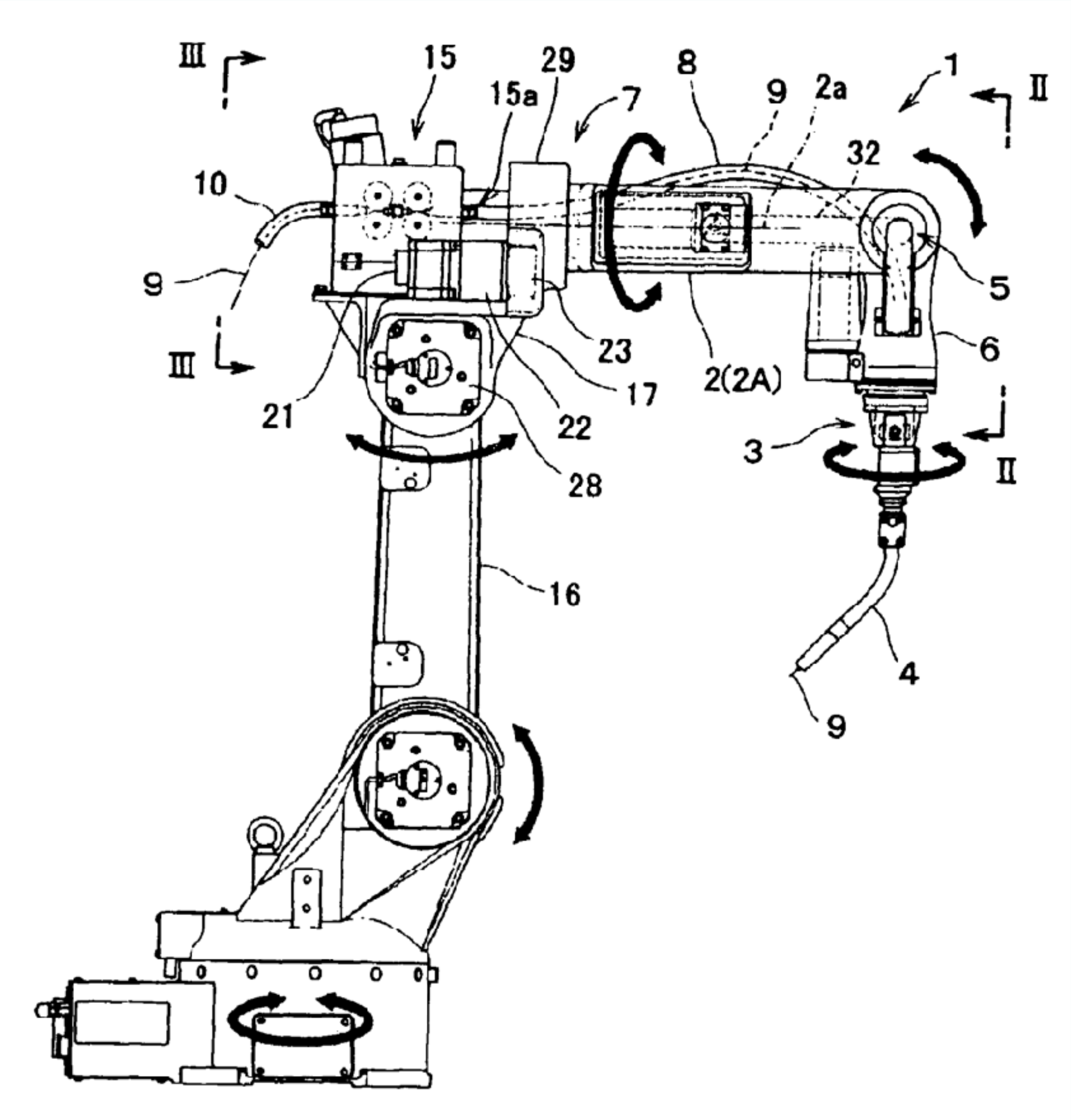
OpenAI

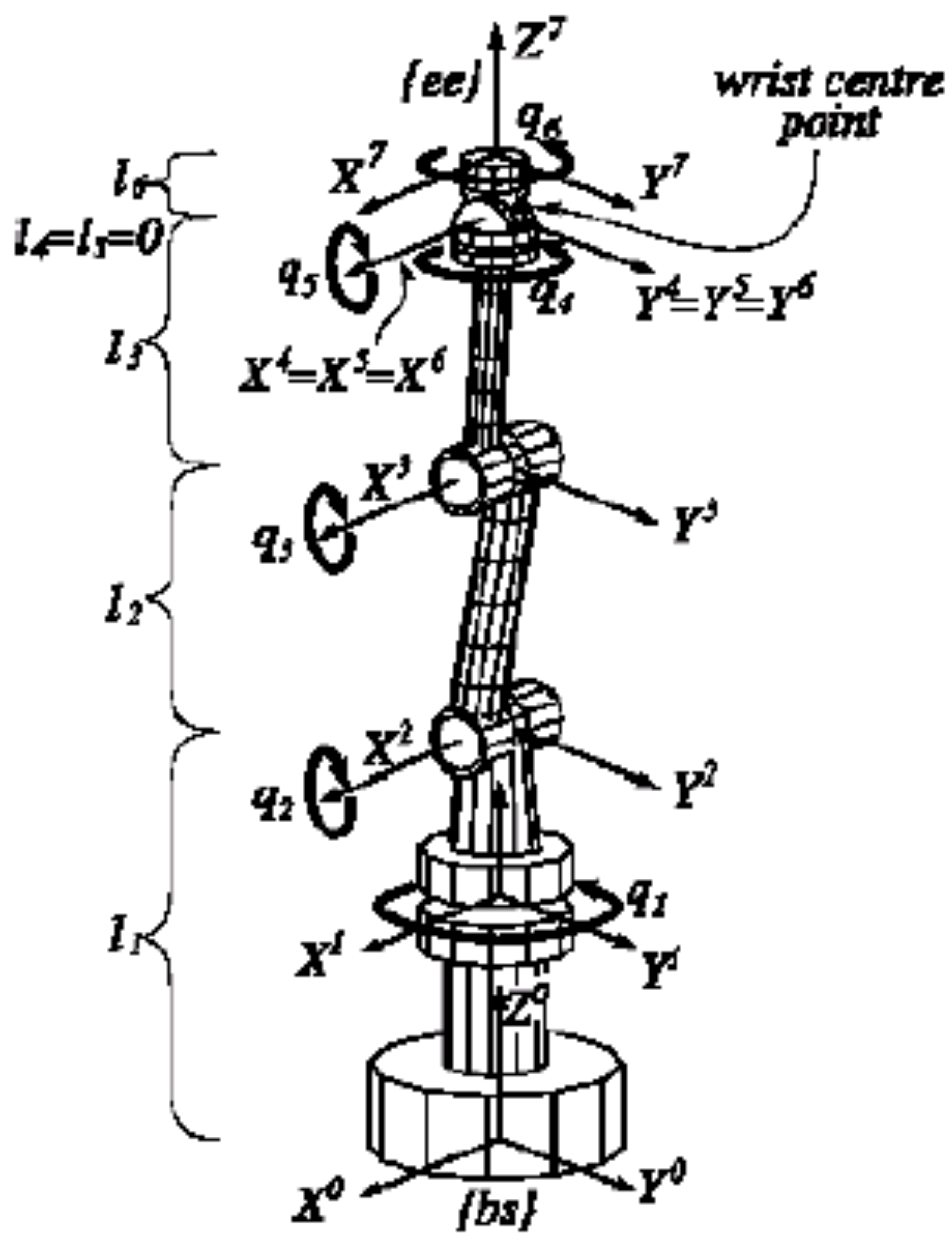
Learning Dexterity

Peter Welinder

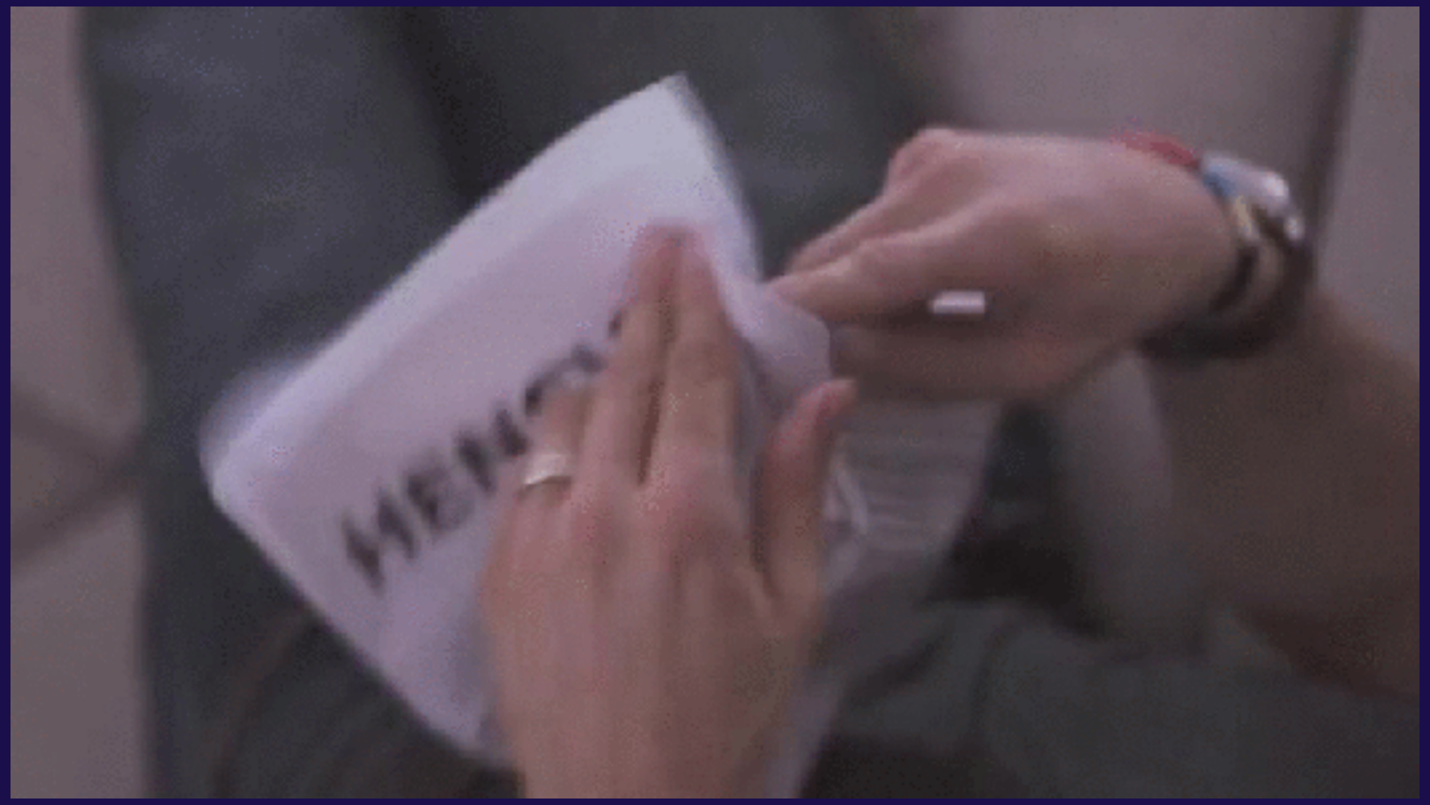
SEPTEMBER 09, 2018


FRAGILE
FRAGILE


III
15
29
8
9
2a
1
II
15a
7
10
32
9
5
23
6
III
21
22
17
2(2A)
3
II
28
16
4
9

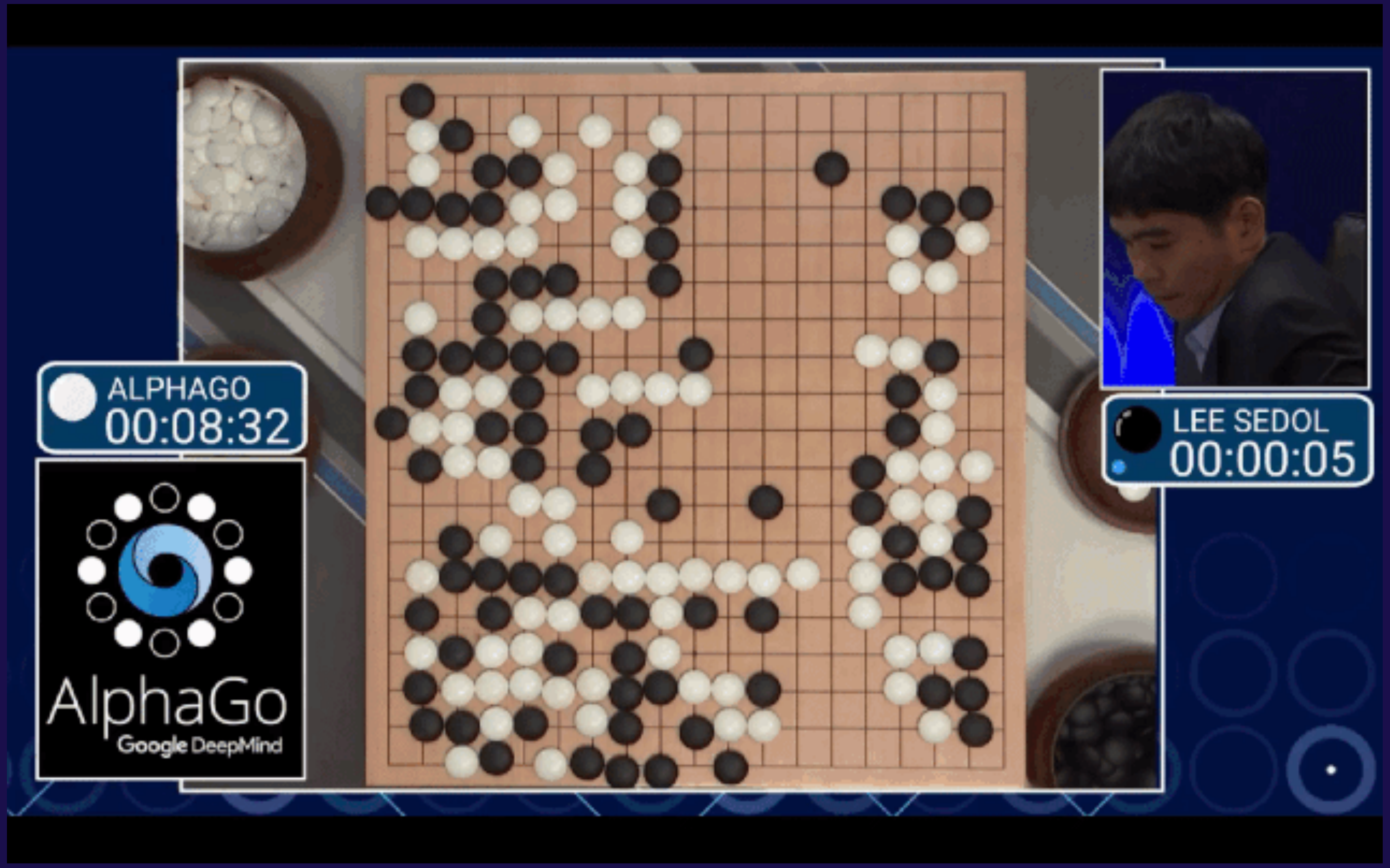

Z^7
{ee}
wrist centre point
q_6
l_6
X^7
Y^7
$l_4=l_5=0$
q_5
q_4
$Y^4=Y^5=Y^6$
l_3
$X^4=X^5=X^6$
X^3
q_3
Y^3
l_2
X^2
q_2
Y^2
q_1
l_1
X^1
Y^1
Z^0
X^0
{bs}
Y^0

