2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Dynamic Object Catching with Quadruped Robot Front Legs

André Schakkal, Guillaume Bellegarda, Auke Ijspeert

***Abstract*— This paper presents a framework for dynamic object catching using a quadruped robot's front legs while it stands on its rear legs. The system integrates computer vision, trajectory prediction, and leg control to enable the quadruped to visually detect, track, and successfully catch a thrown object using an onboard camera. Leveraging a fine-tuned YOLOv8 model for object detection and a regression-based trajectory prediction module, the quadruped adapts its front leg positions iteratively to anticipate and intercept the object. The catching maneuver involves identifying the optimal catching position, controlling the front legs with Cartesian PD control, and closing the legs together at the right moment. We propose and validate three different methods for selecting the optimal catching position: 1) intersecting the predicted trajectory with a vertical plane, 2) selecting the point on the predicted trajectory with the minimal distance to the center of the robot's legs in their nominal position, and 3) selecting the point on the predicted trajectory with the highest likelihood on a Gaussian Mixture Model (GMM) modelling the robot's reachable space. Experimental results demonstrate robust catching capabilities across various scenarios, with the GMM method achieving the best performance, leading to an 80% catching success rate.**

I. Introduction

Quadruped robots are showing impressive abilities to traverse challenging terrains [1]–[3], run at high speeds [4], [5], and locomote over dynamic parkour obstacles [6]–[10]. Recently, to allow quadrupeds to accomplish everyday tasks, there are an increasing number of examples of mounting a manipulator arm on top of a quadruped to perform both locomotion and manipulation (loco-manipulation) [11]–[14]. This results in five (or more) "arms" to perform manipulation and locomotion. On the other hand, bipeds use two feet for locomotion, and typically have two arms for world interactions such as opening doors and moving boxes, also recently emulated with a wheel-legged quadruped balancing on its rear wheeled-legs [15]. Some recent works investigate the possibility of using existing degrees of freedom on typical quadruped robots to perform tasks such as pressing buttons and opening doors [16], or moving boxes by pushing them from different lateral directions using the body [17]–[19]. Another approach for interacting with moving or thrown objects is mounting a net on the robot base, either positioned horizontally and leveraging event cameras to detect and catch an object thrown with high speed [20], or mounted vertically to predict the landing location of an object and then moving the robot to make the catch [21]. Other works show manipulation with multiple legs is possible by resting the quadruped base on the ground, and using the feet in the air to re-orient a ball [22], or resting parts of the robot base on the ground, and using the limbs to push or pick up boxes [23].

This research is supported by the Swiss National Science Foundation (SNSF) as part of project No.197237. The authors are with the BioRobotics Laboratory, Ecole Polytechnique Federale de Lausanne (EPFL). {firstname.lastname}@epfl.ch

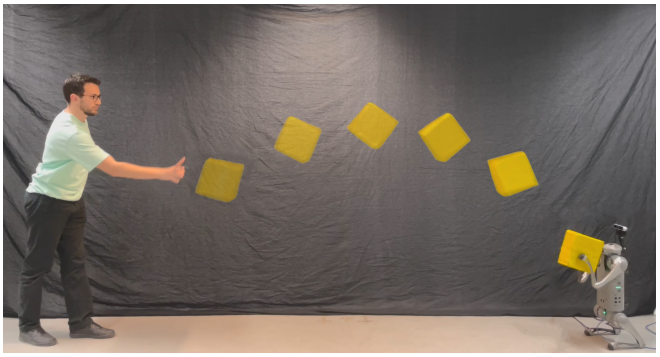


Fig. 1: The Unitree Go1 quadruped robot elevated on its rear legs, detecting a thrown object, predicting a suitable catching position, and successfully catching it with its front legs.

In contrast to previous works, in this paper we consider the task of using a quadruped's existing degrees of freedom to catch a thrown object, as shown in Figure 1. This necessitates reconfiguring the robot from its nominal standing position on four legs, to standing on its rear legs. Once elevated, this new configuration allows for manipulation capabilities by using its two front limbs together to interact with the world, for example to catch a thrown object. A camera mounted onboard the robot is used to detect the object, and multiple consecutive images are used to predict a suitable catching position. Upon predicting and selecting this catching position and the time for the object to reach this position, the robot can move its limbs to catch the object.

II. Related work

Dynamic object catching stands as a rich area of research interest to demonstrate fast prediction and control capabilities, rooted in early pioneering works [24], [25]. Over the years, researchers have explored diverse methodologies and systems, including the use of humanoids [25]–[27], robotic arms employing various end-effectors such as baskets [28]–[31], grippers [24], [32], and robotic hands [33]–[36]. Quadrupeds have also been employed for catching objects [20], [21]. It is noteworthy that the existing approaches involving quadrupeds significantly differ from the methodology proposed in this paper. Specifically, in [20] and [21], quadrupeds operate on all fours, catching objects with nets/baskets mounted on their backs. To the best of our knowledge, no prior work has demonstrated an object-catching application with a standing quadruped utilizing its front legs.

Dynamic object catching involves several essential subtasks: object detection and tracking, trajectory prediction, and the catching control maneuver. In the following, we review previous existing works in each of these areas.

A. Object Detection and Tracking

Object detection serves as a crucial component in dynamic object catching, enabling the system to determine the location of the thrown object for trajectory prediction and subsequent catching maneuvers. Object tracking methodologies can be broadly classified into two main categories: those requiring depth information in conjunction with the detection module for accurate 3D localization, and those operating without additional depth information.

