基于深度强化学习的 LEAP Hand 单轴手内方块重定向技术报告

Technical Report for In-Hand Cube Reorientation with LEAP Hand

Xuhang Ye

cquleaf@yexuhang.com

github.com/CQULeaf/leap-hand-inhand-reorientation

yexuhang.com/projects/leap-hand-inhand-reorientation

创建于: 2026 年 5 月 17 日

更新于: 2026 年 5 月 17 日

目录

1 项目概述

1.1 摘要

本报告面向作品集的展示，整理并呈现仓库 CQULeaf/leap-hand-inhand-reorientation 中 LEAP Hand 单轴手内方块重定向项目的技术细节，具体的仿真与实机效果视频见 项目展示页。项目目标是在无视觉、无触觉反馈条件下，仅依赖电机本体感知历史与上一时刻目标动作，让 LEAP Hand 在 Isaac Lab 仿真环境中驱动方块围绕 z 轴持续完成分段重定向任务，并提供从强化学习训练、checkpoint 播放到真实硬件部署的工程入口。

报告采用技术报告常见的“背景–方法–实现–评估–讨论”结构。该结构与高校技术报告和科研论文中强调目的、方法、结果与讨论的写法一致 [1, 2]，但表达上更突出项目亮点、系统边界和可复现路径。

定义 1.1: 核心设计判断

该项目的价值不在于堆叠传感器，而在于用简洁的本体感知观测、递归策略网络和渐进式域随机化，构建一条从高并行仿真训练到 LEAP Hand 硬件执行的闭环技术链路。

1.2 任务定义

本项目研究的问题是 dexterous in-hand manipulation 中的方块重定向任务。给定一个位于 LEAP Hand 掌内的立方体，策略需要持续改变手指关节目标，使物体姿态依次接近沿 z 轴旋转后的目标姿态。仓库当前实现聚焦单轴重定向，而不是任意三维姿态控制；这一选择让任务更适合展示强化学习、目标更新、奖励设计和 Sim-to-Real 鲁棒性设计之间的关系。

从控制约束看，策略没有直接使用外部相机、触觉阵列或物体真实状态作为输入。策略观测由关节位置与上一时刻目标动作组成，并保留 3 帧历史。这个约束使任务更接近低成本硬件可部署系统：真实机器人只需读取 Dynamixel 电机状态，再按照与仿真一致的归一化和关节顺序映射生成策略输入。

1.3 项目亮点

表 1: 项目技术亮点

亮点	说明
最小传感输入	策略观测仅使用关节位置和动作历史，不依赖视觉、触觉或物体状态输入。
Isaac Lab 直接环境	基于 `DirectRLEnv` 实现任务逻辑，便于在仿真循环中精确控制动作、观测、奖励与重置。
递归 PPO 策略	`rl_games` 配置采用 actor-critic MLP 与 GRU，适合利用短时历史弥补部分可观测性不足。
自动域随机化	环境实现 ADR，将摩擦、质量、刚度阻尼、初始扰动、动作噪声和动作延迟逐步扩展。
部署链路完整	仓库包含训练、评估、预训练 checkpoint 播放、Python SDK 直连 Dynamixel 和 ROS2 部署入口。

2 技术背景与系统架构

2.1 LEAP Hand 与低成本灵巧操作

LEAP Hand 是一个面向机器人学习的低成本、高效、类人灵巧手平台 [3]。对于作品集项目而言，它的意义在于提供了一个真实硬件可触达的实验对象：一方面，它有足够多的自由度支持手内操作；另一方面，它的电机状态读取和位置控制接口可以被工程化地接入策略部署脚本。

本项目没有重新设计硬件，而是围绕 LEAP Hand 右手 USD 资产、Isaac Lab 任务注册和 Dynamixel 控制链路完成软件系统集成。仓库通过 `assets/leap_hand_v1_right` 管理仿真资产，通过 `deployment_scripts` 管理硬件部署逻辑，使“仿真中的关节顺序”和“实机电机顺序”可以在同一代码库中显式对齐。

2.2 Isaac Lab DirectRLEnv

Isaac Lab 提供了面向强化学习任务的环境抽象，其中 `DirectRLEnv` 适合将奖励、观测、动作应用和重置逻辑直接写入环境类 [4]。本项目的核心环境为 `ReorientationEnv`，配置类为 `LeapHandEnvCfg`，任务通过 Gymnasium id `Isaac-Reorient-Cube-Leap` 注册。