Learning



Trends towards learning-based robotics

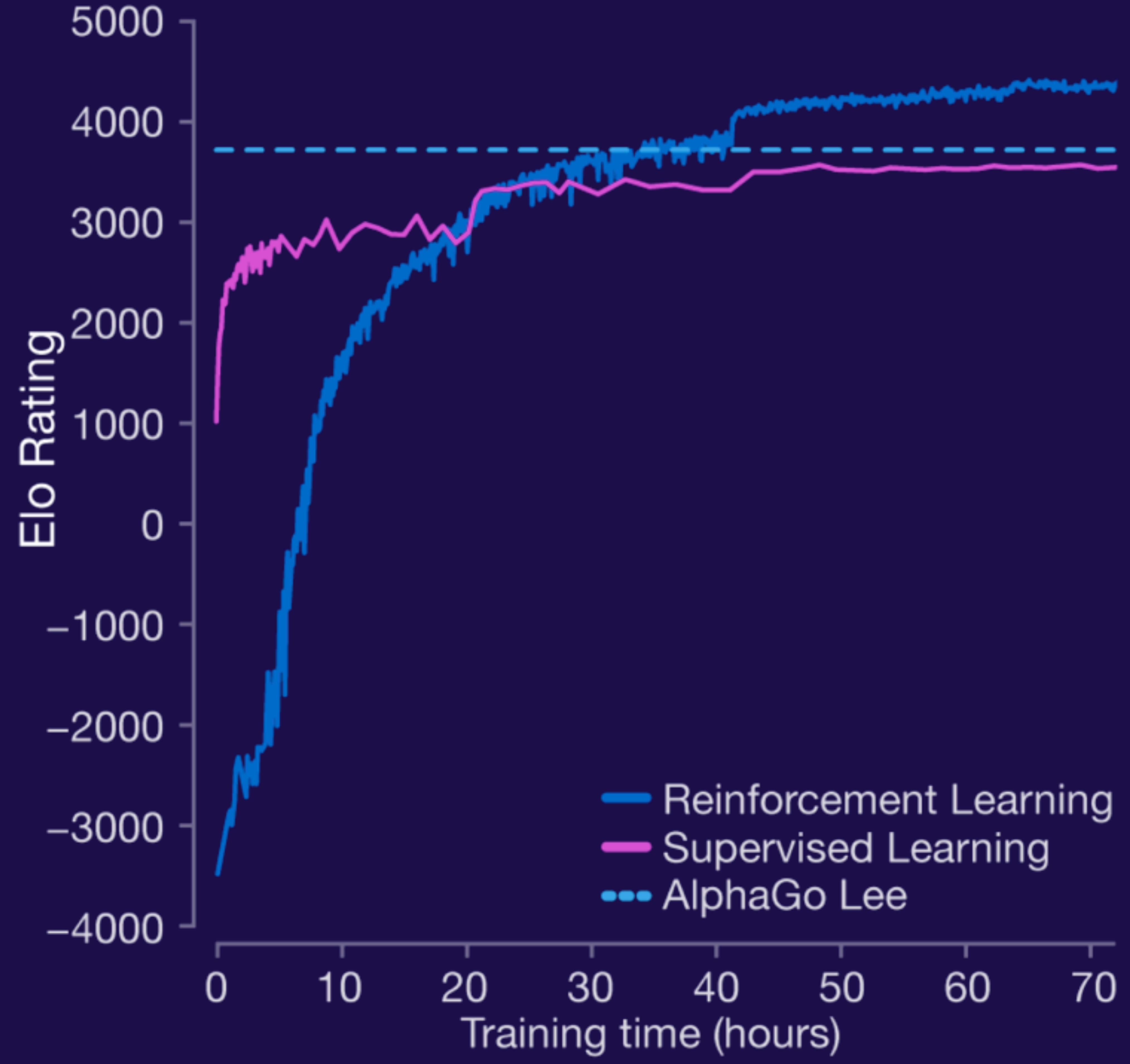
Reinforcement Learning



Go (AlphaGo Zero)

Dota 2 (OpenAI Five)

What about Robotics? RL doesn't work because it uses lots of experience.



