2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

# Scalable Multi-Objective Robot Reinforcement Learning through Gradient Conflict Resolution

Humphrey Munn[1,2], Brendan Tidd[2], Peter Böhm[1,2], Marcus Gallagher[1], David Howard[2]

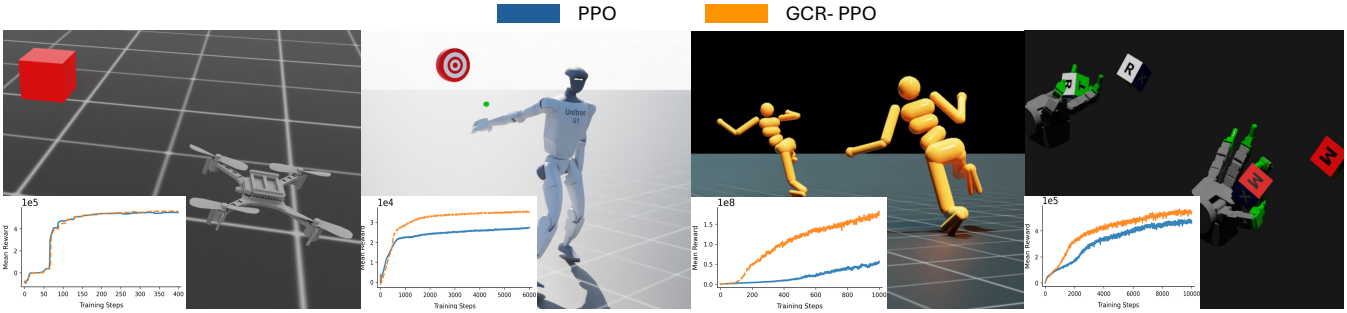


Fig. 1: Comparison of GCR-PPO and massively-parallel PPO across four representative tasks: Quadcopter, Full-Body Throwing (multi-objective), Humanoid Running (multi-objective: gait length, energy usage, arm span, base height), and Allegro-Cube. Performance is similar on the Quadcopter task, where gradient conflict is low. On the other three tasks, which exhibit higher gradient conflict, GCR-PPO achieves improved performance. Overall, these results indicate that GCR-PPO mitigates objective conflicts and scales more effectively than standard parallelised PPO.

***Abstract*— Reinforcement Learning (RL) robot controllers usually aggregate many task objectives into one scalar reward. While large-scale proximal policy optimisation (PPO) has enabled impressive results such as robust real-world robot locomotion, many tasks still require careful reward tuning and remain brittle to local optima. Tuning cost and sub-optimality grow with the number of objectives, limiting scalability. Modelling reward vectors and their trade-offs can address these issues; however, multi-objective methods remain underused in RL for robotics because of computational cost and optimisation difficulty. In this work, we study gradient conflicts that arise when multiple task objectives are combined into a scalar reward. In particular, we explicitly address the conflict between task-based rewards and terms that regularise the policy towards realistic behaviour. We propose GCR-PPO, a lightweight modification to PPO that decomposes actor updates into objective-wise gradients using a multi-headed critic and resolves conflicts according to objective priority. We evaluate GCR-PPO on IsaacLab manipulation and locomotion benchmarks and two additional tasks modified to include many objectives. GCR-PPO demonstrates superior scalability compared to massively-parallel PPO ($p = 0.04$) without significant computational overhead. Across tasks, GCR-PPO improves performance over large-scale PPO by an average of 9.5% (Symmetric Percentage Change), with larger gains on tasks exhibiting higher gradient conflict. Code is available at: https://github.com/humphreymunn/GCR-PPO.**

## I. INTRODUCTION

As robotics is applied to increasingly complex real-world tasks, scalable reinforcement learning (RL) becomes increasingly important. Scalability challenges arise across multiple aspects of robotic RL [1], [2], including sample efficiency, where robots must repeatedly interact with the environment to refine behaviour, and behavioural complexity, which concerns learning policies that satisfy many objectives simultaneously. While recent advances in fast parallel simulators [3], [4] have significantly improved sample efficiency, behavioural complexity remains comparatively underexplored.

Behavioural complexity arises in two primary forms: multi-task and multi-objective learning. Multi-task learning involves solving multiple distinct tasks, such as a kitchen robot that can boil water, open the fridge, cook, and clean [5]. Each task is typically defined by a separate goal or reward specification, and policies are trained to generalise across these tasks. In contrast, multi-objective learning requires simultaneously balancing multiple requirements that may vary over time. This enables more flexible and emergent behaviours without explicit, discrete task definitions. However, existing multi-objective RL approaches for robotics struggle to scale due to difficulties handling conflicting objectives and limited sample efficiency.

We address this challenge by focusing on scalable multi-objective learning, where multiple reward signals are combined to guide a single policy without destabilising optimisation. Our approach builds on gradient repair methods [6] for actor–critic algorithms. We introduce Gradient Conflict Resolution (GCR) and integrate it into Proximal Policy Optimisation (PPO), resulting in the algorithm GCR-PPO.

GCR-PPO enables reinforcement learning agents to learn from many reward signals simultaneously by resolving conflicts between their gradients. This prevents important objectives from being suppressed during scalarisation and improves learning stability and performance when many objectives are present.

Our key novel contributions are:

- Prioritisation-based gradient resolution, extending PCGrad [6] to respect the robotics-specific distinction between task objectives and regularisers.
- A lightweight multi-head critic architecture for efficient

[1]Humphrey Munn, Peter Böhm, and Marcus Gallagher are with the School of Electrical Engineering and Computer Science, University of Queensland, QLD. 4072, Australia {h.munn,p.bohm,marcusg}@uq.edu.au
[2]Humphrey Munn, Brendan Tidd, Peter Böhm, and David Howard are with {humphrey.munn,brendan.tidd,peter.bohm, david.howard} @data61.csiro.au

per-reward advantage estimation that integrates directly into standard PPO pipelines.
- Large-scale comparisons against massively parallel GPU-based PPO on challenging robotic RL benchmarks, including custom multi-objective tasks.

