A Appendix

A.1 Physics Simulation

Robots are simulated using PhysX [18] reduced coordinate articulations. Any individual rigid bodies may be simulated using either maximal coordinate rigid bodies or single-link reduced coordinate articulations. Articulations with a single link and rigid bodies are equivalent and interchangeable.We use the Temporal Gauss Seidel (TGS) [14] solver to compute the future states of objects in our physics simulation. The TGS solver uses the observation that sub-stepping a simulation with a single gauss-seidel solver iteration yields significantly faster convergence than running larger steps with more solver iterations. It folds this process efficiently into the iteration process, calculating the velocity at the end of each iteration and accumulating these velocities (scaled by dt/N, where N is the number of iterations) into a per-body accumulated delta buffer. This delta buffer is projected onto the constraint Jacobians and added to the bias terms in the constraints. This approach adds only a few additional operations to a more traditional Gauss-Seidel solver, producing almost identical performance cost per-iteration. However, it achieves the same effect on convergence as having sub-stepped the simulation without the computational expense. With positional joint constraints, an additional rotational term is calculated for joint anchors to improve handling of non-linear motion to avoid linearization artifacts. This term is not necessary (and in fact undesirable) to add to contacts.

A.1.1 Tendons

We simulate tendons as part of the Shadow Hand environment and describe the details of this simulation here.

A.1.2 Fixed Tendons

Fixed tendons are an abstract mechanism that couple degrees of freedom (DOF) of an articulation. A fixed tendon is composed of a tree of tendon joints, where each joint is associated with exactly one axis of a link's incoming articulation joint. In the following, when we refer to a tendon joint's position, we mean the position of the axis of this associated articulation joint.

In addition, each tendon joint has a coefficient that determines the contribution of the (rotational or translational) joint position to the length of the tendon, which is evaluated recursively by traversing the tree: The length at a given tendon joint is the length at its parent tendon joint plus its joint position scaled by the coefficient.

Given the tendon length at each joint, the tendon applies a spring force (or torque) to the joint's child link that is proportional to the deviation of the tendon length from a desired (tendon-wide) rest length. An equal and opposing force is applied to the parent link of the root tendon joint; conceptually, each tendon joint is a virtual joint drive between the root parent link and the tendon joint's child link. In addition to the spring force, the tendon-joint applies a damping force that is proportional to and acting against the velocity of the virtual root-to-child link joint.

Analogous to the length dynamics, the tendon supports length limits that apply an additional force or torque that is proportional to the deviation from set limits.

A.2 Observations & Rewards

In this section we describe the reward and observations for each environment in detail.

A.2.1 Ant and Humanoid environments

Both the Ant and Humanoid environments use the same reward formulation, namely:

$$\begin{aligned}\mathsf{R} = \mathsf{R}_{\mathsf{progress}} + \mathsf{R}_{\mathsf{alive}} \times \mathbb{1}(\mathsf{torso\_height} \geq \mathsf{termination\_height}) + \mathsf{R}_{\mathsf{upright}} \\ + \mathsf{R}_{\mathsf{heading}} + \mathsf{R}_{\mathsf{effort}} + \mathsf{R}_{\mathsf{act}} + \mathsf{R}_{\mathsf{dof}} \\ + \mathsf{R}_{\mathsf{death}} \times \mathbb{1}(\mathsf{torso\_height} \leq \mathsf{termination\_height})\end{aligned}$$

where

$$
\begin{aligned}
\mathsf{R}_{\mathsf{progress}} &= \mathsf{potential} - \mathsf{prev\_potential}\\
\mathsf{R}_{\mathsf{upright}} &= \mathsf{dot}(\mathsf{torso\_up\_vector}, \mathsf{up\_vector}) > 0.93\\
\mathsf{R}_{\mathsf{heading}} &= \mathsf{heading\_weight} * \begin{cases} 1.0, & \text{if } \mathsf{norm\_angle\_to\_target} \geq 0.8\\ \frac{\mathsf{norm\_angle\_to\_target}}{0.8}, & \text{otherwise} \end{cases}\\
\mathsf{R}_{\mathsf{act}} &= -\sum ||\mathsf{actions}||^2\\
\mathsf{R}_{\mathsf{effort}} &= \sum_{i=1}^{N} \mathsf{actions}_i \times \mathsf{normalized\_motor\_strength}_i \times \mathsf{dof\_velocity}_i\\
\mathsf{potential} &= -\frac{||\mathsf{p}_{\mathsf{target}} - \mathsf{p}_{\mathsf{torso}}||_2}{dt}
\end{aligned}
$$

Ant Reward described in Section A.2.1. Observations detailed in Table 2.