5 million games

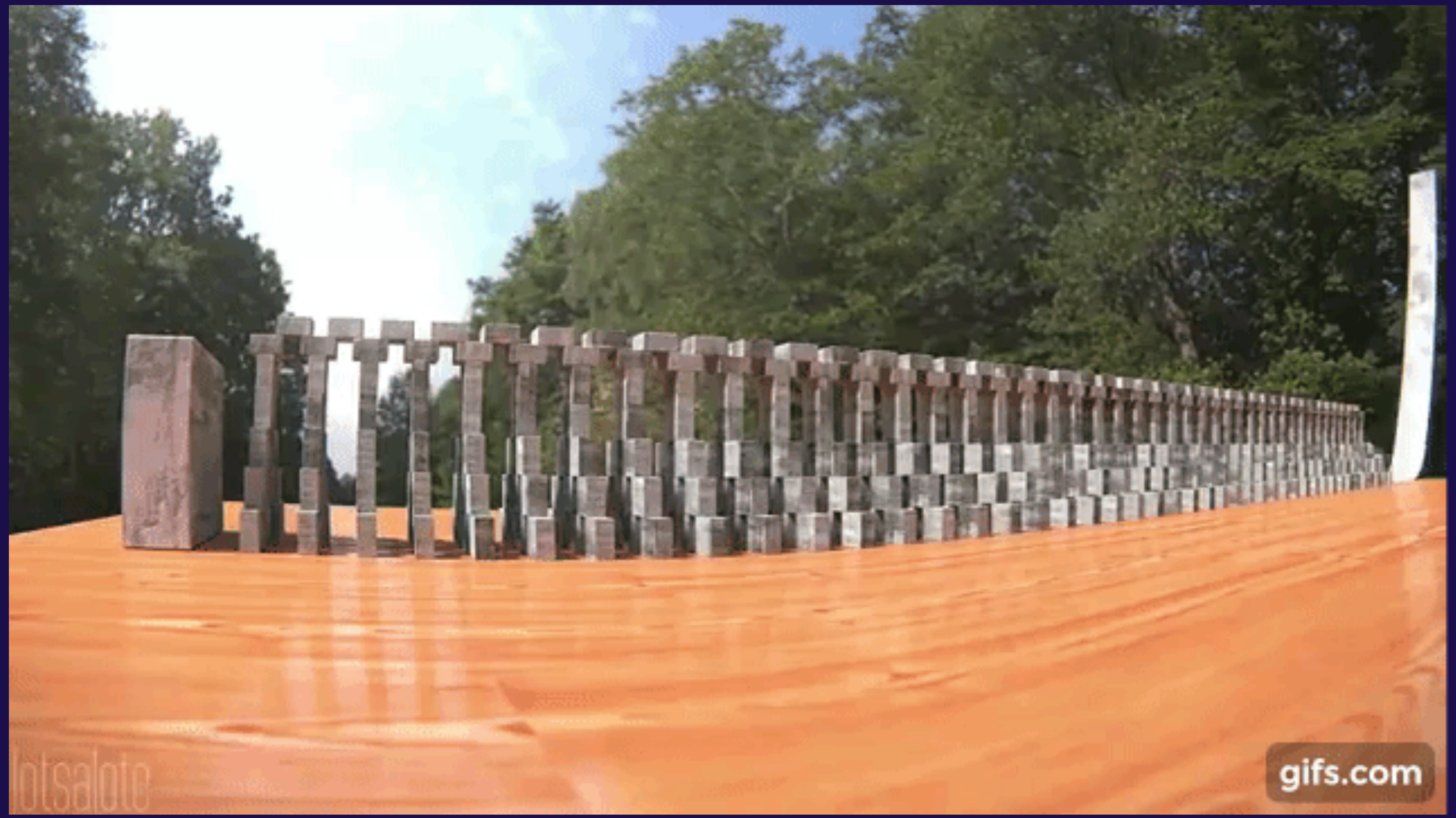
~500 years of playing

Go: 200 years per day

Dota: 200 years per day


r3d29
r3d09
r3d10
r3d10
r3d31
r3d31
r3d16
r3d16
10:35

Simulators

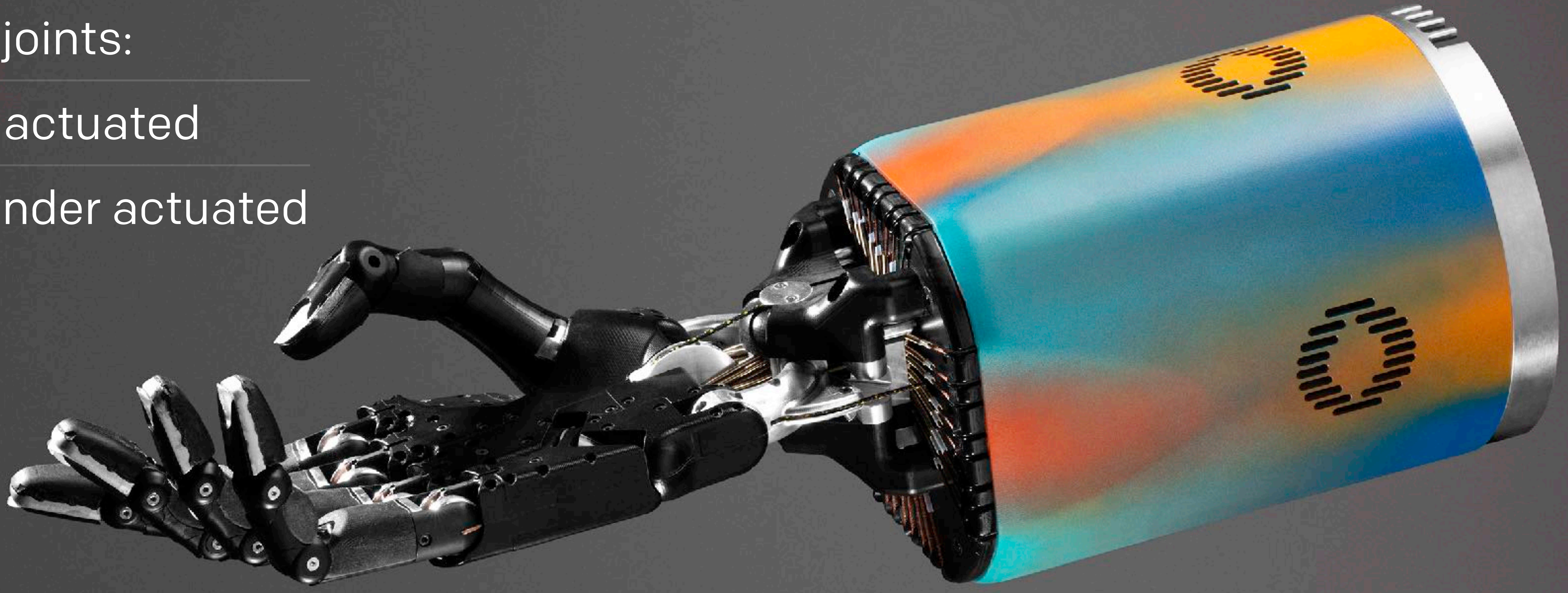


Learning dexterity

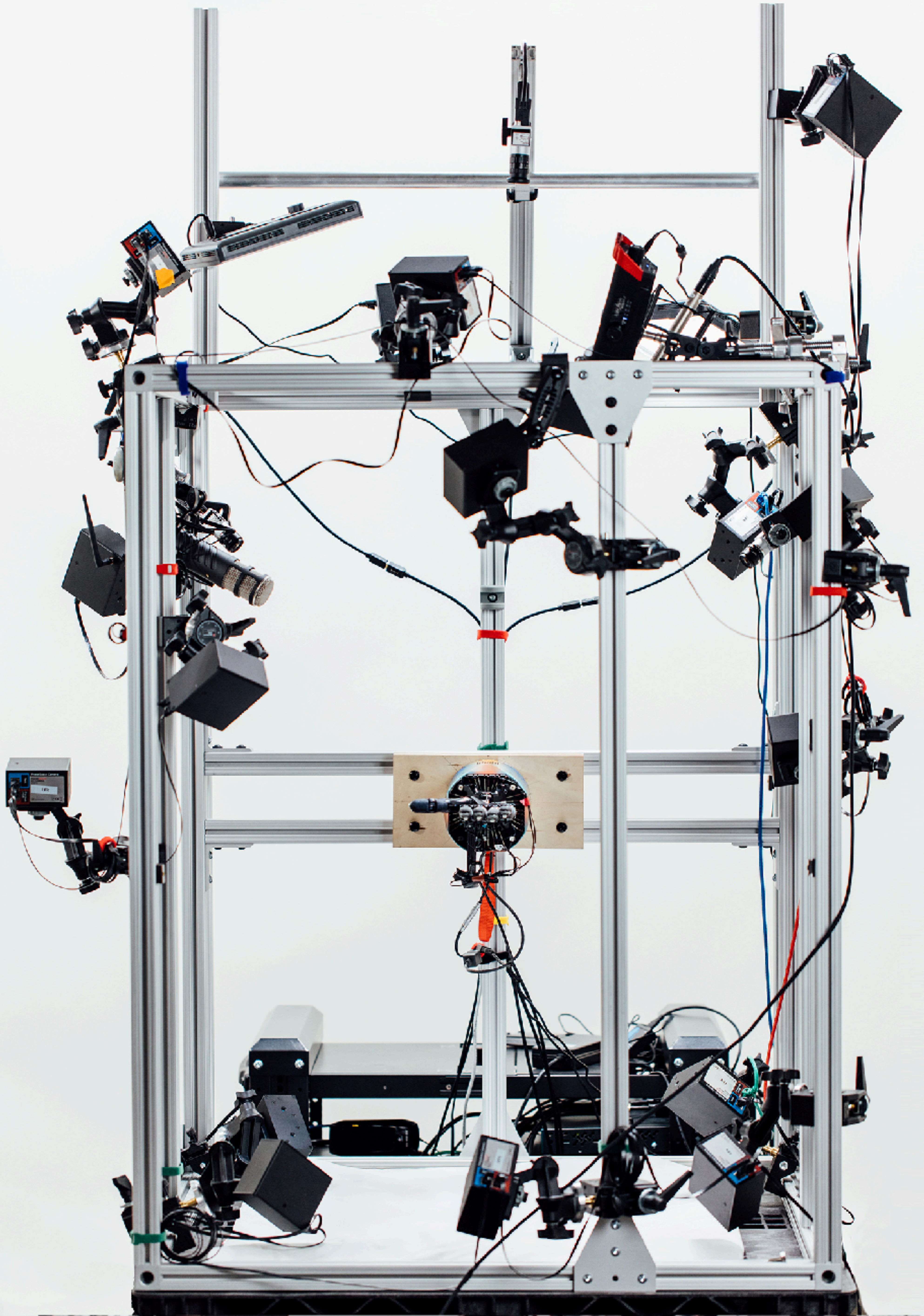
24 joints:

20 actuated

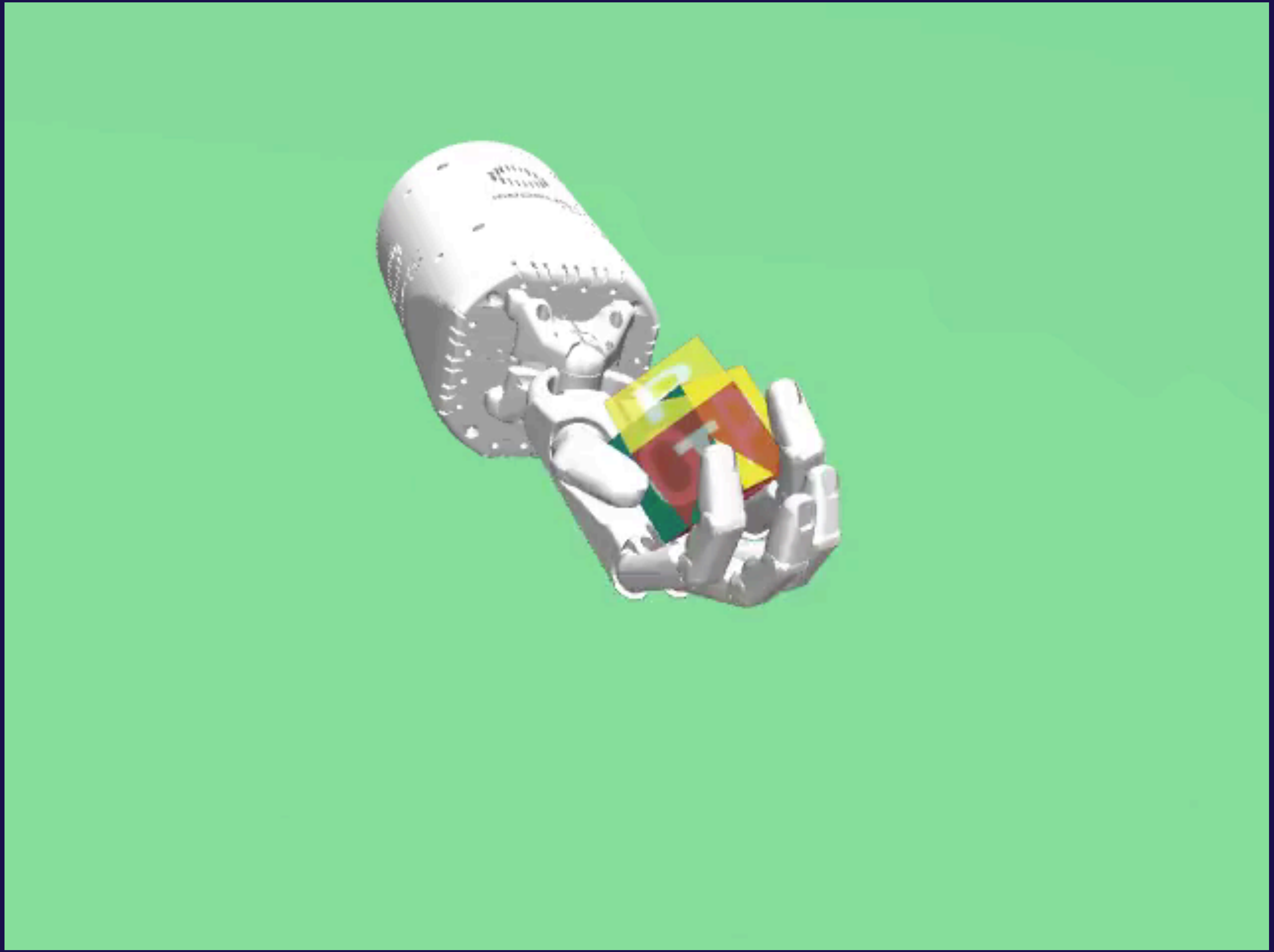
4 under actuated



Shadow
Robot Company



Rotating a block



Challenges

RL in real world

high dimensional control

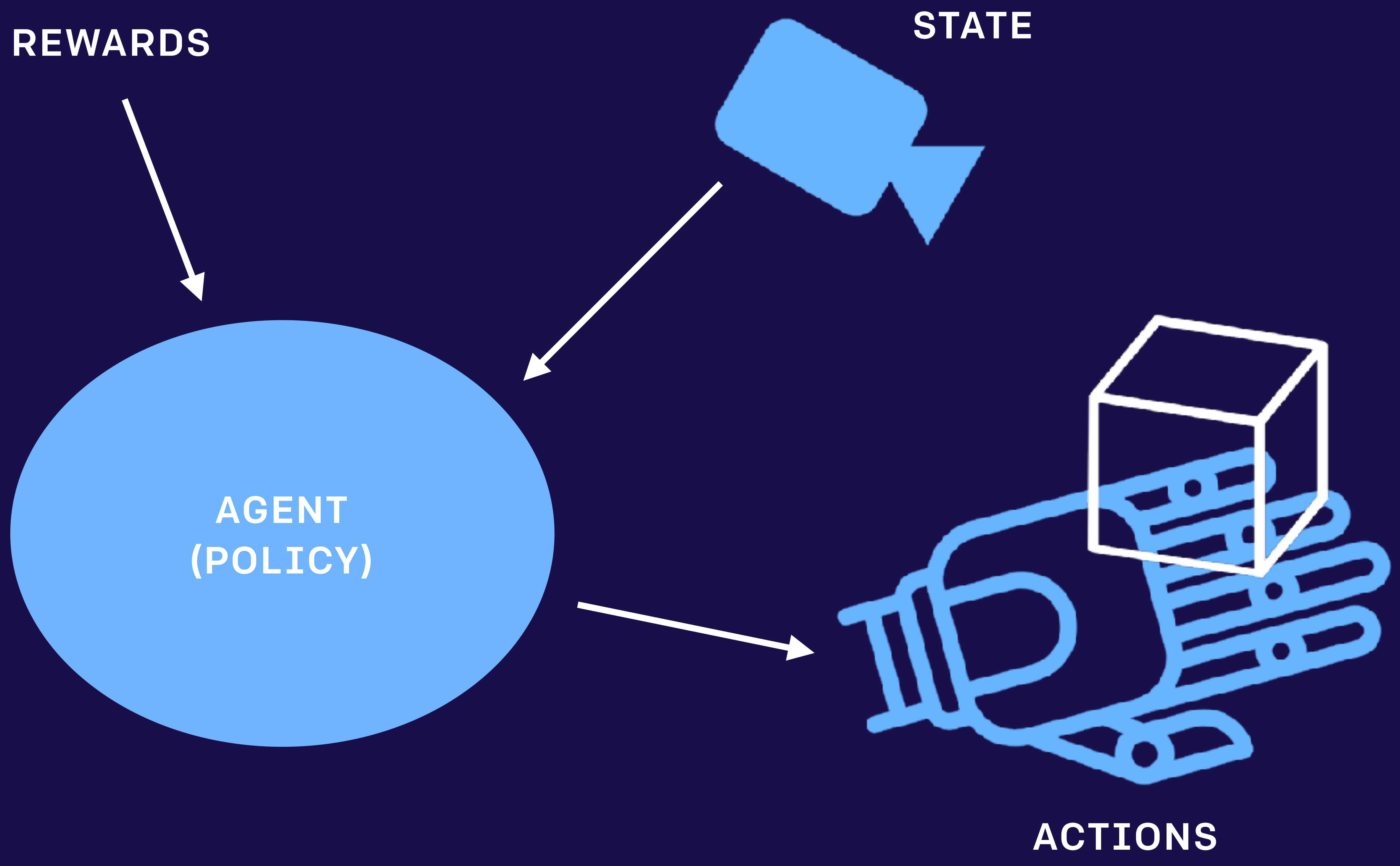
noisy and partial observations

manipulating multiple objects.

Approach

Reinforcement Learning
+
Domain Randomization

Reinforcement Learning



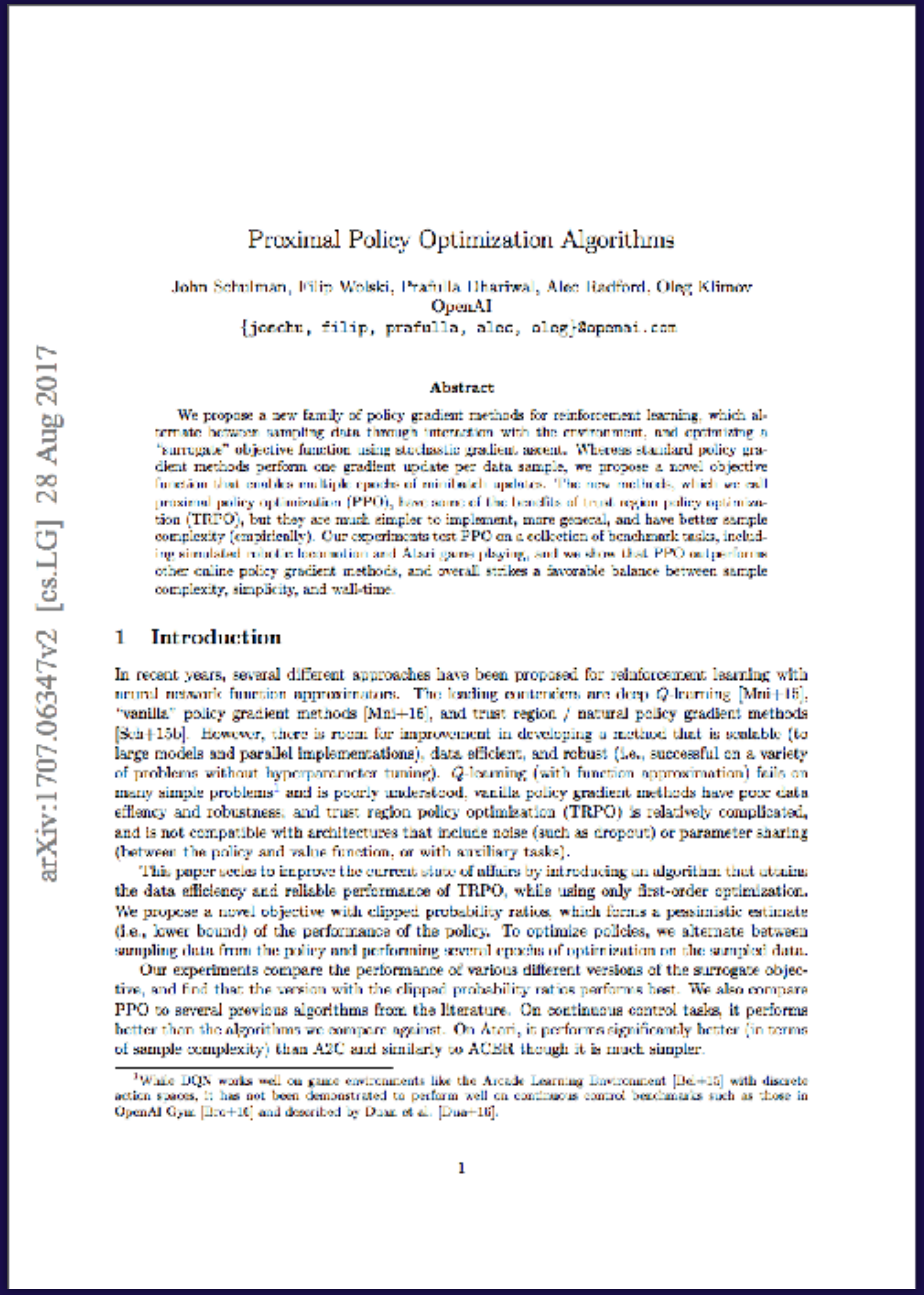
$$\text{action}_t = \text{policy}(\text{state}_t)$$

$$\text{score} = \sum_t \text{reward}(\text{state}_t, \text{action}_t)$$

Reinforcement Learning

$$\theta^* = \arg\max_\theta \sum_{\tau \in \text{episodes}} \text{reward}(\text{policy}_\theta, \tau)$$

Proximal Policy Optimization (PPO)



arXiv:1707.06347v2 [cs.LG] 28 Aug 2017

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

Abstract

We propose a new family of policy gradient methods for reinforcement learning, which alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Whereas standard policy gradient methods perform one gradient update per data sample, we propose a novel objective function that enables multiple epochs of minibatch updates. The new methods, which we call proximal policy optimization (PPO), have some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity (empirically). Our experiments test PPO on a collection of benchmark tasks, including simulated robotic locomotion and Atari game playing, and we show that PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time.