Among the methods not requiring extra depth information, motion capture systems have been employed, as seen in works utilizing the Vicon motion capture system [28], [32], and the Optitrack vision system [33]. While these systems offer high accuracy, they necessitate markers on the thrown object, a complex setup with multiple cameras, can be relatively expensive, and lack mobility and generalizability. Such methods could however serve as benchmarks to assess the performance of the other modules of the system.

Conversely, depth-dependent methods, requiring additional depth information, are generally more cost-effective and compact. Techniques such as simple color detection, coupled with depth data obtained through stereo vision [25], [27], [36], [37], fall into this category. A drawback of this approach lies in its object dependency, requiring the system to be informed of the object's color. Moreover, the absence of the object's color in the background is essential to avoid interference with the detection module. Alternative techniques involve various image processing techniques, such as template matching [38]. Additionally, deep learning algorithms such as YOLO (You Only Look Once) [39] have been employed for object detection [40]. These methods were complemented by depth estimation via stereo vision for correct 3D localization. Finally, an interesting variation involves object detection using an event camera, where the object's depth is subsequently estimated based on perceived width and the camera's focal length, eliminating the need for stereo vision [20].

B. Trajectory Prediction

Predicting the trajectory of a thrown object is crucial for enabling a robotic system to catch it successfully. Various methods have been employed to achieve accurate trajectory prediction.

Traditionally, trajectory prediction has been accomplished through regression-based methods. Recursive Ordinary Least Squares (OLS) regression is employed in some studies [20], while others enhance this approach by incorporating a regularization term to account for gravity [24], [25]. Gaussian Process Regression has also been utilized to model and predict object trajectories [34]. Additionally, simple approaches involving Kalman filters, leveraging known ballistic models, are used for trajectory predictions in various scenarios [21], [27], [28], [31], [38].

In scenarios demanding prediction of more complex, nonlinear trajectories, advanced methods come into play. Deep learning techniques were used, particularly employing bidirectional Long Short-Term Memory (LSTM) networks [41], and conditional generative models [37]. Another avenue explores Dynamical System estimation to model the motion of the object [26], [33], [42].

C. Catching Maneuver

Executing a successful catching maneuver involves two primary capabilities. Firstly, determining the optimal catching position is crucial. This is often accomplished by using the predicted trajectory of the object. A straightforward approach involves identifying the intersection point between the predicted trajectory and a predefined plane where the robot intends to intercept the object [20], [21], [27], [38]. Another effective strategy is to model the reachable space of the robot and select a point on the predicted trajectory within this space. Various models for the reachable space, including geometric shapes, can be used. The point on the predicted trajectory can then be selected by choosing the one closest to the robot's initial position [29], the robot's base [24], or its end effector [32]. Alternatively, Gaussian Mixture Models (GMMs) offer a probabilistic approach, allowing the selection of the catching point based on the highest likelihood within the constraints of the optimization problem [26]. The catching position can also be determined by solving a nonlinear optimization problem which satisfies both robot constraints and object trajectory, and minimizing an energy-based [34] or acceleration-based [35] objective function.

The second critical capability is the motion required to reach and catch the object. Catching maneuvers differ significantly based on the robotic platform employed. Although there have been instances of quadrupeds catching objects, they have involved nets/baskets mounted on the robot's back [20], [21]. Since our focus is on catching objects specifically with the front legs of a quadruped, the strategies are more aligned with those used in humanoid and robotic arm catching scenarios. Catching motion options range from simple PD control to more advanced methods such as Model Predictive Control (MPC) [30], [31], learned controllers [28], [40], and the generation of a second-order Dynamical System to govern the robot's motion [26], [33].

D. Contribution

We present a novel framework that enables quadruped robots to autonomously catch thrown objects using their front legs. The key components of our framework include:

- **Quadruped Elevation**: We optimize a standing up behavior for the quadruped to stand on its rear legs, leaving its front legs available for catching.
- **Object Detection**: We fine-tune a YOLOv8 model with a specifically curated dataset to be able to detect different objects, using an onboard camera.

- **Trajectory Prediction**: Using the detected coordinates of the thrown object, we predict its trajectory using gravity-informed ordinary least squares
- **Catching Maneuver**: To perform catching, we select a suitable catching position by comparing three different strategies:
  - Intersecting the predicted trajectory with a predefined vertical plane
  - Selecting the point on the predicted trajectory that has a minimal distance to the center of the robot's legs in their nominal position
  - Selecting the point on the predicted trajectory that has the highest likelihood on a Gaussian Mixture Model (GMM) modeling the robot's reachable space, which was fit over several catching positions collected by a human directing the robot in passive mode

  Upon selecting the catching position, the robot performs the catch with a Cartesian PD controller.

Using this framework, we observe robust catching capabilities in a variety of scenarios, including an 80% success rate on a test scenario involving 50 throws from a distance of 2 meters in front of the robot.

The rest of this paper is organized as follows. In Section III we present each of the components of our quadruped catching framework and design choices. In Section IV we discuss results and analysis from different catching scenarios, with each of the catching position identification methods. Section V concludes the paper and suggests future directions for further work.

III. Method

In this section we describe our framework for catching objects with a quadruped robot's front legs. A high-level control diagram is illustrated in Figure 2, showing the four important blocks of our pipeline: (A) elevating the quadruped onto its rear legs, and subsequently throwing the object, (B) detecting and extracting the object location from the frame, (C) using the object locations from successive frames to predict the object's trajectory, and (D) controlling the front legs to catch the object at the predicted catching position.

