Preferenced Oracle Guided Multi-mode Policies for Dynamic Bipedal Loco-Manipulation

Prashanth Ravichandar* pr39760@usc.edu
Lokesh Krishna* lkrajan@usc.edu
Nikhil Sobanbabu ns_562@usc.edu
Quan Nguyen quann@usc.edu

Dynamic Robotics and Control Laboratory
University of Southern California
Los Angeles, CA 90089, USA

Abstract:
Dynamic loco-manipulation requires whole-body control and contact-rich interactions with objects and the environment. Existing learning-based approaches train a high-level policy or hand-design a finite state machine to switch between low-level skill policies trained independently, often resulting in quasi-static behaviors. We propose Preferenced Oracle Guided Multi-mode Policies to learn a single policy that masters multiple modes and their preferred sequences of transitions. Designing hybrid automatons as oracles to generate references over different control modes, policy optimization is guided through bounded exploration. With a task-agnostic preference reward, we enforce learning a desired sequence of mode transitions, thereby solving the task effectively. In uni-object loco-manipulation tasks like omnidirectional box moving and soccer, our approach results in whole-body control and smooth transitions, enabling contact-rich dribbling, goal kicks, and ball stops. Leveraging the oracle's abstraction, we solve each loco-manipulation task across diverse robot morphologies, including HECTOR V1, Berkeley Humanoid, Unitree G1, and H1, using the same reward definition and weights. Project website: https://indweller.github.io/ogmplm/

Keywords: Whole-body Control, Loco-Manipulation, Reinforcement Learning

1 Introduction

Robot control for loco-manipulation requires tight integration of locomotion and manipulation to interact extensively with the object and environment. Traditionally, agile locomotion and dexterous manipulation have been studied individually, with the overlapping focus being robustness and recovery behaviors. In locomotion, the robot's contact with its environment is limited to the end-effector while maintaining balance and tracking a desired CoM reference over challenging terrains. In contrast, manipulation requires multiplicity in contact interactions with objects of distinct geometries and dynamics. For effective loco-manipulation, a control policy is thus required to stabilize the underactuated states of the robot and the object, leveraging its whole body to make and break contact to solve the task effectively. For loco-manipulation, model-based optimal control relies on a pre-planned contact schedule and knowledge of an object's physical properties [1, 2]. Selecting an optimal

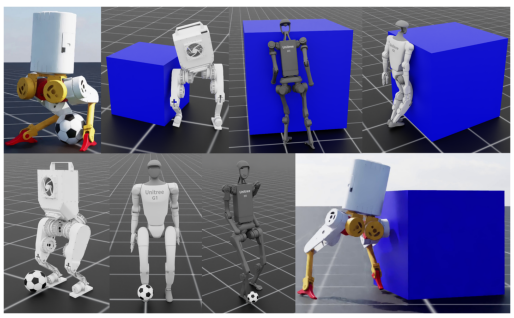


Figure 1: Dynamic contact-rich whole body loco-manipulation tasks on different bipedal and humanoid robots with Preferenced OGMP.

RSS 2025 Workshop on Whole-Body Control and Bimanual Manipulation (RSS2025WCBM).

sequence of contacts is combinatorial, typically addressed through sampling-based [3] or heuristic-based [4] planners, followed by trajectory optimization [5], with recent efforts toward combined optimization [6]. Nonetheless, its online computational tractability remains challenging when augmenting object dynamics, highlighting an open problem in loco-manipulation control.

As a compelling alternative, Deep Reinforcement Learning (RL) for robot control scales to high-dimensional simulation models [7, 8, 9, 10, 11, 12]. Optimizing a policy over stochastic dynamics allows introducing randomization, perturbation, and model uncertainty as seen in a real robot, thereby leading to robust controllers for legged locomotion in the wild [13, 14] and dexterous manipulation of objects with complex geometry [15]. For uni-object quadrupedal loco-manipulation, prior works leverage hierarchical frameworks with a versatile low-level locomotion policy and a high-level policy for arm-control [16] or navigation commands [17]. Analogously, [18] proposes a high-level switching policy and multiple low-level skill policies for agile goalkeeping. With two independently trained policies and a hand-designed finite state machine, [19] showcases dynamic quadrupedal dribbling and recovery. Similarly, [20] learns multiple low-level policies with a subsequent state-initialization strategy for various modes of operation and executes a predetermined sequence of policy switches for humanoid box loco-manipulation. A shared motif of the above methods is to leverage hierarchical control with a learned high-level policy or a hand-designed finite state machine with pre-trained low-level policies. Despite the promising empirical results, it remains unclear whether the hierarchy and learning of multiple policies are a fundamental necessity or a practical convenience for loco-manipulation. Moreover, the amount of task-specific engineering, combined with the lack of abstraction, limits the ability of these approaches to be adopted across different tasks and robots. Alternatively, [21] learns multiple low-level skill policies but finally distills them into a single policy for humanoid soccer. Thus, a single policy network can approximate all the necessary skills, yet a multi-policy training strategy is adopted to 1) ease the reward shaping for each skill and 2) alleviate the local optima arising from undesirable skill transitions. In this work, we learn a single policy that masters multiple skills (or control modes) via guided policy optimization, prioritizing desired mode transitions while maintaining an abstraction extendable to different robots and tasks.

