

A Notations

Table 1: Definition of symbols.

Symbol	Description
v_B	Linear velocity of the base in robot frame
ω_B	Angular velocity of the base in robot frame
g	Gravity direction of the robot’s base
q	Joint positions of the robot
τ	Applied joint torque of the robot
\dot{q}	Joint velocities of the robot
a	Network’s prediction
x_{rot}	Robot attitude (roll, pitch, and height)
v_B^*	Target linear velocity of the base in robot frame
ω_B^*	Target angular velocity of the base in robot frame
q^*	Target joint positions of the robot
x_{rot}^*	Target robot attitude
$\mathbb{1}_{walking}$	True iff target velocities are non-zero.
$\mathbb{1}_{flight}$	True iff fewer than three feet are in contact with the ground.
t_c^j	Contact time of foot j
t_a^j	Air time of foot j
T_{gait}	Gait cycle time
${}^wP^{(i)}$	Point cloud observation
${}^wx_{arm}^{(i)}$	Desired 3D location of the arm end-effector
${}^wR_{arm}^{(i)}$	Desired 3D rotation of the arm end-effector
${}^wx_{leg}^{(i)}$	Desired 3D location of the leg end-effector
${}^wR_{leg}^{(i)}$	Desired 3D rotation of the leg end-effector
${}^wp_{body}^{(i)}$	Desired body pose
$m^{(i)}$	Desired leg mask for manipulation or locomotion use
$g^{(i)}$	Gripper state
${}^w\xi_{arm}^{(i)}$	Arm end-effector trajectory
${}^w\xi_{leg}^{(i)}$	Leg end-effector trajectory
${}^w\xi_{torso}^{(i)}$	Torso end-effector trajectory

B Hardware Setup

The arm-mounted quadruped robot we use in this project is Boston Dynamics’ Spot with a Spot Arm mounted on its back [50]. There are 12 actuators for the Spot, and each leg has three. The arm has six degrees of freedom and has a gripper at the end effector. The system is shown in Figure 9.

In command-level control, we query the Spot API [51] for the real-time Spot joint state estimates. For the reinforcement learning locomotion policy, we input the proprioceptive states concatenated with the torso targets into the neural network trained by RL, represented in ONNX Runtime [52]. The computed joint targets from RL policy and IK planners are masked and combined together as a 19-dimension target joint vector for all Spot leg and arm joints and sent to the robot using the low-level motor control API [51].

For task specifications on the task level, we use the built-in stereo cameras on Spot to get RGBD images and construct the colored point clouds used in the *contact point* and *language instruction* based interfaces.

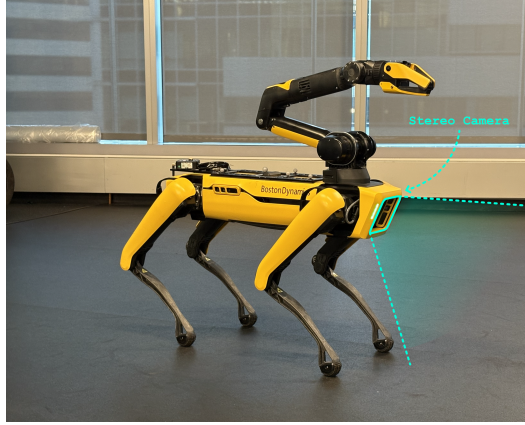


Figure 9: **Robot Hardware.** Boston Dynamics Spot quadruped robot with the Spot Arm mounted on its back. This platform is used for all loco-manipulation experiments presented in this work.

C Task Design

This section details the real-world loco-manipulation tasks used to evaluate the ReLIC framework, as introduced in Section 4.2. As shown in Figure 3, we define a suite of twelve diverse and complex tasks, grouped into three categories based on the nature of interlimb coordination. Some tasks are instantiated in multiple variants tailored to different task interfaces, including **D** for direct target specification, **C** for contact-point-based control, and **L** for language instruction. For details about the task interfaces, please refer to Appendix F.

C.1 Mobile Interlimb Coordination

These tasks require the robot to manipulate large objects using its arm and one leg while navigating the space with the remaining three legs.