Observation space		Degrees of freedom
Torso vertical position		1
Velocity	positional	3
	angular	3
Yaw, roll, angle to target		3
Up and heading vector proj.		2
DOF measurements	position	8
	velocity	8
Sensor forces, torques		24
Actions		8
Total number of observations		60

Table 2: Observations used for Ant training.

Observation space		Degrees of freedom
Torso vertical position		1
Velocity	positional	3
	angular	3
Yaw, roll, angle to target		3
Up and heading vector proj.		2
DOF measurements	position	21
	velocity	21
	force	21
Sensor forces/torques		12
Actions		21
Total number of observations		108

Table 3: Observations used for Humanoid training.

Humanoid Reward described in Section A.2.1. Observations detailed in Table 3.

A.2.2 Locomotion environments

Ingenuity Observations detailed in Table 4. The reward function is as follows:

$$\mathsf{R} = \mathsf{R}_{\mathsf{pos}} \times (1 + \mathsf{R}_{\mathsf{upright}} + \mathsf{R}_{\mathsf{spin}})$$

reaching cost

$$\mathsf{R}_{\mathsf{pos}} = \frac{1}{1 + ||\mathsf{dist\_to\_target}||^2}$$

spinning cost

$$\mathsf{R}_{\mathsf{spin}} = \frac{1}{1 + ||\mathsf{spin\_rate}||^2}$$

upright cost

$$\mathsf{R}_{\mathsf{upright}} = \frac{1}{1 + \mathsf{local\_up\_vector}_z^2}$$

ANYmal Locomotion For the included flat-terrain environment, observations are detailed in Table 5 and the reward function is as follows:

$$\mathsf{R} = c_1 \mathsf{R}_{\mathsf{vel,xy}} + c_2 \mathsf{R}_{\mathsf{vel,yaw}} + c_3 \mathsf{R}_{\mathsf{torque}}$$

Reward terms are defined in Table 7 and symbols in Table 6.

Observation space		Degrees of freedom
Offset from target		3
Rotation		4
Velocity	positional	3
	angular	3
Total number of observations		13

Table 4: Observations used for ingenuity training.

Observation space		Degrees of freedom
Base velocity	positional	3
	angular	3
Body-relative gravity		3
Target X, Y, yaw velocities		3
DOF states	position	12
	velocity	12
Actions		12
Total number of observations		48

Table 5: Observations used for ANYmal training.

For rough terrain locomotion with sim-to-real, we extend the observations with 140 terrain heights around the robot's base and use the more complex reward function:

$$\begin{aligned}\mathsf{R} =\, & c_1\mathsf{R}_{\mathsf{vel,xy}} + c_2\mathsf{R}_{\mathsf{vel,yaw}} + c_3\mathsf{R}_{\mathsf{vel,z}} + c_4\mathsf{R}_{\mathsf{vel,pitch/roll}} + c_5\mathsf{R}_{\mathsf{joint\,vel/acc}} + \\ & c_6\mathsf{R}_{\mathsf{torque}} + c_7\mathsf{R}_{\mathsf{rate}} + c_8\mathsf{R}_{\mathsf{collision}} + c_9\mathsf{R}_{\mathsf{airtime}}\end{aligned}$$

Joint positions	$\mathbf{q}_j$
Joint velocities	$\dot{\mathbf{q}}_j$
Joint accelerations	$\ddot{\mathbf{q}}_j$
Target joint positions	$\mathbf{q}_j^*$
Joint torques	$\boldsymbol{\tau}_j$
Base linear velocity	$\mathbf{v}_b$
Base angular velocity	$\boldsymbol{\omega}_b$
Commanded base linear velocity	$\mathbf{v}_b^*$
Commanded base angular velocity	$\boldsymbol{\omega}_b^*$
Number of collisions	n_c
Feet air time	$\mathbf{t}_{air}$
Environment time step	dt