Through a series of experiments, we show that GCR-PPO scales better than PPO with the number of objectives, enabling more complex behaviours. For example, Humanoid Running discovers a stylised fast gait with long strides at a low base height. In contrast, massively-parallel PPO typically solves only the fundamental running objective and fails to realise the stylised variant. Across 13 IsaacLab tasks, GCR-PPO achieves an average +4.6% symmetric percentage change (SPC) over PPO, with higher gains for tasks exhibiting higher gradient conflict (Spearman $\rho = 0.736$, p = 0.0041). On two custom multi-objective suites, it reaches up to +542% SPC relative to massively-parallel PPO for specific Humanoid configurations. A consistent trend emerges: as more objectives are added, the PPO baseline is prone to poor convergence points that optimise a single term rather than the joint objective set, whereas GCR-PPO more reliably identifies and optimises a feasible subset of objectives jointly, reducing conflicts over training.

## II. RELATED WORK

Managing and integrating potentially conflicting behavioural goals is a central challenge in robot control [7], [8]. In robotic reinforcement learning, objective functions must balance task success with behavioural regularisation such as safety and realism [9], [10]. However, most RL pipelines optimise a single scalar reward, creating a gap between efficient single-objective optimisation and the richer reward structures required for complex behaviour.

### A. Classical Control and Task Prioritisation

Early solutions to conflicting objectives in robotics were developed in classical control, most notably prioritised control and stack-of-tasks methods [11], [12].

These approaches project lower-priority objectives into the null space of higher-priority ones, ensuring that secondary goals do not compromise critical tasks. They have been successfully applied in humanoid and whole-body control, where accurate system models and strict task hierarchies are available. However, these methods depend on exact system models and manually specified task hierarchies, which limit their applicability in modern reinforcement learning.

In RL, objectives are encoded implicitly through reward terms rather than explicit priorities, and interaction is often model-free with noisy gradient feedback. RL is therefore often preferred in high-dimensional and uncertain environments where rigid model-based prioritisation is impractical [13]. Nevertheless, resolving conflicts through projection has influenced more recent gradient-based methods [6], [14], [15] in learning-based control, and underpins our approach to prioritising objectives defined by reward terms.

### B. Single-Objective Reward Combination in Reinforcement Learning

In most reinforcement learning applications for robotics, multiple behavioural objectives are combined into a single scalar reward and optimised using standard algorithms such as PPO [16] with generalised advantage estimation (GAE) [17]. This scalarisation simplifies optimisation but implicitly assumes that reward components are aligned, obscuring conflicts between objectives. Several techniques improve stability in scalar-reward settings. PopArt [18] rescales value predictions to address reward-scale imbalance, while auxiliary-task frameworks such as UNREAL [19] and distributed architectures like IMPALA [20] improve representation learning and sample efficiency. However, these methods do not directly address interference between competing reward components. Reward shaping [21] highlights the importance of reward design but operates at the level of reward construction rather than optimisation. As a result, scalarised RL pipelines provide limited mechanisms for resolving gradient-level conflicts when multiple objectives must be satisfied simultaneously.

### C. Multi-Objective Reinforcement Learning

Multi-objective reinforcement learning (MORL) [10] extends the standard RL setting to optimise multiple, potentially conflicting objectives simultaneously [22], [23]. A typical formulation seeks policies that approximate the Pareto front, i.e., the set of policies that are not dominated in all objectives.

Exact Pareto-optimal methods provide principled guarantees but are computationally intractable in high-dimensional problems [23]. Practical approximations include convex combinations of objectives and gradient-based approaches such as MGDA [24], though these remain costly in high-dimensional settings like robotic RL due to repeated optimisation of vector-valued value functions or constrained subproblems.

Many MORL methods adopt scalarisation strategies to improve scalability, where a weighted sum of objectives is optimised [25]. While computationally efficient, scalarisation hides conflicts and requires carefully chosen weights, which may not generalise across tasks or environments. Other approaches explore decomposition methods, such as learning multiple value functions for each objective [26], but these remain limited by sample complexity and optimisation overhead.

### D. Multi-Task Reinforcement Learning

Multi-task reinforcement learning (MTRL) aims to train a single policy to solve multiple tasks, typically by sharing representations across tasks [27]. While multi-objective RL (MORL) focuses on optimising several reward terms within one task, MTRL treats each task as a separate objective instead.

Recent methods take heuristic approaches: PCGrad [6] projects gradients to reduce interference and shows substantial empirical gains, though without theoretical con-

vergence guarantees. Conflict-averse gradient descent (CA-Grad) [28] adds convergence guarantees at the cost of greater computational complexity. In contrast, our approach extends the lightweight PCGrad framework with priority-based projections tailored to the robotics setting, where task objectives and regularisers must be distinguished. Recent work has also explored projection-based gradient methods directly in reinforcement learning for robotics. PEGrad [29] projects the gradient of an energy minimisation objective orthogonally to the task reward gradient to obtain energy-efficient policies without degrading task performance. While effective for balancing a fixed pair of objectives (task and energy), their approach targets a specific reward structure and two-objective setting. In contrast, our method addresses general multi-objective RL with many reward components by resolving gradient conflicts using priority-based projections within PPO, and employs a multi-headed critic to efficiently estimate per-objective advantages when scaling to larger numbers of objectives.

### E. Connections Between Multi-Task and Multi-Objective RL

In multi-task reinforcement learning, gradients from several distinct tasks are aggregated to update a single policy [27]. Additive-reward RL can be seen as a related formulation: summing reward components into a scalar corresponds to optimising multiple objectives, but with all components sharing the same state–action trajectories rather than distinct task distributions. This difference aside, both settings face similar optimisation challenges, notably destructive interference from conflicting gradients. In additive-reward RL, these conflicts are often hidden by scalarisation.