A. Quadruped Elevation

In this section we briefly describe the trajectory optimization framework to generate the quadruped elevation motion, shown in Figure 2-(A). It has been adapted from the trajectory optimization framework used in our prior work for generating quadruped jumping motions [43], [44]. The robot starts from a standing position with all four feet on the ground, and should end standing on its rear feet, statically balancing with the rear knees in contact with the ground as well. This is accomplished with a time-based two part contact dynamics component: all legs in contact, and then only the rear legs in contact. The discrete time optimization can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{x}_k, \boldsymbol{u}_k;\ k=1\ldots N} \quad & J\big(\boldsymbol{x}_N\big) + h\sum_{k=1}^{N} w\big(\boldsymbol{x}_k, \boldsymbol{u}_k\big) \\ \text{s.t.} \quad & d\big(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}\big) = 0,\ k = 1\ldots N-1 \\ & \phi\big(\boldsymbol{x}_k, \boldsymbol{u}_k\big) = 0,\ k = 1\ldots N \\ & \psi\big(\boldsymbol{x}_k, \boldsymbol{u}_k\big) \leq 0,\ k = 1\ldots N \end{aligned} \tag{1}$$

where $\boldsymbol{x}_k = [p_{x,k};\ p_{z,k};\ \theta_k;\ \boldsymbol{q}_k]$ is the full state of the system at sample k along the trajectory, $\boldsymbol{u}_k$ is the corresponding control input, J and w are final and additive costs to end upright at a particular height while minimizing energy, h is the time between sample points k and $k+1$, and N is the total number of samples along the trajectory. The constraints are specified as follows:

- The function $d(\cdot)$ captures the full-body dynamic constraints, which is discretized from

$$\begin{bmatrix} \boldsymbol{M} & -\boldsymbol{J}_c^T \\ -\boldsymbol{J}_c^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{x}} \\ \boldsymbol{f}_c \end{bmatrix} = \begin{bmatrix} -\boldsymbol{C}\dot{\boldsymbol{x}} - \boldsymbol{g} + \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{S}_f\boldsymbol{\tau}_f \\ \dot{\boldsymbol{J}}_c(\boldsymbol{x})\dot{\boldsymbol{x}} \end{bmatrix},$$

  where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}$ represents the Coriolis and centrifugal terms, $\boldsymbol{g}$ denotes the gravity vector, $\boldsymbol{J}_c$ is the spatial Jacobian expressed at the foot contact, $\boldsymbol{S}$ and $\boldsymbol{S}_f$ are distribution matrices of actuator torques $\boldsymbol{\tau}$ and joint friction torques $\boldsymbol{\tau}_f$, $\boldsymbol{f}_c$ is the spatial force at the foot contact. The dimensions of $\boldsymbol{J}_c$ and $\boldsymbol{f}_c$ depend on the contact phases.
- The function $\phi(\cdot)$ represents equality constraints on initial/final joint and body configurations.
- The function $\psi(\cdot)$ captures inequality constraints including joint angle/velocity/torque limits, friction cone limits, and minimum ground reaction forces.

The optimization produces desired joint angles $(\boldsymbol{q}_d)$, joint velocities $(\dot{\boldsymbol{q}}_d)$ and feed-forward joint torques $(\boldsymbol{\tau}_d)$ at a sampling time of 10 *ms*, which are then linearly interpolated to 1 *ms*. These can be tracked to successfully elevate the quadruped with the following joint PD controller running at 1 *kHz* as:

$$\boldsymbol{\tau}_{\text{ff}} = \boldsymbol{K}_{p,joint}(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{K}_{d,joint}(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) + \boldsymbol{\tau}_d \tag{2}$$

where $\boldsymbol{K}_{p,joint}$ and $\boldsymbol{K}_{d,joint}$ are proportional and derivative gains in the joint coordinates.

B. Object Detection

After successfully elevating the quadruped on its rear legs, the subsequent step involves detecting and tracking the thrown object (Figure 2-(B)). For this purpose, an Intel RealSense™ Depth Camera D455 is mounted on the robot, capturing RGB and depth images at 30 frames per second.

1) YOLOv8 Object Detection: To facilitate object detection and tracking, we use YOLOv8 [39]. YOLOv8 offers high generalizability in object detection by handling diverse objects without constraints on color or shape, and performs onboard detection without requiring any modifications or attachments to the objects themselves. We fine-tuned the small pretrained model `YOLOv8s` from the `ultralytics`

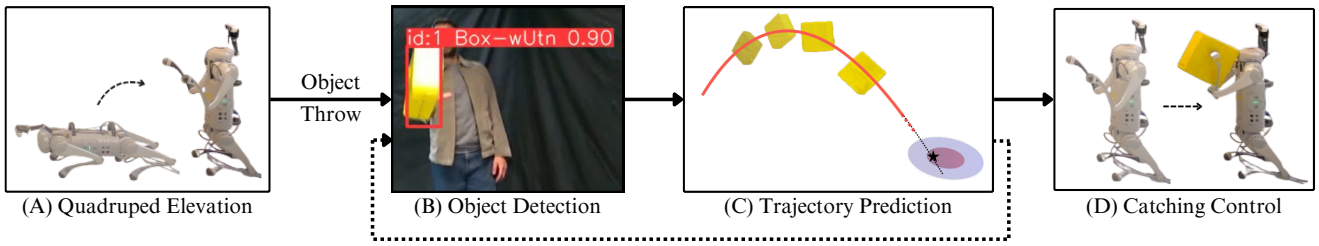


Fig. 2: Control diagram for catching objects with a quadruped robot. (A) the quadruped elevates onto its rear legs with an optimized trajectory, leaving its front legs available to catch an object. The user subsequently throws an object. (B) and (C): an onboard camera is used to detect the object and extract its location, the system iteratively predicts a suitable catching location through successive frames while the object is in the air. (D) as the object reaches the predicted catching location, the quadruped controls its front legs to catch the object.