Table 6: Definition of symbols.

		definition	weight
Linear velocity tracking	$\mathsf{R}_{\mathsf{vel,xy}}$	$\phi(\mathbf{v}^*_{b,xy} - \mathbf{v}_{b,xy})$	$1dt$
Angular velocity tracking	$\mathsf{R}_{\mathsf{vel,yaw}}$	$\phi(\boldsymbol{\omega}^*_{b,z} - \boldsymbol{\omega}_{b,z})$	$0.5dt$
Linear velocity penalty	$\mathsf{R}_{\mathsf{vel,z}}$	$-\mathbf{v}^2_{b,z}$	$4dt$
Angular velocity penalty	$\mathsf{R}_{\mathsf{vel,pitch/roll}}$	$-\lvert\lvert\boldsymbol{\omega}_{b,xy}\rvert\rvert^2$	$0.05dt$
Joint motion	$\mathsf{R}_{\mathsf{joint\,vel/acc}}$	$-\lvert\lvert\ddot{\mathbf{q}}_j\rvert\rvert^2 - \lvert\lvert\dot{\mathbf{q}}_j\rvert\rvert^2$	$0.001dt$
Joint torques	$\mathsf{R}_{\mathsf{torque}}$	$-\lvert\lvert\boldsymbol{\tau}_j\rvert\rvert^2$	$0.00002dt$
Action rate	$\mathsf{R}_{\mathsf{rate}}$	$-\lvert\lvert\dot{\mathbf{q}}_j^*\rvert\rvert^2$	$0.25dt$
Collisions	$\mathsf{R}_{\mathsf{coll.}}$	$-n_{collision}$	$0.001dt$
Feet air time	$\mathsf{R}_{\mathsf{airtime}}$	$\sum_{f=0}^{4}(\mathbf{t}_{air,f} - 0.5)$	$2dt$

Table 7: Definition of reward terms, with $\phi(x) := \exp(-\frac{||x||^2}{0.25})$. The z axis is aligned with gravity.

Adversarial Imitation Learning AMP learns an imitation objective using an adversarial discriminator D, trained to differentiate between motion from the dataset $\mathcal{M}$ and motions produced by the policy π,

$$\underset{D}{\arg\min} \quad -\mathbb{E}_{s,s'\sim p_{\mathcal{M}}(s,s')}\left[\log D(s,s')\right] - \mathbb{E}_{s,s'\sim p_{\pi}(s,s')}\left[\log\left(1 - D(s,s')\right)\right],$$

where $p_{\mathcal{M}}(s,s')$ denotes the likelihood of observing a state transition from s to s' in the motion data, and $p_{\pi}(s,s')$ is likelihood of a state transition under the policy. The discriminator can then be used to specify rewards r_t for training a policy to imitate behaviors shown in the motion data

$$r_t = -\log\left(1 - D(s_t, s_{t+1})\right).$$

This objective, in effect encourages the policy to produce behaviors that fool the discriminator into classifying them as behaviors from the reference motion data.

Observation space		Degrees of freedom
Pelvis vertical height		1
Pelvis rotation		6
Pelvis Velocity	positional	3
	angular	3
DOF measurements	position	52
	velocity	28
Key point position		15
Total number of observations		108

Table 8: Observations used for AMP training with a humanoid character. 3D rotations are represented using a 6D tangent-normal encoding.

Observation space		Degrees of freedom
Joint DOFs	arm position	7 (Joint Torque only)
	eef position	2
EEF pose		7
Cube A pose		7
Cube A to Cube B position		3
Total number of observations		19 / 26

Table 9: Franka Cube Stack observations. Note that pose observations include the global 3-dim cartesian position and 4-dim quaternion orientation, and the arm joint position observations are only provided if using joint torque control.