Recent multi-objective RL literature highlights how scalarisation masks such challenges. Vamplew et al. [30] show that scalarisation can lead to suboptimal policies due to value function interference when heterogeneous rewards are collapsed into a single utility. Dann et al. [31] demonstrate the benefits of treating multiple rewards explicitly rather than averaging them into a single signal.

A related work is CoMOGA [32], which frames CMORL updates as constrained sub-problems to aggregate gradients conflict-free with safety guarantees, yielding tabular guarantees and Pareto coverage. In contrast, we address single-task additive rewards in PPO, resolving conflicts with lightweight priority-based PCGrad projections aimed at scalability and on-policy practicality rather than constrained Pareto sets.

Motivated by these observations, we explicitly decompose the scalar objective into per-component losses and gradients using a multi-head critic architecture. This decomposition makes hidden conflicts observable and allows gradient-surgery techniques from multi-task optimisation (e.g., projection-based conflict resolution) to be applied in the multi-objective setting.

## III. METHOD

We implement GCR-PPO as a lightweight modification of the publicly available RSL-RL PPO implementation [33],

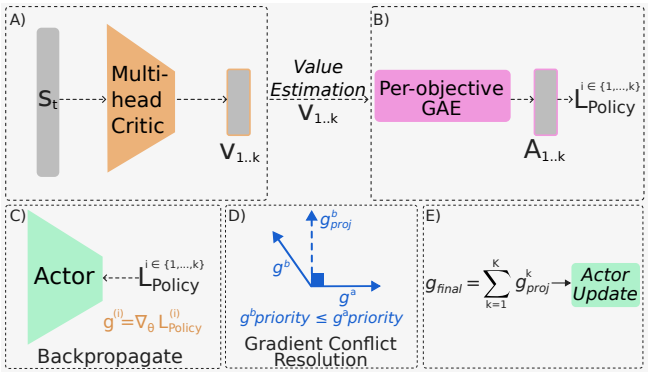


Fig. 2: GCR-PPO assumes an additive reward decomposition $r = r_0 + \cdots + r_k$, each term obtains a surrogate loss, and resolves gradient conflicts for scalability to multi-objective settings. (A) A multi-headed critic estimates values for each reward term. (B) These are processed with GAE and standardised (preserving relative scale) to yield per-term advantages. (C) Surrogate losses are computed, and (D) their gradients are projected with PCGrad so conflicts do not weaken higher-priority terms. (E) The projected gradients are aggregated into a single update for the actor.

preserving the standard PPO training loop and hyperparameters while adding only a multi-head critic and per-component gradient projection in the actor update. The entropy coefficient is tuned as described in Section IV-A.

### A. Reinforcement Learning Formulation

We consider a discounted Markov Decision Process (MDP)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma),$$

with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel $P(s_{t+1} \mid s_t, a_t)$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount factor $\gamma \in (0, 1)$. A stochastic policy $\pi_\theta(a_t \mid s_t)$ maximises the expected discounted return

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{\infty} \gamma^t r_t\Big].$$

We assume an additive per-step reward

$$r(s_t, a_t) = \sum_{k=1}^{K} r(s_t, a_t)^{(k)},$$

where $r_t^{(k)}$ are reward components specified by the practitioner. This construction is common in robotics, often including terms for task completion or success, and regularising terms to control efficiency.

We partition the reward components into two disjoint categories via index sets $\mathcal{I}_\mathcal{T}$ (task-based) and $\mathcal{I}_\mathcal{R}$ (regulariser-based), with

$$\mathcal{I}_\mathcal{T} \cup \mathcal{I}_\mathcal{R} = \{1, \dots, K\}.$$

In practice, this partition follows standard reward design conventions in robotics: task-based components correspond to objectives defining task success (e.g., goal tracking, forward velocity), while regularisers capture stabilising or efficiency constraints (e.g., torque penalties or joint-limit costs). This separation is typically explicit in reward design for legged locomotion and manipulation tasks.

For each component, define the discounted return

$$G_t^{(k)} = \sum_{t'=t}^{\infty} \gamma^{\,t'-t} r_{t'}^{(k)}.$$

Then the total return and objective decompose as:

$$G_t = \sum_{k=1}^{K} G_t^{(k)}, \qquad J(\pi_\theta) = \sum_{k=1}^{K} J^{(k)}(\pi_\theta),$$

where

$$J^{(k)}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\big[G_0^{(k)}\big].$$

This formulation yields a value function for each reward component. With a minor adaptation of generalised advantage estimation (GAE) [34] (Section III-B), we compute component-wise advantages and the corresponding surrogate-policy gradients per objective. These gradients enable the gradient-resolution mechanisms introduced in Section III-D.

### B. Multi-head critic and component-wise advantages

We parameterise a vector-valued critic $\mathbf{V}_\phi : \mathcal{S} \to \mathbb{R}^K$ with $k$-th head $V_\phi^{(k)}(s)$ predicting the discounted value of the $k$-th component $r^{(k)}$, i.e.

$$V_\phi^{(k)}(s_t) \approx \mathbb{E}\left[\sum_{\ell=0}^{\infty} \gamma^\ell\, r_{t+\ell}^{(k)} \;\middle|\; s_t\right], \qquad k = 1, \ldots, K.$$

We train all heads jointly by least squares against bootstrapped per-component targets $\hat{G}_t^{(k)}$:

$$\mathcal{L}_V(\phi) = \tfrac{1}{2}\,\mathbb{E}\left[\sum_{k=1}^{K} \big(\hat{G}_t^{(k)} - V_\phi^{(k)}(s_t)\big)^2\right].$$