Recently [22] proposed a task-agnostic guided policy optimization framework for multi-contact quadrupedal loco-manipulation. A task-specific policy is guided by a pre-computed demonstration from an offline multi-modal planner [5], featuring continuous trajectories and contact schedules with adaptive phase dynamics to handle discrepancies between demonstration and rollout. While [22] is tractable for motion with sparse user-defined contact affordances [5], the scalability to contact-dense bipedal dribbling is not straightforward, highlighting the need for emergent behaviors through exploration. Concurrent work [23] proposes Oracle Guided Multi-mode Policies (OGMP) as a framework for structured exploration by bounding the permissible states to a local neighborhood of a coarse state reference generated by a closed-loop oracle. In [23], a single policy is trained for solving agile bipedal parkour with control modes like leap for gaps, jump for blocks, and pace for flat terrain, traversing arbitrary parkour tracks. Unlike other approaches, the mode transitions in OGMP are implicit and emergent without relying on high-level modules, leading to effective transition maneuvers. While OGMP was validated with oracles using robot state feedback and exogenous time-varying inputs (like height-map), the design of oracles to accommodate external object dynamics is unexplored. Unlike parkour, where terrain geometry uniquely determines the mode transitions, soccer involves multiple possible mode sequences to solve the task, which is the focus of this work. We first introduce a hybrid automata perspective to design multi-mode oracles with continuous reference-generating dynamics and discrete mode switches. With such an oracle, we propose a Preferenced OGMP to synthesize multi-mode control favoring desirable mode transitions. The key contribution of our paper is two-fold:

1. We formalize oracle design using a hybrid automata perspective to model multi-mode reference generation. By designing a single oracle, we solve different uni-object loco-manipulation tasks such as omnidirectional moving box and soccer variants on multiple bipeds and humanoid robots including HECTOR, Berkeley Humanoid, Unitree G1 and H1 (Fig. 1).

2. We introduce a task-agnostic preference reward that directs the policy towards a desired sequence of mode transitions. The resulting Preferenced OGMP solves the loco-manipulation tasks effectively without any robot-specific reward shaping.

The rest of the paper is organized as follows: Sec 2 revisits the OGMP theory, Sec 3 introduces the hybrid automata perspective for oracles, Sec 4 presents the design methodology applied for the tasks of interest followed by the results in Sec 5.

2 Oracle Guided Multi-mode Policies Revisited

Given an oracle Ξ, which serves as a closed-loop state reference generator for solving a task $\mathcal{T}$, [23] presents an oracle guided policy optimization framework to synthesize multi-mode policies to optimize a task objective $J_{\mathcal{T}}$. Specifically for any task variant $\psi_{\mathcal{T}} \in \Psi_{\mathcal{T}}$, Ξ provides a finite-horizon reference which is at most ϵ away from the optimal state trajectory x_t^*, from any given state x_t based on feedback λ_t, as defined below.

$$x^{\Xi}_{t:t+t_H} = \Xi(\lambda_t) \tag{1a}$$

$$\text{s.t.} \quad \exists\, x^{\Xi}_{t:t+t_H} \quad \forall\,(x_t, \psi_{\mathcal{T}}) \tag{1b}$$

$$\|x^{\Xi}_t - x^*_t\|_W < \epsilon \quad \forall\, t \in [0,\, \infty) \tag{1c}$$

where W is a user-defined weight matrix and ϵ is an unknown maximum deviation bound. With such a Ξ, OGMP localizes the exploration to a ρ-neighborhood of x^{Ξ}, ensuring effective learning for an appropriate choice of ρ. Formally,

$$\pi^* := \arg\max_{\pi \in \Pi} J_{\mathcal{T}} \tag{2a}$$

$$\text{s.t.} \quad \|x^{\pi}_t - x^{\Xi}_t\|_W < \rho \quad \forall t \in [0, \infty) \tag{2b}$$

where x^{π} are the states visited while rolling out policy π and ρ is the permissible state-bound. By bounded exploration, OGMP effectively escapes the multitude of local optima in the objective landscape, delivering the desired performance.