library[1] using a dataset of 389 images. These images were extracted from recorded videos capturing users throwing objects towards the quadruped, with intentional variations in the background to help generalization. The dataset was annotated using the `Roboflow` software [45]. The recorded videos were processed to generate individual images. The images were then annotated, where bounding boxes were defined around the thrown object, and the object was classified as a "Box". Various augmentations were applied to the annotated images, including a horizontal flip, a 15° rotation, and Gaussian blur, all targeted at the bounding boxes. This process generated new data to create a more robust and versatile model. Using this augmented dataset, the model underwent fine-tuning with an AdamW optimizer [46]. The fine-tuning process involved a learning rate of 0.002 and a momentum of 0.9, lasting for 100 epochs. With only one class to detect, this configuration achieved a high mean average precision of 0.992 when the IoU threshold is set to 0.5 (mAP@50) on the validation set. This allowed us to achieve good object detection with tight bounding boxes around objects of different colors and dimensions.

2) Coordinate Transformation: During inference, the fine-tuned model takes as input an RGB image and outputs coordinates of bounding boxes over the detected objects, their classes, and their corresponding confidence values. The coordinates of the bounding box are two dimensional coordinates in pixels in the image frame. The pixel coordinates of the center of the bounding box x_p and y_p are then transformed into three dimensional coordinates in the robot frame (x, y, z) (the coordinate system is shown in Figure 4) using the depth of the object returned from the RealSense camera x_{depth}. Leveraging the intrinsic parameters of the RealSense camera—specifically f_x and f_y representing the focal length along the image plane's axes, and pp_x and pp_y indicating the pixel coordinates of the principal point in the image plane—we calculate (x, y, z) with the following formulas:

$$x = x_{depth}\cos\theta - x_{depth}\frac{(y_p - pp_y)}{f_y}\sin\theta \tag{3}$$

$$y = x_{depth}\frac{(x_p - pp_x)}{f_x} \tag{4}$$

$$z = -\left(x_{depth}\sin\theta + x_{depth}\frac{(y_p - pp_y)}{f_y}\cos\theta\right) \tag{5}$$

[1]`https://github.com/ultralytics/ultralytics`

with θ being the angle the camera makes with the horizontal.

C. Trajectory Prediction

With the sequence of object coordinates in the robot frame available, the next step is trajectory prediction (Figure 2-(C)). Accurately predicting the object's path is crucial for determining a suitable catching location.

1) Approach: We choose to employ gravity-informed ordinary least squares for trajectory prediction due to its effectiveness in modeling ballistic trajectories and computational efficiency, making it well-suited for real-time prediction. Assuming that, upon leaving the user's hand, the object experiences only free-fall motion with an initial velocity, and neglecting air resistance, we know that the object's trajectory will consistently follow a three-dimensional parabola with:

$$x(t) = v_{0x}t + x_0 \tag{6}$$

$$y(t) = v_{0y}t + y_0 \tag{7}$$

$$z(t) = -\frac{1}{2}gt^2 + v_{0z}t + z_0 \tag{8}$$

where (x_0, y_0, z_0) is the initial position of the object at the start of the trajectory, (v_{0x}, v_{0y}, v_{0z}) is the initial velocity, and g is gravity.

2) Parameter Estimation: Since measurements inevitably contain some degree of noise, exact parameters for the equations above cannot be determined. Therefore, we solve the following system of equations:

$$x_i(t_i) = a_x t_i + b_x, \quad i = 1, ..., n, \tag{9}$$

$$y_i(t_i) = a_y t_i + b_y, \quad i = 1, ..., n, \tag{10}$$

$$z_i(t_i) = a_z t_i^2 + b_z t_i + c_z, \quad i = 1, ..., n, \tag{11}$$

where a, b, c represent the coefficients to be estimated, and n is the number of measurements. We have that $n \geq 3$ since for Equation 11, we need at least three measurements to have a solution for the parameters of the equation.

For x and y, we find their corresponding parameters by solving the following normal system of equations:

$$\begin{bmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n t_i \\ \sum_{i=1}^n t_i & \sum_{i=1}^n t_i^2 \end{bmatrix} \begin{bmatrix} b_x \\ a_x \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i t_i \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} \sum_{i=1}^n 1 & \sum_{i=1}^n t_i \\ \sum_{i=1}^n t_i & \sum_{i=1}^n t_i^2 \end{bmatrix} \begin{bmatrix} b_y \\ a_y \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n y_i t_i \end{bmatrix} \tag{13}$$

On other hand, since z follows a parabolic trajectory, we determine its corresponding parameters by solving the

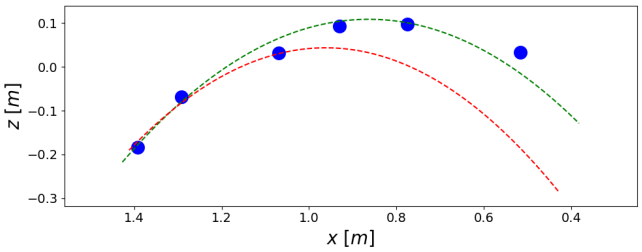


Fig. 3: Example trajectory prediction. The discrete blue points represent observed object positions, the red curve shows the predicted trajectory using the first 3 observed positions, and the green curve shows the predicted trajectory using all 6 observed positions.

following system of normal equations, introducing $\lambda = 1$ to account for gravity:

$$\begin{bmatrix} \sum_{i=1}^{n} 1 & \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} t_i^2 \\ \sum_{i=1}^{n} t_i & \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i^3 \\ \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i^3 & \sum_{i=1}^{n} t_i^4 + \lambda \end{bmatrix} \begin{bmatrix} c_z \\ b_z \\ a_z \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} z_i \\ \sum_{i=1}^{n} z_i t_i \\ \sum_{i=1}^{n} z_i t_i^2 - \lambda \frac{1}{2} g \end{bmatrix} \tag{14}$$

3) Iterative Refinement: Solving this system of equations provides a time-dependent model $\boldsymbol{x}_{pred}(t) = (x_{pred}(t), y_{pred}(t), z_{pred}(t))$. Consequently, we can predict the position of the object for a specific timestep t. Similarly, we can determine the timestep t at which the object will reach a specific (x, y, z).