Component-wise Generalised Advantage Estimation (GAE) is computed with the per-component temporal-difference residuals

$$\delta_t^{(k)} = \tilde{r}_t^{(k)} + \gamma(1-d_t)\, V_\phi^{(k)}(s_{t+1}) - V_\phi^{(k)}(s_t),$$

where $d_t \in \{0,1\}$ indicates termination. For a trajectory segment of length $T$, the component-wise advantages are

$$A_t^{(k)} = \sum_{\ell=0}^{T-1-t} (\gamma\lambda)^\ell \left(\prod_{j=0}^{\ell-1}(1-d_{t+j})\right) \delta_{t+\ell}^{(k)}, \quad k = 1, \ldots, K,$$

and the corresponding TD($\lambda$) targets used in $\mathcal{L}_V$ are

$$\hat{G}_t^{(k)} = A_t^{(k)} + V_\phi^{(k)}(s_t).$$

Stacking across components yields the advantage vector $\mathbf{A}_t = [A_t^{(1)}, \ldots, A_t^{(K)}]^\top$.

### C. Advantage normalisation strategy

Directly summing the raw per-component advantages $\{A_t^{(k)}\}_{k=1}^K$ will lead to imbalance, as reward terms vary in magnitude. Standard PPO implementations—such as Stable Baselines3 [35], RSL_RL [33], and skrl [36]—address this by standardising advantages within each batch to unit variance. However, applying this procedure independently to each component would eliminate their relative scale in the multi-component setting. To avoid this, we adopt a normalisation strategy that preserves inter-component ratios while ensuring the combined advantage has unit variance.

We centre each component advantage and apply a global scale so the summed advantage has unit variance while preserving component ratios:

$$\hat{\mathbf{A}}_t = \frac{\mathbf{A}_t - \mu}{\sqrt{\mathbf{1}^\top C \mathbf{1} + \varepsilon}} \tag{1}$$

where

$$\mu = \frac{1}{N}\sum_{t=1}^{N} \mathbf{A}_t, \qquad C = \frac{1}{N-1}\sum_{t=1}^{N} (\mathbf{A}_t - \mu)(\mathbf{A}_t - \mu)^\top.$$

where $N$ is the total number of samples in the batch. By construction, $\mathrm{Var}\Big(\sum_{k=1}^K \hat{A}_t^{(k)}\Big) \approx 1$. This property provides a stable baseline for subsequent policy-gradient updates.

### D. Gradient resolution strategy

We use the standard PPO clipped surrogate loss for each component with clipping coefficient $\epsilon$. With importance ratio $\rho_t = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ and normalised advantage $\hat{A}_t^{(k)}$, the per-component objective is

$$\mathcal{L}_{\text{policy}}^{(k)}(\theta) = -\,\mathbb{E}\Big[\min\big(\rho_t\, \hat{A}_t^{(k)},\ \mathrm{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\hat{A}_t^{(k)}\big)\Big]. \tag{2}$$

This per-component clipping is not equivalent to clipping the aggregated objective, because gradient resolution is applied only after the individual surrogate losses are formed. In practice, the baseline clipping threshold together with PPO's KL-based learning-rate adaptation was sufficient to stabilise policy updates.

The gradient of each loss,

$$\mathbf{g}^{(k)} \;=\; \nabla_{\boldsymbol{\theta}}\, \mathcal{L}^{(k)}(\boldsymbol{\theta}), \tag{3}$$

represents the update direction associated with the $k$-th reward component. With multiple objectives, these gradient vectors can conflict—i.e., $\mathbf{g}^{(i)\top}\mathbf{g}^{(j)} < 0$ (negative cosine similarity)—so improving one objective can decrease another.

We use the Projected Conflicting Gradient (PCGrad) algorithm [6] to resolve conflicts that are detrimental to task success. Given two component gradients $\mathbf{g}^{(i)}$ and $\mathbf{g}^{(j)}$, if their inner product is negative ($\mathbf{g}^{(i)\top}\mathbf{g}^{(j)} < 0$), we project the *lower-priority* gradient onto the orthogonal complement of the higher-priority one:

$$\mathbf{g}^{(i)} \;\leftarrow\; \mathbf{g}^{(i)} \;-\; \frac{\mathbf{g}^{(i)\top}\mathbf{g}^{(j)}}{\|\mathbf{g}^{(j)}\|^2}\, \mathbf{g}^{(j)}.$$

This removes only the conflicting component, preserving non-conflicting directions and ensuring higher-priority objectives are never weakened. Unlike the original PCGrad, which treats all objectives symmetrically, our variant enforces priority by making the projection asymmetric: lower-priority gradients are adjusted, while higher-priority ones remain unchanged.

Priority is determined by reward type:

- **Task-based gradients** take precedence. If a conflict occurs between a task-based and a regulariser component, the regulariser gradient is projected onto the task-based direction.
- **Task–task conflicts** are handled symmetrically with PCGrad, ensuring neither dominates completely.
- **Regulariser–regulariser conflicts** are also handled symmetrically with PCGrad, as they do not imply conflict with the task.

The final policy update direction is the sum of the adjusted gradients

$$\mathbf{g}_{\text{final}} = \sum_{k=1}^{K} \mathbf{g}_{\text{proj}}^{(\mathbf{k})}.$$

This scheme ensures that task-related objectives drive learning, while regularisers act as soft constraints. Non-deterministic projection maintains diversity of gradient directions across mini-batches, encouraging stable yet task-focused policy improvement.

## IV. EXPERIMENTS AND RESULTS

To evaluate GCR-PPO in multi-objective settings, we consider three problem groups: the **IsaacLab Task Suite** [37], custom variants of **Full-Body Throwing** [38], and custom variants of **Humanoid Running** [37]. Different combinations of these objectives produce 213 distinct multi-objective tasks for each custom task.