3 Multi-mode Oracles as Hybrid Automata

This section establishes a functional view of multi-mode oracles as hybrid automata [24]. Since oracles are reference generators, they have continuous dynamics that evolve the reference states and discrete jumps between the control modes based on environment feedback, leading to the following definition,

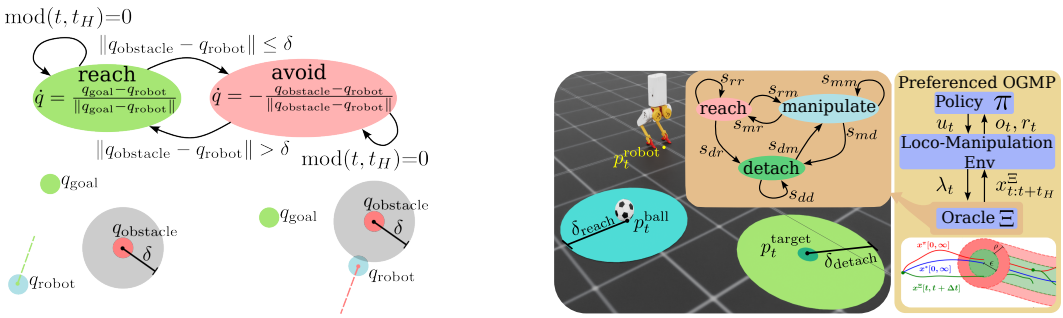
$$\Xi := (\mathcal{M}, \mathcal{X}, \Lambda, f, \mathcal{S}) \tag{3}$$

where $\mathcal{M}$ is a discrete set of control modes, $\mathcal{X}$ is the continuous state space of the reference, Λ is the continuous input space for environment feedback, $f : \mathcal{M} \times \mathcal{X} \times \Lambda \to \mathcal{X}$ is the reference generating dynamics, and $\mathcal{S}$ is a set of permissible transitions with reset, guard, and invariant conditions. Thus, Ξ generates continuous references in $\mathcal{X}$ and discretely jumps between modes in $\mathcal{M}$ through switches in $\mathcal{S}$. As an oracle, these design choices can be coarse, merely guiding optimization as an ansatz rather than offering a high-fidelity solution for the task. We present the following example to elucidate this perspective.

Example - Reach-avoid Task: We take the classical reach-avoid problem [25] to demonstrate the design of an oracle for guided policy optimization. Consider a 2D robot modeled as a double integrator, $\ddot{q}_{\text{robot}} = u$, a parameterized controller, $u = \pi(\,.\,;\theta)$ and a point obstacle and goal with generalized positions $q_{\text{obstacles}}, q_{\text{goal}} \in \mathbb{R}^2$ respectively.

Given the current environment feedback $\lambda_t := [q_{\text{robot}}, q_{\text{obstacle}}, q_{\text{goal}}, t]$, Ξ generates a reference by integrating the continuous dynamics forward in time i.e.,

$$q^{\Xi}_{\tau} = \int_t^{\tau} \dot{q} dt, \quad \forall \tau \in [t,\, t + t_H]$$



(a) (b)

Figure 2: a) A multi-mode oracle designed as a hybrid automaton for the reach-avoid task. Dotted lines show the generated references in reach (green, left) and avoid (red, right) modes. b) Overview of the proposed framework: the environment dynamically queries the multi-mode oracle for a reference. A bounded exploration around the reference is then performed to learn dynamic loco-manipulation effectively.

from the current mode in $\mathcal{M} := \{\text{reach, avoid}\}$. The switching conditions $\mathcal{S}$ ensure that when the obstacle is at least δ m away, Ξ remains in reach mode, generating references toward the goal, Fig. 2a (left). Otherwise, it switches to avoid mode guiding the robot away from the obstacle, Fig. 2a (right). A policy optimization algorithm is expected to learn the optimal θ by exploring the ρ-neighborhood of oracle's references as defined in (2). We now highlight key remarks of this formulation.

Remark 1. *λ_t couples the environment dynamics with the oracle dynamics. Thus, depending on the control provided by π, the sequence of mode transitions within the oracle can vary across rollouts. While the oracle defines all the permissible mode switches, the policy determines the mode-transition probabilities.*

Remark 2. *From Remark 1, we note that an agent might converge to undesirable transition sequences by exploiting dynamics and task objectives. For example, π may learn to avoid an obstacle indefinitely (getting stuck in an "avoid $\rightarrow$ reach $\rightarrow$ avoid" loop) rather than reaching the goal and completing the task. This highlights the need for an explicit mechanism to enforce a preferred sequence of mode transitions, as discussed in Sec 4.3*

Thus, a hybrid automaton view of oracles provides a formal framework to design oracles systematically. Note that the notational complexity does not imply design complexity while scaling to high-dimensional systems, as we will demonstrate in Sec. 4.