相比完全依赖 manager-based 配置的任务，直接环境实现更利于展示本项目的控制细节：目标姿态更新、方块跌落判断、动作延迟、外力扰动和 ADR 更新都能在同一环境类中被追踪。

2.3 训练、评估与部署链路

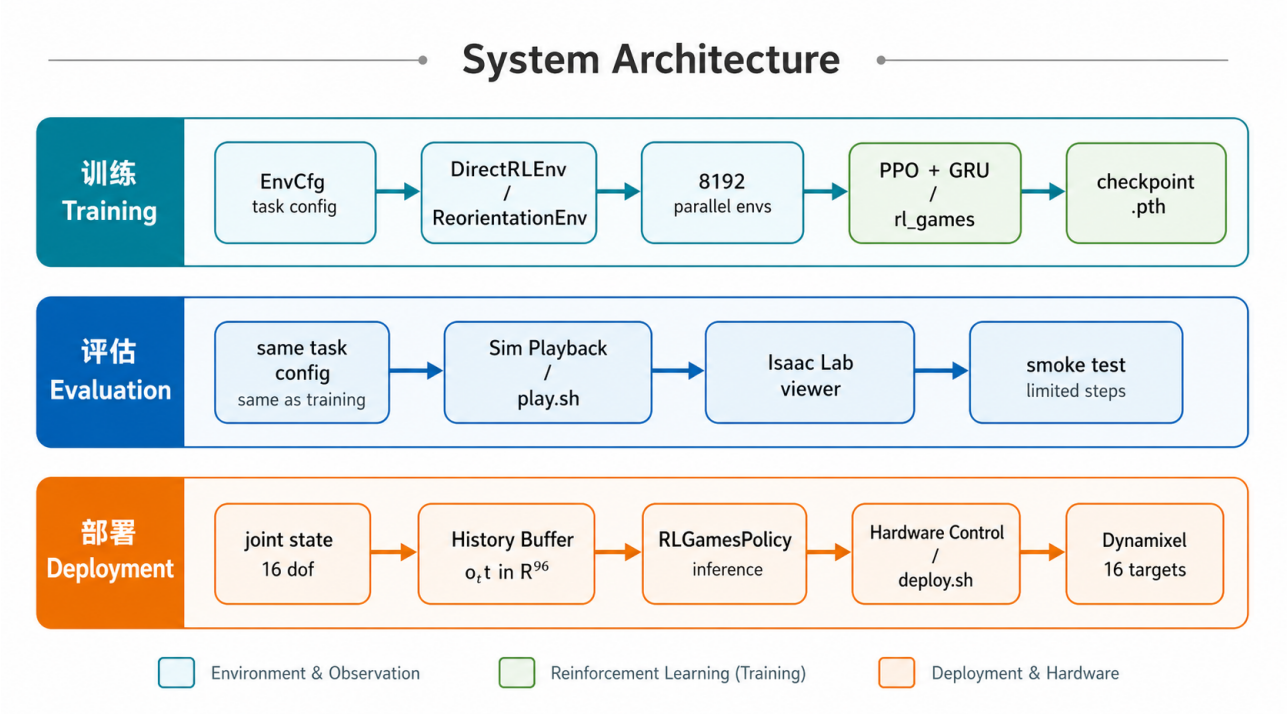

图 1: 项目系统架构：同一任务配置支撑训练、评估和部署。

如图 1 所示，项目可以分成三条链路：训练链路在 Isaac Lab 中启动环境并使用 `rl_games` 更新策略；评估链路加载 checkpoint 并在仿真中播放策略；部署链路调用硬件控制脚本，完成模型加载、观测构造、动作输出和 Dynamixel 电机命令发送。

2.4 工程目录分层

仓库目录按照“入口脚本–Isaac Lab 扩展–资产–工具–部署”分层组织。这样的结构让新读者可以先看 README 和 shell 入口，再进入 `source/LEAP_Isaaclab` 理解任务实现。

表 2: 主要工程模块

路径	职责
`scripts/train`	训练入口，封装本地 profile、checkpoint 恢复和迭代数设置。
`scripts/eval`	仿真播放入口，默认加载预训练权重并支持有限步 smoke test。
`scripts/deploy`	实机部署包装脚本，提供 dry-run、串口、电流限制和步数上限参数。
`source/LEAP_Isaaclab/tasks`	Gym 任务注册、环境配置和 `DirectRLEnv` 实现。
`source/LEAP_Isaaclab/utils`	ADR、观测处理、LEAP Hand 工具函数和 Dynamixel 客户端。
`pretrained`	存放项目默认 checkpoint。