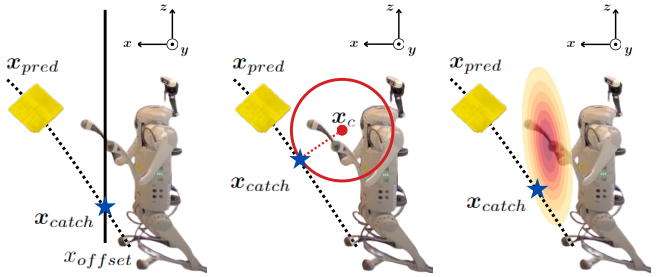
For each new frame captured by the camera providing new (x, y, z) measurements, we re-solve the system of equations to refine the estimation of the regression parameters. The iterative nature of this process ensures that with each new measurement, we enhance the accuracy of the regression parameters by incorporating more data points into the regression. Such improvement in performance is shown in Figure 3, where the addition of more measurements to the prediction module results in predictions closer to the ground truth.

4) Initiating Trajectory Prediction: An important point for correctly starting the trajectory prediction is that the object has been thrown and left the user's hand. This distinction prevents the coordinates of the object, determined when it is in the user's hands, from being considered in the trajectory prediction module, which would lead to incorrect predictions. To address this, the system considers the detected object coordinates for prediction only when the absolute difference between the coordinates of the current and previous detection exceeds a predefined threshold.

D. Catching Maneuver

Having achieved the ability to detect, track, and predict the trajectory of the thrown object, the final submodule essential for a successful catch is the catching maneuver itself (Figure 2-(D)).

1) Catching Position Identification: There are a potentially infinite number of possible positions where the quadruped can catch the thrown object. In this paper, we consider three different methods for determining a suitable



(a) Plane Intersection (b) Min Distance to Center (c) GMM

Fig. 4: Illustrations of the different catching position identification methods.

catching position $\boldsymbol{x}_{catch} = (x_{catch}, y_{catch}, z_{catch})$, each illustrated in Figure 4.

- **Plane Intersection**: The catching position $\boldsymbol{x}_{catch}$ and the time to reach this position t_{catch} are determined by intersecting the predicted trajectory with a vertical plane $x_{offset} = 25cm$ in front of the quadruped, as illustrated in Figure 4a. Specifically, we find the timestep t_{catch} that solves:

$$x_{pred}(t_{catch}) = x_{offset} \Leftrightarrow t_{catch} = \frac{x_{offset} - b_x}{a_x} \tag{15}$$

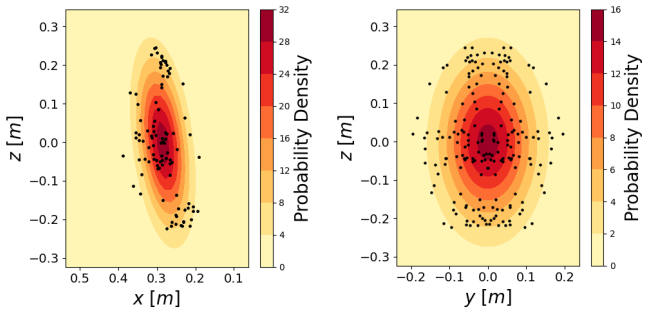
Subsequently, we determine the catching position $\boldsymbol{x}_{catch}$ by substituting this timestep t_{catch} into the trajectory equations:

$$\boldsymbol{x}_{catch} = \boldsymbol{x}_{pred}(t_{catch}) \tag{16}$$

- **Minimum Distance to Center**: The catching position $\boldsymbol{x}_{catch}$ is determined by choosing the point on the predicted trajectory that has the smallest Euclidean distance to the center between the front feet of the quadruped in its nominal configuration $\boldsymbol{x}_c$. The catching position is thus the closest point to the original center between the front feet, which is on the predicted trajectory, as shown in Figure 4b. Finding the catching position $\boldsymbol{x}_{catch}$ and the time to reach this position t_{catch} is equivalent to solving the following optimization problem:

$$\begin{aligned} \boldsymbol{x}_{catch}, t_{catch} = \min_{\boldsymbol{x}, t} \quad & \|\boldsymbol{x}_c - \boldsymbol{x}\|_2 \\ \text{s.t.} \quad & \boldsymbol{x}_{pred}(t) = \boldsymbol{x} \end{aligned} \tag{17}$$

- **Gaussian Mixture Model (GMM)**: The third approach consists of using Gaussian Mixture Models (GMMs). Inspired by [26], we performed 100 catching demonstrations by throwing the object towards the quadruped, and a human manually directed the quadruped to catch the object with its front feet. The final positions where the object was caught were recorded. We then fit a GMM on these points to have a density distribution of these points. The parameters of the GMM $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1:K}$, where K is the number of Gaussians, were determined through expectation maximization. Using the Bayesian information criterion (BIC), the optimal number of Gaussians was determined to be $K = 1$, therefore $\pi_1 = \pi = 1$. The probability density of a catching position $\boldsymbol{x}$ is shown in Figure 5 and is