1 Introduction

In recent years, several different approaches have been proposed for reinforcement learning with neural network function approximators. The leading contenders are deep Q-learning [Mni+15], "vanilla" policy gradient methods [Mni+16], and trust region / natural policy gradient methods [Sch+15b]. However, there is room for improvement in developing a method that is scalable (to large models and parallel implementations), data efficient, and robust (i.e., successful on a variety of problems without hyperparameter tuning). Q-learning (with function approximation) fails on many simple problems[1] and is poorly understood, vanilla policy gradient methods have poor data effiency and robustness; and trust region policy optimization (TRPO) is relatively complicated, and is not compatible with architectures that include noise (such as dropout) or parameter sharing (between the policy and value function, or with auxiliary tasks).

This paper seeks to improve the current state of affairs by introducing an algorithm that attains the data efficiency and reliable performance of TRPO, while using only first-order optimization. We propose a novel objective with clipped probability ratios, which forms a pessimistic estimate (i.e., lower bound) of the performance of the policy. To optimize policies, we alternate between sampling data from the policy and performing several epochs of optimization on the sampled data.

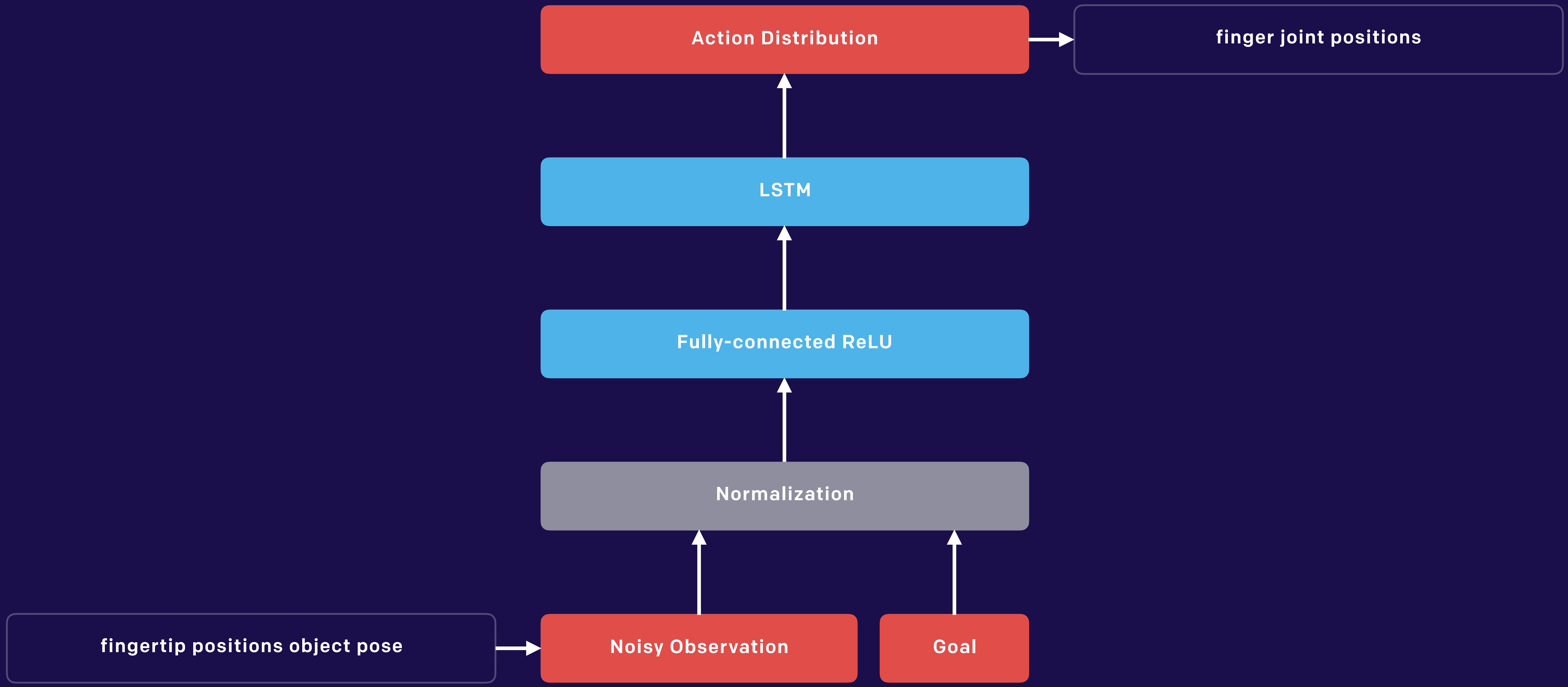
Our experiments compare the performance of various different versions of the surrogate objective, and find that the version with the clipped probability ratios performs best. We also compare PPO to several previous algorithms from the literature. On continuous control tasks, it performs better than the algorithms we compare against. On Atari, it performs significantly better (in terms of sample complexity) than A2C and similarly to ACER though it is much simpler.

[1] While DQN works well on game environments like the Arcade Learning Environment [Bel+15] with discrete action spaces, it has not been demonstrated to perform well on continuous control benchmarks such as those in OpenAI Gym [Bro+16] and described by Duan et al. [Dua+16].

1

Schulman et al. (2017)

Policy

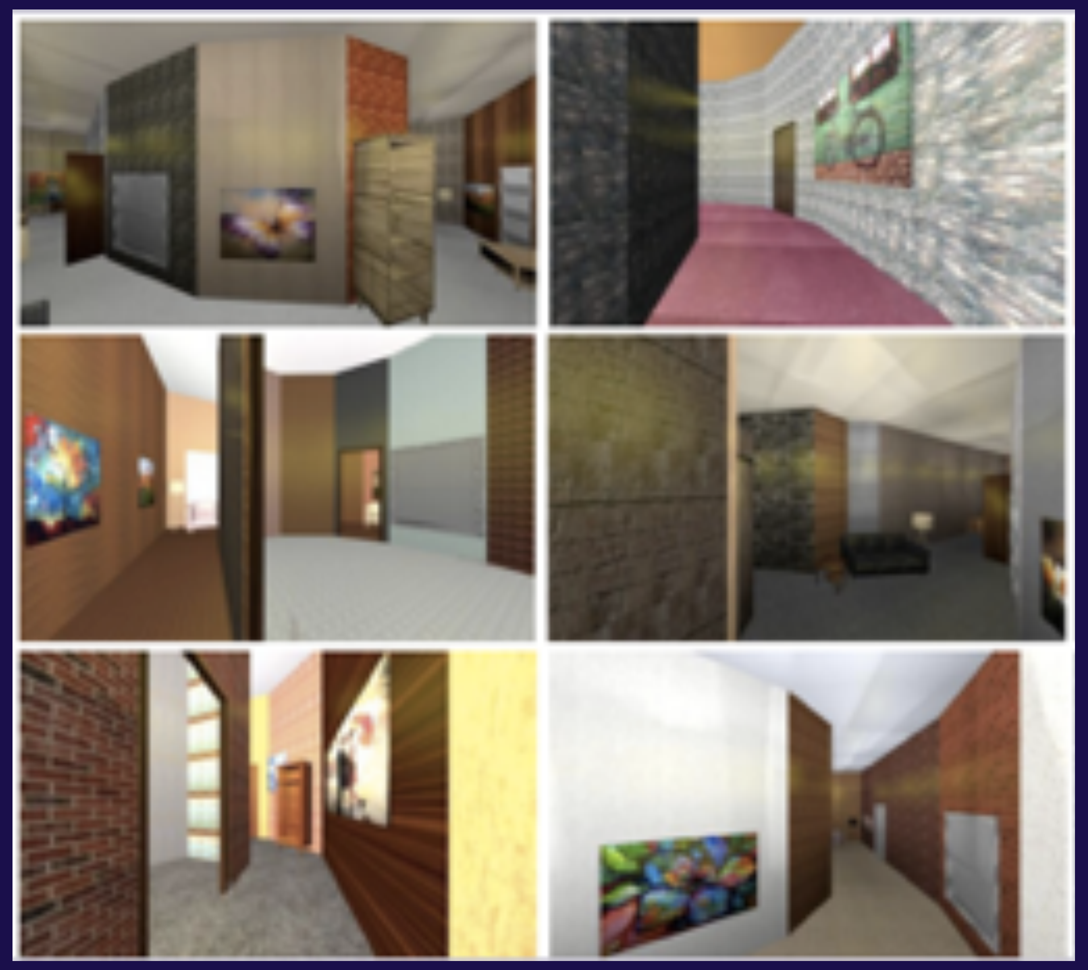


Action Distribution
finger joint positions
LSTM
Fully-connected ReLU
Normalization
fingertip positions object pose
Noisy Observation
Goal

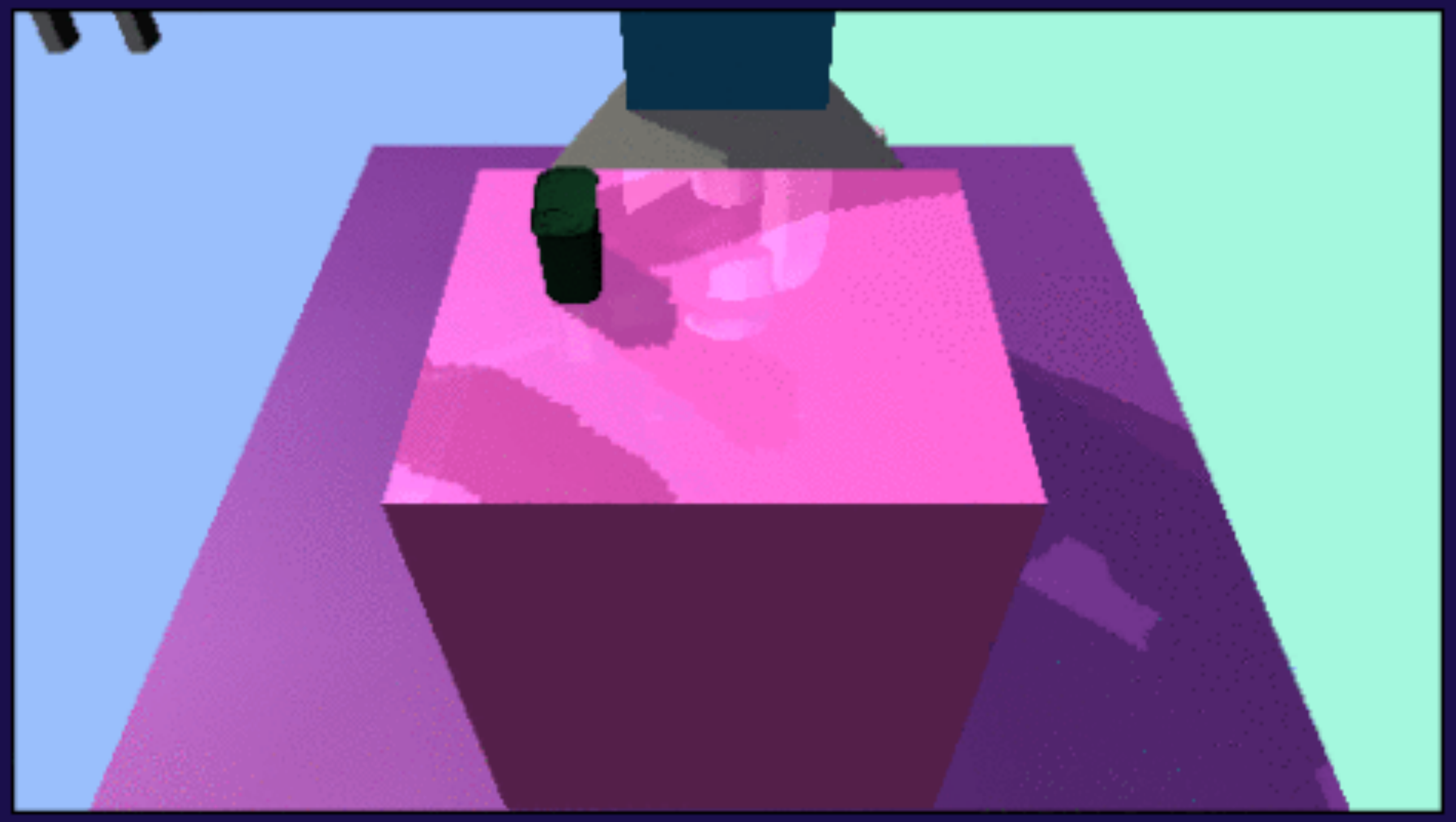

Distributed training with Rapid
Rollout Workers
6,000 CPU Cores
Optimizers
8 GPUs
Policy Parameters