Yoga Ball. A 55 cm yoga ball is placed on a platform in front of the robot. The robot is asked to either (1) pick up the ball and place it into a nearby laundry basket (**D**); or (2) pick up the ball and move backward with it while maintaining balance (**C, L**).

Shipping Box. A large cardboard box is placed randomly in the environment. The robot is required to pick up the box and transport it to a predefined location (**D**).

C.2 Stationary Interlimb Coordination

These tasks involve coordinated arm and leg motion for object manipulation while robustly balancing the torso with the remaining three legs.

Tire Pump. A small manual tire pump is connected to a flat bike tire. The robot stabilizes the pump using the arm and uses one leg to step on the pump repeatedly to inflate the tire (**D**).

Trash Bin. An eight-gallon step-on trash bin is placed in front of the robot. The robot is asked to either (1) pick up a Coke can, step on the pedal to open the lid, and throw the can into the bin (**D**); or (2) with the can already grasped, step on the pedal and dispose of the can (**C**).

Deck Box. A large deck box is placed in front of the robot, with a folded blanket on the ground. The robot grasps the blanket with its arm, lifts the lid of the deck box using a leg, and places the blanket inside while maintaining stability (**D**).

Small Bin. A one-gallon step-on bathroom bin is placed randomly. The robot picks up a paper ball from the floor, moves to the bin, opens the lid using one leg, and throws the ball inside using the arm (**D**).

C.3 Foot-Assisted Manipulation

These tasks are feasible with the arm alone but benefit from the use of a leg to enhance efficiency or robustness.

Tool Chest. A tool chest with one or more drawers is used in multiple variants: (1) pick up a blue tape from the floor, place it into an open drawer, and close the drawer using one leg (**D**); (2) place a pre-held tape into the drawer and close it with a leg (**C**); (3) simultaneously close two drawers using the arm and a leg (**L**).

Storage Bin. Toys are randomly scattered on the floor. The robot picks them up and throws them into a large plastic bin. To improve efficiency, it may drag the bin closer using a hind leg while reaching for toys (**D**).

Chair. The robot must move a chair through a narrow opening partially blocked by a box. It clears the box using a leg while simultaneously dragging the chair with its arm (**D**).

Basket. A shopping basket is placed on the floor. The robot must: (1) pick up a paper towel from a shelf, place it into the basket, and lift the basket using a leg (**D**); or (2) with a toy in hand, place it into the basket, then lift and carry the basket while moving backward (**C**).

Drawer. Multiple toys are scattered on the ground. The robot must clean up by picking up toys one by one and placing them on a small nightstand, opening and closing the drawer with a leg as needed (**D**).

Laundry Bag. The robot picks up a blanket from the ground and places it into a large laundry bag. To improve efficiency, it uses a hind leg to drag the laundry bag closer while grasping the blanket (**D**).

D Implementation Details of Reinforcement Learning

This section outlines the implementation details of our reinforcement learning pipeline, especially regarding the Markov Decision Process (MDP) formulation used to train the reinforcement learning locomotion controller described in Section 3.2. The policy is trained in IsaacLab [44] using Proximal Policy Optimization (PPO) as implemented in RSL-RL [53]. All symbols follow the definitions provided in Table 1.

D.1 Observation Space

Table 2 summarizes the observation terms used as policy input. The observation space consists of two main components: the robot state (proprioception) and the target commands.

The robot state includes base linear and angular velocities, gravity vector, joint positions and velocities, as well as the previous action. To improve generalization, we apply additive noise to these terms during training. The noise models are detailed in Table 2. No scaling or clipping is applied to the individual observation terms.

The target commands specify desired robot behavior, including base linear and angular velocities, body attitude, and joint positions for the arm and legs. These commands are sampled from a pre-computed distribution designed to ensure physical feasibility and collision-free configurations. Linear and angular velocity commands are drawn from $\mathcal{U}(-1.0, 1.0)$ in m/s and rad/s, respectively. Roll and pitch commands are sampled from $\mathcal{U}(-0.3, 0.3)$ radians, and base height from $\mathcal{U}(0.3, 0.7)$ meters.