(a) Side View GMM (b) Front View GMM

Fig. 5: GMM fitted over 100 catching demonstrations. The GMM models the quadruped's reachable space and provides likelihood information for catching positions.

given by

$$\mathcal{P}\left(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right)=\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right). \tag{18}$$

The fitted GMM not only implicitly covers the reachable workspace of the quadruped's front legs and gives suitable catching locations, but it also gives us likelihood information. Regions where the likelihood is high are regions where the human driving the quadruped during the demonstrations caught the object many times. Since the quadruped could possibly catch the object on many positions along the predicted trajectory, this likelihood information leads to choosing a more human-intuitive catching position. Having this probabilistic model, the catching position is chosen to be the point on the predicted trajectory that has the highest likelihood on the probability density of the GMM, as shown in Figure 4c. Therefore, finding the catching position $\boldsymbol{x}_{catch}$ and the time to reach this position t_{catch} is equivalent to solving the following optimization problem:

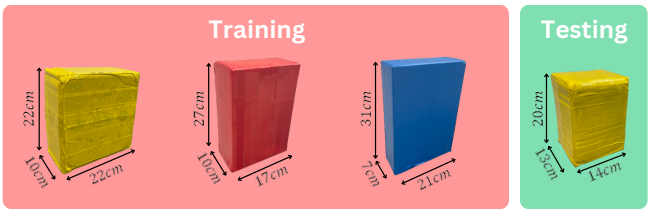
$$\begin{aligned}\boldsymbol{x}_{catch},t_{catch}=\max_{\boldsymbol{x},t}\quad&\mathcal{P}\left(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right)\\ \text{s.t.}\quad&\boldsymbol{x}_{pred}\left(t\right)=\boldsymbol{x}\end{aligned} \tag{19}$$

For each method explained above, $\boldsymbol{x}_{catch}$ and t_{catch} are updated for every new camera frame providing new measurements in order to improve the catching position prediction. Furthermore, using the time to reach the catching position t_{catch} and the current timestep t, we can calculate the time remaining for the object to reach the catching position:

$$t_{remain}=t_{catch}-t \tag{20}$$

2) Catching Control: As the prediction is iteratively refined, we use Cartesian PD control to close the legs around the predicted catching position $\boldsymbol{x}_{catch}$. These coordinates are mapped to leg frame coordinates $\boldsymbol{p}_d = (x_{catch}, y_{catch} \pm y_{opened}, z_{catch})$, where $y_{opened} = 0.15m$ ensures that the front legs are opened around the predicted catching position. These values are first clipped to ensure they remain in the robot workspace in case the prediction makes the catch impossible.

The subsequent step is to determine when to close the feet, and this is done using t_{remain}. Since the feet of the quadruped are always positioned around the predicted catching position, and the time remaining for the object



(a) Object A (b) Object B (c) Object C (d) Object D

Fig. 6: Objects used in experiments for both training and testing.

to reach this position is known, the closing mechanism is activated when t_{remain} drops below a time threshold t_{thresh}. Through experimentation, the optimal time threshold was determined to be $t_{thresh} = 0.15s$.

As soon as t_{remain} drops below t_{thresh}, the feet coordinates in the robot frame are offset to close around the object with $\boldsymbol{p}_d = (x_{catch}, y_{catch} \pm y_{closed}, z_{catch})$, where $y_{closed} = 0.01m$ ensures that the front legs close around the object. The resulting torque control for each front leg is written as follows:

$$\boldsymbol{\tau}=\boldsymbol{J}(\boldsymbol{q})^{\top}\Big[\boldsymbol{K}_p\left(\boldsymbol{p}_d-\boldsymbol{p}\right)-\boldsymbol{K}_d\left(\boldsymbol{v}\right)\Big]-\boldsymbol{K}_{d,joint}(\dot{\boldsymbol{q}}) \tag{21}$$

where $\boldsymbol{J}(\boldsymbol{q})$ is the foot Jacobian at joint configuration $\boldsymbol{q}$, $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are diagonal matrices of proportional and derivative gains in Cartesian coordinates to track the desired foot positions $(\boldsymbol{p}_d)$ with zero desired foot velocity $(\boldsymbol{v})$ in the leg frame. We also add a small joint damping term for stability. We use $\boldsymbol{K}_p = 400\boldsymbol{I}_3$, $\boldsymbol{K}_d = 8\boldsymbol{I}_3$, $\boldsymbol{K}_{d,joint} = \boldsymbol{I}_3$.

E. Experimental Setup

We use the Unitree Go1 quadruped [47], on which we mount a depth camera for object detection and position tracking (Intel® RealSense™ Depth Camera D455), accelerated on a graphics processing unit (GPU) for fast calculations (Nvidia RTX 1050i GPU). Throughout the experiments, we use several objects (shown in Figure 6) with different colors and dimensions, ranging in mass from 150g to 200g. Objects A, B and C were used for training the object detection module and tuning the catching time threshold t_{thresh}. Object D, on the other hand, is used in the following section for testing purposes.

IV. Results

In this section, we discuss results from using our framework to catch various objects thrown towards the robot. Example snapshots of a successful catch are shown in Figure 1. The reader is also encouraged to watch the supplementary video, which illustrates the framework's effectiveness through diverse tests with various objects of different sizes and colors. In particular, with our experiments we seek to answer the following questions:

- How robust is the framework to varying throwing styles (i.e. possible catching locations in the robot workspace), and to different thrown objects?
- What are the effects and benefits of the different catching position intersection methods presented in Section III-D?

TABLE I: Performance of different catching position identification methods.

