OpenAI

Learning Dexterity

Matthias Plappert

SEPTEMBER 6, 2018

OpenAI

“OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence.”

OpenAI

"OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence."

OpenAI

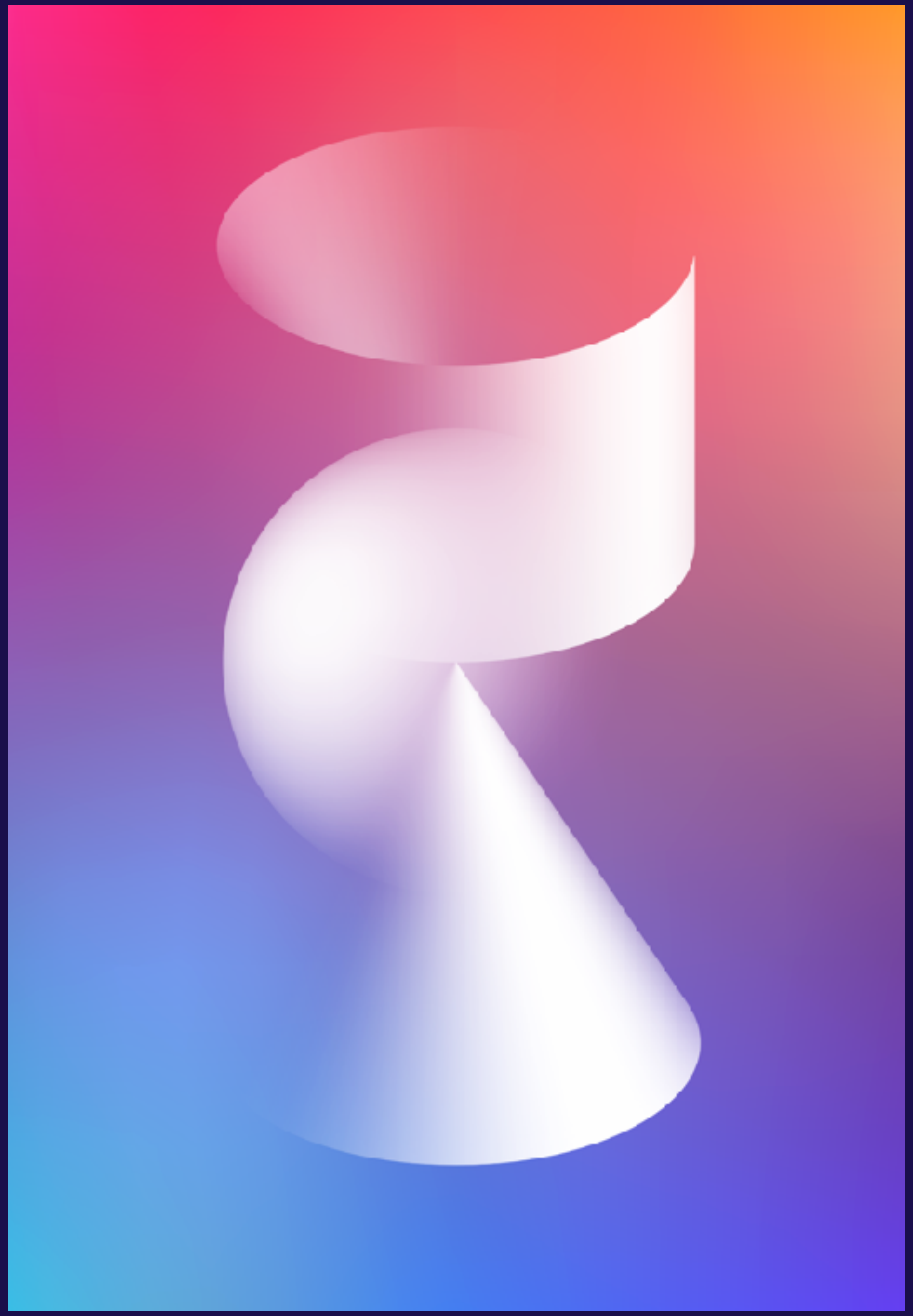
"OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence."

OpenAI

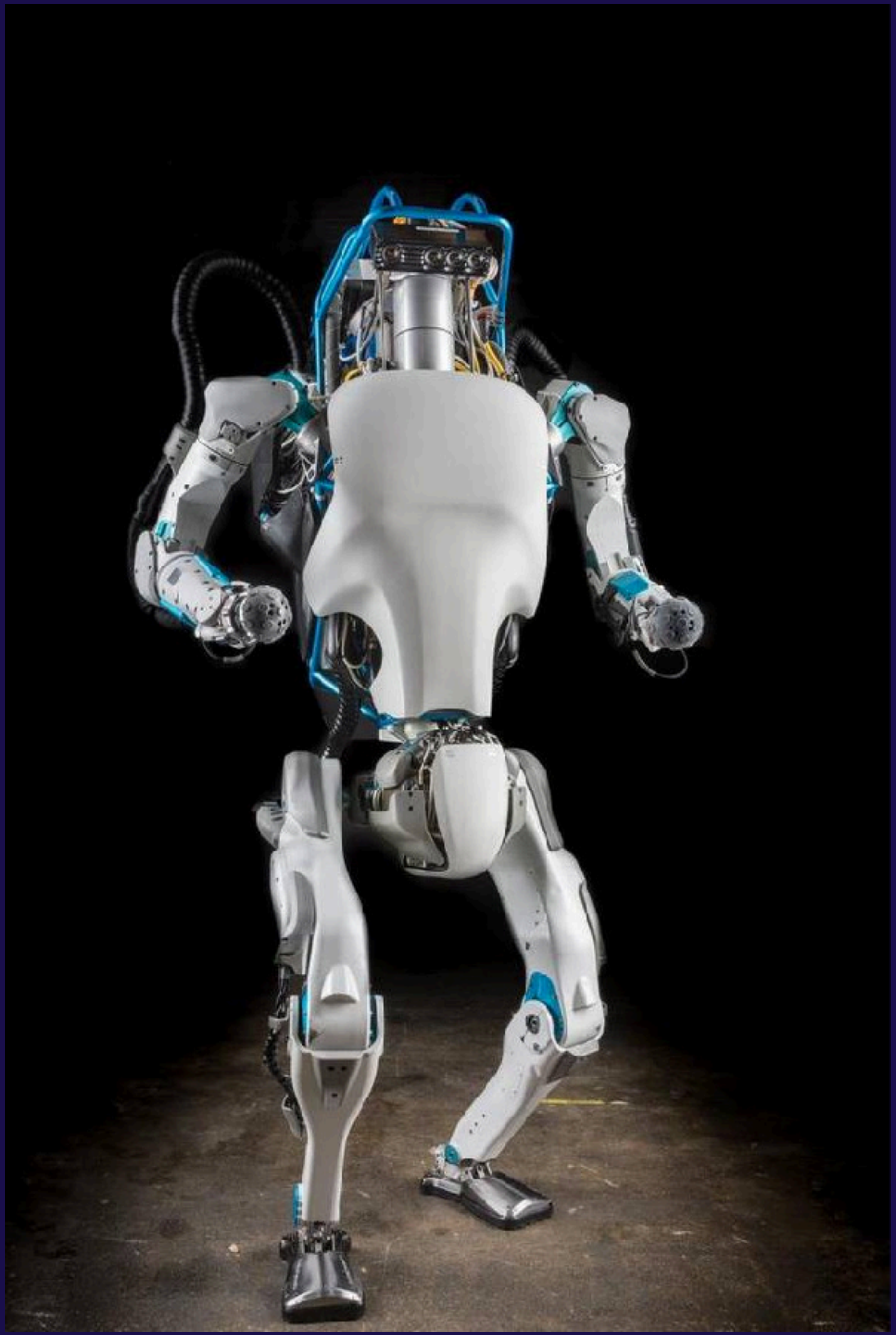
"OpenAI is a non-profit AI research company, discovering and enacting the path to safe artificial general intelligence."

OpenAI Charter

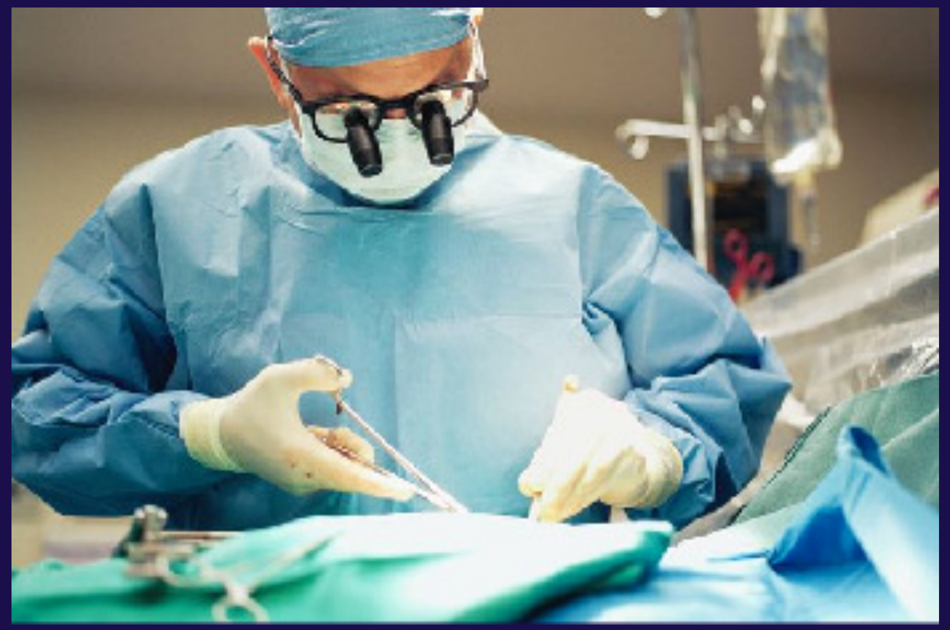
- Broadly Distributed Benefits
- Long-Term Safety
- Technical Leadership
- Cooperative Orientation
- Full text available on our blog: https://blog.openai.com/openai-charter/