3 任务建模与环境实现

3.1 MDP 抽象

从强化学习角度看，该任务可以被建模为部分可观测马尔可夫决策过程。真实状态包含机器人关节状态、物体位姿、接触状态和动力学参数；但策略实际看到的是经过归一化的关节位置和上一时刻目标动作历史。因此，项目使用 3 帧历史和 GRU 网络来补偿缺失的物体状态与接触状态。

形式上，可将任务写作如式 (1) 所示：

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, P, r, \gamma \rangle, \tag{1}$$

其中 $\mathcal{S}$ 包含手部关节状态、物体位姿和接触动力学参数，$\mathcal{A} \subset [-1,1]^{16}$ 为 16 维连续动作空间，$\mathcal{O}$ 是策略实际可见的本体感知观测空间。由于物体位姿不进入策略输入，策略学习的是 $\pi_\theta(a_t \mid o_t, h_t)$，而不是直接依赖完整状态的 $\pi_\theta(a_t \mid s_t)$。

表 3: 任务建模摘要

项目	当前实现
任务 id	`Isaac-Reorient-Cube-Leap`
环境基类	Isaac Lab `DirectRLEnv`
并行环境数	默认 `8192`
仿真步长	`dt = 1/120`，动作 decimation 为 `4`
动作空间	16 维，对应 LEAP Hand 16 个可控关节
观测空间	96 维，来自每帧 32 维观测乘以 3 帧历史
单帧观测	16 维归一化关节位置 + 16 维当前目标关节位置
目标更新	每次成功后目标姿态沿 z 轴前进 $2\pi/16$

3.2 动作接口

环境支持相对动作和绝对动作两种模式，当前配置使用 `relative`。在相对动作模式下，策略输出先被裁剪到 $[-1,1]$，再乘以移动平均系数 `1/24`，叠加到上一时刻目标关节位置上，最后根据关节上下限饱和裁剪。这样的接口把策略输出解释为小幅关节目标增量，有助于保持动作平滑，也更贴近实机位置控制。

相对动作可写为式 (2)：

$$u_t = \mathrm{clip}\left(u_{t-1} + \alpha a_t,\ q_{\min},\ q_{\max}\right), \qquad \alpha = \frac{1}{24}, \tag{2}$$

其中 a_t 是策略输出，u_t 是发送给关节位置控制器的目标，$q_{\min}$ 和 $q_{\max}$ 来自仿真关节限位。

Listing 1: 相对动作更新的核心逻辑

```
targets = prev_targets[:, actuated_dof_indices] + act_moving_average * actions
cur_targets[:, actuated_dof_indices] = saturate(
    targets,
    hand_dof_lower_limits[:, actuated_dof_indices],
    hand_dof_upper_limits[:, actuated_dof_indices],
)
```

部署脚本中保持了同样的动作解释方式：`action_scale = 1/24`，`action_type = "relative"`，并在发送电机命令前完成 sim-to-real 关节顺序映射与角度坐标转换。这一点非常关键，因为训练环境和硬件控制器如果对动作语义理解不一致，checkpoint 即使能在仿真中运行，也难以安全迁移到实机。

3.3 观测构造

策略输入不包含物体真实位置和姿态，而是由手部本体感知构成。环境首先把关节位置按上下限归一化到 $[-1,1]$，再拼接当前目标关节位置，得到 32 维单帧观测。之后，环境维护一个历史缓冲区，将 3 帧观测展平为式 (3) 中的 96 维策略输入。

$$\hat{q}_t = \frac{2q_t - q_{\max} - q_{\min}}{q_{\max} - q_{\min}}, \qquad x_t = [\hat{q}_t, u_t] \in \mathbb{R}^{32}, \qquad o_t = \mathrm{vec}(x_{t-2}, x_{t-1}, x_t) \in \mathbb{R}^{96}. \tag{3}$$

Algorithm 1: BuildProprioceptiveHistory

Input: joint position q_t, previous target u_t, history buffer H_{t-1}, limits $q_{\min}, q_{\max}$

Output: policy observation $o_t \in \mathbb{R}^{96}$ and updated buffer H_t

$\hat{q}_t \leftarrow (2q_t - q_{\max} - q_{\min})/(q_{\max} - q_{\min})$ // normalize to $[-1,1]$