To ensure feasibility and trackability, target commands are sampled and post-processed offline during training. Joint targets are uniformly sampled within the configuration space and filtered using rejection sampling. Samples are discarded if they result in self-collision or ground collision. Additionally, we compute the support polygon and center of gravity (CoG) for each sampled configura-

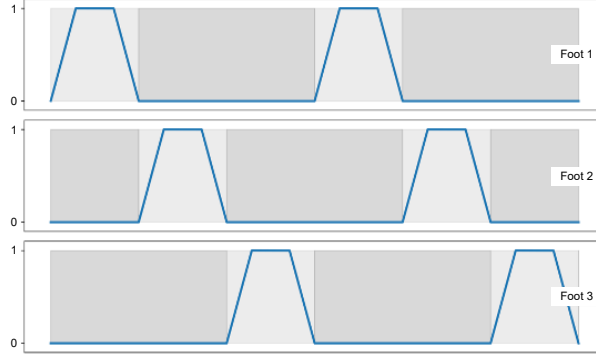


Figure 10: **Gait regularization.** Desired contact timing among the remaining three supporting legs when one leg is lifted for manipulation. The phases are cyclically staggered to enable stable dynamic bouncing.

tion and retain only those in which the CoG lies within the support polygon. A total of 1.0×10^7 valid command samples are generated offline for training.

Table 2: Observation terms and additive noise models used during training. No scaling or clipping is applied to individual terms.

Observation Term	Symbol	Additive Noise
Base Linear Velocity	v_B	$\mathcal{U}(-0.1, 0.1)$
Base Angular Velocity	ω_B	$\mathcal{U}(-0.2, 0.2)$
Gravity Vector	g	$\mathcal{U}(-0.05, 0.05)$
Joint Positions	q	$\mathcal{U}(-0.05, 0.05)$
Joint Velocities	\dot{q}	$\mathcal{U}(-0.5, 0.5)$
Previous Action	a	None
Target Linear Velocity	v_B^*	None
Target Angular Velocity	ω_B^*	None
Target Joint Positions (Arm & Leg)	q^*	None
Target Base Orientation	x_{rot}^*	None

D.2 Reward Function

Table 3 summarizes the reward terms and their corresponding weights. To accelerate policy convergence, we adopt a reward scheduling strategy in which regularization terms (the lower half of Table 3) are initialized at zero and linearly increased over the first 10,000 training steps, after which they remain constant.

The gait reward enforces distinct locomotion patterns depending on the number of supporting legs. For four-leg locomotion, the policy is encouraged to produce a symmetric trotting gait. The reward is computed based on contact time t_c^j and air time t_a^j of each foot $j \in \text{FL, FR, HL, HR}$, following the formulation in [44]. Specifically, we penalize mismatches in contact and air durations between diagonal foot pairs, and between each foot and the air duration of its non-diagonal counterpart:

$$\begin{aligned}
 r_{four-leg} = & ||t_c^{FL} - t_c^{HR}|| + ||t_c^{FR} - t_c^{HL}|| + ||t_a^{FL} - t_a^{HR}|| + ||t_a^{FR} - t_a^{HL}|| \\
 & + ||t_c^{FL} - t_a^{FR}|| + ||t_c^{HL} - t_a^{HR}|| + ||t_a^{FL} - t_c^{FR}|| + ||t_a^{HL} - t_c^{HR}||
 \end{aligned} \quad (\text{D.1})$$

For three-leg locomotion, we enforce a cyclic bouncing gait among the remaining supporting feet. The target contact phases are visualized in Figure 10. When one leg (e.g., FL) is lifted for manipulation, the remaining feet (FR, HR, and HL) are assigned to *foot 1*, *foot 2*, and *foot 3* respectively in clockwise order. A fixed gait cycle duration of $T_{\text{gait}} = 0.4$ seconds is used to define desired contact

Table 3: Reward Terms Summary. The environment scales the reward weights with the time-step dt [53]. For brevity, we drop the time-step t from individual quantities.