4 Methodology

This section presents our design methodology for solving uni-object bipedal loco-manipulation tasks. We first define the tasks of interest and the chosen modes of the oracle. Next, we outline the oracle design and propose a preferenced oracle-guided policy optimization, as shown in Fig. 2b.

4.1 Uni-object Loco-manipulation

We divide the uni-object loco-manipulation task into three modes - reach, manipulate, and detach. With this 3-mode abstraction, we aim to learn policies for multiple robots with varying dynamic and kinematic properties. We consider two tasks with different requirements in the finesse of manipulation: *soccer* and *move box*. The task variants are listed in Table 1, along with the robots trained for each task.

4.1.1 Soccer task

The robot is tasked with approaching a ball, dribbling it to a designated target, and detaching from it. The ball is arbitrarily initialized about a fixed distance from the robot, as seen in Fig. 2b (left).

Table 1: Variants of uni-object loco-manipulation tasks

Tasks	Regularize	Robots
soccer-stop	×	HECTOR v1
soccer-kick	✓	Berkeley Humanoid, Unitree G1 and H1
move box	×	HECTOR v1
move box	✓	Berkeley Humanoid, Unitree G1 and H1

Upon reaching the target position, the robot is expected to halt with two variations: *stop* - bringing the ball to rest from omnidirectional dribbling, and *kick* - shooting it towards the goal after unidirectional dribbling. Guided by the oracle's coarse reference, the policy is expected to emergently learn contact-rich interactive behaviors to control the soccer ball.

4.1.2 Move box task

The move box task is analogous to soccer but only comprises the *stop* variant for the detach mode. We choose boxes of dimensions comparable to each robot's nominal height to ensure whole-body loco-manipulation behaviors, continuously making and breaking contact with the box to "negotiate" it towards the goal.

4.2 Oracle design for loco-manipulation

We design a single oracle for all loco-manipulation tasks to provide references for the robot base and object states across different modes. Following (1), we define an oracle Ξ with modes $\mathcal{M} := \{$reach, manipulate, detach$\}$ and permissible mode transitions $\mathcal{S}$ as shown in Fig. 2b. The oracle receives an environment feedback, $\lambda_t := [p_t^{\text{robot}}, p_t^{\text{object}}, p_t^{\text{target}}, t]$, where $p_t^{\square}$ denotes the position of $\square$, and t is the current time in the episode. We chose the oracle reference horizon $t_H = 1$ sec and found the choice of t_H does not affect performance, as shown in [23]. The reference state is defined as the CoM states of the robot and object, i.e. $x^{\Xi} := [p_{\Xi}^{\text{robot}}, \theta_{\Xi}^{\text{robot}}, v_{\Xi}^{\text{robot}}, \omega_{\Xi}^{\text{robot}}, p_{\Xi}^{\text{object}}, v_{\Xi}^{\text{object}}]$, where θ, v, and ω denote the orientation, linear velocity, and angular velocity respectively. All the references are generated as linear interpolations between the current and target positions. Similar to warm-starting trajectory optimization [26], we found that linearly interpolated references were sufficient to solve our tasks as RL optimizes the policy subject to the full robot dynamics in the simulation. This property makes our approach simple and scalable for loco-manipulation to multiple robots, eliminating the need for dynamically feasible demonstrations [22] or carefully re-targeted kinematic references [27, 28]. By only having references for a subspace of the full state, the joint states are free for the policy to explore, thereby generating distinct behaviors for each task.

4.3 Preferenced Oracle Guided Policy Optimization

Our proposed preferenced oracle guided policy optimization framework for dynamic loco-manipulation is visualized in Fig. 2b. At each environment step, the multi-mode oracle provides a t_H-long reference in the active mode determined by the environment feedback. The policy optimization is guided by the oracle's reference using permissible state terminations as in Section 2 and rewards in Table 3. With this OGMP setup, we observed that the policy could exploit the automaton and converge to undesirable mode transition sequences. To overcome such local optima, we propose assigning mode preference as discussed below.

