)$	0.5
Pos. tracking	$\exp(-10\boldsymbol{c}^2_{xy})$	1.0
Yaw tracking	$\exp(-10c^2_z)$	0.5
Linear velocity (z)	v^2_z	-2.0
Angular velocity (xy)	$\boldsymbol{\omega}^2_{xy}$	-0.05
Orientation	$(g^{\text{proj}}_x)^2 + (g^{\text{proj}}_y)^2$	-40.0
Joint accelerations	$(\frac{\dot{\mathbf{q}}_t - \dot{\mathbf{q}}_{t-1}}{\Delta t})^2$	-4×10^{-7}
Joint power	$\lvert\tau\rvert \times \lvert\dot{\mathbf{q}}\rvert$	-2×10^{-4}
Collision	$\sum_k \mathbf{1}\{\|\mathbf{f}_k\| > 0.1\}$	-1.0
Action rate	$(\mathbf{a}_{t-1} - \mathbf{a}_t)^2$	-0.01
Action smoothness	$(\mathbf{a}_t - 2\mathbf{a}_{t-1} - \mathbf{a}_{t-2})^2$	-2×10^{-3}
Torque	$\boldsymbol{\tau}^2_i$	-2×10^{-4}
Joint position limits	$\Delta\mathbf{q}$	-10.0
Joint velocity	$\dot{\mathbf{q}}^2$	-1.5×10^{-3}
Stand still position	$\lvert\mathbf{q}_i - \mathbf{q}_{i,\text{default}}\rvert$, if dist. < 0.1	-10.0
Base height	$(z_{\text{root}} - z_{\text{target}})^2$	-10.0
Feet air time	r_{ft}	3.0
Task	r_t	100
Feet acceleration	$\Delta {v_f}^2$	-1×10^{-4}
Exploration	r_e	1
Stalling penalty	r_{sp}	1
Stop yaw velocity	r_{sy}	-0.1
feet height	$\exp(-10(h_f - 0.1)^2)$	-2.0

Table I shows the reward terms we used. Most task reward functions are about spatial tracking and time alignment, where the detailed formulation follows[10], [20]. Safety and smoothness terms refer [3]. We normalize each term to comparable ranges and weight task terms higher than auxiliaries, but keep auxiliary weights sufficiently large to prevent catastrophic behaviors, such as falling, sinking, and excessive torques. Notably, there's no specific velocity direction term [9] to train the robot immediately turning to face the target position and then sprinting to the target position upon the robot getting the target position. However, as shown in the experiments section, an emergent behavior is observed, where the trained policy drives the robot rapidly, turning to the direction where the target position is at first and runs towards the position, as displayed in Fig. 5. This emergent behavior demonstrates high agility in time-aware and position-conditioned locomotion training fashion. In addition, to alleviate the sim-to-real gap and enhance the robustness of our policy, we implement the domain randomization for the training, where the detailed terms are shown in Table II.

To enhance the Learning efficiency of the agent, we adopted the *Curriculum Learning* strategy and carried out progressive expansion for the range of motion instructions, such as target position and orientation. In the initial stage, the distance between body position and target point, and the orientation of the target point are uniformly and randomly sampled from the range $[-0.5, 1.0]$ m and $[-1.5, 1.5]$ rad, respectively. If trained within this initial range, the robot will not exhibit abnormal movements, such as turning around and walking backwards. As the performance of the agent improves, the instruction range gradually expands

TABLE II: Domain Randomization Terms

Randomization Term	Range	Unit
Friction	$[0.5, 1.25]$	-
Base mass	$[-1.0, 1.0]\times$ nominal value	Kg
Robot push	$[-1.0, 1.0]$ $/15s$	m/s
CoM of base	$[-1.0, 1.0]^3$	cm
Motor strength factor	$[0.9, 1.1]$	-
Motor offset	$[-2.0, 2.0]$	cm
Joint K_p factor	$[0.8, 1.2]$	N·rad
Joint K_d factor	$[0.8, 1.2]$	N·rad/s

Fig. 3: Experiment Setup.

to $[-5.0, 7.5]$ m and $[-\pi, \pi]$ rad. Specifically, when the agent performs well within the current instruction range, the sampling interval of the target distance and angle is automatically increased to promote its exploration of a wider range of motion capabilities. Moreover, to encourage the discovery of emergent motion skills for more efficient motions, we employ the Random Network Distillation technique (RND) as the intrinsic reward[2] in our training to encourage exploration for the state-action space.