### A. IsaacLab Task Suite

Thirteen IsaacLab tasks were selected, spanning manipulation and locomotion across several robot platforms. Table I reports mean return and gradient conflict levels for each task. Conflict is measured as the average number of objective gradient pairs with angle $> 90^\circ$ during training, with the average shown. Task descriptions are available at [37].

During training we observed that GCR-PPO tends to reduce the learned action standard deviation $\sigma$, affecting policy entropy. To ensure a fair comparison, the entropy coefficient $\lambda$ was tuned separately for each method and task. We sampled $\lambda$ from $[10^{-4}, 0.03]$ (log scale), evaluated three seeds for half the training iterations, and selected the best-performing value. All other hyperparameters follow the original IsaacLab configurations. Rare numerical instabilities ($\leq 2\%$ of runs) were rerun with new seeds when the reward fell more than three standard deviations below the mean.

The average symmetric percentage change (SPC) across tasks was 4.55% (Table IV), increasing to 9.5% when including the custom tasks in Section IV-B. SPC is defined as $SPC(a,b) = \frac{b-a}{\frac{1}{2}(a+b)}$, which avoids asymmetry present in standard percentage change. Because most tasks have unbounded reward scales, SPC alone may not fully reflect relative performance differences. We therefore also report win rates across tasks.

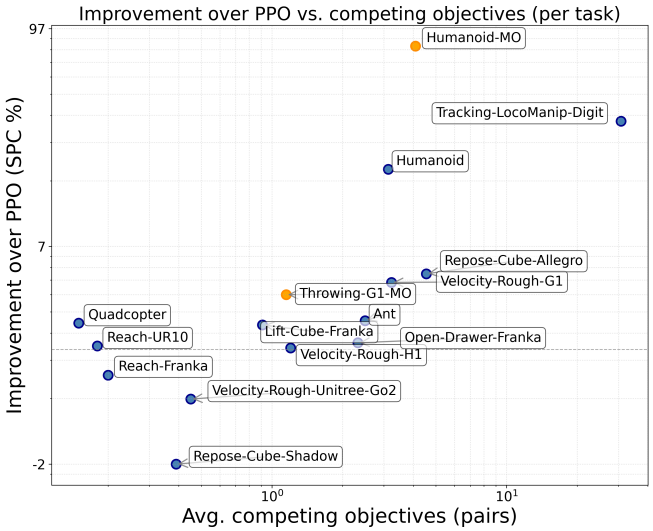


Fig. 3: Relative improvement (symmetric percentage change) of GCR-PPO over PPO, averaged across 10 seeds each, as a function of the average number of competing objective pairs per task. Blue points are standard IsaacLab tasks; orange points are our custom multi-objective variants (Section IV-B).

Using paired t-tests (95% confidence), GCR-PPO significantly improves performance on Humanoid ($p = 2.4\times10^{-4}$), Velocity-Rough-G1 ($p = 0.022$), Repose-Cube-Allegro ($p = 0.024$), and Tracking-LocoManip-Digit ($p = 0.028$), while Velocity-Rough-Unitree-Go2 shows a significant drop ($p = 0.043$). Overall, GCR-PPO wins 10 of 13 IsaacLab tasks; under a two-sided binomial test against a 0.5 null win probability, this corresponds to $p = 0.046$.

Figure 3 plots, for each task, the mean SPC improvement of GCR-PPO over PPO (averaged over 10 seeds per method) against the mean number of conflicting objective pairs measured during GCR-PPO training (also averaged over seeds). Across the 13 IsaacLab tasks, the Spearman rank correlation is 0.736 ($p = 0.0041$), indicating a strong monotonic relationship between task-level conflict and improvement. Conflict is measured as the average number of objective pairs with gradient angle $> 90^\circ$ across training. We use this metric rather than simply counting reward terms, as tasks with the same number of objectives can exhibit very different conflict levels. We do not normalise by the number of objectives, since doing so would downplay the added difficulty of resolving conflicts as the number of objectives increases.

### B. Custom Multi-Objective Benchmarks

Because IsaacLab environments are heavily tuned for PPO, we designed additional benchmarks to stress multi-objective optimisation. We extend two tasks—**Full-Body Throwing** [38] and **Humanoid Running** [37]—with additional style-based objectives. Each objective provides a

TABLE I: Per-task comparison of GCR-PPO against the PPO baseline. SPC denotes Symmetric Percent Change, and Avg. conflict is the mean number of conflicting objective pairs per run (angle $> 90°$) measured during GCR-PPO training, averaged over seeds.

| Task | Baseline Reward $\mu \pm \sigma$ | GCR-PPO Reward $\mu \pm \sigma$ | Max Reward (Baseline \| Ours) | SPC (%) | Avg. conflict |
|---|---|---|---|---|---|
| Quadcopter | 64.205 ± 1.610 | **64.911 ± 1.583** | 67.126 \| 66.446 | **1.09** | 0.15 |
| Reach-UR10 | 0.740 ± 0.006 | **0.741 ± 0.016** | 0.747 \| 0.765 | **0.13** | 0.18 |
| Reach-Franka | **0.749 ± 0.011** | 0.743 ± 0.013 | 0.759 \| 0.766 | -0.80 | 0.2 |
| Repose-Cube-Shadow | **9452.69 ± 205.21** | 9230.76 ± 410.37 | 9759.18 \| 9906.31 | -2.36 | 0.39 |
| Velocity-Rough-Unitree-Go2 | **36.84 ± 0.25** | 36.34 ± 0.65 | 37.23 \| 37.35 | -1.37 | 0.45 |
| Lift-Cube-Franka | 157.25 ± 4.54 | **158.83 ± 0.63** | 159.65 \| 159.85 | **1.00** | 0.91 |
| Velocity-Rough-H1 | 35.54 ± 0.14 | **35.56 ± 0.34** | 35.71 \| 36.01 | **0.06** | 1.2 |
| Open-Drawer-Franka | 98.73 ± 1.03 | **98.98 ± 0.63** | 99.84 \| 99.70 | **0.25** | 2.32 |
| Ant | 195.12 ± 10.13 | **197.48 ± 14.24** | 208.67 \| 214.48 | **1.20** | 2.49 |
| Humanoid | 276.58 ± 27.25 | **329.79 ± 24.79** | 309.08 \| 355.73 | **19.24** | 3.13 |
| Velocity-Rough-G1 | 46.47 ± 1.18 | **48.10 ± 1.66** | 48.50 \| 50.58 | **3.47** | 3.23 |
| Repose-Cube-Allegro | 167.70 ± 6.94 | **174.81 ± 5.91** | 178.99 \| 183.02 | **4.12** | 4.55 |
| Tracking-LocoManip-Digit | 23.45 ± 3.78 | **33.06 ± 11.36** | 29.23 \| 49.97 | **34.20** | 30.71 |