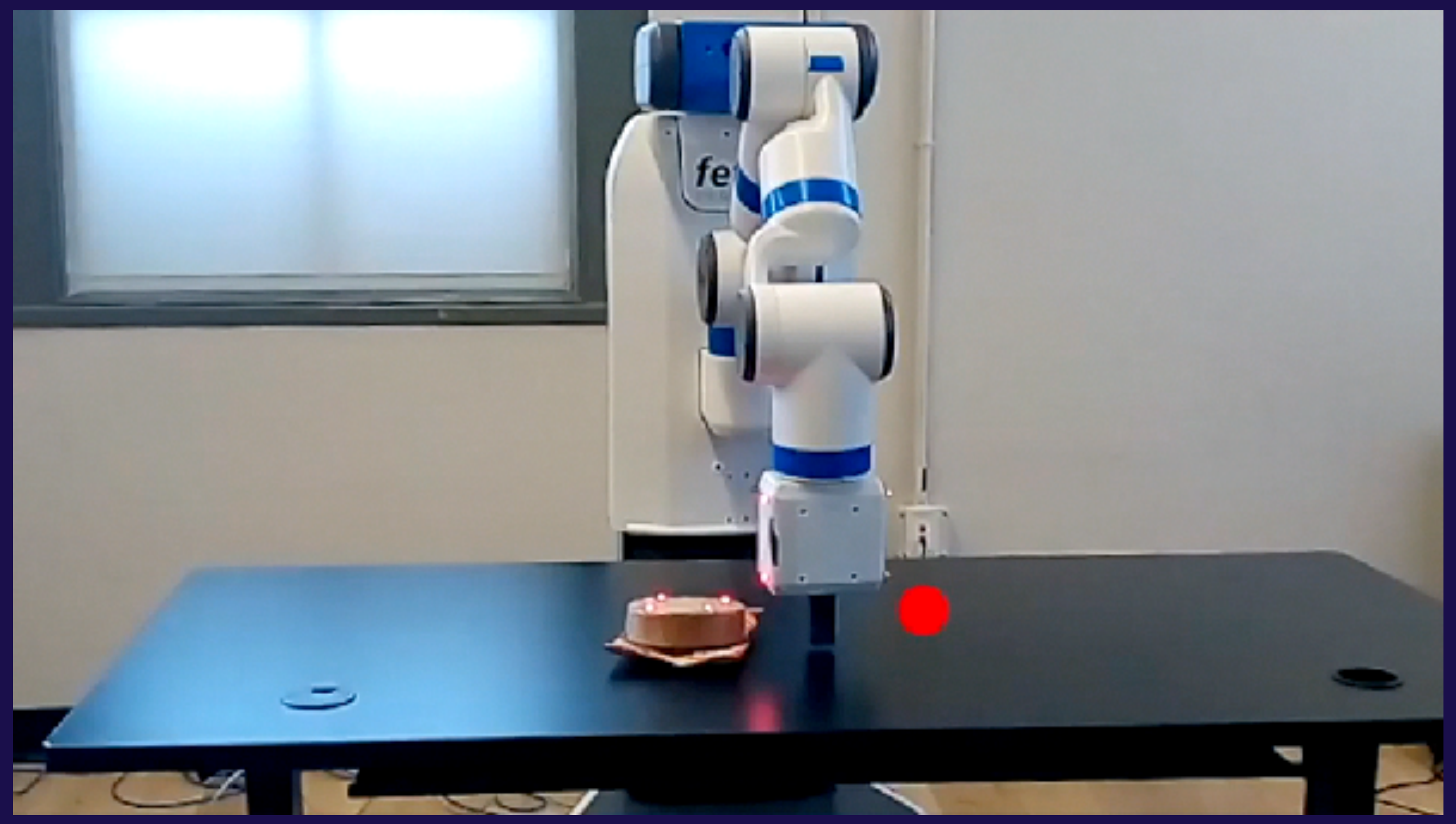

O
A

Domain Randomization



F Sadeghi, S Levine (2017)



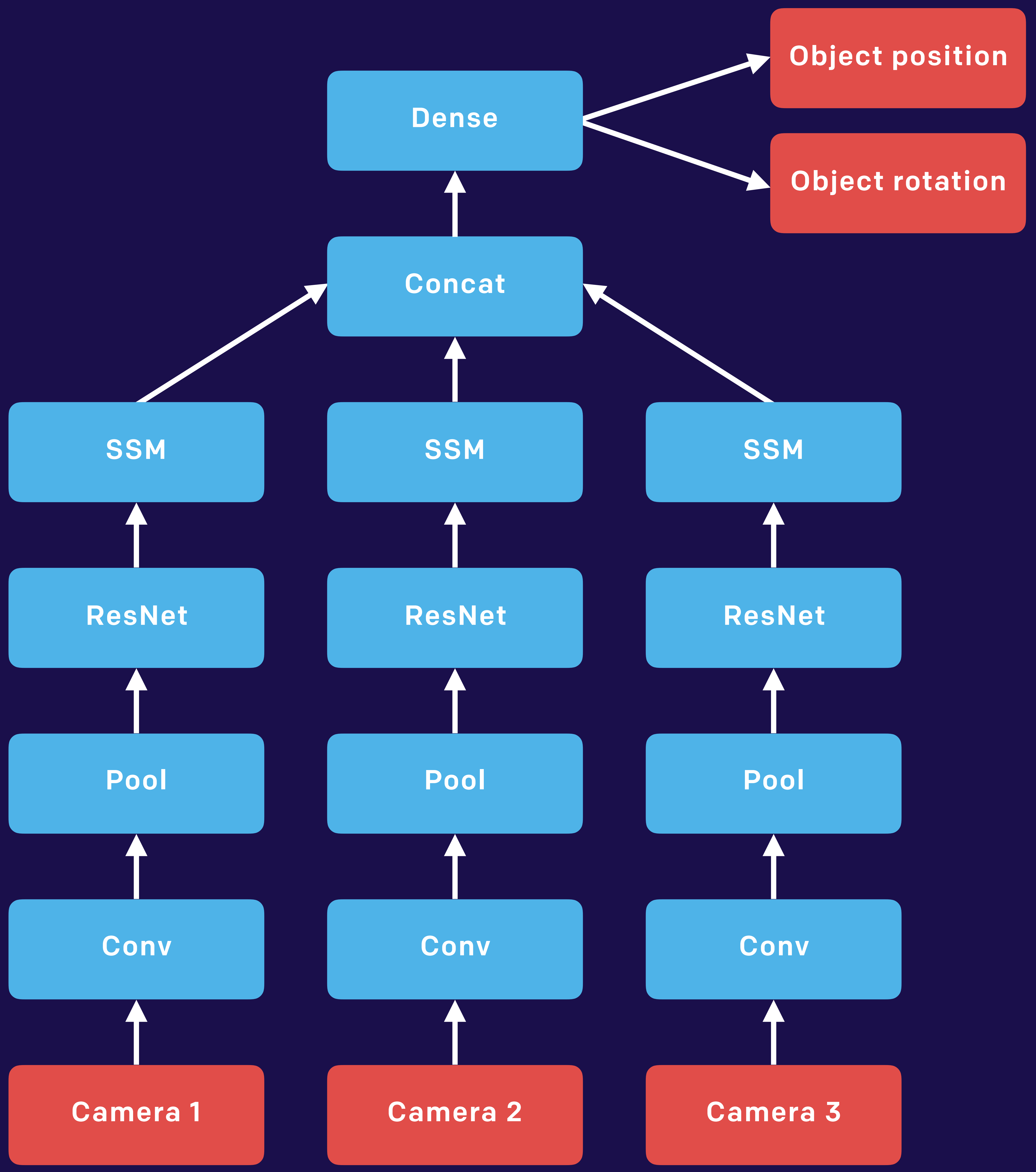


Peng et al. (2018)

Tobin et al. (2017)

Physics Randomizations

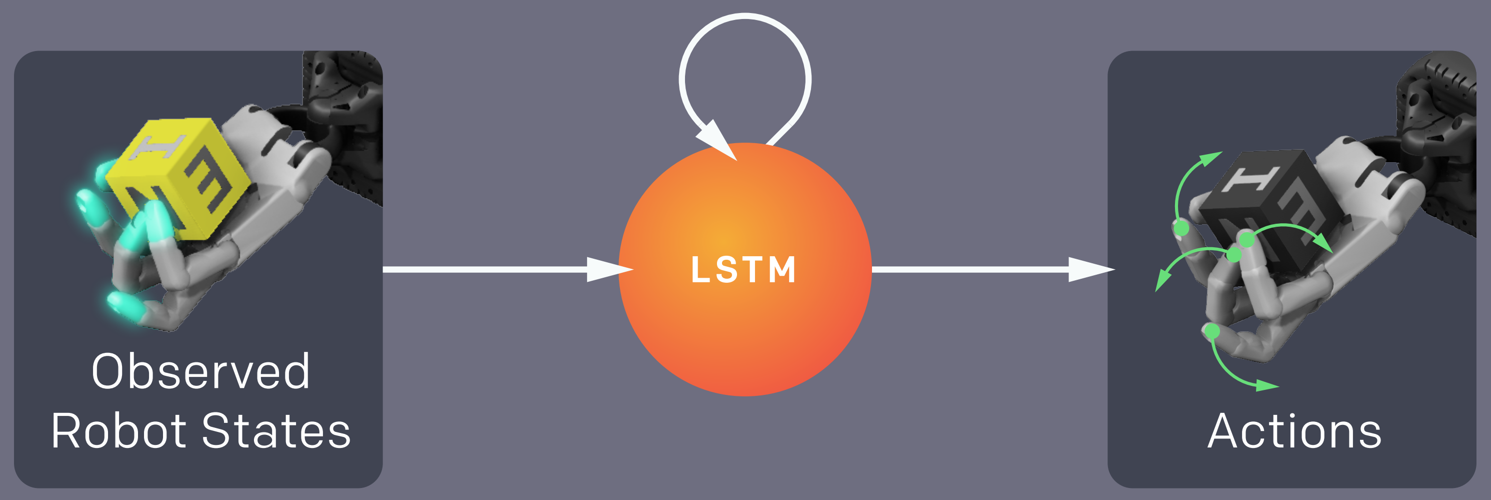
- object dimensions
- object and robot link masses
- surface friction coefficients
- robot joint damping coefficients
- actuator force gains
- joint limits
- gravity vector


Object position
Object rotation
Dense
Concat
SSM
SSM
SSM
ResNet
ResNet
ResNet
Pool
Pool
Pool
Conv
Conv
Conv
Camera 1
Camera 2
Camera 3

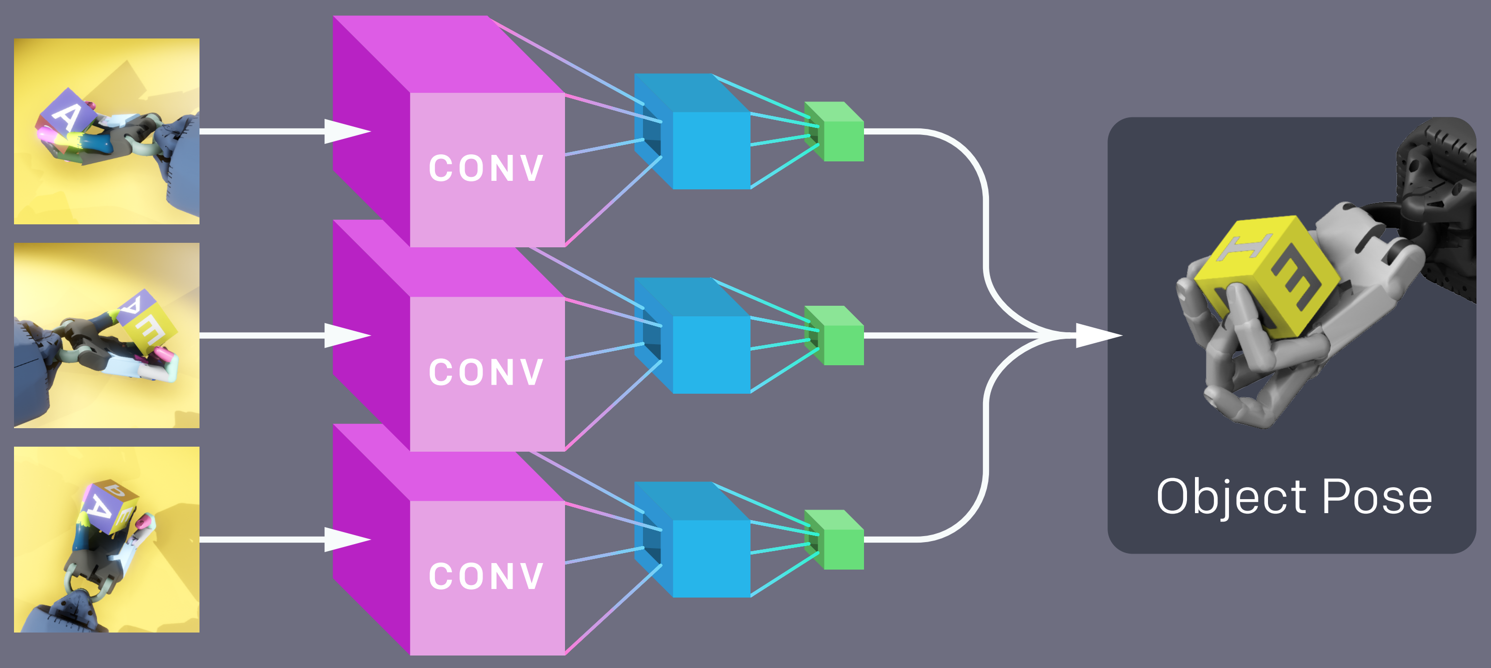
Train in Simulation

A Distributed workers collect experience on randomized environments at large scale.

