Simulation-Based Reinforcement Learning Policy Optimization for Tactile Manipulation: A Case Study on the Eyesight Hand

by

Ethan Chang

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2025

Authored by: Ethan Chang
Department of Mechanical Engineering
May 9, 2025

Certified by: Pulkit Agrawal
Associate Professor of Computer Science, Thesis Supervisor

Accepted by: Daniel Frey
Professor of Mechanical Engineering
Undergraduate Officer, Department of Mechanical Engineering

Simulation-Based Reinforcement Learning Policy Optimization for Tactile Manipulation: A Case Study on the Eyesight Hand

by

Ethan Chang

Submitted to the Department of Mechanical Engineering
on May 9, 2025 in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE

ABSTRACT

Robotic manipulation remains a complex and unsolved challenge due to the need for adaptability across diverse objects and tasks. In this work, we explore how to train effective manipulation policies using reinforcement learning in simulation for the Eyesight Hand: a novel, low-cost, tactile-enabled robotic hand. We implement a range of experiments in MuJoCo to evaluate the impact of controller types, observation spaces, reward formulations, and curriculum strategies on policy performance. Our findings highlight the benefits of delta position control, a carefully selected observation space including joint states, control vectors, object pose, and contact forces, and success-driven curriculum learning. Our study establishes baseline strategies for training robust, tactile-based policies on this in-house hardware.

Acknowledgments

I would like to thank Lars Ankile and Branden Romero for their guidance on this exciting project, as well as the fantastic members of the Improbable AI Lab for their advice in navigating this new field. I also want to thank all my friends, family, and professors who supported me throughout this journey, helping me complete my degree in the most eclectic and fulfilling way.

Thesis supervisor: Pulkit Agrawal
Title: Associate Professor of Computer Science

Contents

List of Figures

1. Introduction

Robotic manipulation is the skill for robots to interact with and control the objects around them. It's a fundamental skill in order to scale the robotics presence in our society in instances like home-caring and industrial automation. Allowing robots to manipulate objects that were originally designed for humans unlocks diverse scenarios for humans and robots to cooperate in the same space without constraints.

However, manipulation remains one of the hardest problems to solve in robotics. Unlike locomotion or navigation, manipulation requires precise control of forces and dexterous control of multiple actuators. Due to its high degree of freedom and range of grasping poses, it is difficult to deterministically decide a simple approach to tackle the range of manipulation tasks, including grasping, rotation, picking, and placing [1, 2]. Traditional rigid grippers are limited in their ability to handle objects of varying shapes, sizes, and material properties without careful planning or custom end-effectors. Even with more dexterous robotic hands, achieving human-level adaptability is difficult due to the complexity of high-dimensional control, uncertainty in object dynamics, and the difficulty of robustly perceiving contact events. Simulated training and reinforcement learning algorithms have shown promise in learning manipulation strategies, but transferring these skills to real-world hardware (the "sim-to-real" gap) [3] remains an ongoing obstacle, particularly when sensitive contact interactions are involved.

In this work, we investigate learning-based tactile manipulation on a low-cost, novel robotic hand design that is made in house with the intention of creating fast manipulation policies. We focus on training robust grasping policies in simulation using Proximal Policy Optimization (PPO) [4], aiming to leverage tactile feedback to achieve stable and adaptable object grasps. This thesis aim to understand the fundamental limitations to these simulation based training methods and provide guidelines in similar training tasks.

2. Previous Work

The goal of this project is to successfully train an adaptable policy on the in-house designed robotic hand: the Eyesight Hand. Following previous successful instances on zero-shot deployed algorithms we look into possible routes to best train on this new hardware hand, which has different features than these commercial robot hands.

2.1. Eyesight Hand

The EyeSight Hand [5] is a cost-effective, 7-DoF humanoid hand designed with careful consideration for actuation, tactile sensing, and kinematics, with a target price of under 2,500. It distinguishes itself from existing commercial humanoid hands in two key ways: first, through the incorporation of highly compliant, back-drivable actuators, and second, by integrating a novel vision-based tactile sensor. These innovations enable the EyeSight Hand to execute new movement trajectories, particularly in grasping tasks. Due to its high compliance and tactile feedback, the hand can rapidly approach and interact with surfaces, such as desks,