Learning Dexterity

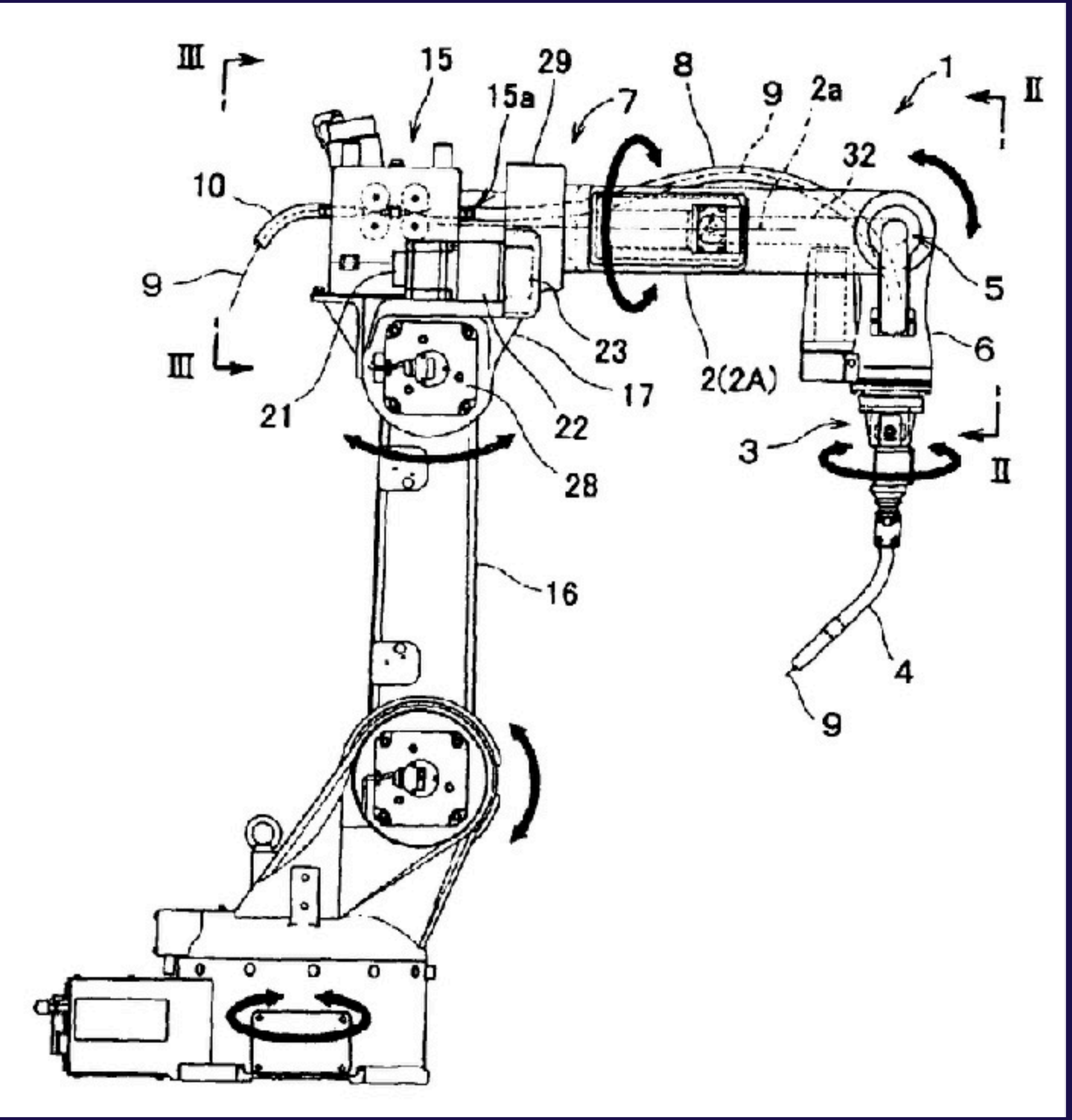



FRAGILE
FRAGILE

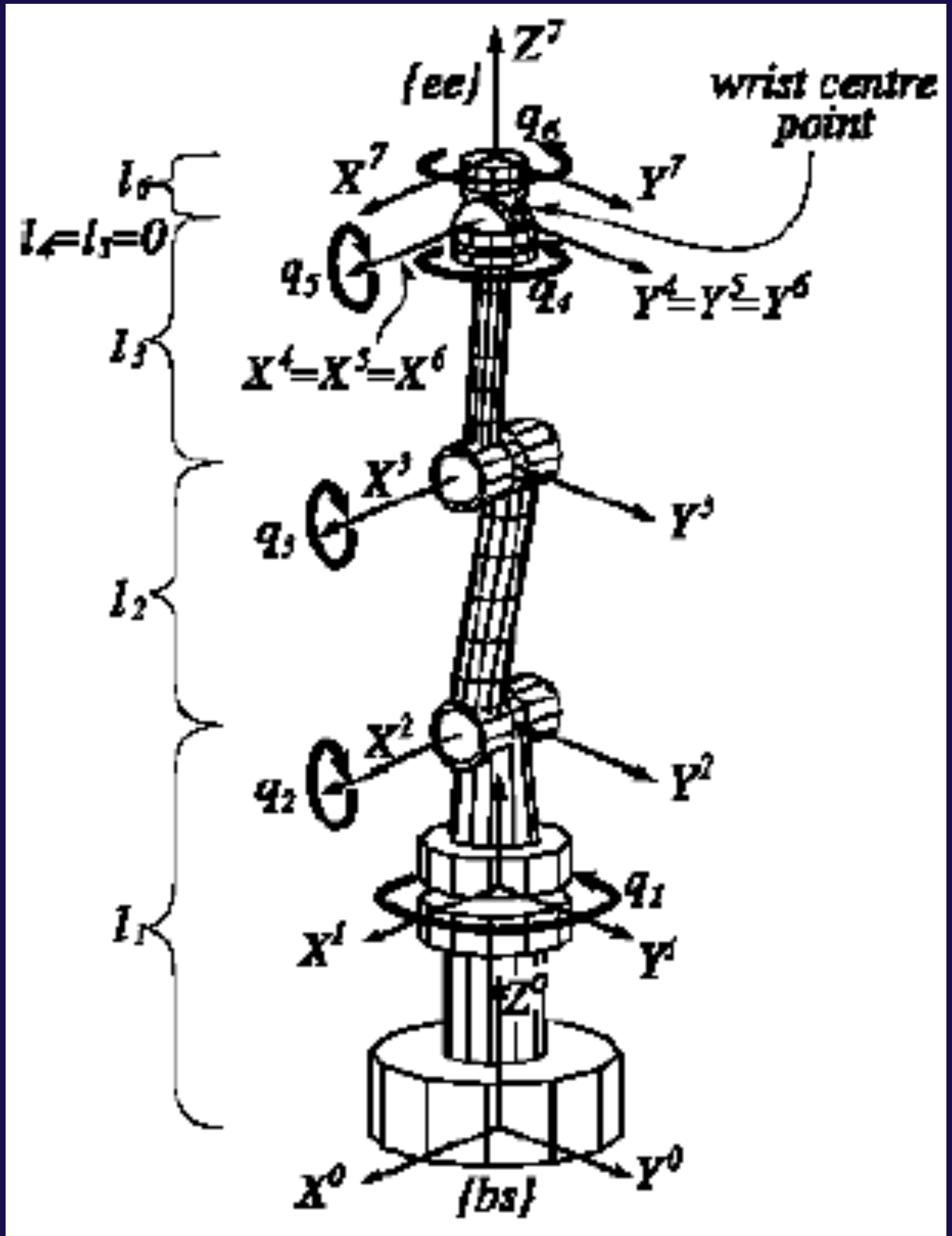






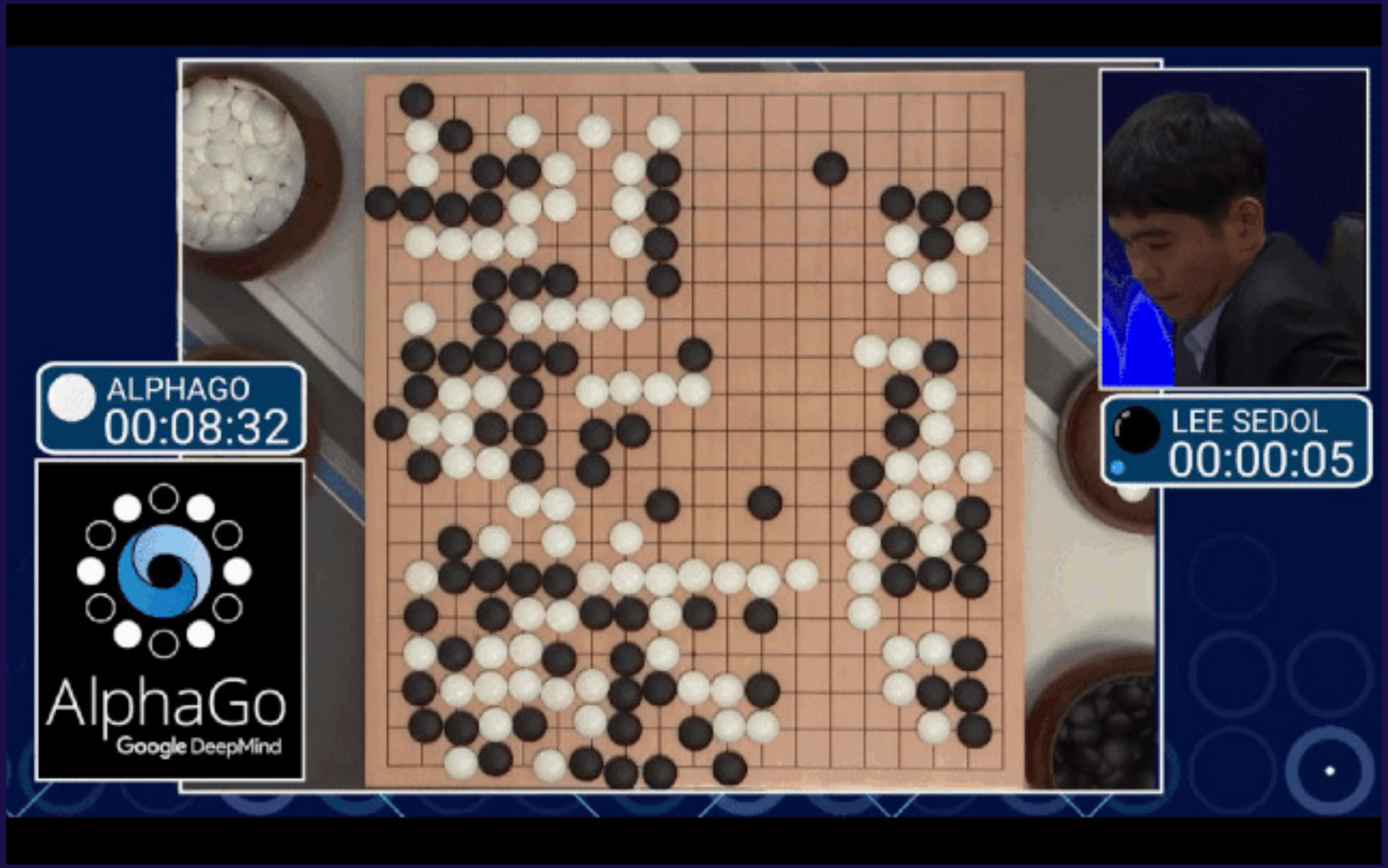


III
15
29
8
9
2a
1
II
15a
7
10
32
9
5
23
6
III
21
22
17
2(2A)
3
II
28
16
4
9



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