	Plane Intersection	Min Distance to Center	GMM
Success rate [%]	70	78	80
Mean total power [W]	21.89	17.70	15.13

A. General Performance

To assess the overall performance of our system, we conducted catching experiments using object D which was not included in the YOLOv8 model's training data during finetuning. Object D was used for testing due to its smaller catching surface, enabling us to evaluate the different catching position identification methods outlined in Section III-D. For each of the three methods, we threw the object 50 times from a distance of 2 meters aiming at the center of the quadruped's front legs. A successful catch was defined as the quadruped closing its front legs to hold the thrown object without it falling to the ground. Table I shows the success rates and the average total power consumed for successful catches, across the 50 throws for each of the three methods.

We observe an 8% difference in performance between the Plane Intersection method and the Minimum Distance to Center method, and a 2% difference in performance between the Minimum Distance to Center and the GMM method. Clearly, the latter two methods significantly enhance the catching performance of the quadruped compared with the simpler Plane Intersection baseline method. Moreover, comparing the Plane Intersection method with the Minimum Distance to Center and GMM methods shows a 19% and 31% decrease in average power consumption, respectively. In summary, we find that for "easy-to-catch" throws aimed at the center, the Minimum Distance to Center and GMM methods outperform the Plane Intersection method. Additionally, these two methods exhibit comparable performance, with the GMM method showing slightly better performance in terms of catching success rate and average power consumption.

B. Challenging Catching Scenario Case Study

To assess the system performance under challenging conditions, we evaluated the catching success rate of the different methods with "harder-to-catch" throws using object D. We conducted multiple throws aimed at the limits of the robot's reachable space and observed the success rate of each method. Throws directed toward the rightward, leftward, and upward limits of the workspace yielded similar results to those presented in Table I. However, throws aimed at the lower limit proved to be more difficult to catch. To evaluate the performance of the methods with these throws, we executed 10 throws aimed lower than 15 cm below the quadruped's shoulders for each method and recorded the success rate. This type of low throw is illustrated in Figure 4, where the object is aimed at a low point of the quadruped's reachable space. Using the Plane Intersection and Min Distance to Center methods, the quadruped failed to catch any low throws. In contrast, employing the GMM method resulted in a 60% success rate, with the quadruped successfully catching 6 out of the 10 low throws. The GMM method notably outperforms the other methods with hard-to-catch throws. Its success can be attributed to its probabilistic modeling of the workspace of the front legs and providing insights into more human-intuitive catching locations. Consequently, this method selects the most intuitive catching position on the predicted trajectory, as shown in Figure 4c. Conversely, the failure of the Plane Intersection method to catch low throws can be attributed to the predicted trajectory intersecting the catching plane lower than the front legs' reachable space limits as depicted in Figure 4a. Additionally, the Minimum Distance to Center method, lacking information on catching intuition or the reachable space of the front legs, may select a catching position closest to the nominal center, but in a location that is challenging to catch the object, as shown in Figure 4b.

V. Conclusion

In conclusion, this paper presents a comprehensive framework for enabling a quadruped robot to autonomously catch thrown objects using its front legs. The system involves multiple submodules, including quadruped elevation, object detection, trajectory prediction, catching position selection, and the catching control maneuver. The YOLOv8-based object detection module, fine-tuned for the specific task, demonstrated effective detection capabilities with tight bounding boxes. Trajectory prediction using an ordinary least squares approach showcased iterative refinement of regression parameters, resulting in accurate predictions with increasing numbers of measurements. To select a suitable catching position, we evaluated three different strategies: 1) trajectory intersection with a vertical plane, 2) the closest point on the predicted trajectory to the nominal center of the feet, and 3) the point on the predicted trajectory which has the highest likelihood on a Gaussian Mixture Model modelling the robot's reachable space. We found that the GMM offered the best performance (80% success rate and lowest power consumption) with easy-to-catch throws, and additionally allowed catches in harder scenarios with boxes thrown at the extreme limits of the robot workspace.

Limitations include the need for a more controlled experimental setup to fully quantify performance, and consideration of the thrown object's orientation on catching performance. Future work could focus on improving these limitations, generalization to catch arbitrary objects, and developing more sophisticated control methods (beyond the Cartesian PD closing mechanism, such as MPC) to catch objects thrown at an angle.

References

[1] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[2] G. Bellegarda and A. Ijspeert, "CPG-RL: Learning central pattern generators for quadruped locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 547–12 554, 2022.

[3] G. Bellegarda, M. Shafiee, and A. Ijspeert, "Visual CPG-RL: Learning central pattern generators for visually-guided quadruped locomotion," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 1420–1427.

[4] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.

[5] G. Bellegarda, Y. Chen, Z. Liu, and Q. Nguyen, "Robust high-speed running for quadruped robots via deep reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 10 364–10 370.

[6] M. Shafiee, G. Bellegarda, and A. Ijspeert, "Viability leads to the emergence of gait transitions in learning agile quadrupedal locomotion on challenging terrains," *Nature Communications*, vol. 15, no. 1, p. 3073, 2024.

[7] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv preprint arXiv:2309.14341*, 2023.

[8] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *Conference on Robot Learning (CoRL)*, 2023.

[9] M. Shafiee, G. Bellegarda, and A. Ijspeert, "Puppeteer and marionette: Learning anticipatory quadrupedal locomotion based on interactions of a central pattern generator and supraspinal drive," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1112–1119.

[10] M. Shafiee, G. Bellegarda, and A. Ijspeert, "ManyQuadrupeds: Learning a single locomotion policy for diverse quadruped robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 3471–3477.

[11] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: Learning a unified policy for manipulation and locomotion," in *Conference on Robot Learning (CoRL)*, 2022.