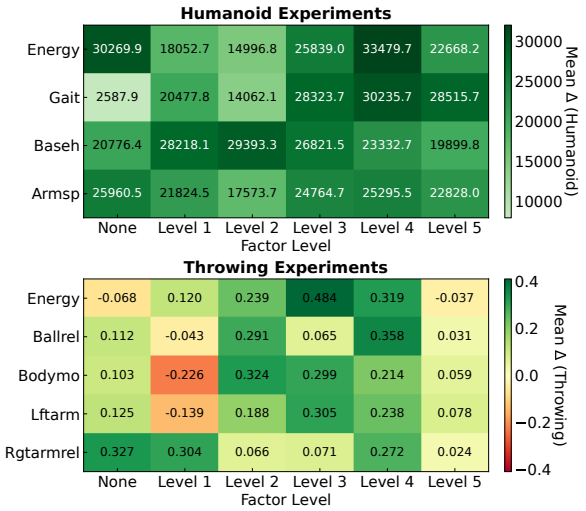


Fig. 4: Mean final reward difference (GCR-PPO - PPO) for the custom multi-objective benchmarks. Rows correspond to objectives and columns to the five threshold ranges defined in Section IV-B. Each cell reports the average performance difference across task configurations containing that objective range. 'None' indicates configurations where the objective was not included. Green indicates improvement over PPO and red indicates worse performance.

threshold-based reward: the agent receives a fixed bonus when behaviour satisfies the specified condition. Thresholds are partitioned into five non-overlapping ranges, with one randomly selected per experiment. For each objective count, 20 random subsets were sampled, producing a diverse set of task configurations. For additional task objectives, scales are set to 1 for objectives rewarded once per episode and `step_dt` for those rewarded at every step. Reward scales were set so that the reward magnitude was roughly proportional to the percentage of objective satisfaction, with additional objectives contributing at most a combined twofold increase over the original reward.

For **Full-Body Throwing** (Throwing-MO), six additional objectives were added: energy consumption, hip height, ball release timing, torso velocity at release, and left/right hand height. The implementation follows [38] with additional regularisation terms for action rate, torque, and joint acceleration.

TABLE II: Win rates (%) of GCR-PPO over PPO across objective threshold ranges for Full-Body Throwing. Values in parentheses denote the number of task configurations.

| Objective | L0 | L1 | L2 | L3 | L4 | NA |
|---|---|---|---|---|---|---|
| ENERGY | 70.6 (17) | 80.0 (10) | 90.0 (10) | 100.0 (14) | 84.6 (13) | 54.3 (35) |
| BASEH | 57.1 (14) | 90.0 (10) | 85.7 (7) | 92.3 (13) | 90.9 (11) | 63.6 (44) |
| BALLREL | 72.7 (11) | 66.7 (12) | 88.2 (17) | 66.7 (9) | 69.2 (13) | 73.0 (37) |
| BODYMO | 84.6 (13) | 75.0 (12) | 81.2 (16) | 73.3 (15) | 62.5 (8) | 68.6 (35) |
| LFTARM | 76.9 (13) | 58.3 (12) | 72.7 (11) | 66.7 (12) | 80.0 (15) | 77.8 (36) |
| RGTARM | 90.0 (10) | 72.7 (11) | 84.6 (13) | 58.3 (12) | 80.0 (10) | 69.8 (43) |

TABLE III: Win rates (%) of GCR-PPO over PPO across objective threshold ranges for Humanoid Running. Values in parentheses denote the number of task configurations.

| Objective | L0 | L1 | L2 | L3 | L4 | NA |
|---|---|---|---|---|---|---|
| ENERGY | 100 (9) | 100 (12) | 100 (11) | 100 (9) | 100 (9) | 100 (24) |
| GAIT | 100 (6) | 100 (13) | 100 (10) | 100 (6) | 100 (16) | 100 (23) |
| BASEH | 100 (12) | 100 (5) | 100 (8) | 100 (15) | 100 (8) | 100 (26) |
| ARMSP | 100 (10) | 100 (8) | 100 (5) | 100 (7) | 100 (13) | 100 (31) |

For **Humanoid Running** (Humanoid-MO), four objectives were introduced: energy consumption, stride length, hip height, and arm–pelvis distance. Objectives are activated through threshold ranges sampled per experiment, ensuring comparable weighting across objectives.

Figure 3 shows average conflict and SPC improvement for the custom tasks: **Humanoid Running** (79.7% SPC, 4.09 conflicts) and **Full-Body Throwing** (2.6% SPC, 1.15 conflicts). Both lie above the IsaacLab trend, with Humanoid Running a strong outlier, likely due to reduced reward shaping optimised for standard PPO. Within-task analyses show no significant correlation between conflict and SPC, as many objective combinations are mutually infeasible (e.g., minimising energy while maximising gait). When feasible, however, GCR-PPO reduces gradient conflict over training (Figure 5).