Mode preference : Given a set of oracle modes and rewards, multiple solutions exist with some mode transitions resulting in undesirable behavior. For example, in Fig. 2b, the agent can exploit the loop $s_{rr} \to s_{rm} \to s_{mr} \to s_{rr}$, leading to kicking the ball once at high velocity in the manipulate mode and continuously chasing it in the reach mode. Such "creative" behaviors hinder finding the preferred behavior, such as $s_{rr} \to s_{rm} \to s_{mm} \to s_{md} \to s_{dd}$. Removing transitions (e.g., s_{mr}) to break the loop would be conservative as adversarial perturbations may drive the ball away from the robot, necessitating a switch to reach mode. Hence, we need to retain all necessary transitions but minimize the number of undesired transitions. Since it is infeasible to specify all optimal

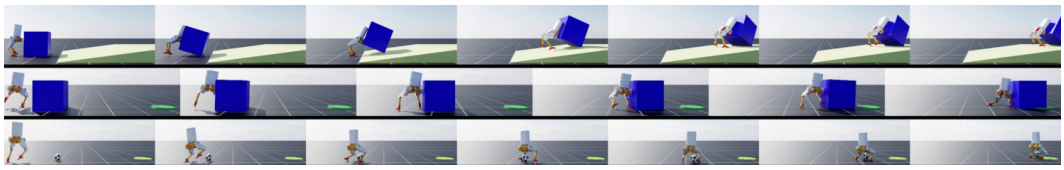


Figure 3: Keyframes from the simulation: a) First row - moving the box along the slope. b) Second row - moving the box omnidirectionally along the plane c) Third row - dribbling a soccer ball in the soccer-stop task. In each case, the robot learns to manipulate the object to the target location (green circle).

mode sequences apriori (as it is state-dependent), we propose assigning ranks to each mode. Ensuring increasing ranks point toward the preferred direction of transitions, we introduce the following preference reward :

$$r_{\text{preference}} = \begin{cases} -1 & m_t < \max\limits_{0 \le \tau \le t} (m_\tau) \\ 0 & \text{otherwise} \end{cases}$$

where t is the current time step in the episode, and m_t is the rank of the mode at t. Indicating preference as a penalizing reward allows yet minimizes transitions in the "wrong direction" and results in subsequent recovery to the "right direction". Specifically, we set ranks: reach-0, manipulate-1, and detach-2 and penalize any transition that decreases the mode rank compared to the highest rank observed thus far in the current episode. In the example above, switching from manipulate to reach (rank change: $2 \rightarrow 1$) is allowed but penalized, forcing the policy to learn the necessary recovery maneuvers. Note that the preference reward offers flexibility, allowing different modes to share the same rank and enabling multiple optimal behaviors. Unlike densely shaped reward terms, the above-proposed preference reward is sparse by definition and is left untuned across tasks.

Training details: For training, we use the PPO [29] implementation of RSL RL [30] with the actor/critic network architectures being 1 RNN cell with 128 hidden units followed by an MLP with hidden units [200, 100]. The remaining hyperparameters are retained from the default configuration for RSL RL provided by Isaac Lab [31]. For all tasks specified in Table 1, the policy observation comprises the proprioceptive robot states, relative distances to the object and target, and a phase indicating the current step in the reference. The policy outputs actuator PD targets, to be tracked by a low-level PD controller to compute torques. Our rewards are split into task and regularization terms, as seen in Table 3. Along with the base tracking from [23], object-tracking and simple task rewards are added. Apart from the maximum episode length, we terminate based on the position errors of the robot and object relative to the current oracle reference. For the permissible state bounds we use $\rho_x^{\text{robot}} = 0.4$, $\rho_y^{\text{robot}} = 0.4$ and $\rho_z^{\text{robot}} = 0.2$ and $\rho_x^{\text{object}} = 0.4$. To ensure realistic ball-robot dynamics in the soccer tasks, we incorporate a drag force represented by the equation $F_{drag} = -0.5 v_{ball}^2$, similar to [19].

5 Results

In this section, we present our results and performance analysis across the loco-manipulation tasks with different robots listed in Table 1. We train on four robots that vary in morphology, actuation, and degrees of freedom - HECTOR v1, Berkeley Humanoid, Unitree G1, and H1. Since HECTOR is under active development, we train without regularization for simulation results. For other robots, we add regularization rewards considering the sim-to-real concerns, thus resulting in smoother motions. Our codebase with all the above robot and task variants is open-sourced [32]. It is worth noting that during inference, the policies are independent of the oracle, and the reference-based terminations are deactivated. We present a qualitative overview followed by quantitative comparisons with relevant baselines.