[12] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, "Roloma: Robust loco-manipulation for quadruped robots with arms," *Autonomous Robots*, vol. 47, no. 8, pp. 1463–1481, 2023.

[13] J.-P. Sleiman, F. Farshidian, and M. Hutter, "Versatile multicontact planning and control for legged loco-manipulation," *Science Robotics*, vol. 8, no. 81, p. eadg5014, 2023.

[14] E. Arcari, M. V. Minniti, A. Scampicchio, A. Carron, F. Farshidian, M. Hutter, and M. N. Zeilinger, "Bayesian multi-task learning mpc for robotic mobile manipulation," *IEEE Robotics and Automation Letters*, 2023.

[15] C. Schwarke, V. Klemm, M. van der Boon, M. Bjelonic, and M. Hutter, "Curiosity-driven learning of joint locomotion and manipulation tasks," in *7th Annual Conference on Robot Learning*, 2023.

[16] X. Cheng, A. Kumar, and D. Pathak, "Legs as manipulator: Pushing quadrupedal agility beyond locomotion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[17] A. Rigo, Y. Chen, S. K. Gupta, and Q. Nguyen, "Contact optimization for non-prehensile loco-manipulation via hierarchical model predictive control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9945–9951.

[18] M. Sombolestan and Q. Nguyen, "Hierarchical adaptive loco-manipulation control for quadruped robots," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 156–12 162.

[19] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo, "Learning whole-body manipulation for quadrupedal robot," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 699–706, 2023.

[20] B. Forrai, T. Miki, D. Gehrig, M. Hutter, and D. Scaramuzza, "Event-based agile object catching with a quadrupedal robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[21] Y. You, T. Liu, X. Liang, Z. Xu, M. Zhou, Z. Li, and S. Zhang, "Run and catch: Dynamic object-catching of quadrupedal robots," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 743–750.

[22] F. Shi, T. Homberger, J. Lee, T. Miki, M. Zhao, F. Farshidian, K. Okada, M. Inaba, and M. Hutter, "Circus anymal: A quadruped learning dexterous manipulation with its limbs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2316–2323.

[23] W. J. Wolfslag, C. McGreavy, G. Xin, C. Tiseo, S. Vijayakumar, and Z. Li, "Optimisation of body-ground contact for augmenting the whole-body loco-manipulation of quadruped robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3694–3701.

[24] W. Hong and J.-J. E. Slotine, "Experiments in hand-eye coordination using active vision," in *Experimental Robotics IV: The 4th International Symposium, Stanford, California, June 30–July 2, 1995*. Springer, 2005, pp. 130–139.

[25] M. Riley and C. G. Atkeson, "Robot Catching: Towards Engaging Human-Humanoid Interaction," *Autonomous Robots*, vol. 12, no. 1, pp. 119–128, Jan. 2002.

[26] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.

[27] J. Kober, M. Glisson, and M. Mistry, "Playing catch and juggling with a humanoid robot," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 875–881.

[28] 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)*. IEEE, 2020, pp. 6718–6725.

[29] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2001, pp. 1623–1629.

[30] T. Gold, R. Römer, A. Völz, and K. Graichen, "Catching objects with a robot arm using model predictive control," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 1915–1920.

[31] Y. Huang, M. Hofer, and R. D'Andrea, "Offset-free model predictive control: A ball catching application with a spherical soft robotic arm," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 563–570.

[32] D. Carneiro, F. Silva, and P. Georgieva, "Robot anticipation learning system for ball catching," *Robotics*, vol. 10, no. 4, p. 113, 2021.

[33] S. S. M. Salehian, M. Khoramshahi, and A. Billard, "A dynamical system approach for softly catching a flying object: Theory and experiment," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 462–471, 2016.

[34] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3719–3726.

[35] B. Bäuml, 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*. IEEE, 2010, pp. 2592–2599.

[36] B. Huang, Y. Chen, T. Wang, Y. Qin, Y. Yang, N. Atanasov, and X. Wang, "Dynamic handover: Throw and catch with bimanual hands," in *Conference on Robot Learning*. PMLR, 2023, pp. 1887–1902.

[37] S. Gomez-Gonzalez, S. Prokudin, B. Schölkopf, and J. Peters, "Real time trajectory prediction using deep conditional generative models," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 970–976, 2020.

[38] S.-T. Kao and M.-T. Ho, "Ball-catching system using image processing and an omni-directional wheeled mobile robot," *Sensors*, vol. 21, no. 9, p. 3208, 2021.

[39] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[40] X. Huang, Z. Li, Y. Xiang, Y. Ni, Y. Chi, Y. Li, L. Yang, X. B. Peng, and K. Sreenath, "Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2715–2722.

[41] Y. Zhao, R. Yang, G. Chevalier, R. C. Shah, and R. Romijnders, "Applying deep bidirectional lstm and mixture density network for basketball trajectory prediction," *Optik*, vol. 158, pp. 266–272, 2018.

[42] S. Kim and A. Billard, "Estimating the non-linear dynamics of free-flying objects," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1108–1122, Sep. 2012.

[43] G. Bellegarda, C. Nguyen, and Q. Nguyen, "Robust quadruped jumping via deep reinforcement learning," *arXiv preprint arXiv:2011.07089*, 2020.

[44] Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo, and S. Kim, "Optimized jumping on the mit cheetah 3 robot," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7448–7454.

[45] B. Dwyer, J. Nelson, J. Solawetz *et al.*, "Roboflow (Version 1.0)," 2022.

[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2018.

[47] Unitree Robotics. Go1. https://www.unitree.com/products/go1/.