Term Name	Definition	Weight
Robot Base Linear Velocity Tracking	$\exp - \frac{\ \mathbf{v}_B - \mathbf{v}_B^*\ }{0.25}$	7.0
Robot Base Angular Velocity Tracking	$\exp - \frac{\ \boldsymbol{\omega}_B - \boldsymbol{\omega}_B^*\ }{0.25}$	3.5
Robot Base Attitude Tracking	$\ \mathbf{x}_{rot} - \mathbf{x}_{rot}^*\ ^2$	-120
Gait	$\mathbb{1}_{walking} \cdot (r_{four-leg} + r_{three-leg})$	-5
Robot Joint Velocity	$\ \dot{\mathbf{q}}\ ^2$	-1.0×10^{-5}
Robot Joint Acceleration	$\ \ddot{\mathbf{q}}\ ^2$	-1.0×10^{-6}
Robot Applied Joint Torque	$\ \boldsymbol{\tau}\ ^2$	-2.0×10^{-4}
Action Rate	$\ \dot{\mathbf{a}}\ ^2$	-0.1
Flight Penalty	$\mathbb{1}_{flight}$	-10.0

timing. The corresponding reward penalizes deviation from the target phase alignment:

$$r_{three-leg} = \mathbb{1}_{t_a^1 > 0} \cdot (||t_c^2 - t_a^1 - \frac{T_{gait}}{3}|| + ||t_c^3 - t_a^1||) \quad (\text{D.2})$$

D.3 Termination and Reset

During training, an episode terminates either when the agent reaches a time limit of 20 seconds or when the robot’s body makes contact with the ground. Upon termination, the environment resets the robot to a randomly sampled initial state. The base height is sampled from $\mathcal{U}(0.2, 1.3)$ meters, while roll and pitch are sampled from $\mathcal{U}(-0.5, 0.5)$ radians. The xy position and yaw are uniformly sampled across the workspace. Initial base linear and angular velocities are drawn from $\mathcal{U}(-0.5, 0.5)$ in both m/s and rad/s, respectively. Joint positions for both the arm and legs are uniformly sampled within their respective configuration spaces.

E Implementation Details of Sim-to-Real Transfer

This section details the techniques used to enable successful transfer of the learned locomotion policy from simulation to the real robot. We describe the domain randomization strategies employed during training, followed by the motor calibration procedure that accounts for discrepancies in actuator dynamics between simulation and hardware.

E.1 Domain Randomization

To enhance the robustness of the locomotion policy under real-world variability, we apply extensive domain randomization during training. Specifically, we randomize physical properties such as link and actuator friction coefficients, actuator stiffness and damping parameters, and terrain mesh geometry. In addition, we apply periodic random external pushes to the base to simulate unexpected disturbances.

E.2 Motor Calibration

As discussed in Section 3.3, accurate actuator torque limits are critical for successful zero-shot deployment on hardware. In practice, we found that calibrating only the knee joints using torque-velocity constraints is sufficient. Figure 11 presents the torque-velocity profiles collected from the initial policy deployed on the real robot.

The calibration process begins with manual estimation of joint limits, followed by automatic refinement using CMA-ES [48]. To optimize the limits, we replay the same policy command sequence in simulation and minimize the Wasserstein distance between the rollout trajectories from the real

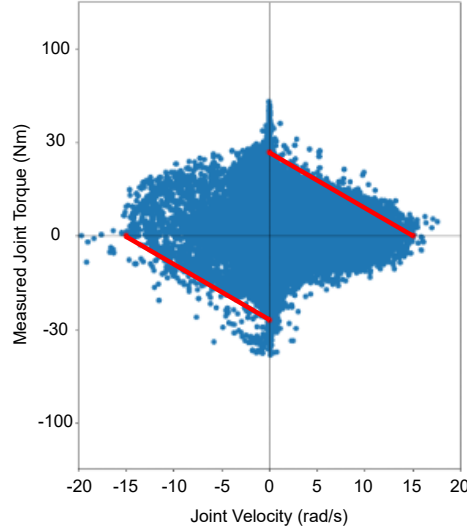


Figure 11: **Torque-Velocity Calibration.** Empirical torque versus joint velocity measurements for the knee joints, collected on the real robot. Each blue point corresponds to a sensor-recorded sample. The red lines denote the initial uncalibrated torque limits used in simulation.

world and simulation. In our experiments, a single round of this procedure yielded satisfactory results.