$x_t \leftarrow \mathrm{concat}(\hat{q}_t, u_t)$ // $x_t \in \mathbb{R}^{32}$

$H_t \leftarrow \mathrm{ShiftLeft}(H_{t-1})$

$H_t[:, \mathrm{last}] \leftarrow x_t$

$o_t \leftarrow \mathrm{Flatten}(\mathrm{Transpose}(H_t))$

return o_t, H_t

这种观测设计使策略没有“作弊式”读取物体位姿。物体状态只参与奖励、终止条件和目标判断，而不直接进入策略输入。对于作品集展示，这是一个值得强调的设计点：系统在训练时利用仿真状态提供监督信号，但部署时策略输入仍保持低成本和可实机获得。

3.4 奖励函数

奖励函数由位置约束、姿态对齐、动作正则、姿态偏差正则、成功奖励和跌落惩罚构成。方块到掌内参考位置的距离越小，奖励越高；物体姿态与目标姿态的旋转距离越小，奖励越高；动作幅度和偏离默认抓取姿态会被惩罚。当旋转误差和位置误差同时满足阈值时，环境触发目标更新并给出成功奖励。

姿态误差使用式 (4) 所示的四元数相对旋转距离：

$$d_R(R_o, R_g) = 2\arcsin\left(\left\|\mathrm{vec}\left(q_o q_g^{-1}\right)\right\|_2\right), \tag{4}$$

其中 q_o 和 q_g 分别表示物体姿态和目标姿态。总奖励可概括为式 (5)：

$$r_t = w_p\|p_o - p_{\text{hand}}\|_2 + \frac{w_R}{|d_R| + \epsilon} + w_a\|a_t\|_2^2 + w_q\|u_t - q_{\text{grasp}}\|_2^2 + b_{\text{success}} + b_{\text{fall}}. \tag{5}$$

表 4: 奖励与终止项分解

项目	配置/条件	作用
位置距离奖励	`dist_reward_scale = -10.0`	约束方块留在掌内目标区域附近。
姿态对齐奖励	`rot_reward = 1.0` `rot_eps = 0.1`	鼓励物体姿态接近当前目标姿态。
动作惩罚	`-0.0002`	抑制过大的关节目标增量。
抓取姿态偏差	`-0.3`	限制策略偏离默认抓取姿态过远。
成功奖励	`250`	当姿态和位置同时达标时推动连续重定向。
跌落惩罚	`fall_dist = 0.07, fall_penalty = -10`	方块离开掌内区域时惩罚并终止 episode。

3.5 目标更新与终止条件

单轴重定向通过连续更新目标姿态实现。环境初始化时把目标姿态设置为物体当前 yaw 的下一个 z 轴分段目标；当成功条件满足时，目标姿态继续沿 z 轴旋转 $2\pi/16$。因此，一个完整旋转被拆成 16 个局部目标，策略只需要持续完成短距离姿态推进。

目标更新可以写成式 (6)：

$$q_{k+1}^g = q_z\left(\frac{2\pi}{16}\right) \otimes q_k^g, \tag{6}$$

其中 $\otimes$ 表示四元数乘法，$q_z(\cdot)$ 表示绕 z 轴的增量旋转。

终止条件主要包括两类：方块离掌心参考位置过远，以及 episode 达到随机化后的最大长度。此外，环境还检查方块是否发生明显翻转，即物体局部 z 轴与目标局部 z 轴夹角过大。该判断避免策略通过不符合任务意图的翻转姿态获得局部奖励。

4 强化学习训练与鲁棒性设计

4.1 PPO 与递归策略网络

训练配置位于任务目录下的 `agents/rl_games_ppo_cfg.yaml`。配置使用 `rl_games` 的连续动作 actor-critic 模型，算法名为 `a2c_continuous`，并启用 PPO 训练选项。`rl_games` 是面向高性能强化学习训练的框架，Isaac Lab 通过专门的 wrapper 将仿真环境接入其向量化训练接口 [5, 6]。

网络结构由共享 MLP 和 GRU 组成。MLP 单元数为 `[512, 256, 128]`，激活函数为 ELU；RNN 使用 1 层、256 单元的 GRU，并设置 `before_mlp = true` 与 layer normalization。对于只有本体感知历史、没有物体位姿输入的任务，递归网络可以从动作和关节位置变化中隐式估计接触状态与物体运动趋势。