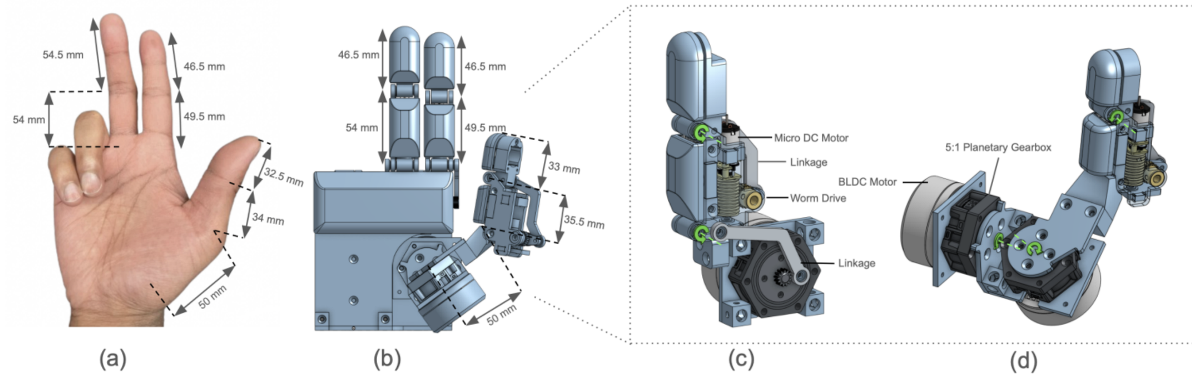

Figure 2.1: (a-b) Comparison of the dimension of a human hand with the dimensions of the EyeSight Hand. (c-d) Diagram illustrating the kinematics of the (c) index/middle finger and (d) thumb as well as the placements of the motors and transmissions. Image adapted from original paper. [5]

while allowing direct collision during object manipulation. With these enhanced capabilities, we anticipate developing novel grasping policies that prioritize fast and adaptive actions. The hand will be able to collide with objects and explore various grasp poses dynamically. The initial stages of training will be conducted in a simulated environment to understand the dynamics of the EyeSight Hand. Subsequently, we will implement sim-to-real transfer and real-world reinforcement learning to evaluate the hand's performance in grasping tasks and optimize its behavior in physical environments.

2.2. Reinforcement Policy Training with Demos

Reinforcement Learning has shown huge potential in controlling complex tasks, no matter using visual inputs or state-based inputs.[6, 7] However, training complex robotic hands using direct PPO algorithms won't lead to results that is easily generalized to diverse tasks due to reward hacking and simulation errors. To account for that, methods to expand on behavior cloning (BC) algorithms and reduce the amount of data collected is developed. This line of work explores how to combine methods that uses demonstration of tasks like Imitation Learning (IL) with Reinforcement Learning methods.

Ball et al. propose RLPD [8], a simple but effective method for incorporating offline data into online RL without needing extra regularization or pretraining steps. Their method emphasizes balanced sampling between online and offline data and introduces techniques like LayerNorm to prevent value overestimation. Hu et al. introduce Imitation Bootstrapped Reinforcement Learning (IBRL) [9], which tightly integrates an imitation learning (IL) policy with an RL agent. Instead of using demonstrations just to initialize the replay buffer, they actively involve the IL policy during both exploration and training by selecting whichever action (from IL or RL) has a higher predicted value. This approach improves training stability and sample efficiency, especially for tasks with sparse rewards. Ankile et al. present ResiP, a residual RL method for precise manipulation tasks like visual assembly. They build on a behavior-cloned base policy and use a residual policy trained with PPO to fine-tune actions for more accurate performance. This structure avoids modifying the base model directly and is particularly useful for fine corrections in high-precision tasks [10].

2.3. Physics-based Control Methods

Directly controlling all joints of a robotic hand is often unreliable or requires extra simulation time. Rather than learning end-to-end behaviors from scratch, many approaches layer reinforcement learning on top of more traditional or physics-informed control systems to improve training efficiency, safety, and sim-to-real transfer.

DextrAH-G proposes a system that combines reinforcement learning with geometric fabrics — a control framework that handles safety constraints, joint limits, and natural motion shaping [11]. A policy trained in simulation outputs high-level action goals, which are then executed by the fabric controller. This separation helps the robot learn faster and deploy safely in the real world. Their pipeline uses teacher-student distillation to transfer from privileged simulation states to a depth-only real-world policy. Geometric fabric control, as detailed by Van Wyk et al, offers a way to embed rich, constraint-aware dynamics directly into the control architecture. These second-order artificial dynamics guide the robot's behavior while still allowing the policy to issue meaningful actions, simplifying reward design and improving transfer performance [12]. UniDexGrasp takes a two-stage approach to dexterous grasping: a generative model samples diverse grasp poses, and a goal-conditioned policy executes them using point clouds and proprioception. They introduce techniques like curriculum learning to help the policy generalize across hundreds of objects. Their pipeline also uses a teacher-student structure for efficient training with realistic inputs [13].

3. Experiment Setup

The training of the eyesight hand manipulation policy is set up in the Mujoco environment, an open-sourced physics-based environment that is suitable for modeling contacts, meshes, and movements developed by Google researchers [14, 15]. The robot hand is modeled as an URDF file in the MuJoCo environment. The hand features 9 individual position-controlled joints, from the palm, with the palm having an extra 6 degrees of freedom in x,y,z, and roll, pitch, yaw; resulting in a total of 15 DOF of the hand.

As shown in Figure 3.1, the modeled eyesight hand has three fingers, each has three joints controlling the movement of the respective parts. On the thumb, the are two actuators controlling the curling of the thumb, with one actuator controlling a rotational degree of freedom orthogonal to the curling axis. On the index and middle finger, two actuators also control the curling of the joints, while one actuator allows the fingers to perform side-to-side swaying movements.