Franka Cube Stack Observations are detailed in Table 9. The reward function used is as follows:

$$\mathsf{R} = \max\left(\mathsf{R}_{\mathsf{stack}}, \mathsf{R}_{\mathsf{align}} + \mathsf{R}_{\mathsf{lift}} + \mathsf{R}_{\mathsf{reach}}\right)$$

where:

$$\begin{aligned}
\mathsf{R}_{\mathsf{stack}} &= \mathsf{w}_{\mathsf{stack}} \times \mathbb{1}((\mathsf{height}_{\mathsf{cubeA}} > \mathsf{height}_{\mathsf{cubeB}})\&(\mathsf{cubeA\_aligned\_cubeB})\&(\mathsf{gripper\_away\_from\_cubeA})),\\
\mathsf{R}_{\mathsf{align}} &= \mathsf{w}_{\mathsf{align}} \times (1 - \tanh(10 \times \mathsf{cubeA\_to\_B\_xy\_dist})) \times \mathbb{1}(\mathsf{cubeA\_is\_lifted}),\\
\mathsf{R}_{\mathsf{lift}} &= \mathsf{w}_{\mathsf{lift}} \times \mathbb{1}(\mathsf{cubeA\_is\_lifted}),\\
\mathsf{R}_{\mathsf{reach}} &= \mathsf{w}_{\mathsf{reach}} \times \left(1 - \tanh\left(\frac{10}{3} \times (\mathsf{dist}(\mathsf{cubeA}, \mathsf{gripper}) + \mathsf{dist}(\mathsf{cubeA}, \mathsf{lfinger}) + \mathsf{dist}(\mathsf{cubeA}, \mathsf{rfinger})\right)\right)\right)
\end{aligned}$$

We set $\mathsf{w}_{\mathsf{stack}} = 16.0$, $\mathsf{w}_{\mathsf{align}} = 2.0$, $\mathsf{w}_{\mathsf{lift}} = 1.5$, and $\mathsf{w}_{\mathsf{reach}} = 0.1$

A.2.3 Robotic Hands

Shadow Hand The reward function for Shadow Hand is as follows:

$$\mathsf{R} = \mathsf{w}_{\mathsf{dist}}\mathsf{R}_{\mathsf{dist}} + \mathsf{R}_{\mathsf{rot}} + \mathsf{w}_{\mathsf{act}}\mathsf{R}_{\mathsf{act}}$$

distance cost

$$\mathsf{R}_{\mathsf{dist}} = ||\mathsf{p}_{\mathsf{obj}} - \mathsf{p}_{\mathsf{target}}||_2$$

orientation cost

$$\mathsf{rot\_dist} = 2 \times \arcsin(\max(1, ||\mathsf{q}_{\mathsf{obj}} * \overline{\mathsf{q}_{\mathsf{target}}}||_2))$$

$$\mathsf{R}_{\mathsf{rot}} = \frac{1}{|\mathsf{rot\_dist}| + 0.1}$$

action smoothness cost

$$\mathsf{R}_{\mathsf{act}} = \sum ||\mathsf{actions}||^2$$

where $\mathsf{w}_{\mathsf{dist}} = -10$ and $\mathsf{w}_{\mathsf{act}} = -2e - 4$.

```
@torch.jit.script
def compute_hand_reward(
    object_pos, object_rot, target_pos, target_rot, actions,
    dist_reward_scale: float, rot_reward_scale: float, rot_eps: float,
    action_penalty_scale: float, success_tolerance: float, reach_goal_bonus: float,
    fall_dist: float,fall_penalty: float):

    #dist_reward_scale: -10.0
    #rot_reward_scale: 1.0
    #rot_eps: 0.1
    #action_penalty_scale: -0.0002
    #reach_goal_bonus: 250
    #fall_distance: 0.24
    #fall_penalty: 0.0

    # Distance from the hand to the object
    goal_dist = torch.norm(object_pos - target_pos, p=2, dim=-1)

    # Orientation alignment for the cube in hand and goal cube
    quat_diff = quat_mul(object_rot, quat_conjugate(target_rot))
    rot_dist  = 2.0 * torch.asin(torch.clamp(torch.norm(quat_diff[:, 0:3], p=2, dim=-1),
     max=1.0))

    # Orientation reward
    rot_rew = 1.0/(torch.abs(rot_dist) + rot_eps)

    # action smoothness reward
    action_penalty = torch.sum(actions ** 2, dim=-1)
    # Total reward is: position distance + orientation alignment + action regularization
     + success bonus + fall penalty
    reward = goal_dist * dist_reward_scale + rot_rew * rot_reward_scale
           + action_penalty * action_penalty_scale

    # Find out which envs hit the goal and update successes count
    goal_resets = torch.where(torch.abs(rot_dist) <= success_tolerance,
                              torch.ones_like(reset_goal_buf), reset_goal_buf)

    # Success bonus: orientation is inside the `success_tolerance` of goal orientation
    reward = torch.where(goal_resets == 1, reward + reach_goal_bonus, reward)

    # Fall penalty: distance to the goal is larger than a threashold
    reward = torch.where(goal_dist >= fall_dist, reward + fall_penalty, reward)

    return reward
```