表 5: 训练配置摘要

配置项	当前值
随机种子	`42`
折扣因子	`gamma = 0.99`
GAE 参数	`tau = 0.95`
学习率	`3e-4`，adaptive schedule
PPO clip	`e_clip = 0.2`
rollout horizon	`horizon_length = 32`
minibatch	`32768`
mini epochs	`5`
RNN 序列长度	`seq_length = 4`
保存策略	`save_best_after = 100`，`save_frequency = 200`

4.2 高并行仿真训练

环境配置默认 `scene.num_envs = 8192`，这意味着每个训练更新可以同时收集大量交互样本。并行环境数与 GPU 物理仿真缓冲区共同决定训练吞吐。项目也提供 `--physx-smoke-buffers`、`--physx_gpu_contact_count` 和 `--physx_gpu_patch_count` 等调试参数，便于在不同显存环境下进行 smoke test。

训练入口会把环境配置和 agent 配置 dump 到日志目录：

Listing 2: 训练入口示例

```
bash scripts/train/stage1.sh --profile local-5060

bash scripts/train/stage1.sh \
  --checkpoint pretrained/leap_hand_reorient.pth \
  --max-iterations 1200 \
  --run-name resume_reorientation
```

这种日志组织方式对复现实验尤其重要：同一个 checkpoint 是否可比较，取决于当时的环境参数、训练超参数和随机化范围是否一致。

4.3 自动域随机化

项目实现了 Automatic Domain Randomization, ADR。ADR 的核心思想是：训练初期使用较窄的动力学范围，策略达到一定重定向速度后，再逐步增加随机化强度。当前配置设置 `num_increments = 25`，每次满足条件后增加一档参数范围。

表 6: ADR 参数范围

随机化对象	范围/终值	目的
机器人摩擦	静摩擦、动摩擦固定为 `1.0`，恢复系数上界到 `0.5`	覆盖接触反弹差异。
关节刚度阻尼	刚度 `2.5-3.1`，阻尼 `0.05-0.15`	降低对单一执行器参数的依赖。
物体摩擦	`0.3-1.5`	增强不同表面接触条件下的鲁棒性。
物体质量	`0.9-1.3` 倍缩放	覆盖物体质量估计误差。
物体尺寸	pre-startup 缩放 `1.1-1.25`	让策略适应不同方块尺寸。
初始位姿	平移扰动最高 `0.01 m`，roll/pitch 最高 `0.1 rad`	增强重置状态多样性。
动作噪声	`0.1-0.2`	模拟控制命令误差。
动作延迟	最高 3 个 policy step	缓解仿真与硬件控制延迟差异。
外力扰动	最大线加速度 `0.5-5.0`	模拟不可预测扰动。

例子 4.1: 为什么 ADR 适合这个项目

手内操作高度依赖接触动力学。真实硬件上的摩擦、延迟、质量和电机响应很难与仿真完全一致。ADR 把这些不确定性从固定超参数改成逐步扩张的训练分布，使策略先学会基本技能，再逐步面对更难的物理变化。

4.4 噪声、延迟与外力扰动

动作噪声在物理步预处理阶段注入，动作延迟通过动作历史缓冲实现，外力扰动在动作应用阶段按固定步频重采样并施加到物体上。三者分别模拟控制命令不精确、系统响应滞后和外界接触扰动。

观测延迟配置当前范围为 `0.0-0.0`，即目前该项目没有启用观测延迟随机化，说明当前鲁棒性重点放在动作侧、物理参数侧和初始状态侧。若后续真实硬件日志显示状态读取存在明显延迟，可以把观测延迟作为下一轮训练的扩展项。

4.5 Sim-to-Real 接口

部署脚本中的 `LEAPHandController` 负责把仿真策略接到真实电机。它完成四件事：加载 `rl_games` checkpoint，读取并归一化关节位置，维护与仿真一致的 3 帧观测历史，将策略动作转换成目标关节位置并发送到 Dynamixel。

实机部署默认控制频率为 `30 Hz`，默认 P/D gain 分别为 `800` 和 `200`，默认电流限制为 `500 mA`。

5 评估、部署与复现

5.1 现有评估入口

仓库已经提供预训练 checkpoint，并在 README 中给出仿真播放命令。

Listing 3: 仿真播放与 smoke test

```
bash scripts/eval/play_stage1.sh --num-envs 1

bash scripts/eval/play_stage1.sh \
  --num-envs 1 \
  --max-steps 20 \
  --headless \
  --physx-smoke-buffers
```