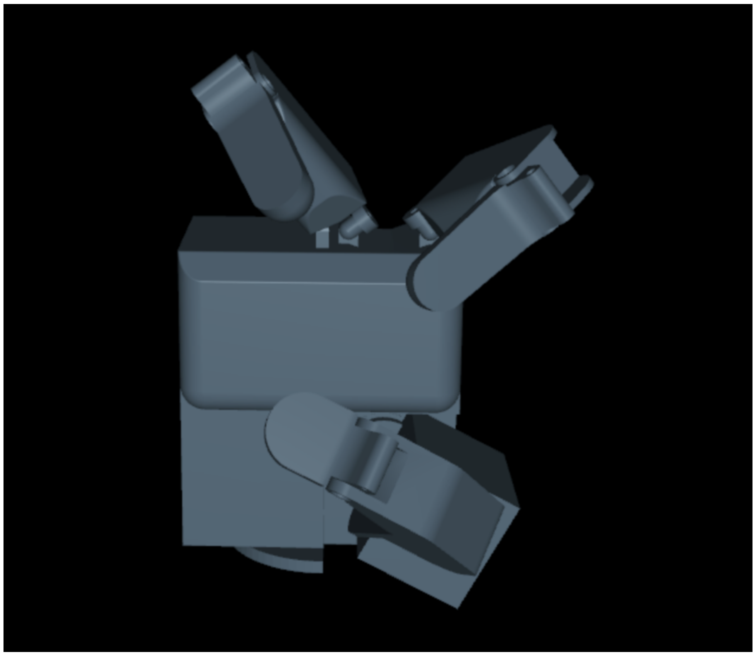

Figure 3.1: Eyesight hand modeled in the MUJOCO environment

3.1. Environment Tuning

To ensure our custom mesh reflects the contact situation in the real world, as shown in Table 3.1, the parameters are selected to reduce the timestep, increase iterations to model contact situations better after penetration is observed between the hand and the environment. We used the conjugate gradient (CG) solver, which is efficient for large, sparse systems and supports the fine tolerances needed in our setup. We also set the cone="pyramidal" option, which models friction cones as pyramids (instead of smooth cones), allowing for faster and more stable resolution of sticking vs. slipping contacts — important in tactile control scenarios where small slips need to be reliably detected.

On the physical property end of contact modeling, we configured MuJoCo's default joint and contact properties to support stable, high-fidelity tactile interactions. A light joint damping of 0.05 was applied globally to avoid oscillations without overly constraining motion, allowing responsive yet smooth finger movement. For all geometries, we used condim=3 to enable full 3D frictional contact (normal + 2 tangential directions), essential for simulating realistic stick-slip behavior in object grasps. The friction model was tuned with coefficients 3 1, providing high static friction and lower dynamic resistance to simulate rubber-like contact surfaces. Contact softness was modeled using solimp="0.99 0.99 0.001" and solref="0.0002 1", providing near-rigid contact with slight compliance to prevent chattering. These parameters were chosen after observing undesirable behaviors in simulation including oscillation, sliding and slow movements of the hand.

Table 3.1: MuJoCo physics solver configuration used in our environment, compared to typical defaults.

Parameter	Our Value	Default Value	Rationale
`timestep`	0.001 (1 ms)	0.002	Higher resolution to capture fast contact events and tactile signals.
`iterations`	150	50	More solver steps improve contact force convergence for complex grasps.
`noslip_iterations`	30	0–10	Enhances handling of frictional constraints and stick-slip behavior.
`tolerance`	1e–12	1e–5	High numerical precision ensures stability in fine contact dynamics.
`impratio`	10	1	Increases priority of contact constraints over equality constraints.
`solver`	CG	Newton / PGS	Conjugate gradient is efficient and stable for large sparse systems.
`cone`	pyramidal	elliptic	Pyramidal cones offer faster, more stable friction modeling.

Table 3.2: Default contact and joint parameter settings used in the MuJoCo environment.

Parameter	Value	Description / Rationale
`joint damping`	0.05	Low damping for stable, responsive joint movement.
`geom condim`	3	Enables full 3D frictional contact (normal + 2 tangents).
`friction`	3 1	High static / lower dynamic friction simulates rubber-like behavior.
`solimp`	0.99 0.99 0.001	Soft contact model with near-rigid compliance.
`solref`	0.0002 1	Fast contact convergence with soft spring-damper dynamics.
`margin`	1e–5	Prevents unstable contact switching near zero gap.
`contype` / `conaffinity`	1 / 1	All default geoms allowed to participate in contact.

3.2. Training Parameters

The PPO training code of this project is adapted from the cleanRL repository [16]. We used a Neural Network to convert an observation tensor to an activation tensor that acts as the policy. Each policy is trained for 20 million environment timesteps. The agent was trained across 16 parallel environments, each running episodes of 50 steps at a control frequency of 25 Hz, which is 2 seconds for each episode. The policy and value networks shared a similar architecture consisting of two hidden layers with 512 units each, using ReLU activations. We used orthogonal initialization for all layers and fixed the policy's log standard deviation (logstd=0.0).