5.1 Qualitative Analysis of Performance

As shown in Fig. 3 and 4, the policies trained for each task successfully complete the intended loco-manipulation objective. Training a single multi-mode policy allows for implicit learning of

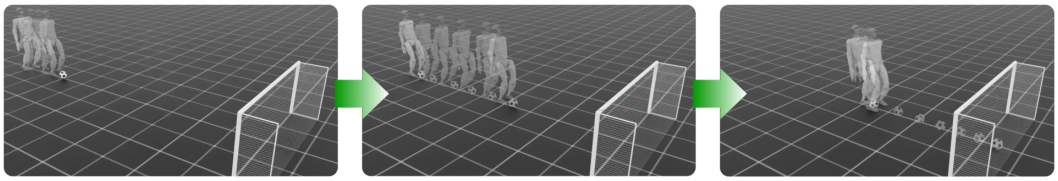


Figure 4: Motion trace of the Unitree G1 policy, transitioning through reach, manipulation, and detach modes.

non-trivial transition maneuvers. This is highlighted in Fig. 5 (left, center) for soccer-kick task, where the robot crouches and decelerates as it approaches the ball to prevent undesirable contacts. Similarly, before kicking, it accelerates to impart momentum to the ball, as shown in Fig. 5 (right). As seen in the supplementary video, during the manipulate mode, HECTOR exhibits dribbling behaviors both when the ball is within and outside the support polygon spanned by its feet. It extensively uses both legs, shuffling the ball between its feet, kicking with one foot, and stopping it with the other. Despite using the same oracle and reward weights, we note the resulting policies for each robot differ in their strategies, adhering to the kinematic and dynamic limits. For instance, in soccer-kick, Berkeley Humanoid kicks using both legs, while G1 and H1 favor single-leg kicks. In the move box task, all robots employ their entire body to establish a "stable" support plane on the object and manipulate it toward the target location. Rather than always facing the box, the policies prefer to exploit the robot morphology to cling to the object with non-trivial configurations. Hence, across all tasks, emergent behaviors involving contact-rich whole-body interactions are observed. This includes dynamic adjusting of contact points and non-trivial on-and-off contacts to stabilize and reactively "negotiate" with the object. Thus, without additional robot-specific hyperparameter tuning, our approach enables successful policy synthesis across multiple robots and task variants within the abstraction of dynamic uni-object loco-manipulation.

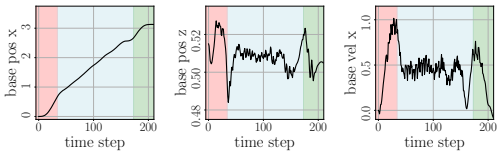


Figure 5: Selected CoM states in the soccer-kick task for Berkeley Humanoid. Colored regions indicate active modes-reach (red), manipulate (blue), kick (green).

5.2 Comparisons and Quantitative Analysis

5.2.1 Performance Metrics

Table 2 presents the quantitative evaluation for the soccer-kick task. Over 100 episodes, we randomize the initial joint states between $(-0.05, 0.05)$. We declare the episode to be successful when the ball reaches the goal and the robot stands within the detached region. The average ball contacts represent the number of contacts the robot makes with the ball in the manipulate mode. The fall percentage indicates the fraction of episodes in which the robot height falls below a threshold. Despite being trained without initial randomization, our approach succeeds in 98% of the episodes with at least 9 contacts made with the ball in the manipulate mode in each episode. We compute the resulting transition probabilities from each mode by counting the transitions and normalizing them over the total number of transitions. This provides the mode transition distributions the trained policy converged to, as shown in Fig. 6. Self-transitions dominate the distribution due to the receding horizon design of oracles — having to stay in a mode until a different switching condition is met.

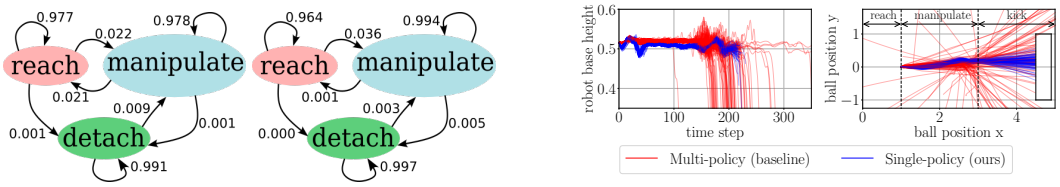
Table 2: Metrics over 100 episodes for the soccer-kick task

Policy	Success %	Avg. ball contacts	Fall %
multi-policy (baseline)	26.0	**12.46**	69.0
single-policy w/o pref	28.0	5.11	**0.0**
single-policy (ours)	**98.0**	9.76	**0.0**