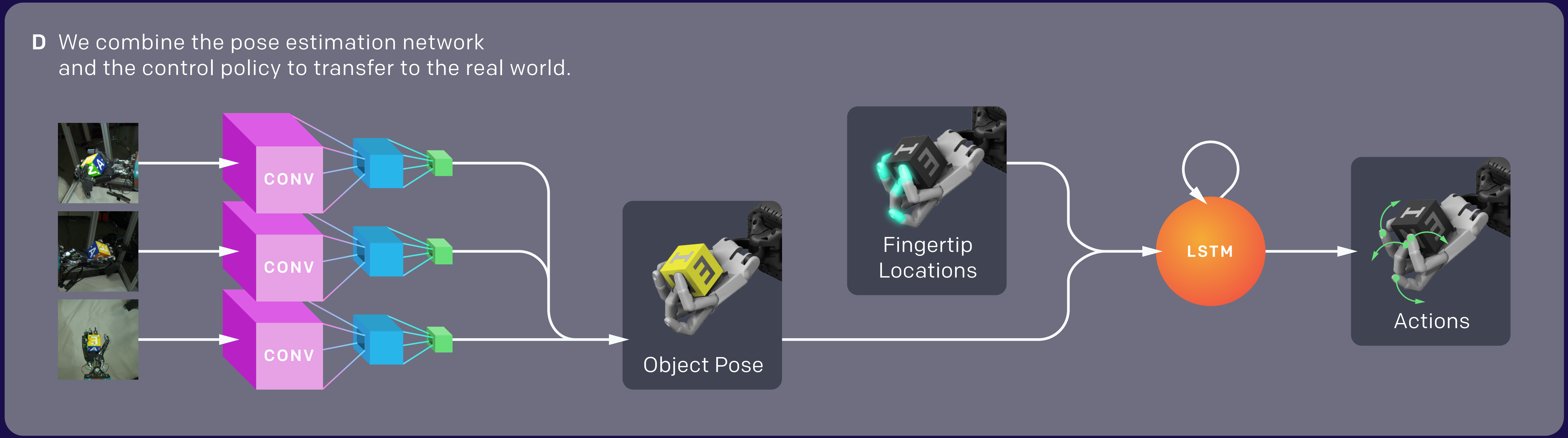
B We train a control policy using reinforcement learning. It chooses the next action based on fingertip positions and the object pose.



C We train a convolutional neural network to predict the object pose given three simulated camera images.



Transfer to the Real World

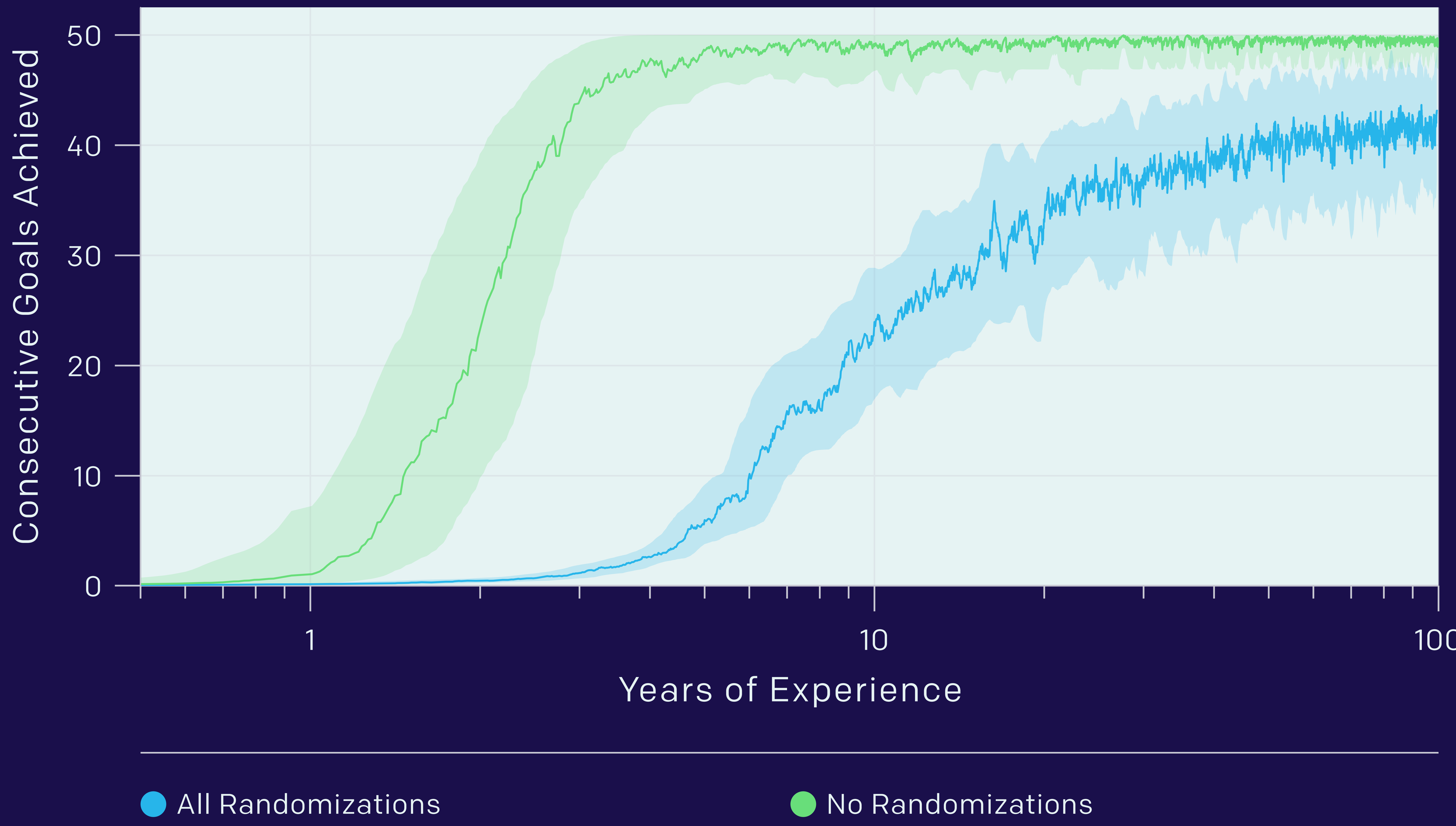


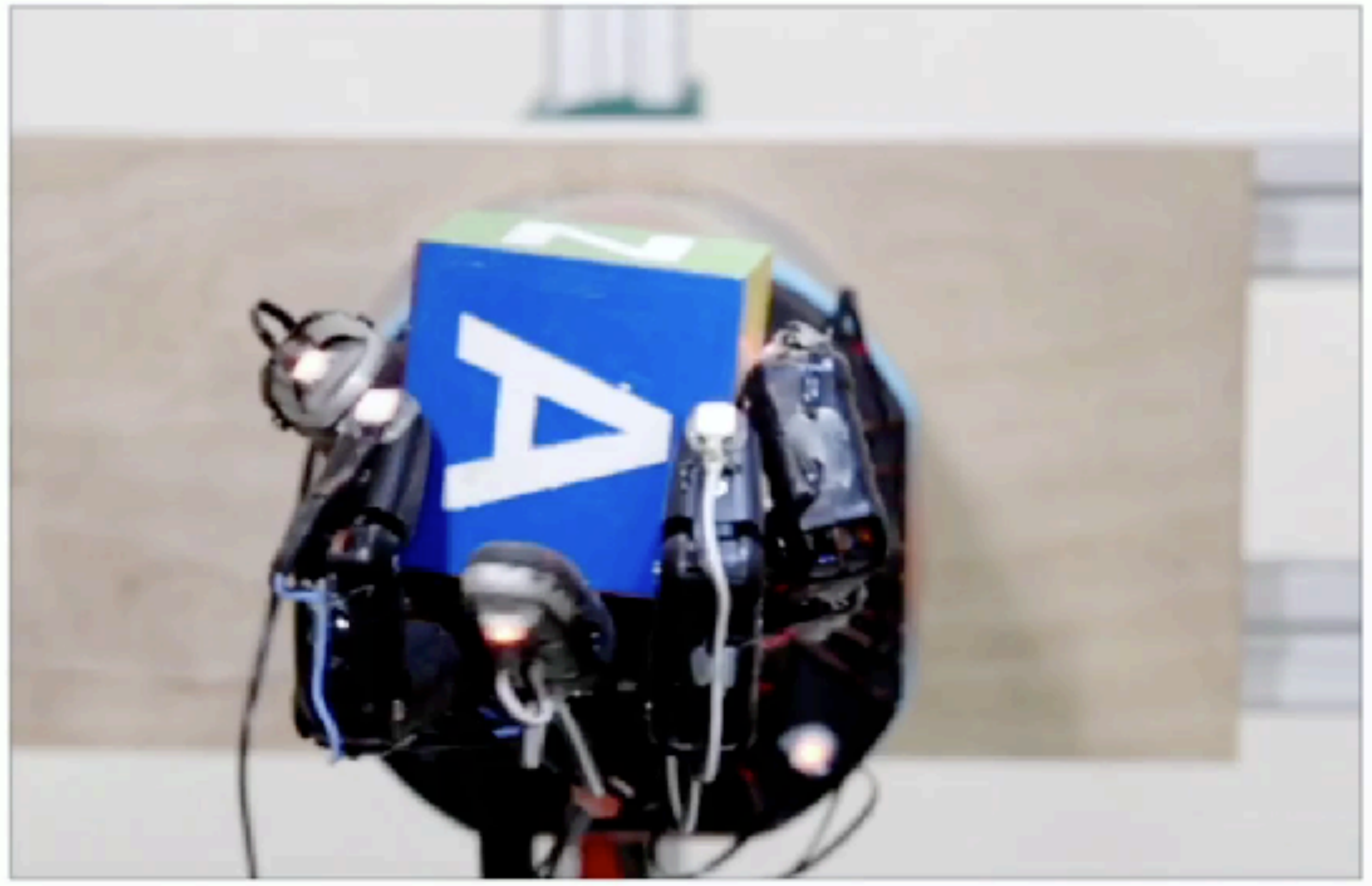
Results

Results

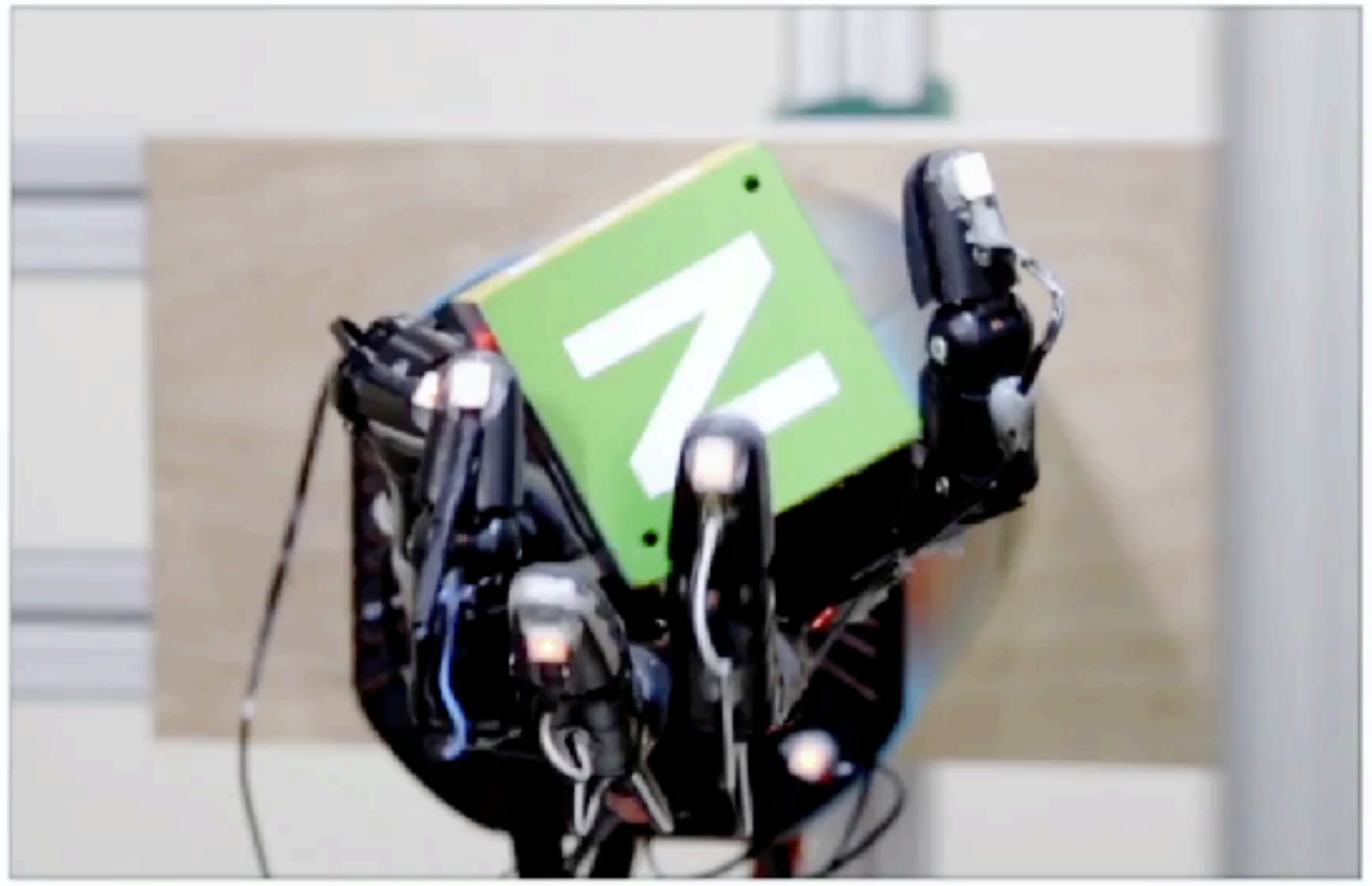
RANDOMIZATONS	OBJECT TRACKING	MAX NUMBER OF SUCCESSES	MEDIAN NUMBER OF SUCCESSES
All	Vision	46	11.5
All	Motion tracking	50	13
None	Motion tracking	6	0