IV. Experiment

In this section, we compare our approach to the velocity tracking approach, namely the baseline, with simulation and physical sim-to-real experiments. The velocity tracking policy is trained with the reward terms in Table I with different scales, except tasks according to rewards, and the domain randomization in Table II. For baseline, we use the linear controller defined as equations (4) and (5), to guide the robot to walk towards the target point,

$$K_p(\boldsymbol{P}_d - \boldsymbol{P}_r) = \boldsymbol{v}_{xy}^{cmd} \tag{4}$$

$$K_p(\theta_d - \theta_r) = \omega^{cmd} \tag{5}$$

where K_p is a coefficient; $\boldsymbol{P}_d$, θ_d are desired position and angle; $\boldsymbol{P}_r$, θ_r are the current robot position and orientation. After tuning, we find that $K_p = 2.5$ has the best performance for the velocity tracking approach. Our position-conditioned method is achieved by providing task-related rewards in RL training without any further tuning.

A. System Setup

The hardware setup consists of a Unitree Go2 quadruped robot with a $22 \times 28 \times 9$ cm^3 size basket, which means that the acceptable error range for us is within 17.8 cm, and an onboard NVIDIA Jetson Orin Nano for locomotion policy inference. The perception system utilizes two stationary cameras. An overhead RGB fisheye camera, calibrated for undistortion, provides a global view for localizing the robot via an AprilTag on its back. A second Intel RealSense D415 camera is positioned near the ball's release point, with its field of view encompassing the flight path and the base coordinate system defined by the AprilTag. The whole experiment setup is illustrated in Fig. 3.

The software pipeline on an external PC uses YOLOv13n to detect the ball from the RealSense D415 feed. A basic Kalman filter with a state vector of $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ and positional measurements (x, y, z) as inputs is used for trajectory prediction, estimating the ball's landing point and time.

The robot's onboard computer receives only the coordinates of the predicted target landing point. The control policy subsequently uses this target point, alongside the robot's proprioceptive data, to generate the locomotion commands for interception.

B. Evaluation with Simulation

We conducted the Sim-to-sim in MuJoCo[24] to preliminarily test our method and the baseline. To simulate the ball-throwing task, we use the oblique projectile model and constrain the landing point within the 2 m range of the robot with a time limit between 1.15 and 1.35 seconds. The landing point of the ball in the simulation is shown in Fig. 6, which is all around the robot. As we find that the robot wastes much time on turning, when the desired turning angle is larger than 60 degrees, we give it a 120-degree command to make turning faster.

TABLE III: Sim-to-sim Catch Success Rate (Catch S.R.)

Catch **S.R.** (%)	< 0.5	0.5–1	1–2
baseline	67.39%	67.11%	16.28%
position conditioned	86.73%	22.45%	35.63%

We conducted 300 simulation experiments for each method. We define successful catching as the ball being thrown into the basket on the back of the robot, including those that are thrown into the basket and then bounce out. The result of the simulation experiments is shown in Table III. Also, we find that the extreme ball catching distance of our method is 1.78 m and 1.37 m for the baseline. These tables show that our method performs better ball-catching ability in short range and long range, and omnidirectional mobility capability. Our method also has higher extreme agility compared with the baseline. However, the baseline performs better in the medium range. Because our policy prefers the robot to turn at first, then rush towards the target position. But the velocity tracking approach leads the robot to move laterally to the target position when the target position is on the side of the robot. In the short range, the command of baseline is very small, which makes the robot move slowly or even stand still. In the medium range, the laterally moving

Fig. 4: Sim-to-real tennis ball catching experiment.

time is shorter than the sum of the turning and rushing forward time. In the larger run, turning and rushing is more effective for the robot to complete the tight time-constrained task.

C. Real World Validation

To assess the sim-to-real transferability of our approach, we deployed both our policy and the baseline policy on a Unitree Go2 robot by the open-source deployment framework[17]. In the simulation test, the velocity-based baseline achieved its best performance with k_p=2.5. However, we found this setting caused insufficient actuation for the robot to move when catching a ball at a short distance, as we mentioned in the simulation experiment. Therefore, during deployment, we adjusted k_p=5.0 when the distance between the target position and robot position was smaller than 0.5 m to improve the baseline's agility without changing its policy structure. Before the formal experiments, we conducted preliminary experiments. We found that for our policy, the robot performed better with landing points to its front-lateral side, whereas lateral landing points were easier to catch for the velocity-based method. We fixed the original

Ours Baseline

Fig. 5: Comparison of different motion patterns. Our method demonstrates the emergent behavior, namely the turning and approaching, while the baseline makes the robot just move laterally.

robot orientation forward, and threw a tennis ball randomly within a 0–2 m range near their *comfortable catching zones*, between 0.8 and 1.2 s. For fairness, each policy was tested under identical conditions with 100 trials, using the same models that achieved the best performance in simulation. Experiments were grouped by landing distance (<0.5 m, 0.5–1.0 m, and 1.0–2.0 m). The distribution of ball landing points across all trials is shown in Fig. 6, confirming that throws were randomized within the designated 0–2 m range.