F Implementation Details of Task Interface

This section provides implementation details for the three task interface modalities used in our experiments, including direct target specification, contact-point-based decomposition, and language-driven task grounding via vision-language models (VLMs). These interfaces correspond to the modalities described in Section 3.4 and are used to generate the target-driven task representation for the ReLIC controller.

F.1 Direct Targets

The *Direct Target* interface supports teleoperation by specifying targets for selected limbs and the torso in real time. This is achieved through a compact teleoperation setup that combines off-the-shelf input devices for intuitive control. A Sony PS5 joystick is used to control the robot’s torso velocity, allowing for responsive and agile base movement. For manipulation, we use a 3Dconnexion SpaceMouse to set precise 6-DoF pose targets for the arm or any of the legs, enabling full spatial control over end-effector positioning and orientation. These inputs are sampled at a fixed rate and interpolated into smooth Cartesian-space trajectories using linear and spherical interpolation.

F.2 Contact Points

In the contact point based interface, we use a contact-based task decomposition to specify the contact points for different end-effectors to solve the tasks. We specify the target contact point on the object, the pre- and post-contact directions, and the selected limb (arm or one of the legs).

Contact Point Selection. At the start of stage i , the system fuses RGB-D images from the camera sensors with the robot’s proprioceptive data to produce a 3D point cloud observation ${}^w P^{(i)} \in \mathbb{R}^{n \times 3}$, where n is the total number of points. From this point cloud, we derive the corresponding targets $\mathcal{T}^{(i)}$ for the current stage by selecting two key points (when we use two limbs for manipulation):

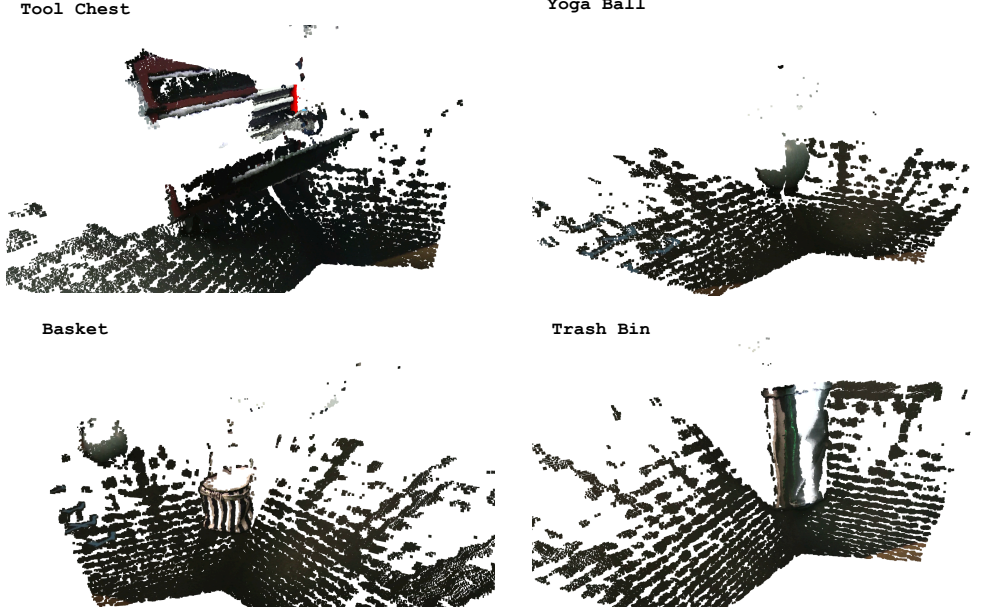


Figure 12: **Point Cloud for Contact Point Interface.** Point cloud constructed using Spot’s onboard stereo cameras in the contact point interface. Users annotate the scene by selecting arm and leg contact points, along with pre- and post-contact directions.