Reward function for cube orientation for Shadow Hand experiments.

There are two different variants of observations used. In the Shadow Hand **Standard** environment, the observations are as shown in Table 10. In The ShadowHand **OpenAI** environment, in order to compare to compare to [21], we use observations as shown in Table 11. Further details of the Shadow Hand environments are available in Section A.4

Observation space		Degrees of freedom
Finger joints	position	24
	velocity	24
	force	24
Cube pose	translation	3
	quaternion	4
	linear velocity	3
	angular velocity	3
Cube rotation relative to goal	quaternion	4
Goal pose	translation	3
	quaternion	4
5 × Finger tips	position	3
	quaternion	4
	linear velocity	3
	angular velocity	3
	force	3
	torque	3
Previous action output from policy		20
Total number of observations		211

Table 10: Observations for the Shadow Hand **Standard** environment.

Observation space		Degrees of freedom
5 × Finger joints	position	3
Cube pose	translation	3
Cube rotation relative to goal	quaternion	4
Previous action output from policy		20
Total number of observations		42

Table 11: Observations for the Shadow Hand **OpenAI** environment. The observations of the critic are the same as for Shadow Hand Standard (see Table 10).

Observation space		Degrees of freedom
Finger joints	position	9
	velocity	9
Cube pose	translation	3
	quaternion	4
Goal pose	translation	3
	quaternion	4
Previous action output from policy		9
Total number of observations		41

Table 12: Trifinger Actor Observations.

Observation space		Degrees of freedom
Actor Observations (see Table 12)		41
Cube pose	linear velocity	3
	angular velocity	3
3 × Finger tips	position	3
	quaternion	4
	linear velocity	3
	angular velocity	3
	force	3
	torque	3
Finger joints	terque	9
Total number of observations		113

Table 13: Trifinger Critic Observations

Trifinger Our total reward is defined as:

$$\mathsf{R} = \mathsf{w_{og}} \mathsf{R_{object\_goal}} + \mathsf{w_{fo}} \mathsf{R_{fingertip\_object}} \times \mathbb{1}(\mathsf{timesteps} \leq 5e7) + \mathsf{w_{fv}} \mathsf{R_{fingertip\_velocity}}$$

reposing cost

$$\mathsf{R_{object\_goal}} = \mathcal{K}(||\mathsf{t_{curr}} - \mathsf{t_{target}}||_2) + \frac{1}{3 \times |\mathsf{rot\_dist}| + 0.01}$$

fingertips interaction cost

$$\mathsf{R_{fingertip\_object}} = \sum_{i \in \mathsf{fingertips}} \Delta_i^t$$

fingertips smoothness cost

$$\mathsf{R}_{\mathsf{fingertip\_velocity}} = \sum_{i \in \mathsf{fingertips}} ||\mathsf{fingertip\_speed}_i||^2$$

Δ_i^t denotes the change across the timestep of the fingertip distance to the centroid of the object and was found to be helpful in [1]. Formally, $\Delta_i^t = ||\mathsf{ft}_{\mathsf{i,t}} - \mathsf{t}_{\mathsf{curr,t}}||_2 - ||\mathsf{ft}_{\mathsf{i,t-1}} - \mathsf{t}_{\mathsf{curr,t-1}}||_2$, where $\mathsf{t}_{\mathsf{curr,t}}$ is position of the cube centroid and ft_{i} denotes the position of the i-th fingertip at time t.