5.2.2 Effect of preference reward

We compare the effect of our proposed preference reward in increasing the number of desirable transitions as noted in Remark 2. From Fig. 6a, we observe that the policy without preference

reward has comparable transition probabilities between reach → manipulate (0.022) and manipulate → reach (0.021). Thus, instead of an effective manipulation strategy, the policy exploits oracle dynamics, getting stuck in a reach → manipulate → reach loop, where the robot kicks the ball away and then chases it, as shown in the video. In contrast, the addition of the preference reward (Fig. 6b) ensures the transition probability from reach → manipulate is greater (0.036) than that of manipulate → reach (0.001). As shown in Table 2, training with preference results in a higher number of ball contacts. While both policies have a low fall percentage, the preference-trained policy stands after kicking, while the non-preference policy avoids falling by perpetually walking behind the ball. Hence, adding preference results in 3x more successful goal kicks than the one without, as seen in Table 2. Thus, the preference reward helps disambiguate the desired optima corresponding to *reach* → *manipulate* → *detach* instead of other locally optimal mode transitions.



(a) w/o preference reward (b) preference reward (c) base height and ball trajectories

Figure 6: a) and b) Mode transition probabilities of the policy in the soccer-stop task. c) Robot's base height and ball trajectories over 100 episodes from randomized joint state initialization for the Berkeley Humanoid in the soccer-kick task.

5.2.3 Single-policy vs. Multi-policy approaches

We compare the performance of our single multi-mode policy with the established multiple uni-mode policies approach. As a baseline, we train three separate policies, one per mode in the soccer-kick task, using the same mode-specific rewards. During inference, we use the oracle as the finite state machine to switch between the policies similar to prior approaches [20, 19]. When trained without initial randomization, the baseline policies fail to transition between each other, as seen in the attached video. To alleviate this, we randomize the initial floating base and joint states for the baseline manipulate and kick policies. As shown in Table 2, the multi-policy baseline falls in 69% of the episodes due to fragile inter-policy transitions, while our single-policy results in stabilizing base motions through robust inter-mode transitions, also seen in Fig. 6c (left). Comparing the dribbling performance, the baseline has 30% more contacts during the manipulate mode. Despite the contact-rich manipulation, the multi-policy baseline fails to transition to the kick policy, as seen in Fig. 6c (right). In contrast, having a comparable number of ball contacts, our approach results in a successful transition to kick mode and a task success rate of 98% — 3x more than the baseline approach. Despite having 0.3x the learnable parameters of the multi-policy approach, our single multi-mode policy outperforms the baseline.

6 Conclusion

This work presents Preferenced Oracle Guided Multi-mode Policies and is applied to dynamic bipedal loco-manipulation. By designing reference-generating hybrid automata as coarse oracles with continuous reference dynamics and discrete mode switches, we effectively guide policy optimization. To minimize the number of undesired transition sequences, we introduce a novel mode preference reward, enhancing performance in tasks with multiple optimal mode transitions. The proposed approach was successfully validated over task variants like move box, soccer-stop, and soccer-kick with multiple robots of varying form factors, including HECTOR v1, Berkeley Humanoid, Unitree G1, and H1 in simulation. Compared with existing approaches, we demonstrate the advantage of training a single multi-mode policy with mode preference, resulting in non-trivial dynamic loco-manipulation. Ongoing work involves the sim-to-real transfer, with future efforts aimed at extending to more challenging tasks.

Appendix

Table 3: Rewards

Term	Weight	Expression
Task Rewards		
Base Pos.	0.3	$\exp(-5.0\|p^{\text{robot}} - p^{\text{robot}}_{\Xi}\|)$
Base Ori.	0.3	$\exp(-5.0[1 - (\theta^{\text{robot}} \cdot \theta^{\text{robot}}_{\Xi})^2])$
Base Lin. Vel.	0.15	$\exp(-2.0\|v^{\text{robot}} - v^{\text{robot}}_{\Xi}\|)$
Base Ang. Vel.	0.15	$\exp(-2.0\|\omega^{\text{robot}} - \omega^{\text{robot}}_{\Xi}\|)$
Mode Preference	-5.0	$\mathbb{1}(m_t < \max_{0 \le \tau \le t}(m_\tau))$
Object Proximity	0.5	$\mathbb{1}(\|p^{\text{object}} - p^{\text{robot}}\| \le h^{\text{robot}})\mathbb{1}(m_t = 1)$
Object Pos.	1.0	$\exp(-2.0\|p^{\text{object}} - p^{\text{object}}_{\Xi}\|)\mathbb{1}(m_t = 2)$
Object Lin. Vel.	1.0	$\exp(-2.0\|v^{\text{object}} - v^{\text{object}}_{\Xi}\|)\mathbb{1}(m_t = 2)$
Ball Rest Penalty	-0.5	$\mathbb{1}(\|v^{\text{object}}\| \le 0.05)\mathbb{1}(m_t = 1)$
Regularization Rewards		
Torque Mag.	0.1	$\exp(-10.0\|\tau\|)$
Torque Rate	0.1	$\exp(-10.0\|\tau - \tau_{\text{prev}}\|)$
Joint Vel.	0.1	$\exp(-10.0\|\dot{q}\|)$
Default Joint Pos.	-0.1	$\sum_i \|q_i - q_i^{\text{default}}\|$

τ, q, $\dot{q}$ denotes joint torques, positions and velocities, h^{robot} denotes the nominal height of the robot and $\|\,.\,\|$ denotes the l^2-norm divided by corresponding normalizing constant.