有限步数播放命令适合作为迁移或环境配置后的 smoke test：它可以验证 Isaac Lab 能启动、任务能注册、checkpoint 能加载、策略和环境 step 链路能跑通。

5.2 真实硬件部署

真实硬件部署路径通过 `scripts/deploy/reorient_z.sh` 进入。该脚本最终调用 `LEAPHandController`，可选择 dry-run 模式，也可以连接实际串口并写入电机控制参数。由于该路径会设置控制模式、扭矩状态、P/D gain 和电流限制，因此正式运行时要提前检查安全条件。

Listing 4: 离线部署链路检查

```
bash scripts/deploy/reorient_z.sh \
  --checkpoint pretrained/leap_hand_reorient.pth \
  --device cuda:0 \
  --dry-run \
  --max-steps 5
```

Listing 5: 真实硬件命令模板

```
bash scripts/deploy/reorient_z.sh \
  --checkpoint pretrained/leap_hand_reorient.pth \
  --port /dev/ttyUSB0 \
  --hz 30 \
  --kp 800 \
  --kd 200 \
  --curr-lim 500 \
  --max-steps 600 \
  --disable-torque-on-exit
```

假设 5.1: 安全边界

真实硬件运行前应确认 LEAP Hand 固定方式、电源和电流限制、急停方案、串口设备、checkpoint 来源和最大运行步数。

5.3 复现清单

表 7: 复现路径与文件层级

目标	命令/文件	可证明内容
资产完整性	`scripts/tools/check_assets.py`	USD 资产存在且与期望 hash 一致。
任务注册	`scripts/tools/list_tasks.py`	Gym registry 中存在 LEAP Hand 任务。
仿真播放	`scripts/eval/play_stage1.sh`	checkpoint 能接入仿真环境。
训练入口	`scripts/train/stage1.sh`	训练脚本、Hydra 配置和 rl_games runner 能启动。
部署 dry-run	`scripts/deploy/reorient_z.sh`	模型加载、观测历史和动作循环不依赖电机即可检查。

5.4 当前局限

第一，策略输入不包含物体状态，这符合低成本部署目标，但也意味着系统对接触状态的估计完全依赖历史和递归网络，失败模式可能不易解释。

第二，当前任务只关注 z 轴单轴重定向。它适合作为手内操作技术展示，但距离任意三维姿态重定向还有差距。若未来扩展到三轴目标，奖励函数、终止条件、目标生成和观测历史长度都可能需要重新审视。

6 结论

本项目展示了一条完整的 LEAP Hand 手内方块重定向技术路径：在 Isaac Lab 中实现直接强化学习环境，用 `rl_games` 训练递归 PPO 策略，通过 ADR 增强接触动力学鲁棒性，并保留仿真播放与真实硬件部署接口。项目的特点是约束明确、链路完整、工程边界清楚，适合作为机器人学习方向作品集中展示“任务建模、训练策略、鲁棒性设计和部署接口”的代表项目。

参考文献

[1] The University of Melbourne. *Technical Reports*. URL: https://students.unimelb.edu.au/academic-skills/resources/reading%2C-writing-and-referencing/reports/technical-reports (visited on 05/17/2026).

[2] Carnegie Mellon University. *IMRD: The Parts of a Research Paper*. URL: https://www.cmu.edu/student-success/other-resources/resource-descriptions/imrd.html (visited on 05/17/2026).

[3] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. “LEAP Hand: Low-Cost, Efficient, and Anthropomorphic Hand for Robot Learning”. In: *Robotics: Science and Systems* (2023). URL: https://arxiv.org/abs/2309.06440.

[4] Isaac Lab Project Developers. *Isaac Lab Environment API Documentation*. 2026. URL: https://isaac-sim.github.io/IsaacLab/v2.3.0/source/api/lab/isaaclab.envs.html (visited on 05/17/2026).

[5] rl-games Contributors. *rl-games: A High Performance Reinforcement Learning Framework*. URL: https://github.com/Denys88/rl_games (visited on 05/17/2026).

[6] Isaac Lab Project Developers. *Isaac Lab RL Wrappers Documentation*. 2026. URL: https://isaac-sim.github.io/IsaacLab/v2.0.2/source/api/lab_rl/isaaclab_rl.html (visited on 05/17/2026).