Training time

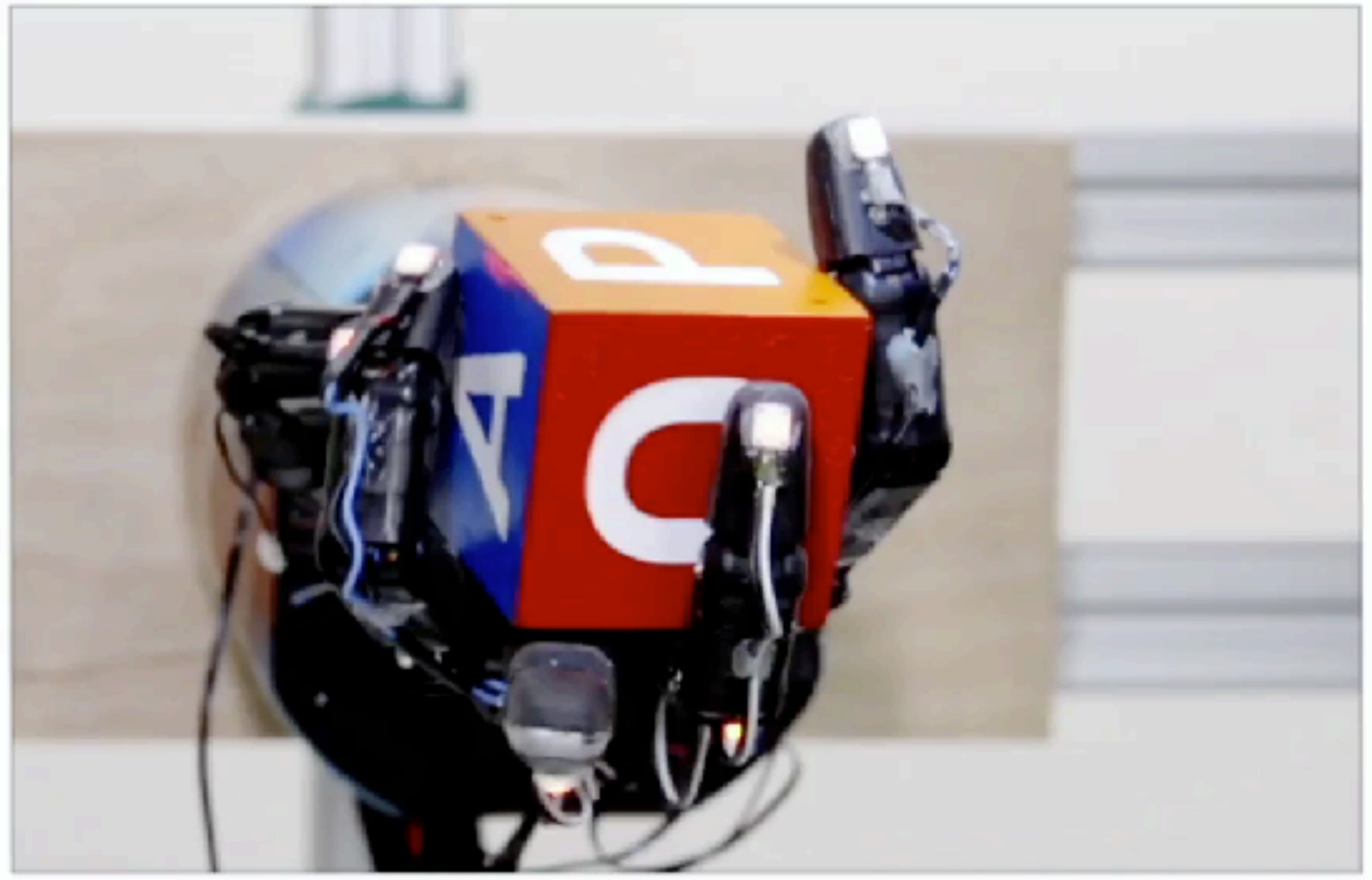




FINGER PIVOTING

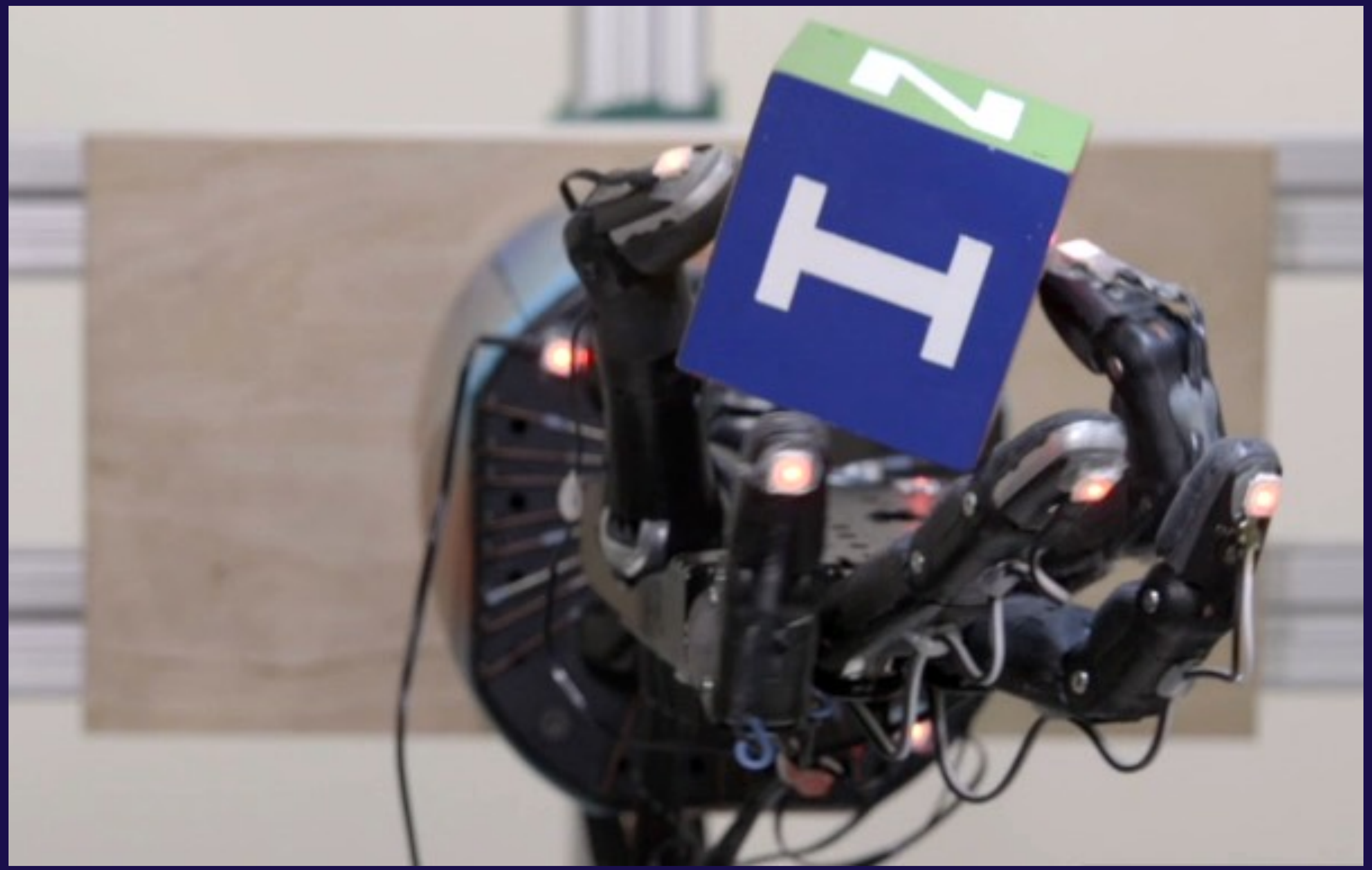


SLIDING

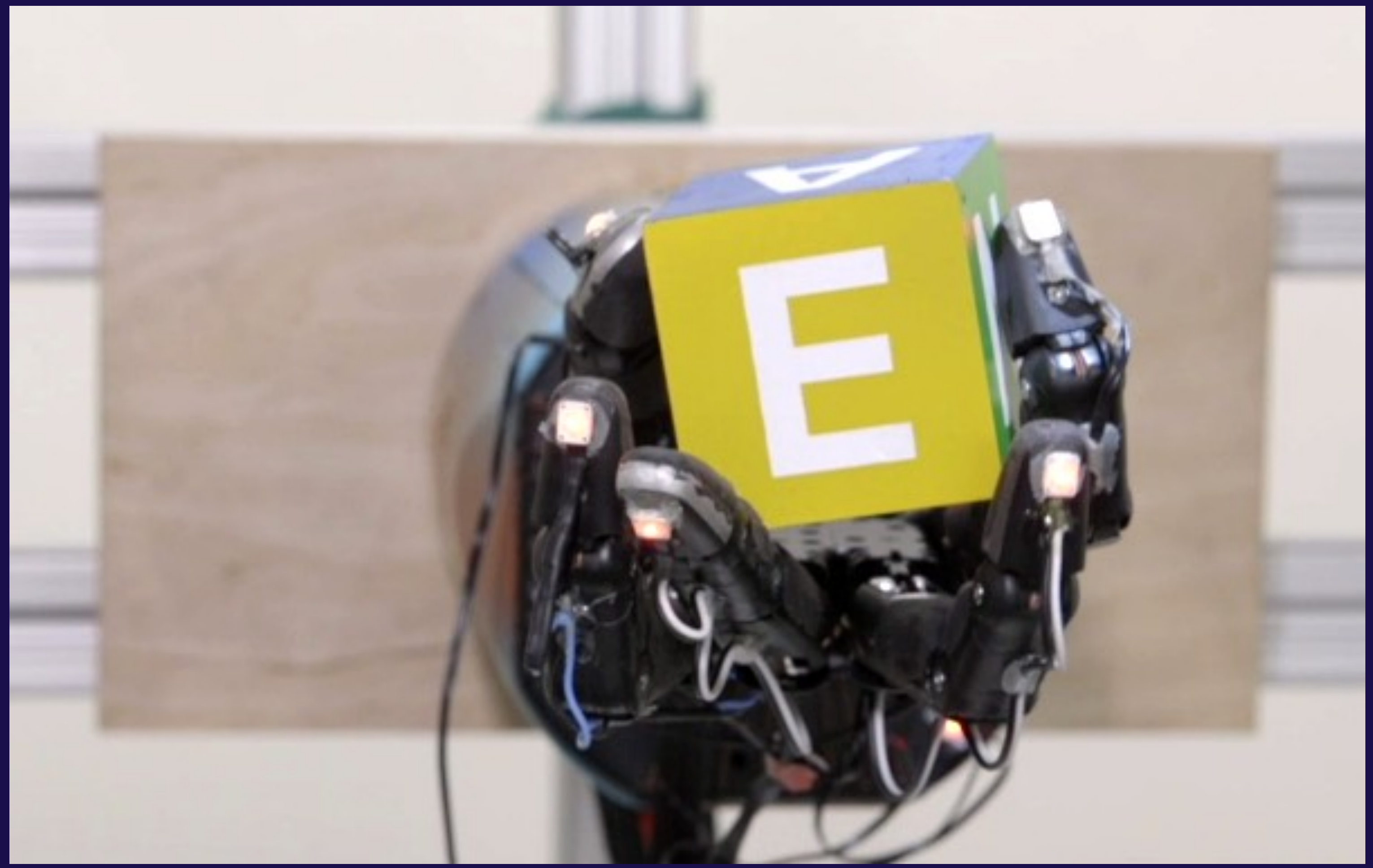


FINGER GAITING

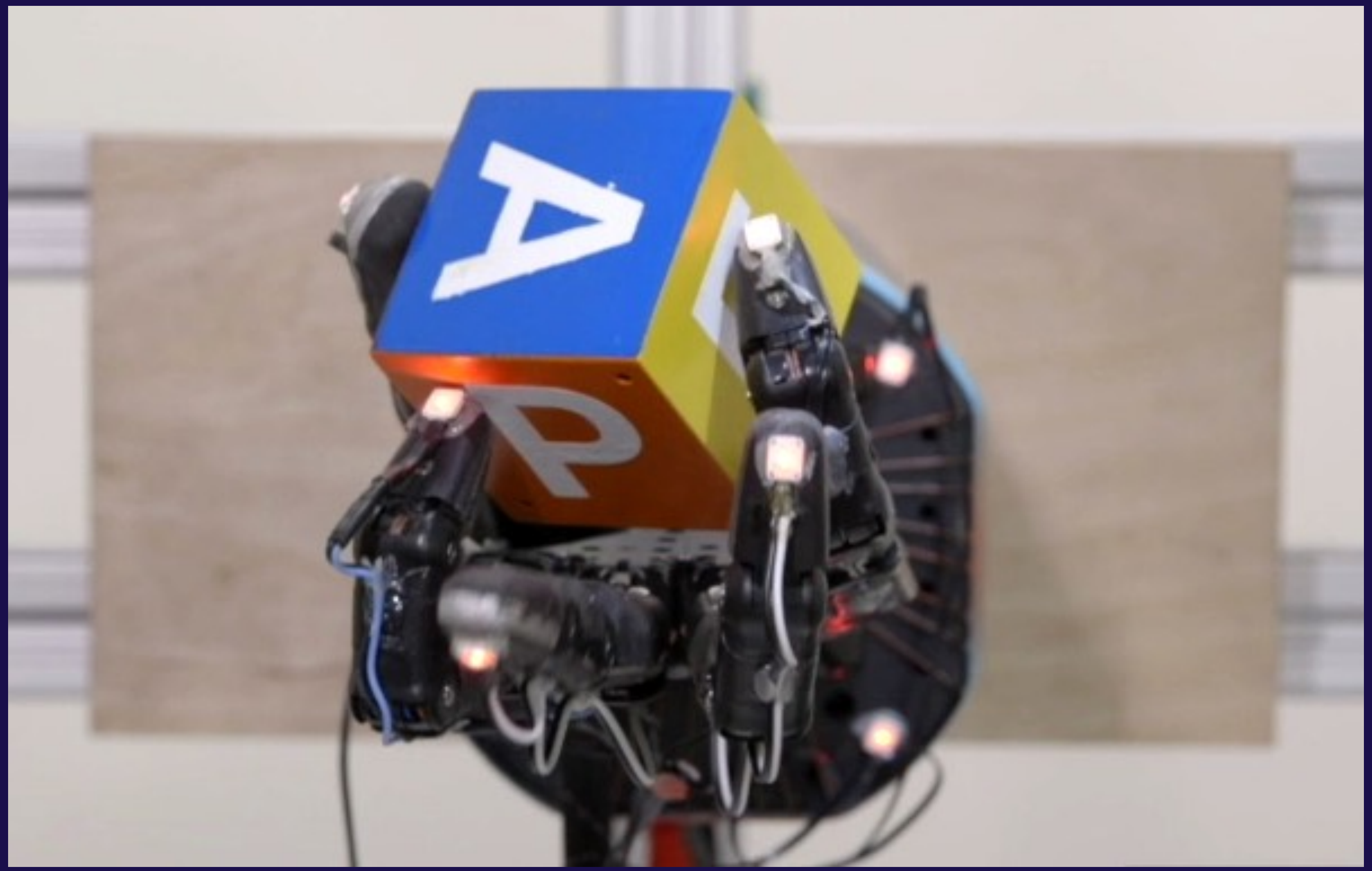
Tip Pinch