References

[1] J. Li and Q. Nguyen. Multi-contact mpc for dynamic loco-manipulation on humanoid robots. In *2023 American Control Conference (ACC)*, pages 1215–1220, 2023. doi:10.23919/ACC55779.2023.10156397.

[2] Picking up momentum. https://bostondynamics.com/blog/picking-up-momentum/.

[3] H. Zhu, A. Meduri, and L. Righetti. Efficient object manipulation planning with monte carlo tree search. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10628–10635, 2023. doi:10.1109/IROS55552.2023.10341813.

[4] S. J. Jorgensen, M. Vedantam, R. Gupta, H. Cappel, and L. Sentis. Finding locomanipulation plans quickly in the locomotion constrained manifold. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6611–6617, 2020. doi:10.1109/ICRA40945.2020.9197533.

[5] J.-P. Sleiman, F. Farshidian, and M. Hutter. Versatile multicontact planning and control for legged loco-manipulation. *Science Robotics*, 8(81):eadg5014, 2023. doi:10.1126/scirobotics.adg5014. URL https://www.science.org/doi/abs/10.1126/scirobotics.adg5014.

[6] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake. Shortest paths in graphs of convex sets. *SIAM Journal on Optimization*, 34(1):507–532, Feb. 2024. ISSN 1095-7189. doi:10.1137/22m1523790. URL http://dx.doi.org/10.1137/22M1523790.

[7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[8] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450. IEEE, 2024.

[9] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

[10] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[11] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[12] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024.

[13] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst. Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning. In *RSS*, 2021.

[14] 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*, 2022.

[15] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023. doi:10.1126/scirobotics.adc9244. URL https://www.science.org/doi/abs/10.1126/scirobotics.adc9244.

[16] I. Dadiotis, M. Mittal, N. Tsagarakis, and M. Hutter. Dynamic object goal pushing with mobile manipulators through model-free constrained reinforcement learning, 2025. URL https://arxiv.org/abs/2502.01546.

[17] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo. Learning whole-body manipulation for quadrupedal robot. *IEEE Robotics and Automation Letters*, 9(1):699–706, 2024. doi:10.1109/LRA.2023.3335777.

[18] 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, 2022. URL https://arxiv.org/abs/2210.04435.

[19] Y. Ji, G. B. Margolis, and P. Agrawal. Dribblebot: Dynamic legged manipulation in the wild. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5155–5162, 2023. doi:10.1109/ICRA48891.2023.10160325.

[20] J. Dao, H. Duan, and A. Fern. Sim-to-real learning for humanoid box loco-manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16930–16936, 2024. doi:10.1109/ICRA57147.2024.10610977.

[21] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess. Learning agile

soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89): eadi8022, 2024. doi:10.1126/scirobotics.adi8022. URL https://www.science.org/doi/abs/10.1126/scirobotics.adi8022.

[22] J. P. Sleiman, M. Mittal, and M. Hutter. Guided reinforcement learning for robust multi-contact loco-manipulation. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 531–546. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/sleiman25a.html.

[23] L. Krishna, N. Sobanbabu, and Q. Nguyen. Ogmp: Oracle guided multi-mode policies for agile and versatile robot control, 2024. URL https://arxiv.org/abs/2403.04205.

[24] T. Henzinger. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, 1996. doi:10.1109/LICS.1996.561342.

[25] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, page 11–20, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334334. doi:10.1145/2728606.2728612. URL https://doi.org/10.1145/2728606.2728612.

[26] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018. doi:10.1109/LRA.2018.2798285.

[27] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning Agile Robotic Locomotion Skills by Imitating Animals. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi:10.15607/RSS.2020.XVI.064.

[28] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *The International Journal of Robotics Research*, 0(0):02783649241285161, 0. doi:10.1177/02783649241285161. URL https://doi.org/10.1177/02783649241285161.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[30] rsl_rl: Fast and simple implementation of rl algorithms, designed to run fully on gpu. https://github.com/leggedrobotics/rsl_rl.

[31] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi:10.1109/LRA.2023.3270034.

[32] https://github.com/DRCL-USC/ogmp_isaac.