We set the learning rate to 1e-4, with linear annealing over the course of training. The PPO hyperparameters followed standard settings: we used $\gamma = 0.99$ for the discount factor, $GAE - \lambda = 0.95$for advantage estimation, and a clipping coefficient of 0.2 to stabilize updates. Policies were updated every 1000 steps with 10 epochs per update using a single mini-batch.

3.3. Task Reduction

To make the problem more tractable for initial learning and policy development, we simplified the manipulation task to focus on fundamental grasping skills. Instead of training the policy to handle fast, reactive manipulation of arbitrary objects, we constrained the task to lifting a single block from a fixed initial position to a designated target point in space. This reduced setting eliminates the need for dynamic regrasping, object recognition, or complex trajectory planning, allowing us to focus on learning robust grasp initiation and contact stability using tactile feedback.

We also omit the robot arm extension of the hand, as we can reverse the work coordinates of the hand to eventually figure out the joint coordinates on the connected robot arm. A visual input is also omitted due to the potential need of a visual decoder. The true state of the block is feed to the model as input, which is common in the teacher-student type model training structure.

3.4. Reward Formation

Learning from DextrAH-G [11], the reward functions of the RL algorithm is formed as follows. We designed a shaped reward function that encourages the agent to lift a small block from a fixed position to a designated target in space. The primary reward component is the height reward, which provides a smooth, distance-based signal based on how close the block is to the target position. This term uses a Gaussian function to give higher reward as the block nears the goal, and is rescaled for numerical stability. We also include a lifted reward term, which penalizes the agent if the block remains in contact with the ground, encouraging upward motion. The environment includes tactile contact sensing, the contact reward is currently assigned as whether at least one finger maintains contact with the object, and can be reintroduced to promote more reliable grasping strategies. A sparse success reward is defined as an optional terminal bonus when the block reaches within 8 mm of the target.

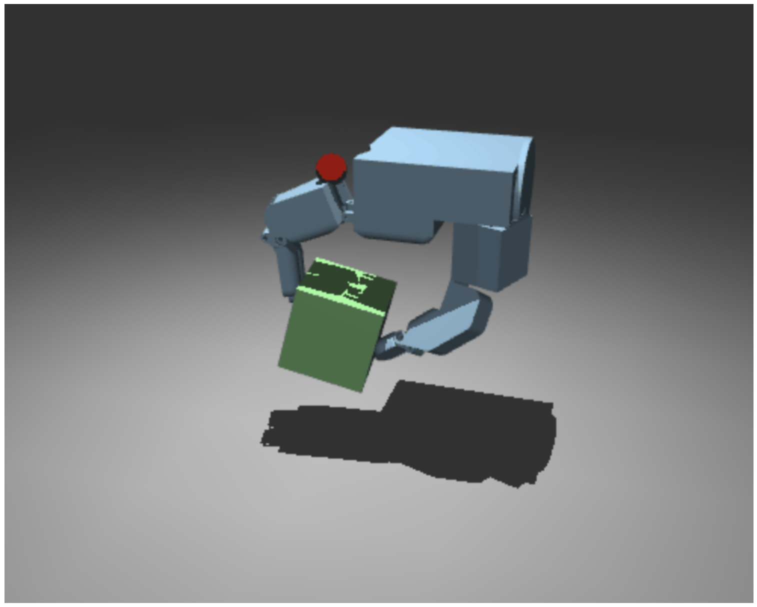

Figure 3.2: Reduced task setup for eyesight hand and a static block. The red dot indicated the target destination of the block.

The final reward is a weighted sum of these components, with most emphasis placed on progress toward the target location, supporting smooth and stable learning in this simplified manipulation task.

4. Simulation Results

More than 100 different training runs are performed to identify the most suitable training method of the eyesight hand. These are fundamental experiments aiming to understand the characteristics, training time, and performances of different setups that have shown to be adaptable on other robotic systems. While we are not able to collect accurate demonstrations for behavior cloning and imitation learning algorithms, through these ground up PPO investigations, we can identify how to best tune our algorithms if provided certain baseline Behavior Cloning algorithms, or better, how this system can adapt without demos in a generalized manner.