one for the desired 3D location of the arm end-effector $w x_{\text{arm}}^{(i)} \in {}^w P^{(i)}$, and another for the desired 3D location of the foot $w x_{\text{leg}}^{(i)} \in {}^w P^{(i)}$. The desired $SE(3)$ pose of the end-effector, $w p_{\text{arm}}^{(i)} = \begin{bmatrix} w x_{\text{arm}}^{(i)} & w R_{\text{arm}}^{(i)} \end{bmatrix}^\top$, combines the chosen position $w x_{\text{arm}}^{(i)}$ with a specified rotation $w R_{\text{arm}}^{(i)} \in \mathbb{R}^3$. Meanwhile, the desired foot location is simply $w p_{\text{leg}}^{(i)} = w x_{\text{leg}}^{(i)}$. The inverse kinematics (IK) module then calculates the desired body pose $w p_{\text{body}}^{(i)} = \text{IK}(w p_{\text{arm}}^{(i)}, w p_{\text{leg}}^{(i)})$. Additionally, we specify a gripper state $g^{(i)}$ and a desired leg mask $m^{(i)}$, resulting in the complete set of stage targets $\mathcal{T}^{(i)}$.

As shown in Figure 12, the interactive point-cloud selection interface will appear every time the robot has reached the previous targets, allowing the user to click on point cloud ${}^w P^{(i)}$ and specify $w x_{\text{arm, leg}}^{(i)}$ and $w R_{\text{arm}}^{(i)}$, followed by manual input of $g^{(i)}$ and $m^{(i)}$. An autonomous VLM agent can also replace the selection from human annotation, as introduced in Appendix F.3.

Waypoints Generation. Once the targets $\mathcal{T}^{(i)}$ are specified, the next step is to generate dense waypoints guiding the robot from its current pose to the desired configuration. While any off-the-shelf trajectory optimization (TO) algorithm could be used, here we employ a straightforward approach: linear interpolation for 3D translation and spherical linear interpolation (Slerp) for 3D rotation. Specifically, we produce three separate trajectories for each target set $\mathcal{T}^{(i)}$: the arm end-effector trajectory $w \xi_{\text{arm}}^{(i)} \in \mathbb{R}^{H \times 6}$, the leg end-effector trajectory $w \xi_{\text{leg}}^{(i)} \in \mathbb{R}^{H \times 3}$, and the torso trajectory $w \xi_{\text{torso}}^{(i)} \in \mathbb{R}^{H \times 6}$. Based on the current pose, the target pose, and the maximum linear and angular velocities, we determine the required number of waypoints for each trajectory (*i.e.*, H_{arm} , H_{leg} , and H_{torso}). The overall trajectory length H is then defined as $H = \max(H_{\text{arm}}, H_{\text{leg}}, H_{\text{torso}})$.

Following these trajectories, the robot tracks each successive waypoint at every timestep using the low-level RL controller described in the next section. When all trajectories have been executed, we check whether the difference between the robot’s final pose and the target configuration is below predefined thresholds. If it is, the targets \mathcal{T}^i are considered “reached”; otherwise, a new set of trajectories is generated from the current pose.