$\mathsf{rot\_dist}$ is the angluar difference between the current and target cube pose, $\mathsf{rot\_dist} = 2 \times \arcsin(\min(1.0, ||\mathsf{q}_{\mathsf{diff}}||_2)), \mathsf{q}_{\mathsf{diff}} = \mathsf{q}_{\mathsf{curr}}\mathsf{q}^*_{\mathsf{target}}$. Following [10], a logistic kernel is used to convert tracking error in euclidean space into a bounded reward function, with $\mathcal{K}(x) = (e^{ax} + b + e^{-ax})^{-1}$, where a is a scaling factor; we use $a = 50$. See [4] for a more thorough motivation and description of these reward terms.

Allegro The reward formulation is identical to that used in Shadow Hand - see Section A.2.3. The observations are also identical, save for the change in number of fingers.

A.3 Hyperparamters for Training PPO

Environment	# Environments	KL Threshold	Mini-batch Size	Horizon Length	# PPO Epochs	Hidden Units	Training Steps
Ant	4096	8e-3	32768	16	4	256, 128, 64	32M
Humanoid	4096	8e-3	32768	32	5	400, 200, 100	327M
Ingenuity	4096	1.6 e-2	32768	16	8	256, 256, 128	32M
ANYmal	8192	1e-2	32768	16	5	256, 128, 64	65M
ANYmal Terrain	4096	1e-2	24576	24	10	512, 256, 128	150M
AMP	4096	2e-1	16384	32	8	1024, 512	39M
Franka	16384	1.6 e-2	131072	32	4	256, 128, 64	786M
SH Standard	16384	1.6 e-2	32768	8	5	512, 512, 256, 128	655M
SH OpenAI	16384	1.6 e-2	32768	8	5	400, 400, 200, 100	1310M

Table 14: Hyperparameters used for training in each environment. Allegro shares the parameters for Shadow Hand OpenAI. The hidden units are ELU for every environment except AMP, where ReLU units are used. Additionally, every environment uses an adaptive learning rate with a KL divergence target specified in the *KL Threshold* column, except for AMP which uses a fixed learning rate of 2e-5 and fixed KL theshold of 2e-1.

A.4 Shadow Hand Details

As mentioned previously, we implemented two variants of the Shadow Hand environment. The **Standard** variant uses privileged policy observations and no Domain Randomization, in order to provide a quick training example to test Reinforcement Learning algorithms on. The **OpenAI** variant uses asymmetric observations, such that it would be possible to transfer the policy to the real world, mimicing the setup in [21].

A.4.1 Randomizations

Isaac Gym implements a high-level API that simplifies setting up physics domain randomization parameters and schedule in yaml configuration files and is very extensible. Here we detail the randomization parameters that we used.

Random forces on the object. Following [21] unmodeled dynamics is represented by applying random forces on the object. The probability p that a random force is applied is sampled at the beginning of the randomization episode from the loguniform distribution between 0.1% and 10%. Then, at every timestep, with probability p we apply a random force from the 3-dimensional Gaussian distribution with the standard deviation equal to $1\ m/s^2$ times the mass of the object on each coordinate and decay the force with the coefficient of 0.99 per 50ms.

Runtime physics randomizations. Physical parameters like friction, joint and tendon properties, as well as correlated noise parameters are randomized after resets once a minimum of 720 steps have passed and held fixed otherwise. Table 15 lists all physics parameters that are randomized.