Figure 4 and Tables II-III show returns and win-rates, with all combinations exceeding 50%. Performance depends on

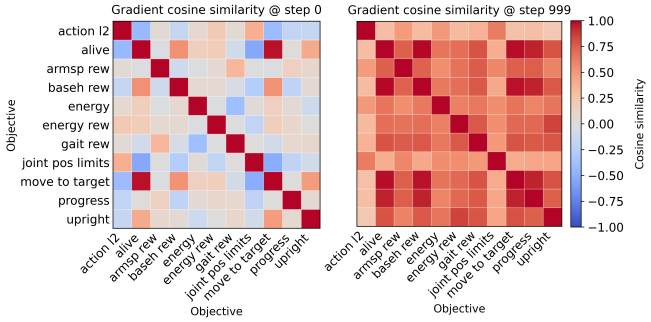


Fig. 5: Conflict evolution in the Humanoid Running custom task, measured by cosine similarity between component gradients: left at training start, right at training end. With GCR-PPO, gradient conflicts decrease over training.

objective ranges: infeasible ones (e.g., minimal energy in Throwing) hurt performance, while feasible ones (e.g., moderate base height, sufficient momentum, large gait) improve it. Some infeasible ranges (e.g., minimal-energy Humanoid Running) still outperform PPO, as GCR-PPO leverages other objectives, whereas PPO overfits the conflicting reward. For each objective we define five threshold ranges (L0–L4). Win rates are reported over randomly sampled task configurations containing that objective range; values in parentheses denote the number of configurations. NA indicates that the objective was not included in the sampled task.

### C. Ablations

We evaluate three variants against PPO: (i) a multi-head critic baseline that computes per-term GAE and surrogate losses, (ii) GCR-PPO-NoPriority, which treats all objectives equally, and (iii) GCR-PPO with task-regulariser prioritisation. Results are shown in Table IV. The multi-head critic yields a modest 2.1% average improvement on IsaacLab (7/13 tasks) and a 12.04% drop on the custom tasks. GCR-PPO-NoPriority improves over PPO on IsaacLab (4.8% SPC vs. 4.6% for GCR-PPO), but performs poorly on custom tasks—30.5% improvement on Humanoid Running versus 79.7% for GCR-PPO, and a 65.0% drop on Throwing. Its lower variance reflects policies solving the primary task but failing to satisfy added objectives. Averaging across all 15 tasks, GCR-PPO achieves the strongest overall performance.

TABLE IV: Ablation: average improvement (SPC) by method on IsaacLab, custom multi-objective (MO), and overall.

| Method | IsaacLab | Custom (Humanoid,Throwing) | Overall |
|---|---|---|---|
| Multi-head PPO | 2.1% | -12.04% | 0.21% |
| GCR-PPO-NoPriority | **4.8%** | -17.25% | 1.86% |
| GCR-PPO | 4.6% | **41.2%** | **9.5%** |

### D. Training Performance

Figure 6 reports iteration time for PPO and GCR-PPO on IsaacLab. Almost all overhead (>99%) arises from gradient projection, while per-component loss computation is negligible. Tasks with many reward terms therefore show larger slowdowns, with a worst case of 54% (≈47 minutes extra training time). The projection cost grows with the number of conflicting pairs, though potential pairs increase quadratically with reward terms. Overall, GCR-PPO improves returns with only moderate runtime overhead.

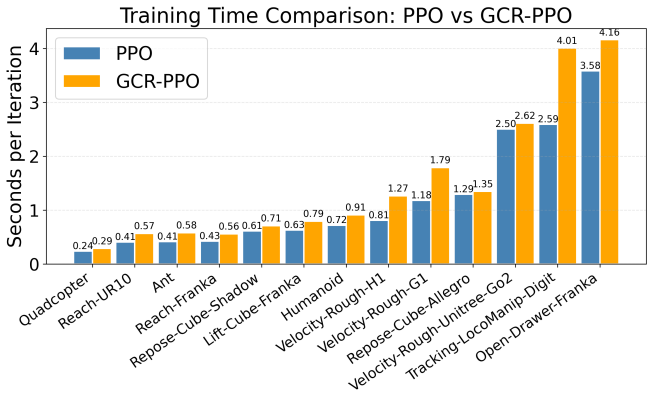


Fig. 6: Time per training iteration (s) on a single NVIDIA H100 GPU, including data collection and model updates.

### E. Qualitative Real-World Deployment

To assess whether GCR-PPO policies remain deployable on hardware, we performed qualitative real-world tests on a Unitree G1 humanoid for three tasks from the `unitree_rl_lab` library: walking, `gangnam_style`, and `dance102`. For each task, a single GCR-PPO policy was deployed repeatedly (10 trials per task). Across all three tasks, the deployed policies were stable and exhibited tracking behaviour visually comparable to or better than, the corresponding PPO policies, with no qualitative degradation in robustness observed during repeated execution. While these trials do not constitute a quantitative real-world benchmark, they indicate that the modifications introduced by GCR-PPO do not impair sim-to-real deployability in these settings.

## V. Conclusions, Discussion and Future Work

We presented GCR-PPO, an on-policy method that decomposes additive rewards into per-objective advantages and resolves conflicts via projection with task–regulariser prioritisation. Across 13 IsaacLab tasks and two custom multi-objective benchmarks, GCR-PPO improves returns and win rates over PPO, with larger gains in settings exhibiting higher conflict. The method integrates easily into PPO pipelines with modest training-time overhead.

Results highlight that some of the difficulty in multi-objective control stems from hidden interference between reward components. Exposing per-objective signals and applying lightweight conflict resolution improves stability without costly multi-objective solvers. Limitations include increased seed variability and added training time from the gradient surgery step. Our asymmetric prioritisation assumes task gradients should dominate regularisers; while empirically effective, a formal analysis of when this stabilises learning compared to symmetric projections remains an open question. Our priority scheme encodes designer intent but does not provide Pareto guarantees. Unlike many MORL formulations that approximate Pareto fronts, our focus is on scalable optimisation of additive rewards within a single

policy update, prioritising practical integration with PPO over explicit Pareto coverage.