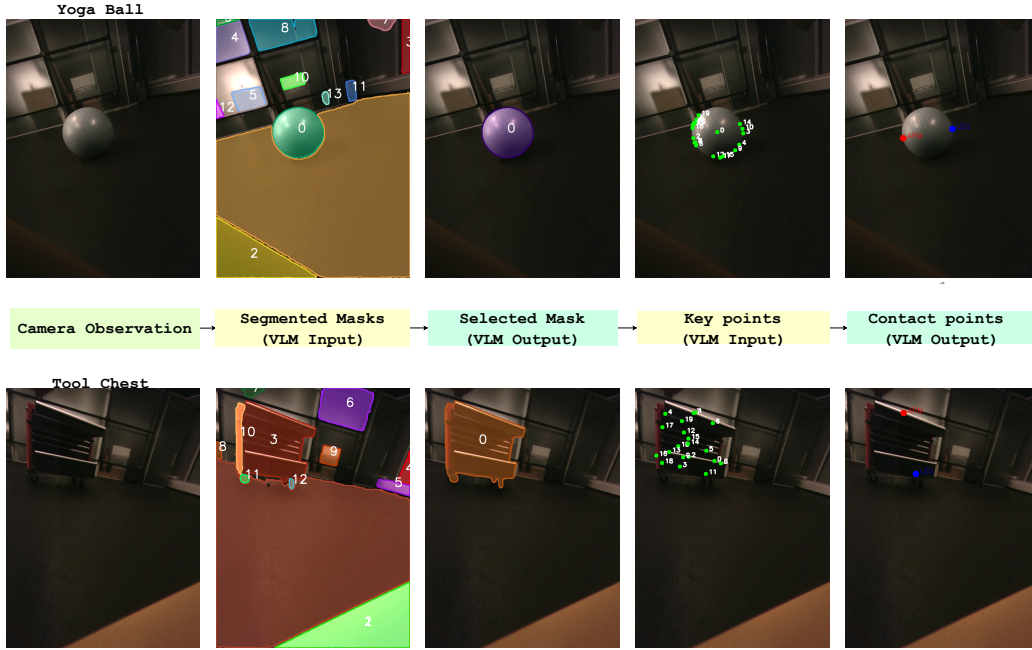


Figure 13: **Language Instruction Task Interface.** Overview of the vision-language pipeline used to automatically infer end-effector contact points from free-form task descriptions. Given an RGB observation, SAM2 segments the scene into object masks, which are then passed to GPT-4o for semantic reasoning. The model selects relevant masks based on task semantics, identifies keypoint candidates, and outputs final contact points for the arm and leg. Examples are shown for the two tasks, *Yoga Ball* (top row) and *Tool Chest* (bottom row).

F.3 Language Instructions

The *Language Instruction* interface enables natural, flexible task specification by grounding free-form language to contact points via vision-language reasoning. We use GPT-4o [49] to process RGB images captured from the robot’s front camera. The scene is first segmented into object masks using SAM2 [54]. In the first stage, GPT-4o selects the task-relevant masks from among these segments, identifying objects that are directly involved in the task. In the second stage, keypoints are sampled within the selected regions, and GPT-4o selects appropriate contact points for the arm and leg end-effectors based on the task description and robot capabilities.

The complete vision-language grounding pipeline is illustrated in Figure 13. Predefined contact directions are assigned according to task semantics. For instance, in the *Yoga Ball* task, the arm and leg move toward each other. In the *Tool Chest* task, both end-effectors move outward from the robot body toward the object.

Example task-specific prompts. Below are the prompt template for language instruction interfaces.

Prompt for task-specific mask selection

I’m showing you an image where different objects have been segmented and numbered.
Based on this task description:

Task details:

Using the arm and leg to close the opening tool chest drawers.

Objects of interest: tool chest drawers.

Robot capabilities:

- Leg: end effector provide contact with objects
- Arm: end effector provide contact with objects

Constraints:

- Use the arm to close the upper opening drawer of the tool chest
- Use the leg to close the lower opening drawer of the tool chest
- ground is not related to the task, so you should ignore it when choosing the contact points

Please identify which numbered regions/objects are most relevant for this task. Choose the most related masks.

Respond with ONLY the number of the relevant mask in this format:

MASK: *mask number*

DO NOT include any explanation or reasoning.

Prompt for contact points selection based on provided key points

Based on this task: “Tool Chest Closing Task”, select TWO points for the robot:

1. One point where the ARM/GRIPPER should contact the object
2. One point where the LEG should contact or interact with the object

Task details:

Using the arm and leg to close the opening tool chest drawers.

Objects of interest: tool chest drawers.

Robot capabilities:

- Leg: end effector provide contact with objects
- Arm: end effector provide contact with objects

Constraints:

- Use the arm to close the upper opening drawer of the tool chest
- Use the leg to close the lower opening drawer of the tool chest
- ground is not related to the task, so you should ignore it when choosing the contact points

Choose points that will allow for stable and effective manipulation.