4.1. Controller Selection

To determine an effective action space for our policy, we experimented with several control strategies inspired by recent work on bi-level and high-frequency control. Following the framework from EvoControl [17], we compared position-based control, delta (relative) position control, and velocity-based control for issuing actions to the robotic hand. Position-based control offered simplicity but was prone to instability due to overshooting and sensitivity to noise, which results in oscillatory movements and delays in policy reflection on situations. Delta position actions provided smoother trajectories by issuing relative movement commands,

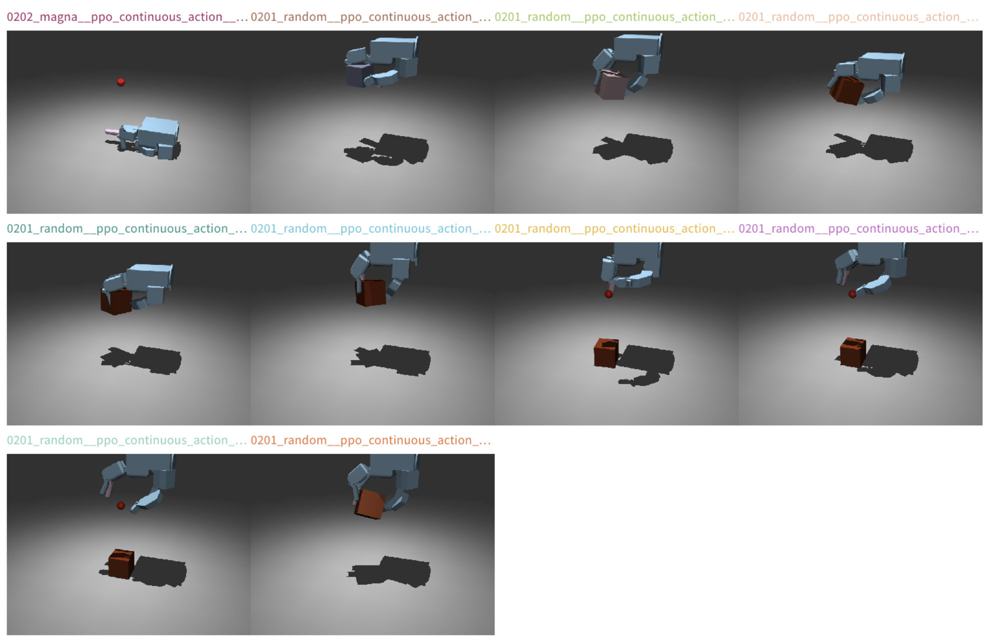

Figure 4.1: Policy is viewed after training on different types of controllers, then chosen by observing behavioral patterns.

which helped the policy generalize across different grasp postures. We also explored velocity control, but found it harder to tune and less effective in achieving reliable grasps. These comparisons revealed that delta control struck a practical balance between responsiveness and stability, making it the preferred method for our grasping setup. While we did not implement a fully learned low-level controller like in EvoControl, our choice of action space was informed by its analysis of credit assignment and expressiveness under different control frequencies.

4.1.1 Delta Control Action Mapping

The policy outputs a normalized action vector $\mathbf{a} \in [-1, 1]^n$, where each element corresponds to a control dimension (e.g., joint actuators or base movement). We use a delta position control scheme, where actions are interpreted as relative displacements added to the previous control signal. The updated control command at time t is given by:

$$\mathbf{u}_t = \mathbf{u}_{t-1} + \Delta\mathbf{a}_t \tag{4.1}$$

To ensure stability during grasping and simplify the learning problem, we apply the following preprocessing:

$$a_t^{(x)} = 0 \quad \text{(disable lateral base motion)} \tag{4.2}$$

$$a_t^{(y)} = 0 \quad \text{(disable forward/backward base motion)} \tag{4.3}$$

The final action is clipped and unnormalized using actuator-specific scaling:

$$\mathbf{u}_t = \mathbf{u}_{t-1} + \mathcal{U}\left(\text{clip}(\mathbf{a}_t, -1, 1)\right) \tag{4.4}$$

where $\mathcal{U}(\cdot)$ denotes the unnormalization function that maps normalized values to the robot's control space. This formulation allows the policy to issue smooth, bounded updates to the joint targets and has shown better stability and convergence in our experiments compared to direct position control.

4.2. Observation Space

The same selection can also be done on the input observation space for the model. We constructed the observation vector as a concatenation of multiple state components, allowing us to flexibly experiment with different combinations of inputs to the policy. The full observation space can include the robot's joint state (12*1), contact force readings from tactile sensors (3*1), torque values on each joint (15*1), the object state (7*1, including position and quaternation orientation of the block), a control vector representing the previous actuator commands (12*1), and optionally a history buffer (12*n) containing past observations. These components are normalized or scaled appropriately, such as dividing sensor forces by a constant to improve numerical stability. The final observation vector is assembled at runtime based on the configuration string and passed directly to the policy network.