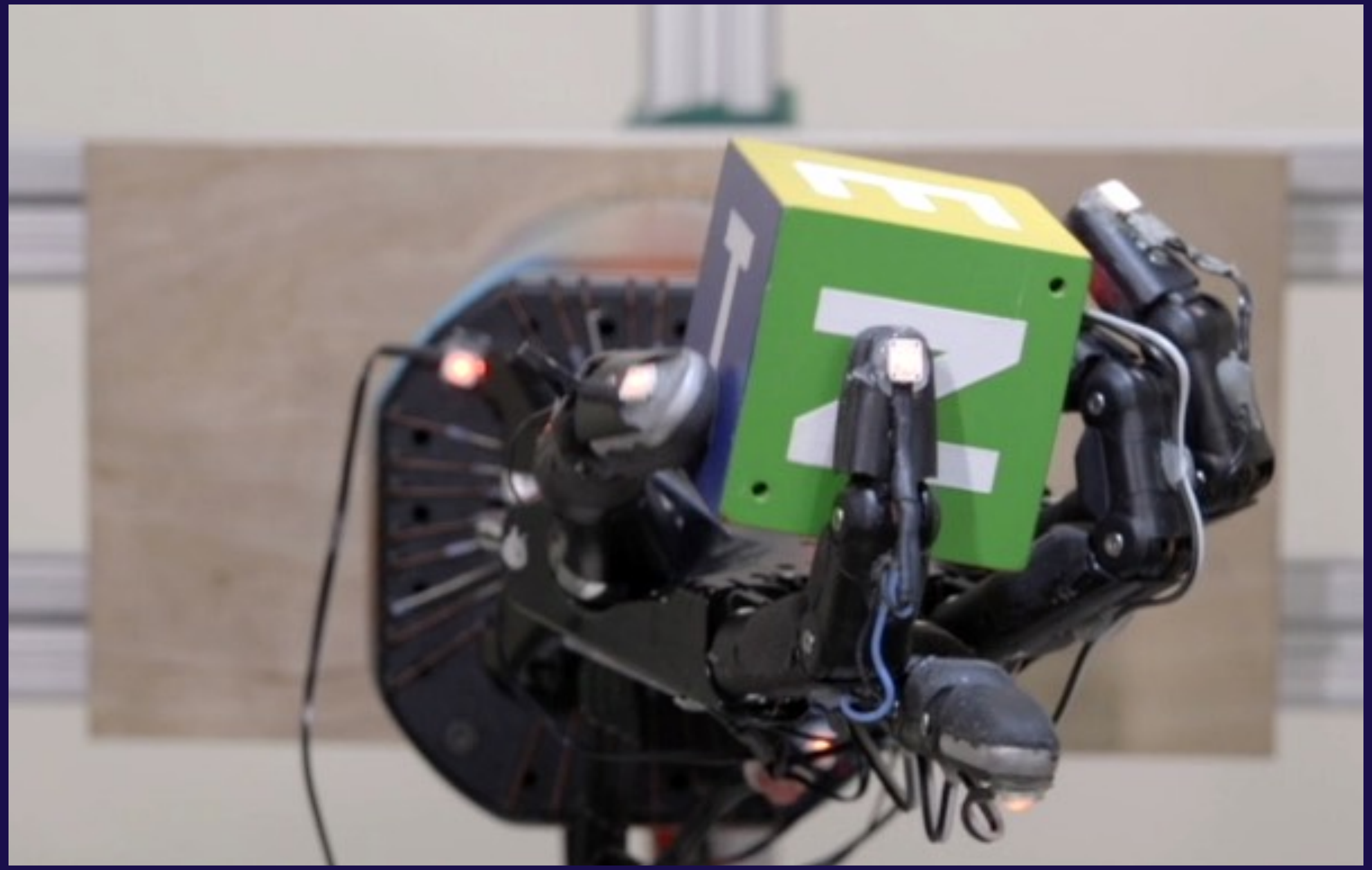
Palmar Pinch



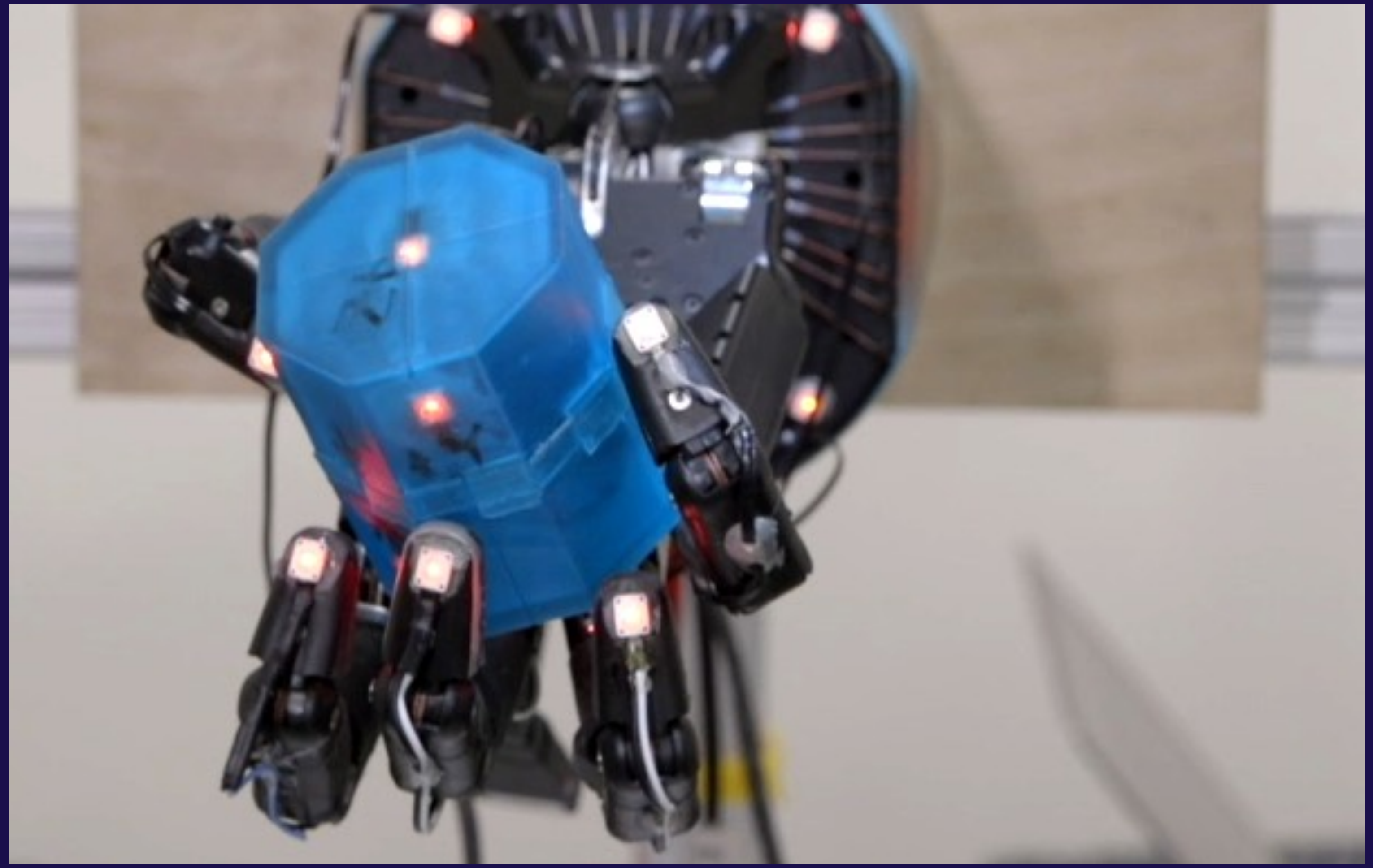
Tripod



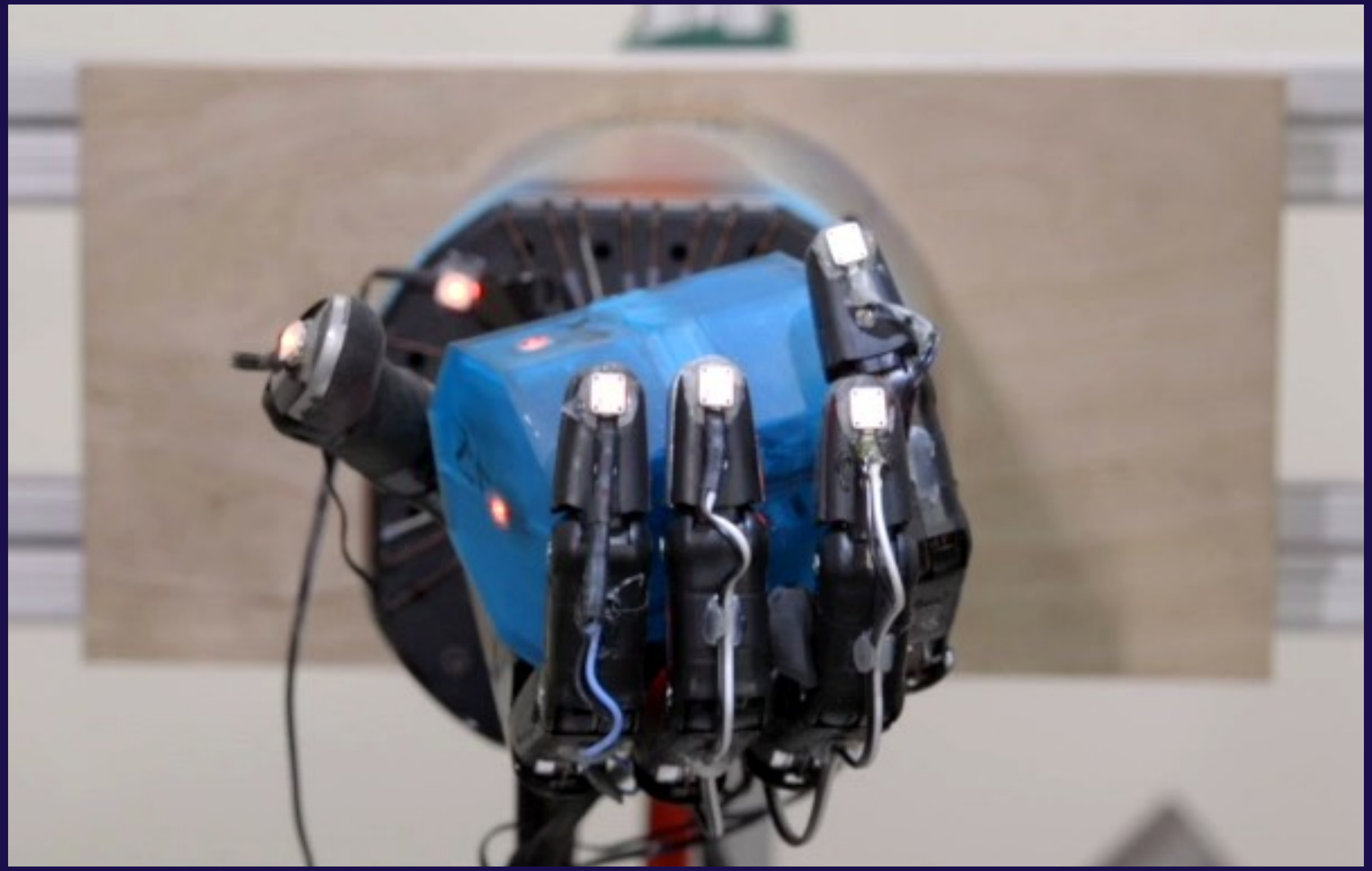
Quadpod



Power Grasp



5-finger Precision Grasp



Thank You

Visit openai.com for more information.

FOLLOW @OPENAI ON TWITTER