PPOrphIt: PPO Robot Manipulator Morphology Design

Steve Gillet
University of Colorado Boulder Robotics
Email: steve.gillet@colorado.edu

Jay Vakil
Email: Jay.Vakil@colorado.edu

***Abstract*—This paper aims to address the cost and overactuation of robotics used in manufacturing by leveraging reinforcement learning (RL) to optimize robot morphologies to specific use cases. RL is the right tool for this job because it can optimize over the large search space of different robot morphologies, effectively exploring a large variety of iterations from all over that search space. The core methodology involves using a PPO model to iterate on the different morphologies and MuJoCo to evaluate their effectiveness in simulation. The evaluation signals will be successes, timing, and manipulation measures and these will be used to show the effectiveness of the PPO model at creating morphologies that are faster, more effective, and more efficient for particular tasks.**

***Index Terms*—reinforcement learning, robot morphology, manipulator design**

I. Introduction and Motivation

High costs, lack of tailorability, and lack of retrofitting are some of the most cited problems for the adoption of the robotics in industrial settings [1]. The most commonly used robotic manipulator in manufacturing is FANUC series of 6-DOF manipulators which cost from $17,500 to $400,000 depending on their capacity and customizations [2] [3]. Meanwhile [4] and [5] show that most manipulators are overactuated, less cost-efficient, and less effective for most tasks they are used in. We want to address this problem by creating an automation pipeline where you feed in a use case and you get out a robot specifically optimized for that use case. By matching the robot to the task we can ensure that robot can perform that task and only the parts needed for that task are used, reducing cost and increasing efficiency. Here we describe the process by which robot manipulator morphologies are optimized to particular tasks using a PPO agent.

We chose reinforcement learning because morphology design is an optimization problem over a complex, high-dimensional search space and RL can explore and learn optimal designs through trial-and-error in simulation [6]. We chose PPO because it is robust and on-policy and excels in continuous action spaces which is ideal for morphology design where there are continuous parameters like link lengths [7]. This is more effective than imitation learning because imitation learning can only use morphologies that already exist severely limiting sample space and innovation [8]. Supervised learning suffers from a very similar issue where labelled data would have to be provided which would be infeasible and there is no mechanism for exploration [9].

The core method is to use MuJoCo to use stable_baselines3 PPO implementation to iterate on the design parameters (number of links, joint types (X, Y, Z hinge, and Z actuation), and link lengths), use MuJoCo to simulate that iteration on a simple task using RRT* and numerical optimization IK for the path planning, feed the results (success, time, manipulability measure) back into the PPO model. The evaluation signals will be successes, time, and manipulability measure. Hopefully the better morphologies will be able to do the tasks more successfully, efficiently, and quicker and then we can vary the tasks slightly to find better morphologies for particular tasks. The tasks will be kept simple but varied in ways to test different types of movements like moving near the base, moving from near to far, different elevations, more linear movements. You can imagine that a lot of pick and place tasks are mostly linear and so might be largely done by linear actuators more efficiently.

The research questions we seek to answer: Can PPO be used to generate more efficient morphologies? What are the best evaluation signals to use? How does varying evaluation signals yield different results (optimizing for time yields simpler manipulators while optimizing for manipulability measure yields more complex manipulators?)?

II. Background and Related Work

Most of the research in this area codesigns morphology and control of modular, atomic robots like [10] which uses PPO, [11] which uses TD3, [12], and [13] which uses PPO and A3C for configuration and control of a reconfigurable, modular robot. More pertinent [14] uses DDQN to co-optimize morphology and control for modular robotic manipulators and [15] which uses soft actor critic for morphology and control of quadraped robots to avoid evalutating performance in sim or real to save time. Our method will focus on manipulators with simple controllers to allow us to focus on morphology design. The method will use standard 6-DOF morphology as a baseline and contrast with PPO and Dreamer generated morphologies. PPO has been used effectively in some of the relevant research and allows for searching over the continuous space of manipulator parameters instead of the discrete space of different modules. Dreamer uses a 'world model' to predict outcomes and plan ahead which improves sample efficiency which might be even more effective by reducing computation needs [16].

III. Methods

The idea is to use PPO to iteratively optimize robot manipulator morphology parameters like number of joints, joint types (hinge X/Y/Z or slide in Z), and link lengths through simulation in MuJoCo. The agent will propose designs, evaluate them on tasks using IK and RRT* path planning, and update the policy based on rewards for task success, efficiency, and manipulability.