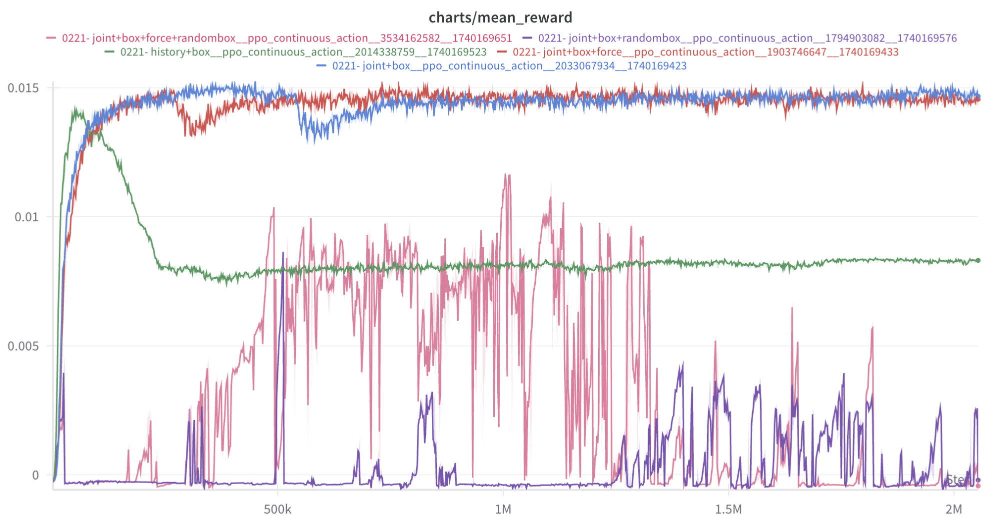

Figure 4.2: Mean reward of policy through a training episode of different observation space, higher and stable rewards are preferred.

As shown in Fig 4.2, we found that certain observation choices significantly impacted training stability and performance. Some of the observation states results in lower training rewards in multiple trails. For example, comparing the blue and green line in Fig 4.2, including a history of past observations tended to degrade performance, likely due to the increased input dimensionality and the temporal ambiguity it introduced. In contrast, using only the most recent state yielded more stable learning. Adding contact force data improved grasp robustness and helped the policy better reason about object interactions, like the pink line and purple line in Fig 4.2. Similarly, including the control vector, the previous action sent to the actuators, proved beneficial. We hypothesize that the combination of the control signal and the current joint state allows the policy to implicitly estimate whether the joints are under load or free-moving, thus improving feedback control.

Based on these findings, we adopted a final observation space composed of joint state, control vector, box state, and contact forces, which provided the best balance between learning efficiency and policy performance.

4.3. Randomization and Curriculum

To evaluate the generalization ability of the learned policy, we introduced domain randomization by perturbing both the position and size of the grasped object during training and evaluation. Specifically, the cube's spawn location and dimensions were sampled randomly at the start of each episode. This exposes the policy to a wider variety of initial conditions and object configurations, encouraging robustness to variations it might encounter in real-world deployment.

4.3.1 Progressive Randomization

To improve generalization, we progressively randomized the initial *position* and *size* of the cube throughout training. At each PPO iteration k, the sampling ranges for these parameters were scaled by a curriculum factor:

$$d_k = d_{\max} \cdot \min\left(1, \alpha \cdot \frac{k}{K}\right), \quad s_k = s_{\max} \cdot \min\left(1, \alpha \cdot \frac{k}{K}\right) \tag{4.5}$$

where:

- d_k and s_k are the sampling ranges for the cube's position and size at iteration k,
- $d_{\max}$ and $s_{\max}$ are the maximum deviation limits,
- α is the curriculum rate (set to 0.1),
- K is the total number of PPO iterations.

This ensures the policy first trains on easier configurations and gradually encounters a wider distribution of task conditions, supporting improved robustness during deployment. As shown in Figure 4.3, it is seen that randomization of box size and location leads to drastic collapse of training rewards during runs. However, the policy is able to relearn to be more general as the training continues at the cost of longer training time. Observe that the blue and red line (randomized training runs) has more fluctuation in terms of rewards and isn't still quite stable at the end of 1M timesteps. This shows that this fixed rate curriculum is too steep and requires sweeping of parameters to be adaptable.