The results are summarized in Table IV, organized separately by strict criterion and tracking criterion. When rim

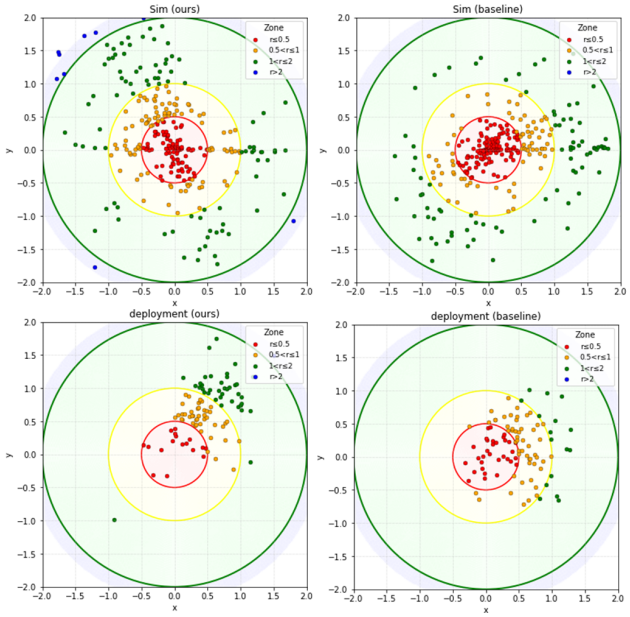

Fig. 6: Distribution of landing points. Comparison of landing positions for our method (left) and the baseline (right) in both simulation (top row) and real-world (bottom row). Note that the robot is facing along the y direction.

TABLE IV: Sim-to-real Success Rate (S.R.) Comparison

Track **S.R.** (%)	<0.5 m	0.5–1 m	1–2 m
baseline	25.00%	51.92%	43.75%
ours	**62.50%**	**64.44%**	**63.16%**
Catch **S.R.** (%)	<0.5 m	0.5–1 m	1–2 m
baseline	6.25%	9.62%	25.00%
ours	**37.50%**	**24.44%**	**36.84%**

contacts are considered successful, which is reflected as the track S.R., the position-conditioned policy S.R. consistently exceeds 60%, while the baseline performance is bad. Under the strict definition, where only direct hitting the backet bottom is counted as successful, that is the catch S.R., the performance gap becomes even more significant: the velocity base baseline achieves less than 10% success rates in <0.5 m and 0.5 m - 1 m cases, and gets 25% success rates in 1 m - 2 m case, essentially failing to catching in real-world settings. In addition, the extreme ball catching distance of our method is 1.70 m, and 1.23m for the baseline. The sim-to-real gap may be due to the soft terrain, which makes the robot encounter considerable resistance when turning around

quickly. Also, there exist the inevitable problems, such as communication delay, motor strength variation due to power voltage changing, and so on.

These findings confirm two points. First, the performance advantage of position conditioning observed in simulation persists in real deployment, despite the added perception and actuation uncertainty. Second, the velocity-conditioned baseline suffers a dramatic degradation during sim-to-real transfer, suggesting that explicit position information is crucial for robust interception in practice.

To conclude, through the extensive simulation test and real-world experiment, we validate the effectiveness of the proposed method to achieve highly agile perceptive locomotion and the whole integrated system. Further, we demonstrate the superiority of our method over the baseline method with the quantitative comparative experiments. In addition, we deploy the policy trained by the proposed method and enable the successful sim-to-real transfer, verifying the practicability from the ultimate end.

V. Conclusion and Future Work

In this work, we proposed a position-based framework that enables the quadruped robot to move agilely to catch the flying ball with considerable success rates compared with the velocity-based approach. Also, we integrated a perception module to detect and track the position of the tennis ball relative to the robot. In addition, a Kalman Filter based object trajectory prediction module is implemented to provide the object landing position and timing for quadruped locomotion. We fed the position and time observations as input to the robot. We developed an RL position-conditioned RL training framework to train a local navigation policy for direct position tracking and ball catching with zero-shot sim-to-real transfer, by decoupling the task into perception and reinforcement learning. The experiment results show that our training framework achieves high precision and success rates for catching balls and outperforms the velocity-based baseline. We deployed policies on a physical robot for both frameworks and conducted real-world experiments to validate the effectiveness and superiority. The results indicate that our approach significantly outperformed the velocity-based baseline and enabled successful sim-to-real transfer. A obvious limitation is that the robot can only run within the field of view of the global camera. Future work could be enabling onboard perception to remove this restriction, thereby improving the policy's adaptability and scalability to larger environments.

References

[1] Anonymous. Run and catch: Dynamic object-catching of quadrupedal robots. *arXiv preprint arXiv:2405.xxxxx*, 2024. (preprint, replace with actual citation when available).