The algorithmic structure will follow a standard RL loop: At each episode the PPO agent samples a morphology configuration from its policy (parameterized as a Gaussian distribution over continuous link lengths and discrete joint types and numbers via Gumbel-Softmax for differentiability). The configuration will be instantiated in a dynamically generated MuJoCo XML model, simulated for a pick-and-place task (ie. moving from start to goal positions with varying elevations and distances), and controlled using numerical IK (using SciPy's minimize with L-BFGS-B) for joint angles and OMPL's RRT* for path planning in joint space. Rewards are then computed and the policy is updated.

The object is to maximize expected cumulative rewards, defined as $r = w_1 \cdot s + w_2 \cdot \frac{1}{t} - w_3 \cdot c + w_4 \cdot m$, where s is task success (1 if goal reached within tolerance, else 0), t is trajectory time, c is morphological complexity in number of joints, m is the manipulability measure (joint Jacobian determinant [17]), and w_i are the tunable weights. PPO minimizes the clipped surrogate loss:

$$\mathcal{L}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \max(1-\epsilon, \min(1+\epsilon, r_t(\theta)))\hat{A}_t \right) \right] + S[\pi_\theta](s_t) - \beta \cdot \mathbb{E}_t[(\hat{V}_\theta(s_t) - V_t^{\text{targ}})^2] \quad (1)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is the advantage estimated via Generalized Advantage Estimation (GAE) [18], $\epsilon = 0.2$ bounds the ratio for stability, S is an entropy bonus, and β balances the value loss [7]. For Dreamer the world model consists of a Recurrent State Space Model (RSSM) with encoder-decoder architecture to predict latent states and rewards, allowing imagination rollouts for policy updates [16].

Architectures use multi-layer perceptrons from Stable Baselines3. The policy network has 2 hidden layers of 64 units each with tanh activation functions for actor and critic. In Dreamer the world model adds a GRU recurrent layer (256 units) for dynamics prediction.

Update rules involve collecting trajectories over N episodes, computing advantages, and performing K epochs of minibatch SGD on the PPO loss with a clip ratio of 0.2. For Dreamer, updates alternate between model fitting and actor-critic optimization on imagined trajectories.

The baseline will involved standard 6-DOF manipulators to contrast with RL's exploration. As an ablation we will test removing the manipulability term from the reward function to assess the impact on design dexterity versus simplicity.

IV. Experimental Setup

We are going to evaluate our approach using simulated pick-and-place type tasks in MuJoCo physics engine sourced from the official MuJoCo library (version 3.3.7). Tasks will include:

1) Basic pick-and-place: moving the end-effector from an average height, average distance start position (e.g., [-0.5, 1.0, 1.0]) to a similar height, similar distance goal position (e.g., [0.6, 0.75, 1.0]) with minimal obstacles
2) Linear movement: moving the end-effector in a very linear way to try and illicit more linear actuation joints (e.g., [0.25, 0.0, 0.5] to [1.25, 0.0, 0.5]).
3) Varying elevation: going from high to low or vice versa to test vertical dexterity (e.g., [0.25, 0.25, 0.0] to [-0.25, 0.25, 1.0]).
4) Held-out generalization: similar to basic but with randomized obstacle positions

Observations are parametric (joint type, link length, number of joints) and task-specific (start position, goal position, end-effector position, distance). Actions are continuous (link lengths) and discrete (joint types, number of joints) sampled via the policy network. Rewards are defined by the objective function: $r = 10 \cdot s + 5 \cdot (1/t) - 1 \cdot c + 2 \cdot m$ with episode termination on success or max steps. Episode length is fixed at 200 simulation steps (time horizon $T = 200$) with stochasticity introduced via Guassian noise on initial joint positions and dynamics to simulate real-world uncertainty.

For compute budget and reproducibility, we will use a NVIDIA GeForce RTX 4060 GPU (or CPU fallback 11th Gen Intel(R) Core(TM) i9-11900KF @ 3.50GHz) via Stable Baselines3. The expected total environment frames will be approximately 1e6 (1000 episodes $\times$ 200 steps $\times$ 5 morphologies per evaluation) which corresponds to 4-6 hours wall-clock time based on MuJoCo's 5000 steps/second benchmark on similar hardware. Experiments will use at least 3 random seeds for all training and evaluation with results logged via TensorBoard. We will report means with 95% confidence intervals ($\bar{x} \pm 1.96 \cdot \frac{s}{\sqrt{n}}$ where s is standard deviation and n is number of seeds).

Primary success metrics will be cumulative return (average reward over episodes) and success rate (fraction of episodes reaching goal within 0.01m tolerance). Secondary metrics include sample efficiency (returns over total frames plotted as learning curves) and generalization (success on held-out tasks).