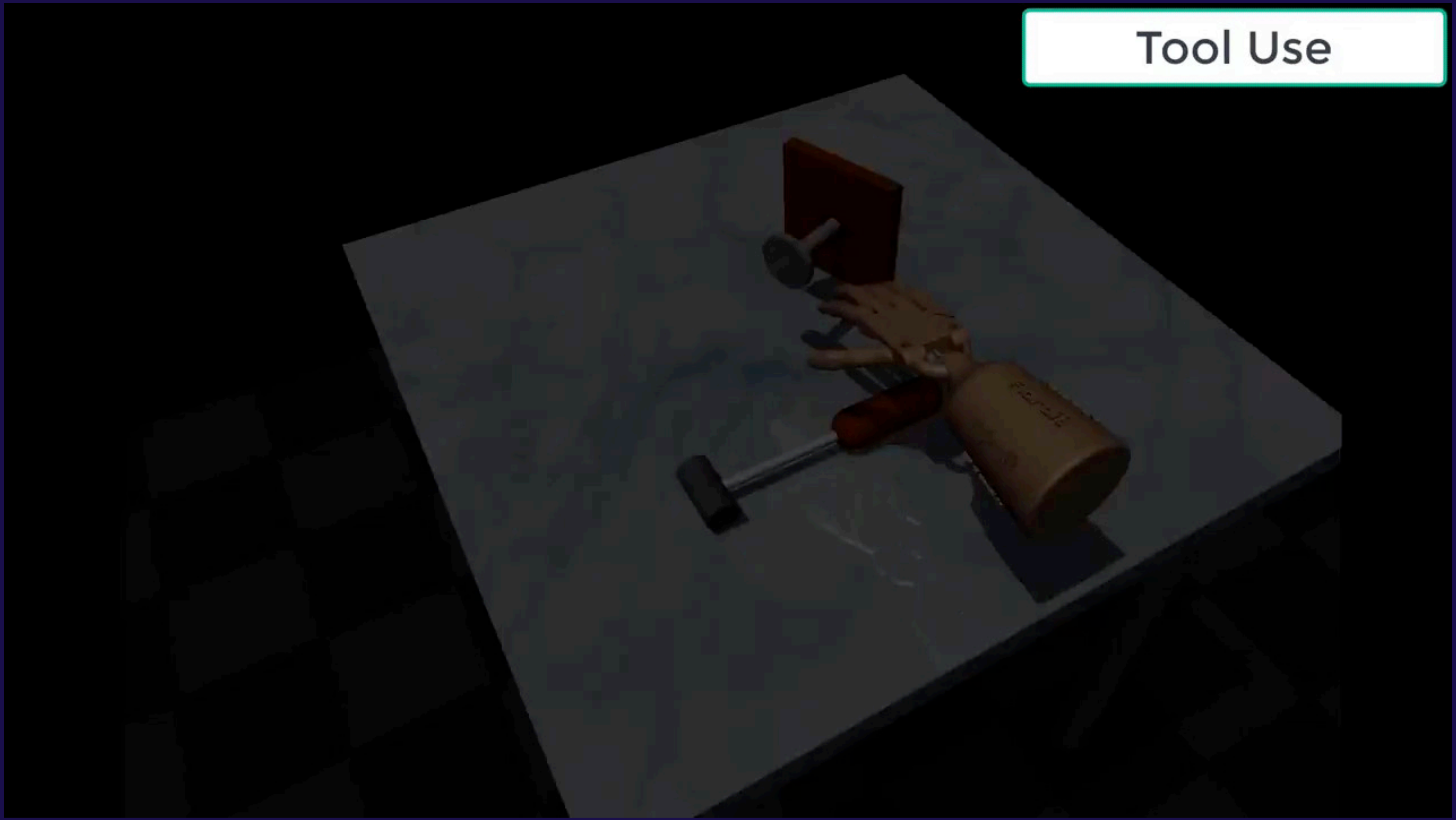
Reinforcement Learning



GO (ALPHAGO ZERO)

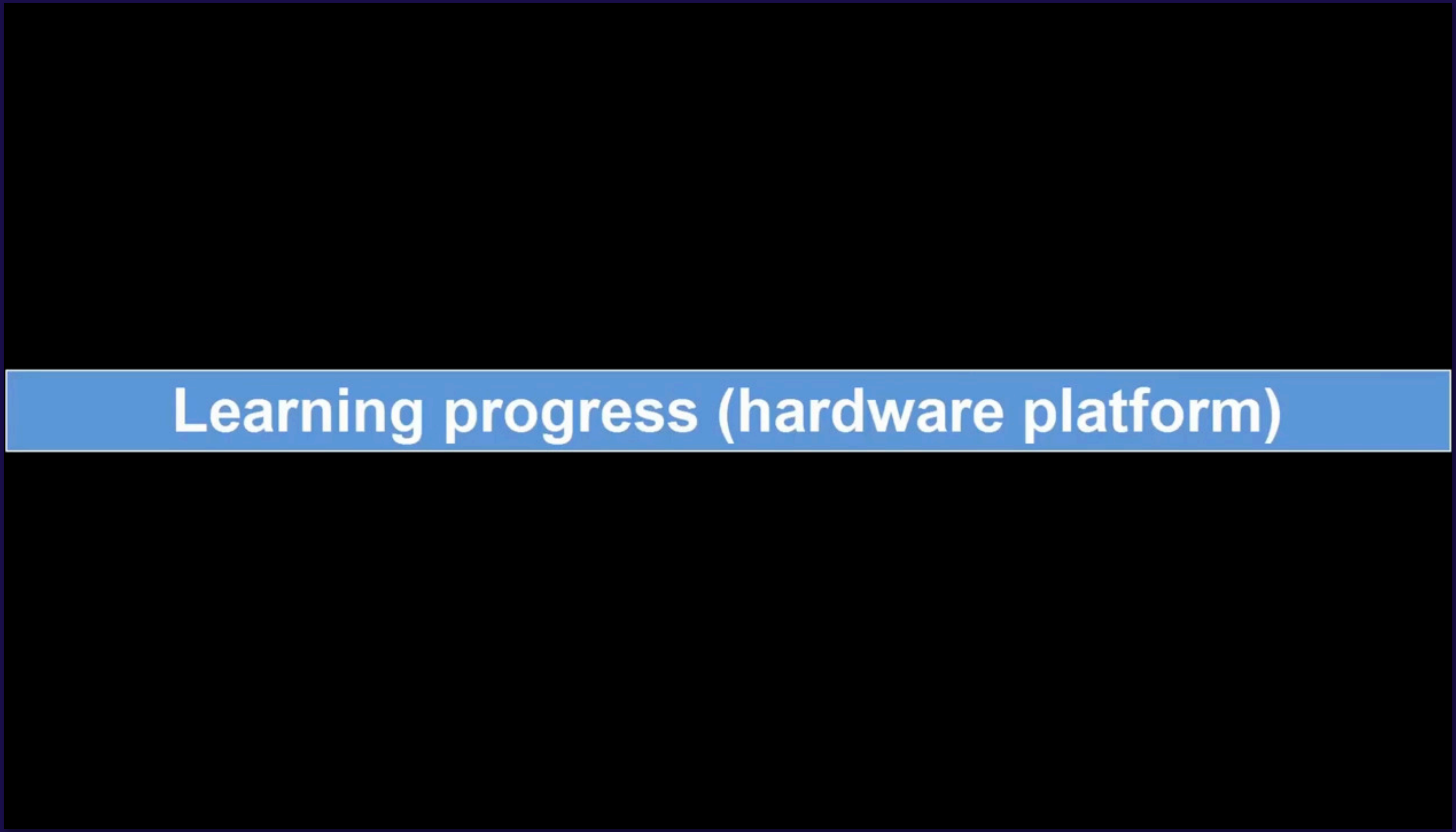
DOTA 2 (OPENAI FIVE)

Reinforcement Learning for Robotics (1)



Rajeswaran et al. (2017)

Reinforcement Learning for Robotics (2)