https://www.overleaf.com/project/680fcab00147d63f13cc4e2b

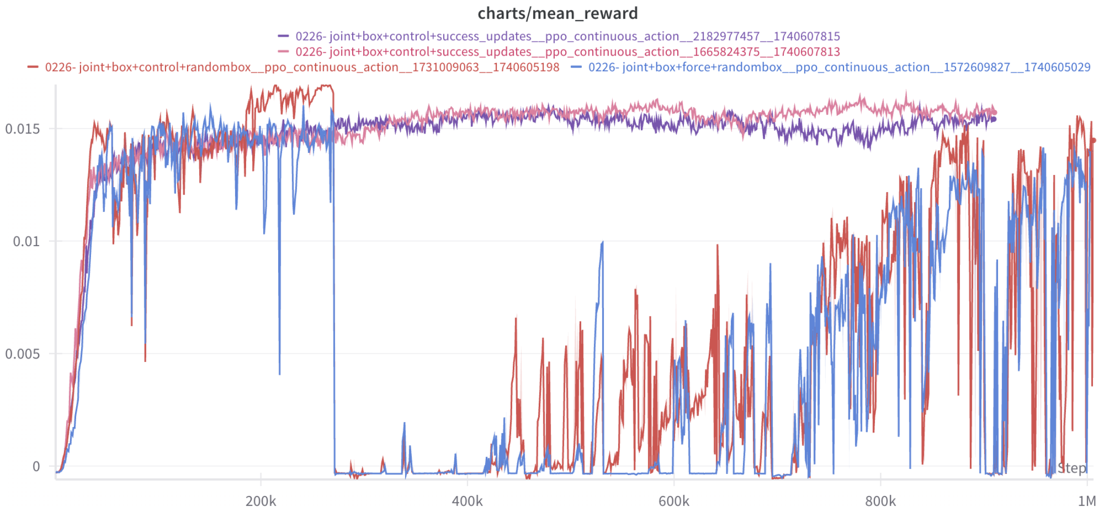

Figure 4.3: Mean reward of policy with and without randomization, pink and purple lines denote runs with a fixed size cube, while red and blue curves denote runs with randomization.

4.3.2 Success-Driven Curriculum

In addition to progressive randomization, we implemented a curriculum that adapts the task difficulty based on the agent's performance. After each training epoch, we evaluate the agent's success rate, defined as the percentage of episodes in which the block is lifted within a threshold distance of the target position. If the success rate exceeds 50%, the environment automatically increases difficulty by shrinking the object and placing it farther from the hand's starting position.

Formally, let r_k be the success rate at epoch k. If $r_k > 0.5$, the box size s_k and spawn radius d_k are updated as:

$$s_{k+1} = \max(s_{\min}, s_k - \delta_s) \tag{4.6}$$

$$d_{k+1} = \min(d_{\max}, d_k + \delta_d) \tag{4.7}$$

where:

- s_k is the cube side length at epoch k,
- d_k is the radial spawn distance from the center,
- δ_s and δ_d are step sizes controlling curriculum granularity,
- $s_{\min}$ and $d_{\max}$ define the bounds of task difficulty.

This success-driven adjustment encourages the policy to continuously adapt to harder tasks, reducing the risk of overfitting to easy training scenarios. An example of this curriculum is shown in Fig. 4.4 where the success rate and mean reward are constantly reset to a new update to block position and size, but are still able to learn a policy that adapts to the new task. Compare to the progressive curriculum with fixed rate, this method is more robust and stable, but also takes longer training time for the eyesight hand to learn a generalized policy.

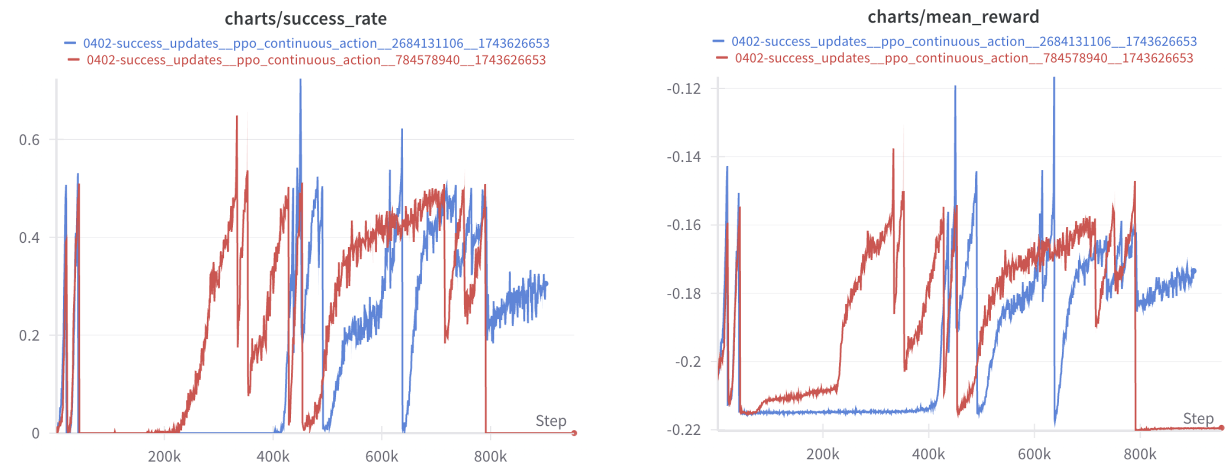

Figure 4.4: Plots of success based curriculum training, comparison of success rate side by side with reward.

5. Discussion

The goal of this study was to explore how to effectively control the Eyesight robotic hand through learning-based methods in simulation. By systematically evaluating different controller interfaces, observation spaces, reward structures, and training curricula, we aimed to identify a robust setup that could serve as a foundation for future real-world deployment.

5.1. Key Findings

Our experiments revealed several important design decisions. First, delta position control emerged as the most stable and generalizable control scheme, offering smooth behavior and easier policy learning compared to direct position or velocity control. In terms of observation space, we found that including the joint state, control vector, box state, and contact forces provided the best balance of training stability and performance. Notably, using a history buffer degraded performance, while the control vector appeared to help the policy infer actuator load indirectly. On the training side, we tested both progressive randomization and a success-based curriculum, showing that adaptively increasing task difficulty led to better generalization and more robust policies.