V. Timeline and Milestones

W1 (Oct 27 - Nov 2): Jay sets up the simulation environment, transitioning from MuJoCo to Isaac Lab if feasible, including dynamic XML/model generation for morphologies and basic pick-and-place tasks with IK/RRT* control; success check: Simulate and validate at least 3 fixed morphologies reaching goals with ¡0.01m error in under 100 steps, owner Jay. W2 (Nov 3 - Nov 9): Steve implements the baseline using demonstrations from standard 6-DOF manipulators in the simulation env, and reproduces PPO training on a simple

fixed morphology; success check: Achieve ¿80% success rate on basic reach task with standard morphologies, and PPO showing stable reward curves over 1e5 frames, owner Steve. W3 (Nov 10 - Nov 16): Integrate PPO with morphology optimization (e.g., sampling link lengths/joint types), with Jay assisting on env wrappers for dynamic configs; success check: PPO generates and evaluates 10 novel morphologies, improving average reward by 20% over random baselines on linear motion task, owners Steve (RL) and Jay (simulation integration). W4 (Nov 17 - Nov 23): Add Dreamer comparison and manipulability ablation (w4=0), running with 3 seeds; Jay handles generalization tasks (e.g., perturbed dynamics); success check: Compare PPO vs. Dreamer on sample efficiency (frames to 90% success), and ablation showing simpler designs without manipulability term (e.g., avg. joints reduced by 1-2), owners Steve (RL experiments) and Jay (task variants). W5 (Nov 24 - Nov 30): Finalize full evaluations on all tasks including held-out splits, generate plots (learning curves, morphology examples), and complete writing; success check: All metrics reported with means/CIs, paper draft ready with visuals of optimized vs. standard manipulators, all.

VI. Risks and Mitigations

Potential risks include sparse rewards, a lot of designs will probably fail completely leading to zero feedback, we hope to mitigate this using shaped rewards (like distance to goal bonuses) and curriculum learning where we start with the simpler tasks first. Another risk is training instability in the RL model resulting from high-dimensional action spaces or inaccuracies in the world models, we hope to address this with hyperparameter sweeps via logging, regularization terms, and earling stopping based on validation rewards. Compute limits may constrain episode counts, we will handle this with initial testing, CPU fallback, trying more sample-efficient methods like Dreamer. Stochasticity may cause variance in results, we counter this with 3+ random seeds and confidence intervals. Lastly, dependency on team roles may risk delays in integration, we hope to mitigate this with weekly meetings and shared code repositories.

VII. Resources

We are using the following software libraries and simulators:

A. Simulators

MuJoCo physics engine version 3.3.7 and Isaac Lab for robot dynamics. Both are open source under Apache 2.0 license.

B. RL Libraries

Stable Baselines3 version 2.7.0 for PPO implementation which is licensed under MIT and DreamerV3 which is open source under Apache 2.0.

C. Optimization & Planning

SciPy version 1.16.2 Copyright (c) 2001-2002 Enthought, Inc. 2003, SciPy Developers. SciPy will be used for inverse kinematics. OMPL version 1.7.0 open source under BSD license for motion planning (shocking).

D. Data Processing

NumPy version 2.2.6 Copyright (c) 2005-2024, NumPy Developers. Used for array operations and reward calculations. Matplotlib version 3.10.6 Matplotlib Development Team license. Used for plotting learning curves.

E. Logging

TensorBoard version 2.20.0 licensed under Apache 2.0. Used for experiement tracking and metrics visualizations.

F. Datasets

No external datasets are used, all data is custom generated via MuJoCo/Isaac Lab.

G. Starter Code

Custom Python scripts for morphology generation and RL integration, inspired by MuJoCo tutorials (available from https://mujoco.readthedocs.io/en/stable/overview.html) and Stable Baselines3 examples (github.com/DLR-RM/stable-baselines3). All modifications are original to this project.

VIII. Ethics and Safety

No human subjects or hardware involved, all experiments conducted in simulation. We want to help advance the robotics community by ensuring our work is reproducible and all code will be made available under MIT license upon project completion. No sensitive data is used, only randomly-generated, simulation data.

IX. Evaluation Plan

We will produce the folowing plots and tables to assess and visualize the results:

A. Learning Curves

Plots of average cumulative return vs. training frames for PPO and Dreamer, with shaded 95% confidence intervals over 3 seeds with separate curves for each task to show convergence.