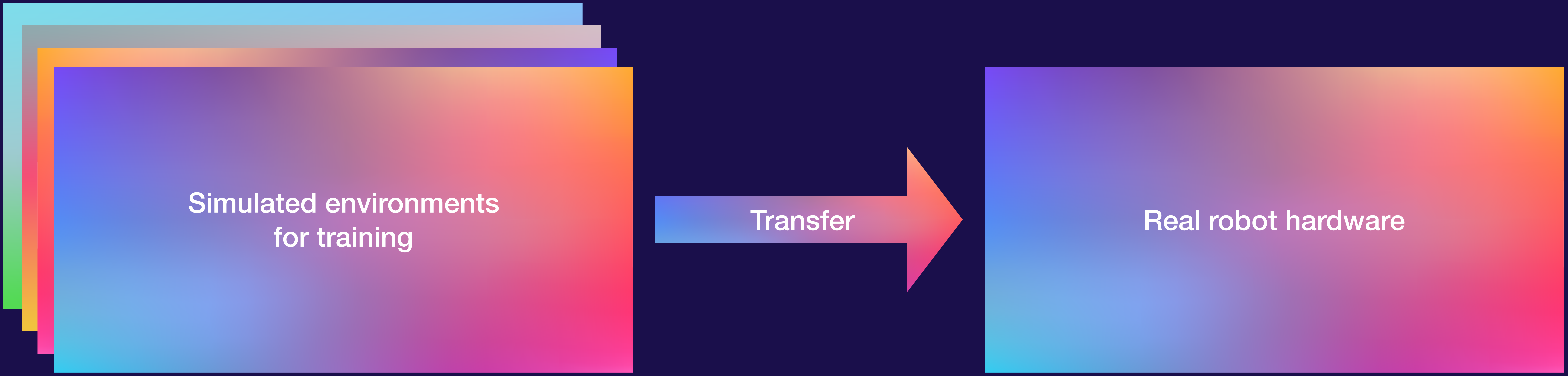


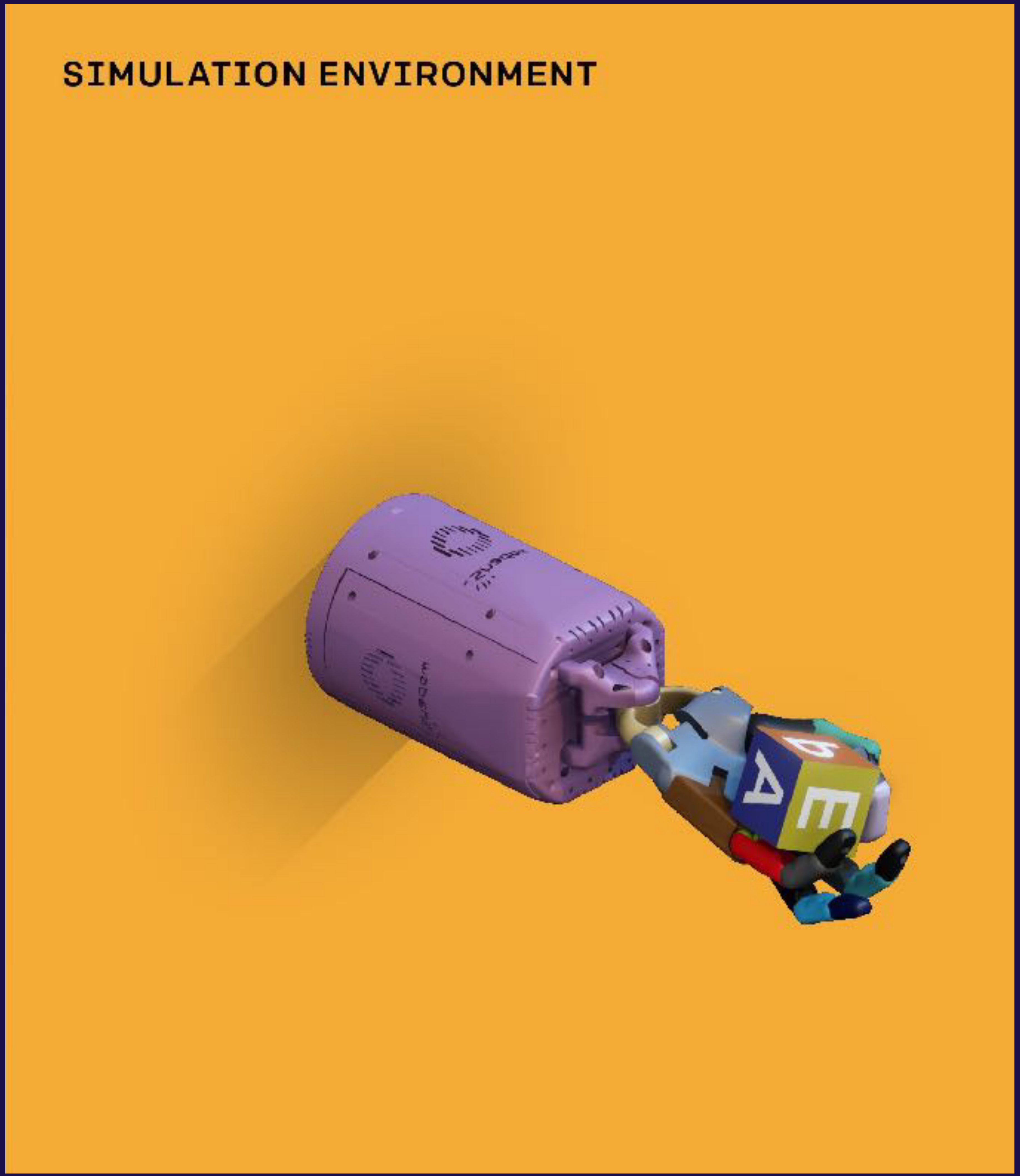
Kumar et al. (2016)

Reinforcement Learning for Robotics (3)

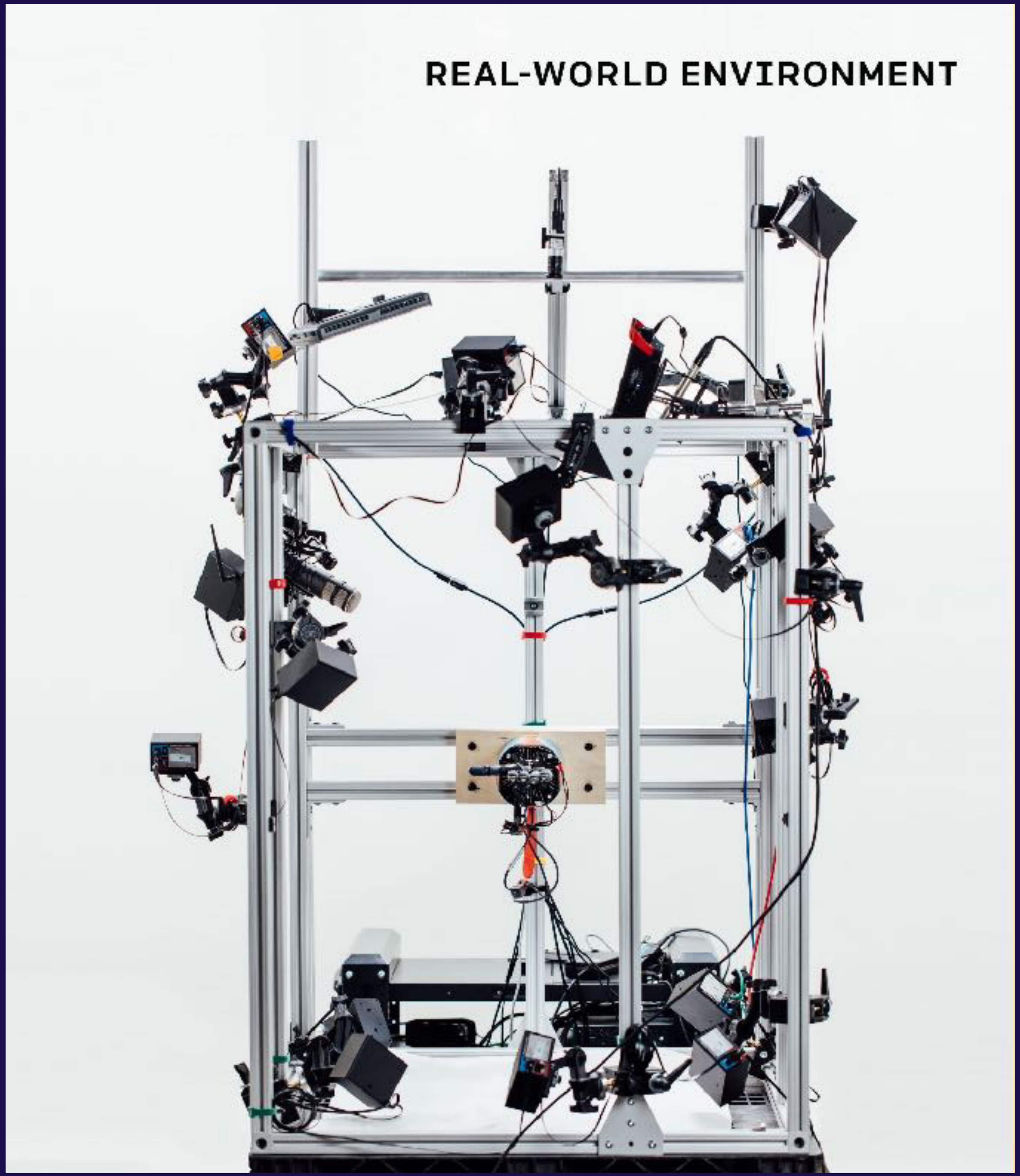
Levine et al. (2018)