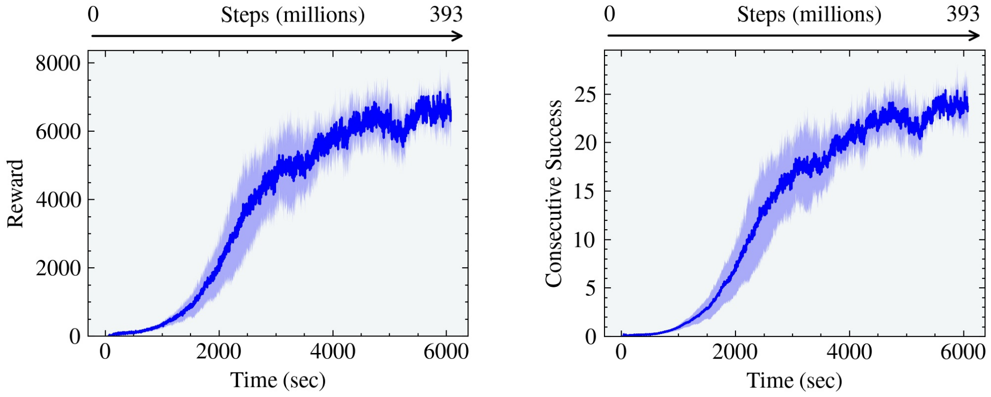
Parameter	Scaling factor range	Additive term range
object dimensions	uniform$([0.95, 1.05])$	
object and robot link masses	uniform$([0.5, 1.5])$	
surface friction coefficients	uniform$([0.7, 1.3])$	
robot joint damping coefficients	loguniform$([0.3, 3.0])$	
actuator force gains (P term)	loguniform$([0.75, 1.5])$	
joint limits		$\mathcal{N}(0, 0.15)$ rad
gravity vector (each coordinate)		$\mathcal{N}(0, 0.4)$ m/s^2

Table 15: Ranges of physics parameter randomizations.

Startup physics randomizations. Object dimensions, and object and robot link masses are randomized once on simulation startup across all simulation environments.

A.4.2 OpenAI Observations

We conduct experiments with Shadow Hand OpenAI observations with a tighter success tolerance of 0.1 radians and show the reward curves as well as the consecutive successes achieved with this training in Figure 14 and 15.



(a) Reward

(b) Consecutive Successes

Figure 14: Training curves for ShadowHand environments with **OpenAI observations** and Feed Forward policy and value functions with a tighter success tolerance of 0.1 rad.

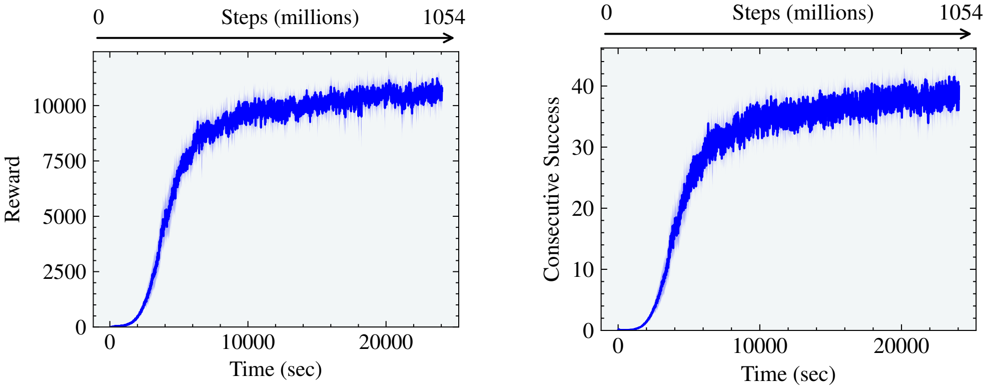
Feed Forward Networks We achieve 20 consecutive successful cube rotations after training in just under 1 hour. This is similar to the performance[2] achieved by OpenAI *et al.* [21] but with a cluster of 384 16-core CPUs and 8 V100 GPUs with training for 30 hours while we only need a single A100.

LSTMs Using sequence networks like LSTMs improve the performance and we find that we are able to achieve 37 consecutive successful cube rotations after training in just under 6 hours. OpenAI *et al.* [21] achieve similar performance in about 17 hours again on a cluster of 384 16-core CPUs and 8 V100 GPUs. We use a sequence length of 4 to train the LSTM. Various other parameters for this set up are in Table 14.

We also note that training with a tolerance of 0.1 rad and testing with a tolerance of 0.4 rad, we are able to even go up to 44 consecutive cube rotations.

---

[2] pp 13, Section 6.5 titled **Sample Complexity & Scale**



(a) Reward

(b) Consecutive Successes

Figure 15: Training curves for ShadowHand environments with **OpenAI observations** and an LSTM based policy and value function with a tighter success tolerance of 0.1 rad.