B. Sample Efficiency Tables

Tables comparing number of frames required to reach 90% success rate across methods (baseline, PPO, Dreamer), with means and confidence intervals. Rows for tasks, columns for methods.

C. Generalization Results

Tables of success rates on held-out tasks (dynamics perturbation with 10% mass variance) reporting means and confidence intervals. Include visualizations of optimized morphologies.

D. Pass Criteria

Methods pass if PPO/Dreamer achieves $> 85\%$ success rate on primary tasks with samples efficiency <5e5 frames, and generalization $> 75\%$ on held-out splits. Negative results (no improvement) will be reported as informative if ablation explains gaps (manipulability term boosts dexterity by gap amount).

X. Expected Results

We pose the following falsifiable hypotheses:

1) PPO generated morphologies will achieve at least 20% higher success rates and 15% lower completion times on pick-and-place tasks compared to standard 6-DOF designs due to exploration of task-specific configurations.
2) Incorporating the manipulability term in the reward function will yield designs with 25% greater dexterity (measured by manipulability index) but potentially higher complexity, testable via ablation.

Success is defined as hypothesis (1) holding true with optimized morphologies showing reduced overkill and generalization $> 75\%$ on held-out dynamics indicating RL's viability for cost-effective manipulator design. An informative negative result would occur if hypothesis (2) fails indicating reward shaping needs refinement or if RL underperforms baseline indicating probably limitations in high-dimensional morphology spaces.

XI. Contributions and Roles

Steve Gillet is responsible for the RL implementation including PPO, Dreamer, and baseline setups, reward design, ablations, and primary experiements on optimization and efficiency He owns the learning curves and sample efficiency tables and RL-specific metrics in evaluations.

Jay Vakil handles the simulation environment including dynamic model generation in MuJoCo/Isaac Lab, task definitions, IK/RRT* integration, and generalization studies with perturbations. He owns the morphology visualization plots and generalization results tables.

Collaboratively one baseline, one ablation, integration testing, and final writing.

References

[1] McKinsey & Company, "Industrial robotics: Insights into the sector's future growth dynamics," McKinsey & Company, Tech. Rep., 2019. [Online]. Available: https://www.mckinsey.com/business-functions/operations/our-insights/industrial-robotics-insights-into-the-sectors-future-growth-dynamics

[2] Standard Bots, "Fanuc robot prices: Cost, models, and buying insights in 2025," 2025. [Online]. Available: https://standardbots.com/blog/fanuc-robot-price

[3] PatentPC, "Top robotics vendors by market share & installations," September 2025. [Online]. Available: https://patentpc.com/blog/top-robotics-vendors-by-market-share-installations

[4] M. Russo, L. Raimondi, X. Dong, and D. Axinte, "Task-oriented optimal dimensional synthesis of robotic manipulators with limited mobility," *Robotics and Computer-Integrated Manufacturing*, vol. 69, p. 102098, 2021.

[5] B. He, S. Wang, and Y. Liu, "Underactuated robotics: A review," *International Journal of Advanced Robotic Systems*, vol. 16, no. 4, pp. 1–14, 2019.

[6] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

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

[8] J. M. D. Delgado and L. Oyedele, "Robotics in construction: A critical review of the reinforcement learning and imitation learning paradigms," *Advanced Engineering Informatics*, vol. 54, p. 101787, 2022.

[9] 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.

[10] J. Bhatia *et al.*, "Reinforcement learning for freeform robot design," *arXiv preprint arXiv:2310.05670*, 2023.

[11] B. Tjanaka *et al.*, "Co-optimization of morphology and behavior of modular robots via hierarchical deep reinforcement learning," in *Robotics: Science and Systems (RSS)*, 2023.

[12] A. Spielberg *et al.*, "Accelerated co-design of robots through morphological pretraining," *arXiv preprint arXiv:2502.10862*, 2025.

[13] M. Kalimuthu, A. A. Hayat, T. Pathmakumar, M. R. Elara, and K. L. Wood, "A deep reinforcement learning approach to optimal morphologies generation in reconfigurable tiling robots," *Mathematics*, vol. 11, no. 18, p. 3893, 2023.

[14] Z. Ding, H. Tang, H. Wan, C. Zhang, and R. Sun, "A modular robotic arm configuration design method based on double dqn with prioritized experience replay," *Symmetry*, vol. 16, no. 6, p. 714, 2024. [Online]. Available: https://www.mdpi.com/2073-8994/16/6/714

[15] K. S. Luck, H. B. Amor, R. Calandra, and J. Peters, "Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2019.

[16] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[17] T. Yoshikawa, "Manipulability of robotic mechanisms," *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

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