Sim2Real

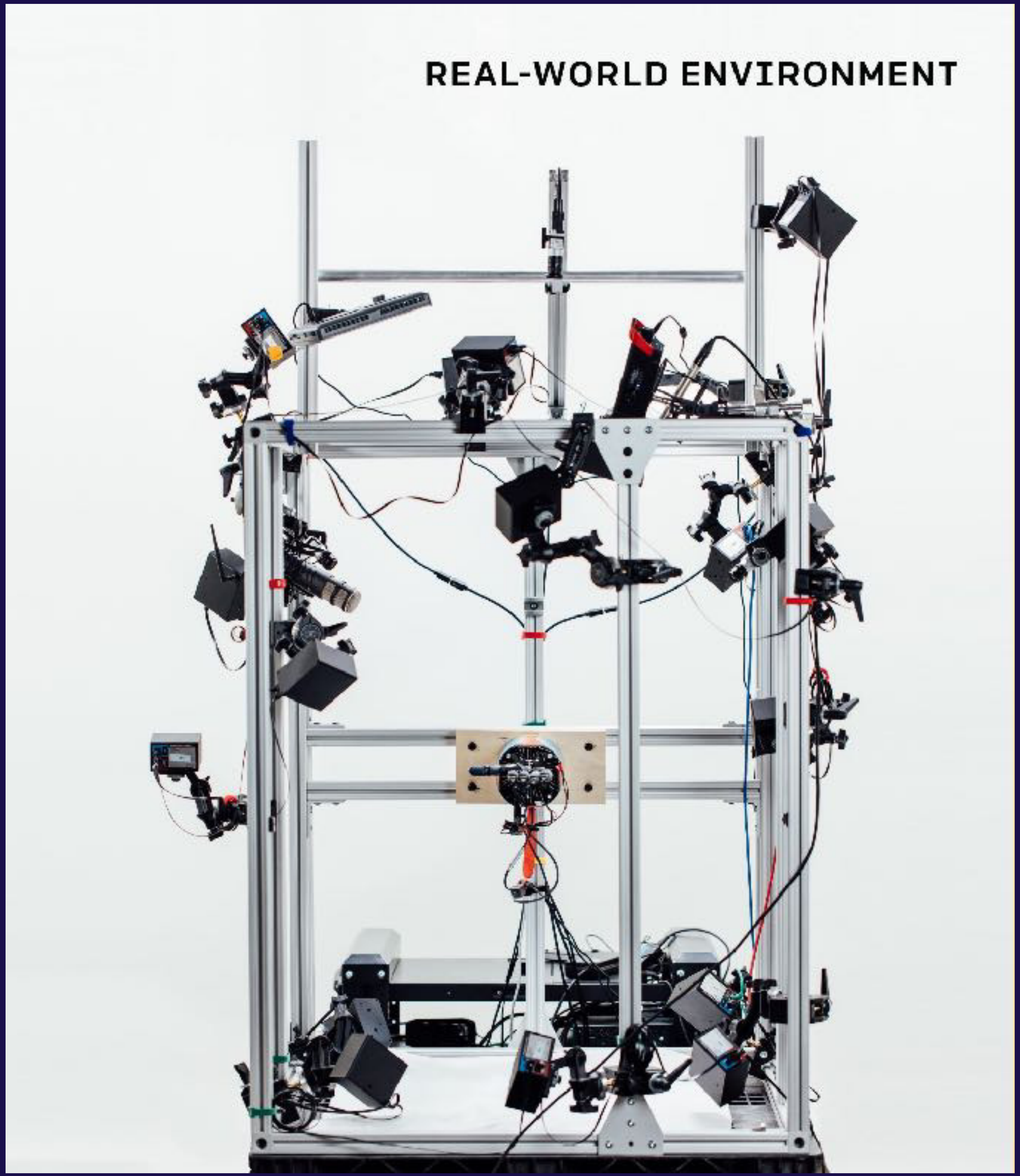

Simulated environments
for training
Transfer
Real robot hardware



Transfer

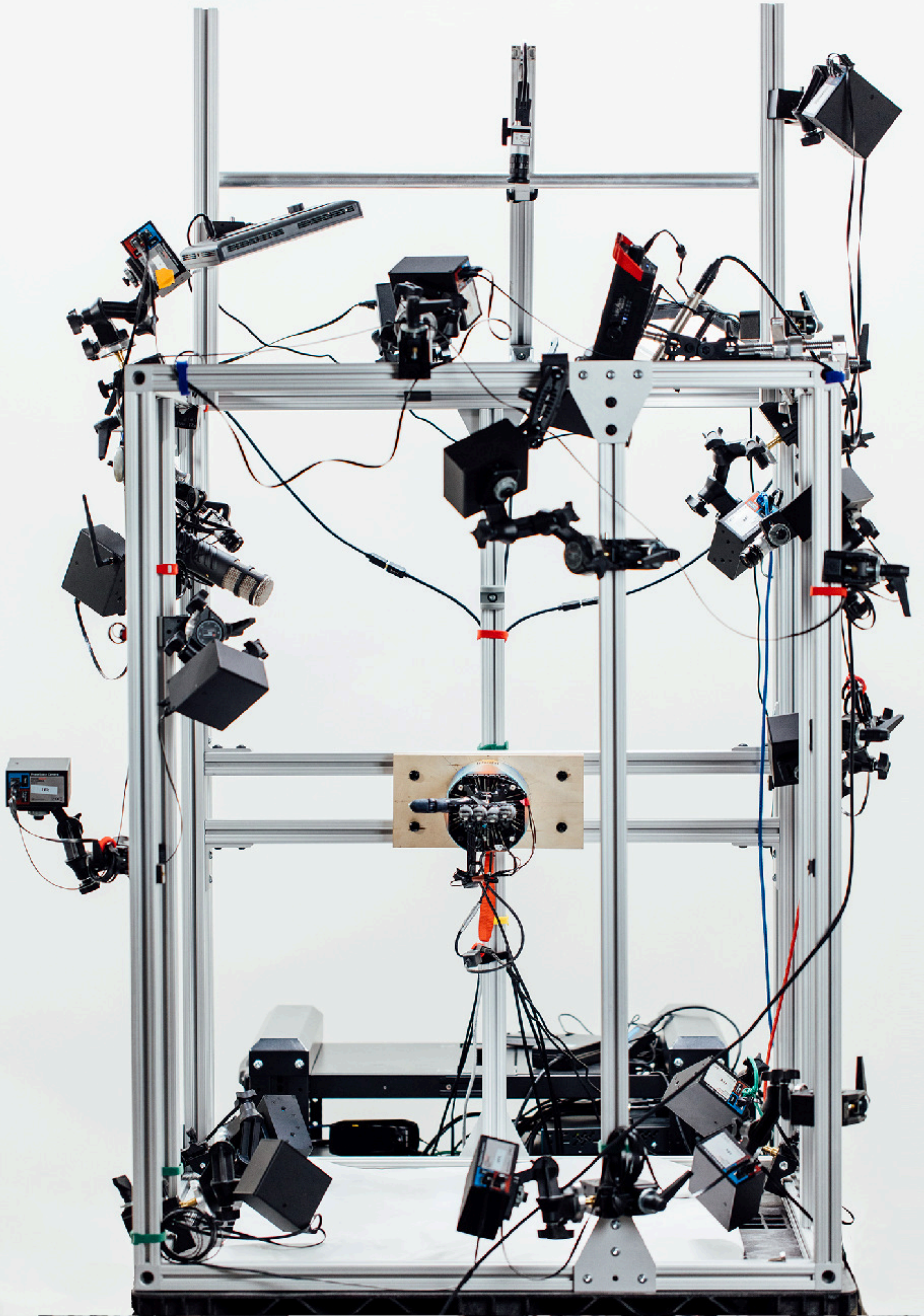


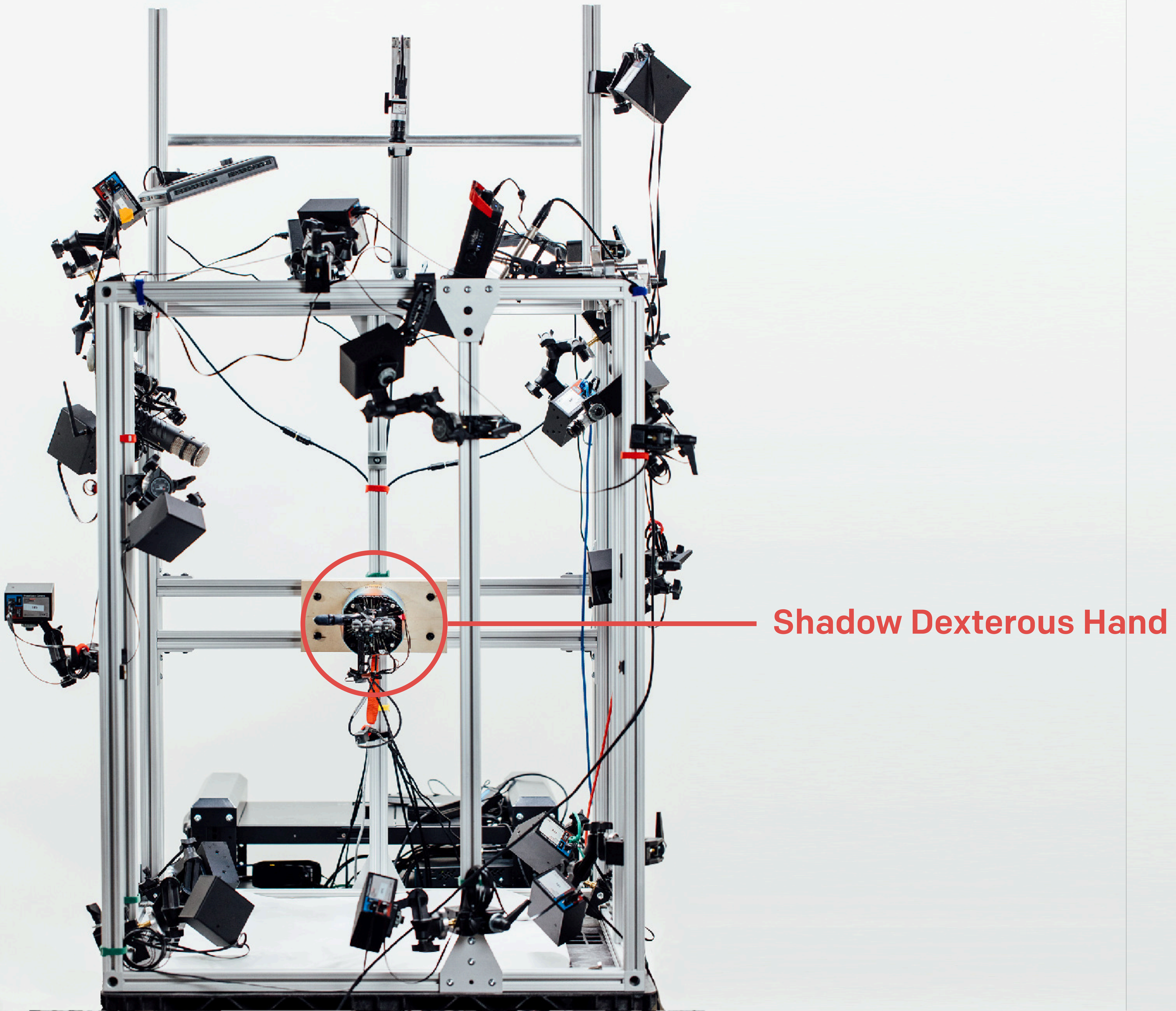
Transfer

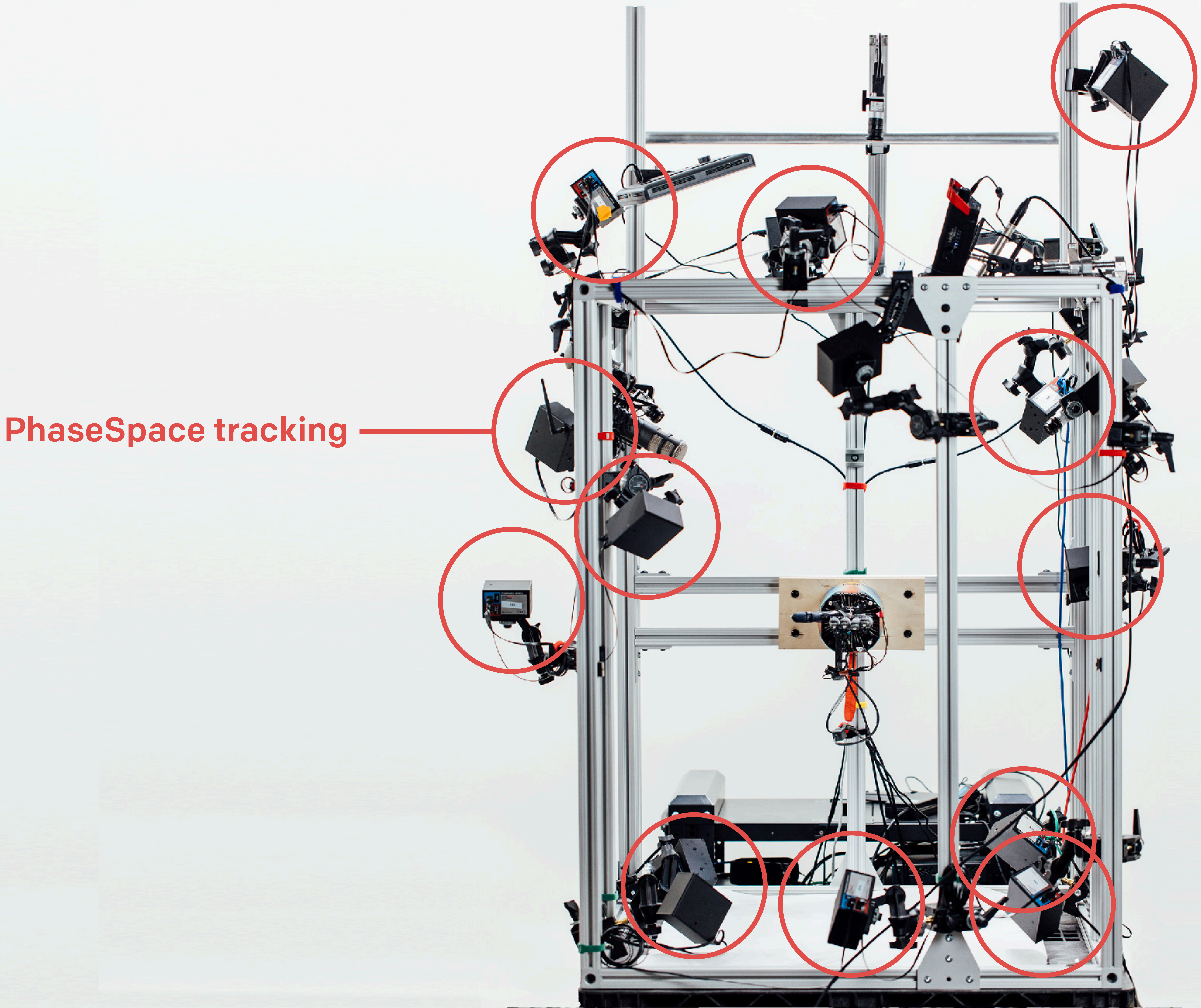


Task & Setup


O
A




Shadow Dexterous Hand


PhaseSpace tracking


Top RGB camera
Right RGB camera
Left RGB camera

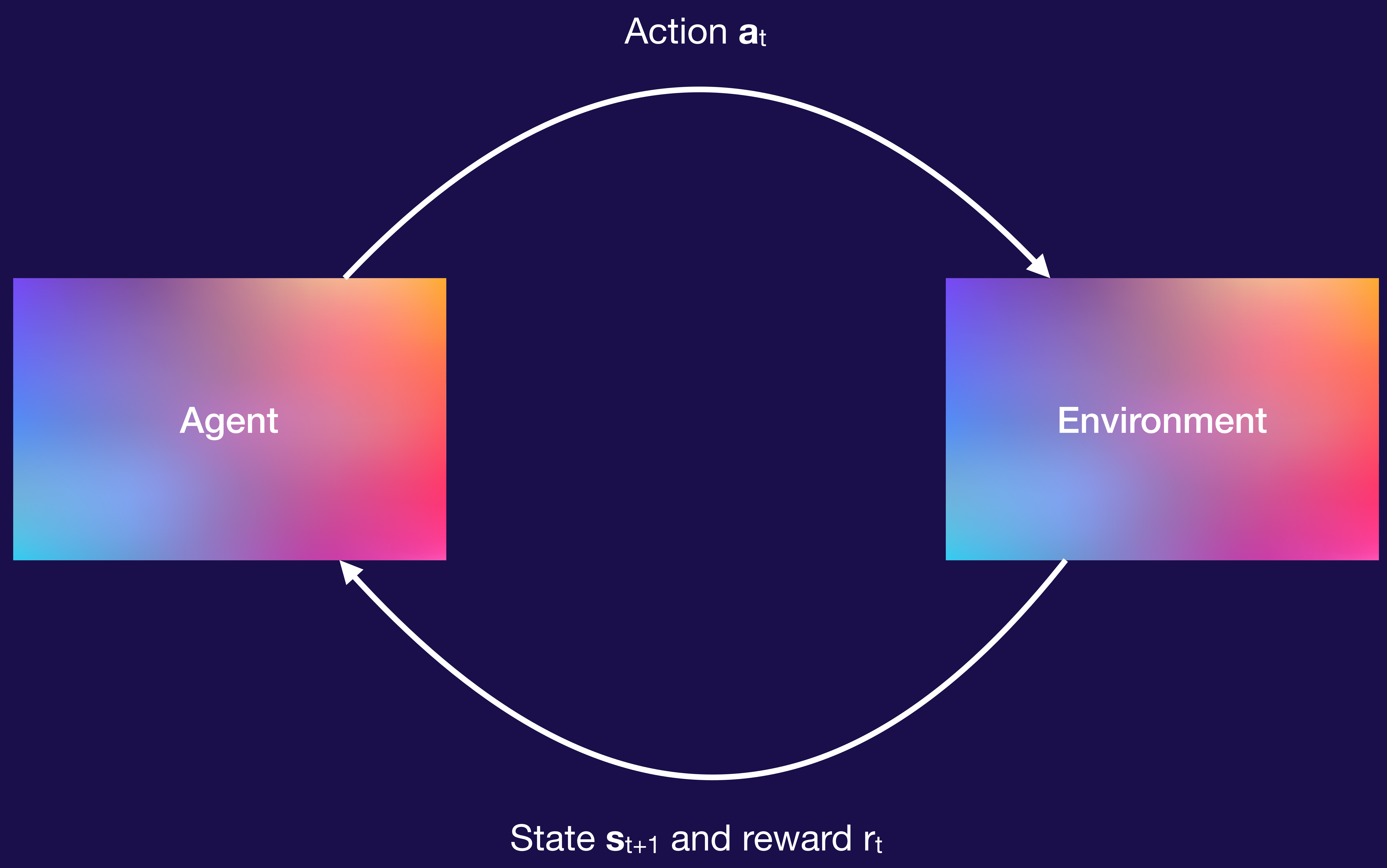
Challenges

- Reinforcement learning for the real world
- High-dimensional control
- Noisy and partial observations
- Manipulating more than one object

Reinforcement Learning
+
Domain Randomization

Reinforcement Learning

Reinforcement Learning (1)



Reinforcement Learning (2)

- Formalize as *Markov decision process*

$$\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, \rho, r)$$

Reinforcement Learning (2)

- Formalize as *Markov decision process*

Set of states

$$\mathscr{M} = (\underbrace{\mathscr{S}}_{}, \mathscr{A}, \mathscr{P}, \rho, r)$$