5.2. Limitations

While the current simulation framework allowed for rapid experimentation, it also introduced several limitations. Most notably, the tactile sensing model is a simplified proxy based on MuJoCo's contact forces, which does not fully capture the noise, latency, or spatial resolution of real tactile sensors. Additionally, the task was restricted to a single object type and a simplified lifting motion, which may not generalize to more dynamic or contact-rich tasks like in-hand manipulation. Although we explored randomization and curriculum learning, we did not evaluate policies under real-world disturbances or mechanical uncertainties, which are key to achieving reliable sim-to-real transfer. To deploy in the real world, these models needs to be adapted or to transfer on to another student model that allows input form the real-world sensors.

5.3. Future Work

Looking forward, several extensions are planned. First, we aim to collect demonstration data and explore behavior cloning or imitation-bootstrapped RL to further accelerate learning. Second, we plan to fine-tune policies on more diverse and dynamic tasks beyond the current fixed-position block lifting. Finally, we will introduce structured perturbations (e.g., sensor noise, actuation delay, mass variation) to better prepare policies for sim-to-real transfer. These steps will help bring us closer to deploying tactile-based manipulation policies on the physical Eyesight hand.

We also acknowledge that the overall training pipeline and architecture may shift significantly with the introduction of a behavior-cloning-based or foundation policy model. Such a model, pretrained on a broad range of manipulation data, could provide strong priors or even render some elements of our current training setup—such as dense reward shaping or curriculum tuning—unnecessary. Future work will need to assess how traditional control and RL techniques integrate or evolve alongside these new paradigms.

6. References

[1] A. Handa et al. *DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality*. 2024. arXiv: 2210.13702 [cs.RO]. URL: https://arxiv.org/abs/2210.13702.

[2] Y. Yao, U. Yoo, J. Oh, C. G. Atkeson, and J. Ichnowski. *Soft Robotic Dynamic In-Hand Pen Spinning*. 2024. arXiv: 2411.12734 [cs.RO]. URL: https://arxiv.org/abs/2411.12734.

[3] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. *Reconciling Reality through Simulation: A Real-to-Sim-to-Real Approach for Robust Manipulation*. 2024. arXiv: 2403.03949 [cs.RO]. URL: https://arxiv.org/abs/2403.03949.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[5] B. Romero, H.-S. Fang, P. Agrawal, and E. Adelson. *EyeSight Hand: Design of a Fully-Actuated Dexterous Robot Hand with Integrated Vision-Based Tactile Sensors and Compliant Actuation*. 2024. arXiv: 2408.06265 [cs.RO]. URL: https://arxiv.org/abs/2408.06265.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG]. URL: https://arxiv.org/abs/1312.5602.

[7] OpenAI et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019. arXiv: 1912.06680 [cs.LG]. URL: https://arxiv.org/abs/1912.06680.

[8] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. *Efficient Online Reinforcement Learning with Offline Data*. 2023. arXiv: 2302.02948 [cs.LG]. URL: https://arxiv.org/abs/2302.02948.

[9] H. Hu, S. Mirchandani, and D. Sadigh. *Imitation Bootstrapped Reinforcement Learning*. 2024. arXiv: 2311.02198 [cs.LG]. URL: https://arxiv.org/abs/2311.02198.

[10] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. *From Imitation to Refinement – Residual RL for Precise Assembly*. 2024. arXiv: 2407.16677 [cs.RO]. URL: https://arxiv.org/abs/2407.16677.

[11] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. V. Wyk. *DextrAH-G: Pixels-to-Action Dexterous Arm-Hand Grasping with Geometric Fabrics*. 2024. arXiv: 2407.02274 [cs.RO]. URL: https://arxiv.org/abs/2407.02274.

[12] K. V. Wyk, A. Handa, V. Makoviychuk, Y. Guo, A. Allshire, and N. D. Ratliff. *Geometric Fabrics: a Safe Guiding Medium for Policy Learning*. 2024. arXiv: 2405.02250 [cs.RO]. URL: https://arxiv.org/abs/2405.02250.

[13] Y. Xu et al. *UniDexGrasp: Universal Robotic Dexterous Grasping via Learning Diverse Proposal Generation and Goal-Conditioned Policy*. 2023. arXiv: 2303.00938 [cs.RO]. URL: https://arxiv.org/abs/2303.00938.

[14] E. Todorov, T. Erez, and Y. Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[15] K. Zakka et al. *MuJoCo Playground*. 2025. arXiv: 2502.08844 [cs.RO]. URL: https://arxiv.org/abs/2502.08844.

[16] S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. Araújo. "CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms". In: *Journal of Machine Learning Research* 23.274 (2022), pp. 1–18. URL: http://jmlr.org/papers/v23/21-1342.html.

[17] S. Holt et al. "Evolving Control: Evolved High Frequency Control for Continuous Control Tasks". In: *CoRL Workshop on Safe and Robust Robot Learning for Operation in the Real World*. 2024. URL: https://openreview.net/forum?id=gzADUWLD9X.