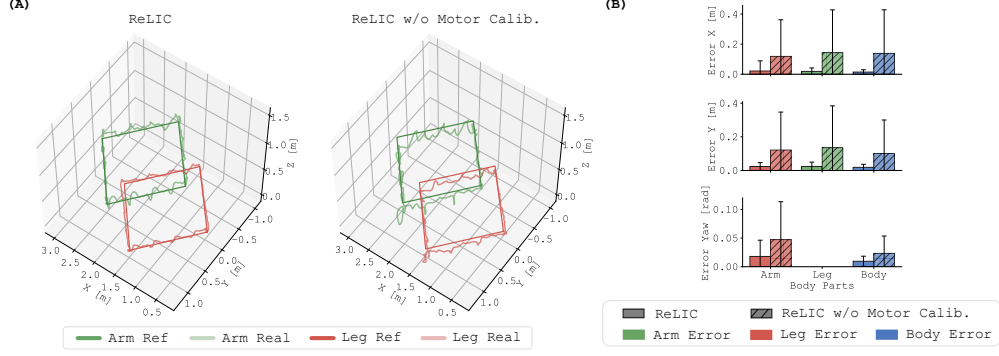


Figure 14: **Ablation Study on Motor Calibration.** Comparison of leg and arm end-effector co-tracking performance between the full *ReLIC* policy and ablated variants. Removing motor calibration (*ReLIC w/o Motor Calib.*) leads to reduced tracking accuracy and stability.

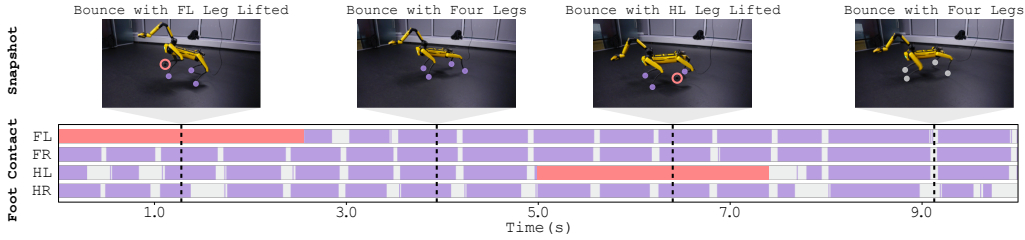


Figure 15: **Ablation Study on Gait transition.** Gait transition for *ReLIC w/o Gait Regular.*: compared to *ReLIC*, the robot has a bouncing gait for both four-legged and three-legged motion, which provides less stability in both tracking and gait switching. Videos can be found on the website.

After the contact points are selected, the target and waypoints generation procedure is the same as the implementation in the contact point interface.

G Ablation Study

Below we discuss the evaluation of the *ReLIC* controller reported in Section 4.1. We present two ablation experiments on the locomotion reinforcement learning controller trained in the simulator. Both experiments are performed using the same reward structure and training setup in simulation. Specifically, we evaluate two ablated variants: **ReLIC w/o Motor Calib.**, trained without motor calibration, and **ReLIC w/o Gait Regular.**, trained without the gait regularization reward.

We assess the importance of motor calibration by measuring the target tracking accuracy of end-effector trajectories for both the arm and legs. As shown in Figure 14, motor calibration substantially improves the stability and precision. The calibrated policy maintains smoother and more synchronized movements across limbs. Furthermore, the calibrated controller demonstrates significantly improved robustness. The robot can remain statically balanced on three legs for over 10 minutes without requiring active adjustment. In contrast, the uncalibrated policy struggles to maintain balance, frequently initiating compensatory motions. A video of the 10-minute static standing demonstration is available on the project website.

We examine the effect of removing the gait regularization term during training by applying the same gait transition protocol used in Figure 4 to the **ReLIC** policy. The resulting gait behaviors are visualized in Figure 15. Without the regularization, the learned gait degenerates into a repetitive bouncing motion, both in quadrupedal and tripodal modes. Despite using motor calibration, this policy exhibits poor stability and consistently fails during the rhombus-shaped trajectory tracking task due to falls. These results highlight that motor calibration alone is insufficient; structured gait priors are critical for achieving stable, flexible locomotion across contact configurations.