Future work includes: (i) analysing why and where conflicts persist; (ii) extending to multiplicative or non-linear rewards; (iii) improving the efficiency of gradient resolution and developing more principled gradient resolution methods; and (iv) developing harder, less-shaped benchmarks.

In summary, GCR-PPO shows that explicit reward geometry and resolving conflicts inside PPO are practical steps toward scalable multi-objective robot learning, with opportunities to generalise to richer objectives and real-world scenarios.

## REFERENCES


[1] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 27, 2025, pp. 28 694–28 698.

[2] A. Farooq and K. Iqbal, "A survey of reinforcement learning for optimization in automation," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2024, pp. 2487–2494.

[3] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on robot learning*. PMLR, 2022, pp. 91–100.

[4] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51 416–51 431, 2021.

[5] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*. PMLR, 2020, pp. 1094–1100.

[6] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[7] P. Althaus and H. I. Christensen, "Behavior coordination in structured environments," *Advanced Robotics*, vol. 17, no. 7, pp. 657–674, 2003.

[8] P. Pirjanian, "Multiple objective behavior-based control," *Robotics and Autonomous Systems*, vol. 31, no. 1-2, pp. 53–60, 2000.

[9] S. Gu, B. Sel, Y. Ding, L. Wang, Q. Lin, A. Knoll, and M. Jin, "Safe and balanced: A framework for constrained multi-objective reinforcement learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[10] S. Huang, A. Abdolmaleki, G. Vezzani, P. Brakel, D. J. Mankowitz, M. Neunert, S. Bohez, Y. Tassa, N. Heess, M. Riedmiller *et al.*, "A constrained multi-objective reinforcement learning framework," in *Conference on Robot Learning*. PMLR, 2022, pp. 883–893.

[11] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, planning and control*. Springer, 2009.

[12] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[13] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[14] S. Guangyuan, Q. Li, W. Zhang, J. Chen, and X.-M. Wu, "Recon: Reducing conflicting gradients from the root for multi-task learning," in *The Eleventh International Conference on Learning Representations*, 2023.

[15] E. Yang, L. Shen, Z. Wang, G. Guo, X. Chen, X. Wang, and D. Tao, "Representation surgery for multi-task model merging," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 56 332–56 356.

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

[17] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *International Conference on Learning Representations (ICLR)*, 2016.

[18] M. Hessel, H. Soyer, L. Espeholt, W. M. Czarnecki, S. Schmitt, and H. van Hasselt, "Multi-task deep reinforcement learning with popart," in *AAAI Conference on Artificial Intelligence*, 2019.

[19] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *International Conference on Learning Representations (ICLR)*, 2017.

[20] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *International Conference on Machine Learning (ICML)*, 2018, pp. 1406–1415.

[21] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99. Citeseer, 1999, pp. 278–287.

[22] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[23] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker, "Empirical evaluation methods for multiobjective reinforcement learning algorithms," *Machine learning*, vol. 84, no. 1, pp. 51–80, 2011.

[24] J.-A. Desideri, "Multiple-gradient descent algorithm (mgda) for multiobjective optimization," in *International Conference on Optimization*, 2012, pp. 1–6.

[25] K. Van Moffaert and A. Nowé, "Multi-objective reinforcement learning using sets of pareto dominating policies," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3483–3512, 2014.

[26] H. Mossalam, Y. M. Assael, D. M. Roijers, and S. Whiteson, "Multi-objective deep reinforcement learning," *arXiv preprint arXiv:1610.02707*, 2016.

[27] N. Vithayathil Varghese and Q. H. Mahmoud, "A survey of multi-task deep reinforcement learning," *Electronics*, vol. 9, no. 9, p. 1363, 2020.

[28] S. Liu, E. Johns, and A. J. Davison, "Conflict-averse gradient descent for multi-task learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[29] S. Peri, A. Perincherry, B. Pandit, and S. Lee, "Non-conflicting energy minimization in reinforcement learning based robot control," *arXiv preprint arXiv:2509.01765*, 2025.

[30] P. Vamplew, C. Foale, and R. Dazeley, "Value function interference and greedy action selection in value-based multi-objective reinforcement learning," *arXiv preprint arXiv:2402.06266*, 2024.

[31] C. Dann, Y. Mansour, and M. Mohri, "Reinforcement learning can be more efficient with multiple rewards," in *International Conference on Machine Learning*. PMLR, 2023, pp. 6948–6967.

[32] D. Kim, M. Hong, J. Park, and S. Oh, "Conflict-averse gradient aggregation for constrained multi-objective reinforcement learning," *arXiv preprint arXiv:2403.00282*, 2024.

[33] C. Schwarke, M. Mittal, N. Rudin, D. Hoeller, and M. Hutter, "Rsl-rl: A learning library for robotics research," *arXiv preprint arXiv:2509.10771*, 2025.

[34] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[35] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of machine learning research*, vol. 22, no. 268, pp. 1–8, 2021.

[36] A. Serrano-Munoz, D. Chrysostomou, S. Bøgh, and N. Arana-Arexolaleiba, "skrl: Modular and flexible library for reinforcement learning," *Journal of Machine Learning Research*, vol. 24, no. 254, pp. 1–9, 2023.

[37] NVIDIA Isaac Lab Project Developers, "Isaac lab documentation," https://isaac-sim.github.io/IsaacLab/, 2025, accessed: Sep. 15, 2025.

[38] H. Munn, B. Tidd, P. Böhm, M. Gallagher, and D. Howard, "Whole-body dynamic throwing with legged manipulators," *arXiv preprint arXiv:2410.05681*, 2024.