[2] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[3] Leixin Chang, Yuxuan Nai, Hua Chen, and Liangjing Yang. Beyond robustness: Learning unknown dynamic load adaptation for quadruped locomotion on rough terrain. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2025.

[4] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. *arXiv preprint arXiv:2309.14341*, 2023.

[5] Ke Dong, Karime Pereida, Florian Shkurti, and Angela P. Schoellig. Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning, 2020.

[6] Balázs Forrai, Takahiro Miki, Daniel Gehrig, Marco Hutter, and Davide Scaramuzza. Event-based agile object catching with a quadrupedal robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[7] Huiqiao Fu, Haoyu Dong, Wentao Xu, Zhehao Zhou, Guizhou Deng, Kaiqiang Tang, Daoyi Dong, and Chunlin Chen. Learning diverse natural behaviors for enhancing the agility of quadrupedal robots. *arXiv preprint arXiv:2503.12345*, 2025.

[8] Neil Guan, Shangqun Yu, Shifan Zhu, and Donghyun Kim. Impedance matching: Enabling an rl-based running jump in a quadruped robot. *arXiv preprint arXiv:2404.15096*, 2024.

[9] Tianyu He, Chen Zhang, Wang Xiao, Guanqi He, Chang Liu, and Guanya Shi. Agile but safe: Learning collision-free high-speed legged locomotion. *arXiv preprint arXiv:2401.17583*, 2024.

[10] David Hoeller, Nikita Rudin, David Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 8(75):eadf9861, 2023.

[11] Lunjun Ji, Jemin Hwangbo, and Marco Hutter. Extreme parkour with legged robots. *arXiv preprint arXiv:2311.01967*, 2023.

[12] Mengqi Lei, Siqi Li, Yihong Wu, Han Hu, You Zhou, Xinhu Zheng, Guiguang Ding, Shaoyi Du, Zongze Wu, and Yue Gao. Yolov13: Real-time object detection with hypergraph-enhanced adaptive visual perception, 2025.

[13] Sicen Li, Yiming Pang, Panju Bai, Zhaojin Liu, Jiawei Li, Shihao Hu, Liquan Wang, and Gang Wang. Learning agility and adaptive legged locomotion via curricular hindsight reinforcement learning. *arXiv preprint arXiv:2503.08961*, 2025.

[14] Zhongyu Li, Xue Bin Peng, and Koushil Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *Nature Machine Intelligence*, 5:332–342, 2023.

[15] Yandong Lin, Chong Zhang, Weiyu Chen, Vladlen Koltun, and Marco Hutter. Learning whole-body loco-manipulation for omni-directional task space pose tracking with a wheeled-quadrupedal-manipulator. *arXiv preprint arXiv:2401.01234*, 2024.

[16] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[17] Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In Karen Liu, Dana Kulic, and Jeffrey Ichnowski, editors, *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 2023.

[18] Arindam Roychoudhury, Shahram Khorshidi, Subham Agrawal, and Maren Bennewitz. Perception for humanoid robots. *Current Robotics Reports*, 2023.

[19] Nikita Rudin, Weiyu Chen, David Hoeller, and Marco Hutter. Motion priors reimagined: Adapting flat-terrain skills for complex quadruped mobility. *arXiv preprint arXiv:2310.04521*, 2023.

[20] Nikita Rudin, Weiyu Chen, Marco Hutter, and Vladlen Koltun. Advanced skills by learning locomotion and local navigation end-to-end. *arXiv preprint arXiv:2209.12827*, 2022.

[21] Abdulrahman Schakkal, Gabriel Bellegarda, and Auke Ijspeert. Dynamic object catching with quadruped robot front legs. *arXiv preprint arXiv:2410.08065*, 2024.

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[23] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[24] 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.

[25] Rejin Varghese and Sambath M. Yolov8: A novel object detection algorithm with enhanced performance and robustness. In *2024 Inter-*

national Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), pages 1–6, 2024.

[26] Hongxi Wang, Haoxiang Luo, Wei Zhang, and Hua Chen. Cts: Concurrent teacher-student reinforcement learning for legged locomotion. *IEEE Robotics and Automation Letters*, 2024.

[27] Hongxiang Yu, Dashun Guo, Huan Yin, Anzhe Chen, Kechun Xu, Zexi Chen, Minhang Wang, Qimeng Tan, Yue Wang, and Rong Xiong. Neural motion prediction for in-flight uneven object catching. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4662–4669, 2021.

[28] Chong Zhang, Weiyu Chen, Jingkai Xu, Nikita Rudin, Marco Hutter, and Vladlen Koltun. Learning agile locomotion on risky terrains. *arXiv preprint arXiv:2311.10484*, 2023.

[29] Yichi Zhang et al. Catch it! learning to catch in flight with mobile dexterous hands. *arXiv preprint arXiv:2409.10319*, 2024.