Reinforcement Learning (2)

- Formalize as *Markov decision process*

$$\mathcal{M} = (\underbrace{\mathcal{S}}_{\text{Set of states}}, \underbrace{\mathcal{A}}_{\text{Set of actions}}, \mathcal{P}, \rho, r)$$

Reinforcement Learning (2)

- Formalize as *Markov decision process*

Transition probabilities: $\boldsymbol{s}_{t+1} \sim \mathscr{P}(\,\cdot \mid , \boldsymbol{s}_t, \boldsymbol{a}_t)$

Set of states

$$\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, \rho, r)$$

Set of actions

Reinforcement Learning (2)

- Formalize as *Markov decision process*

Transition probabilities: $\boldsymbol{s}_{t+1} \sim \mathscr{P}(\,\cdot \mid , \boldsymbol{s}_t, \boldsymbol{a}_t)$

Set of states

$$\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, \rho, r)$$

Set of actions

Initial state distribution: $\boldsymbol{s}_0 \sim \rho(\,\cdot\,)$

Reinforcement Learning (2)

- Formalize as *Markov decision process*

Transition probabilities: $\boldsymbol{s}_{t+1} \sim \mathscr{P}(\,\cdot\mid, \boldsymbol{s}_t, \boldsymbol{a}_t)$

Set of states

Reward function: $r : \mathscr{S} \times \mathscr{A}$

$$\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, \rho, r)$$

Set of actions

Initial state distribution: $\boldsymbol{s}_0 \sim \rho(\,\cdot\,)$

Reinforcement Learning (2)

- Formalize as *Markov decision process*

Transition probabilities: $\boldsymbol{s}_{t+1} \sim \mathscr{P}(\,\cdot \mid , \boldsymbol{s}_t, \boldsymbol{a}_t)$

Set of states

Reward function: $r : \mathscr{S} \times \mathscr{A}$

$$\mathscr{M} = (\mathscr{S}, \mathscr{A}, \mathscr{P}, \rho, r)$$

Set of actions

Initial state distribution: $\boldsymbol{s}_0 \sim \rho(\,\cdot\,)$

- Agent uses a *policy* to select actions:

$$\boldsymbol{a}_t \sim \pi(\,\cdot \mid \boldsymbol{s}_t)$$

Reinforcement Learning (3)

- Let τ denote a *trajectory* with $\boldsymbol{s}_0 \sim \rho(\,\cdot\,), \boldsymbol{a}_t \sim \pi(\,\cdot \mid \boldsymbol{s}_t), \boldsymbol{s}_{t+1} \sim \mathscr{P}(\,\cdot \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$

Reinforcement Learning (3)

- Let τ denote a *trajectory* with $\boldsymbol{s}_0 \sim \rho(\,\cdot\,), \boldsymbol{a}_t \sim \pi(\,\cdot \mid \boldsymbol{s}_t), \boldsymbol{s}_{t+1} \sim \mathscr{P}(\,\cdot \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$
- The *discounted return* is then defined as:

$$R(\tau) := \sum_t \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \qquad \gamma \in [0,1)$$

Reinforcement Learning (3)

- Let τ denote a *trajectory* with $\boldsymbol{s}_0 \sim \rho(\,\cdot\,), \boldsymbol{a}_t \sim \pi(\,\cdot \mid \boldsymbol{s}_t), \boldsymbol{s}_{t+1} \sim \mathscr{P}(\,\cdot \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$
- The *discounted return* is then defined as:

$$R(\tau) := \sum_t \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \qquad \gamma \in [0,1)$$

- We wish to find a policy that maximizes the expected discounted return:

$$\pi^* := \arg\max_\pi \mathbb{E}_\tau \left[R(\tau)\right]$$

Reinforcement Learning (4)

- Depending on the assumptions, many methods exist to find optimal policies. Examples are:
  - Dynamic programming
  - Policy gradient methods
  - Q learning

- This talk will focus on *policy gradient* methods

- Policy gradients are *model-free,* i.e. we do not know the transition distribution

Policy Gradients (1)

- Let's assume a parameterized policy π_{θ}, where $\boldsymbol{\theta}$ is some parameter vector

Policy Gradients (1)

- Let's assume a parameterized policy $\pi_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ is some parameter vector

- Our optimization objective then is:

$$J(\boldsymbol{\theta}) := \mathbb{E}_{\tau}\left[R(\tau)\right]$$

Policy Gradients (1)

- Let's assume a parameterized policy $\pi_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ is some parameter vector

- Our optimization objective then is:

has a dependency on **θ** through τ

$$J(\boldsymbol{\theta}) := \mathbb{E}_{\tau}\left[R(\tau)\right]$$

has no dependency on **θ**

Policy Gradients (1)

- Let's assume a parameterized policy $\pi_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ is some parameter vector

- Our optimization objective then is:

has a dependency on $\boldsymbol{\theta}$ through τ

$$J(\boldsymbol{\theta}) := \mathbb{E}_{\tau}\left[R(\tau)\right]$$

has no dependency on $\boldsymbol{\theta}$

- Simple idea: Let's compute the gradient w.r.t. $\boldsymbol{\theta}$ and do gradient ascent

Policy Gradients (2)

- Goal: Compute the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau} \left[R(\tau) \right]$

Policy Gradients (2)

- Goal: Compute the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau}\left[R(\tau)\right]$
- Expanding the expectation and rearranging we get:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \left[\int R(\tau) p(\tau) d\tau \right]$$

Policy Gradients (2)

- Goal: Compute the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau}\left[R(\tau)\right]$

- Expanding the expectation and rearranging we get:

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \left[\int R(\tau) p(\tau) d\tau \right] \\
&= \int R(\tau) \nabla_{\boldsymbol{\theta}} p(\tau) d\tau
\end{aligned}
$$

Policy Gradients (3)

- Goal: Compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int R(\tau) \nabla_{\boldsymbol{\theta}} p(\tau) d\tau$

Policy Gradients (3)

- Goal: Compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int R(\tau) \boxed{\nabla_{\boldsymbol{\theta}} p(\tau)} d\tau$

Policy Gradients (3)

- Goal: Compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int R(\tau) \boxed{\nabla_{\boldsymbol{\theta}} p(\tau)} d\tau$
- Use the "log derivative trick":

$$\nabla_{\boldsymbol{\theta}} \log p(\tau) = \frac{1}{p(\tau)} \nabla_{\boldsymbol{\theta}} p(\tau) \iff \boxed{\nabla_{\boldsymbol{\theta}} p(\tau)} = \nabla_{\boldsymbol{\theta}} \log p(\tau) p(\tau)$$

Policy Gradients (3)

- Goal: Compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int R(\tau) \boxed{\nabla_{\boldsymbol{\theta}} p(\tau)} d\tau$
- Use the "log derivative trick":

$$\nabla_{\boldsymbol{\theta}} \log p(\tau) = \frac{1}{p(\tau)} \nabla_{\boldsymbol{\theta}} p(\tau) \iff \boxed{\nabla_{\boldsymbol{\theta}} p(\tau)} = \nabla_{\boldsymbol{\theta}} \log p(\tau) p(\tau)$$

- ... and plug back in to obtain:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int R(\tau) \nabla_{\boldsymbol{\theta}} \log p(\tau) p(\tau) d\tau$$

Policy Gradients (3)

- Goal: Compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int R(\tau) \boxed{\nabla_{\boldsymbol{\theta}} p(\tau)} d\tau$
- Use the "log derivative trick":

$$\nabla_{\boldsymbol{\theta}} \log p(\tau) = \frac{1}{p(\tau)} \nabla_{\boldsymbol{\theta}} p(\tau) \iff \boxed{\nabla_{\boldsymbol{\theta}} p(\tau)} = \nabla_{\boldsymbol{\theta}} \log p(\tau) p(\tau)$$

- ... and plug back in to obtain:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \int R(\tau) \nabla_{\boldsymbol{\theta}} \log p(\tau) p(\tau) d\tau \\ &= \mathbb{E}_{\tau} \left[R(\tau) \nabla_{\boldsymbol{\theta}} \log p(\tau) \right] \end{aligned}$$

Policy Gradients (4)

- Goal: Compute $\nabla_{\boldsymbol{\theta}} \log p(\tau)$

Policy Gradients (4)

- Goal: Compute $\nabla_{\boldsymbol{\theta}} \log p(\tau)$

- Probability of a trajectory τ is:

$$p(\tau) = \rho(\boldsymbol{s}_0) \prod_t \mathscr{P}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t) \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$$

Policy Gradients (4)

- Goal: Compute $\nabla_{\boldsymbol{\theta}} \log p(\tau)$

- Probability of a trajectory τ is:

$$p(\tau) = \rho(\boldsymbol{s}_0) \prod_t \mathscr{P}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t) \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$$

$$\log p(\tau) = \log \rho(\boldsymbol{s}_0) + \sum_t \log \mathscr{P}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t) + \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$$

Policy Gradients (4)

- Goal: Compute $\nabla_{\boldsymbol{\theta}} \log p(\tau)$
- Probability of a trajectory τ is:

$$p(\tau) = \rho(\boldsymbol{s}_0) \prod_t \mathscr{P}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t) \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$$

$$\log p(\tau) = \log \rho(\boldsymbol{s}_0) + \sum_t \log \mathscr{P}(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t) + \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$$

- Taking the gradient:

$$\nabla_{\boldsymbol{\theta}} \log p(\tau) = \sum_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$$

Policy Gradients (5)

- Putting it all together:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau} \left[R(\tau) \right]$$

Policy Gradients (5)

- Putting it all together:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau} \left[R(\tau) \right]$$

$$= \int R(\tau) \, \nabla_{\boldsymbol{\theta}} p(\tau) d\tau$$

Policy Gradients (5)

- Putting it all together:

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau}\left[R(\tau)\right] \\
&= \int R(\tau)\, \nabla_{\boldsymbol{\theta}} p(\tau) d\tau \\
&= \mathbb{E}_{\tau}\left[R(\tau)\, \nabla_{\boldsymbol{\theta}} \log p(\tau)\right]
\end{aligned}
$$

Policy Gradients (5)

- Putting it all together:

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau}\left[R(\tau)\right] \\
&= \int R(\tau)\, \nabla_{\boldsymbol{\theta}} p(\tau) d\tau \\
&= \mathbb{E}_{\tau}\left[R(\tau)\, \nabla_{\boldsymbol{\theta}} \log p(\tau)\right] \\
&= \mathbb{E}_{\tau}\left[R(\tau) \sum_{t} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right]
\end{aligned}
$$

Policy Gradients (5)

- Putting it all together:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_\tau \left[R(\tau) \right] \\
&= \int R(\tau) \, \nabla_{\boldsymbol{\theta}} p(\tau) d\tau \\
&= \mathbb{E}_\tau \left[R(\tau) \, \nabla_{\boldsymbol{\theta}} \log p(\tau) \right] \\
&= \mathbb{E}_\tau \left[R(\tau) \sum_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t) \right]
\end{aligned}$$

- Last missing piece: Computing the expectation

Policy Gradients (6)

- Goal: Compute expectation of:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau}\left[R(\tau)\sum_{t}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right]$$

Policy Gradients (6)

- Goal: Compute expectation of:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau}\left[R(\tau)\sum_{t}\nabla_{\boldsymbol{\theta}}\log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right]$$

- Can be estimated using Monte Carlo sampling, which corresponds to rolling out the policy multiple times to collect N trajectories: $\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(N)}$

Policy Gradients (6)

- Goal: Compute expectation of:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau}\left[R(\tau)\sum_{t}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t)\right]$$

- Can be estimated using Monte Carlo sampling, which corresponds to rolling out the policy multiple times to collect N trajectories: $\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(N)}$

- Our final estimate of the policy gradient is thus:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{N}\sum_{n=1}^{N}\left[R(\tau^{(n)})\sum_{t}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t^{(n)} \mid \boldsymbol{s}_t^{(n)})\right]$$

Policy Gradients (7)

1. Initialize **θ** arbitrarily
2. Repeat
   1. Collect N trajectories $\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(N)}$
   2. Estimate gradient:
   $$\hat{\boldsymbol{g}} \leftarrow \frac{1}{N} \sum_{n=1}^{N} \left[R(\tau^{(n)}) \sum_{t} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t^{(n)} \mid \boldsymbol{s}_t^{(n)}) \right]$$
   3. Update parameters:
   $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \hat{\boldsymbol{g}}$$

Proximal Policy Optimization

- Vanilla policy gradient algorithm simple but has several shortcomings
- We use *Proximal Policy Optimization* (PPO) in all our experiments (Schulman et al., 2017)
- Underlaying idea is exactly the same but uses
  - Baselines for variance reduction with generalized advantage estimation
  - Importance sampling to use slightly off-policy data
  - Clipping to improve stability

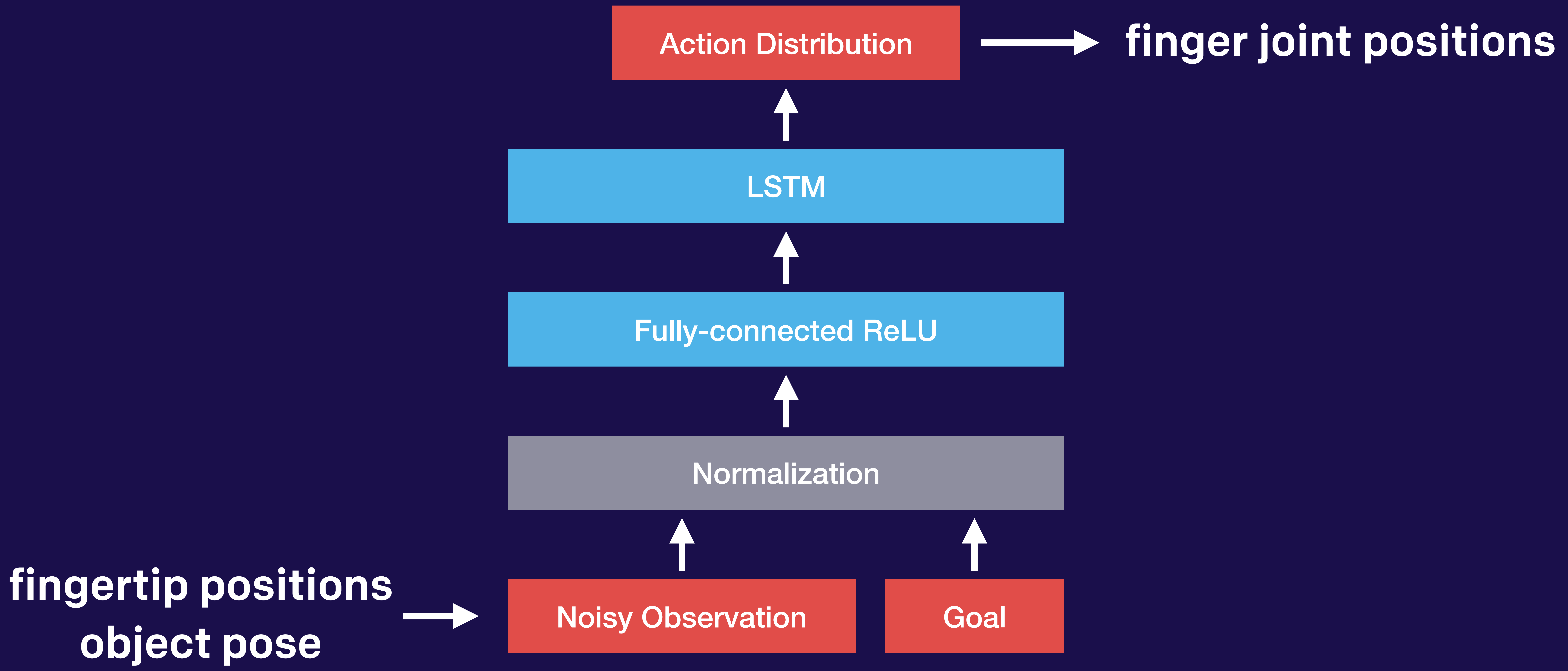
Reward Function

- $r_t = d_t - d_{t+1}$,
  where d_t and d_{t+1} denote the rotation angle between the desired and current orientation before and after the transition

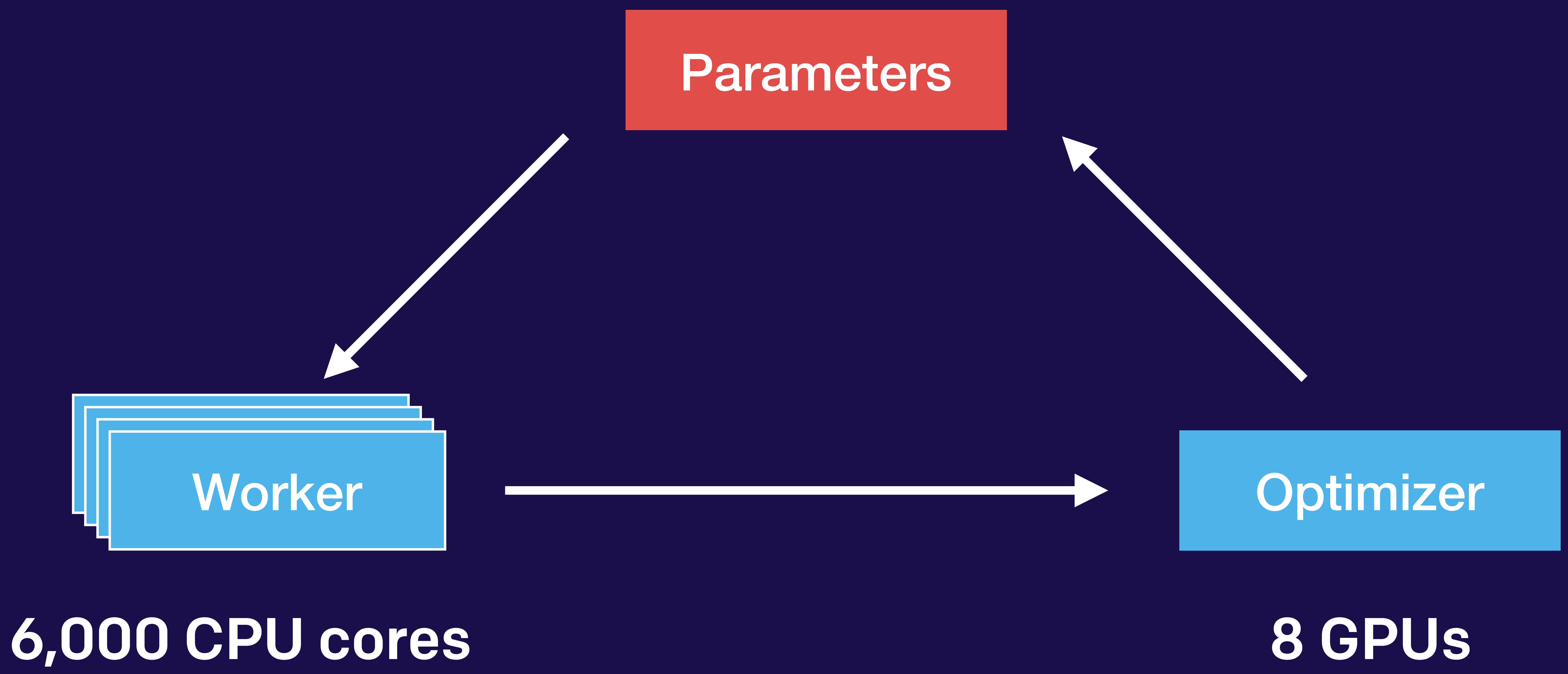
- On success: +5

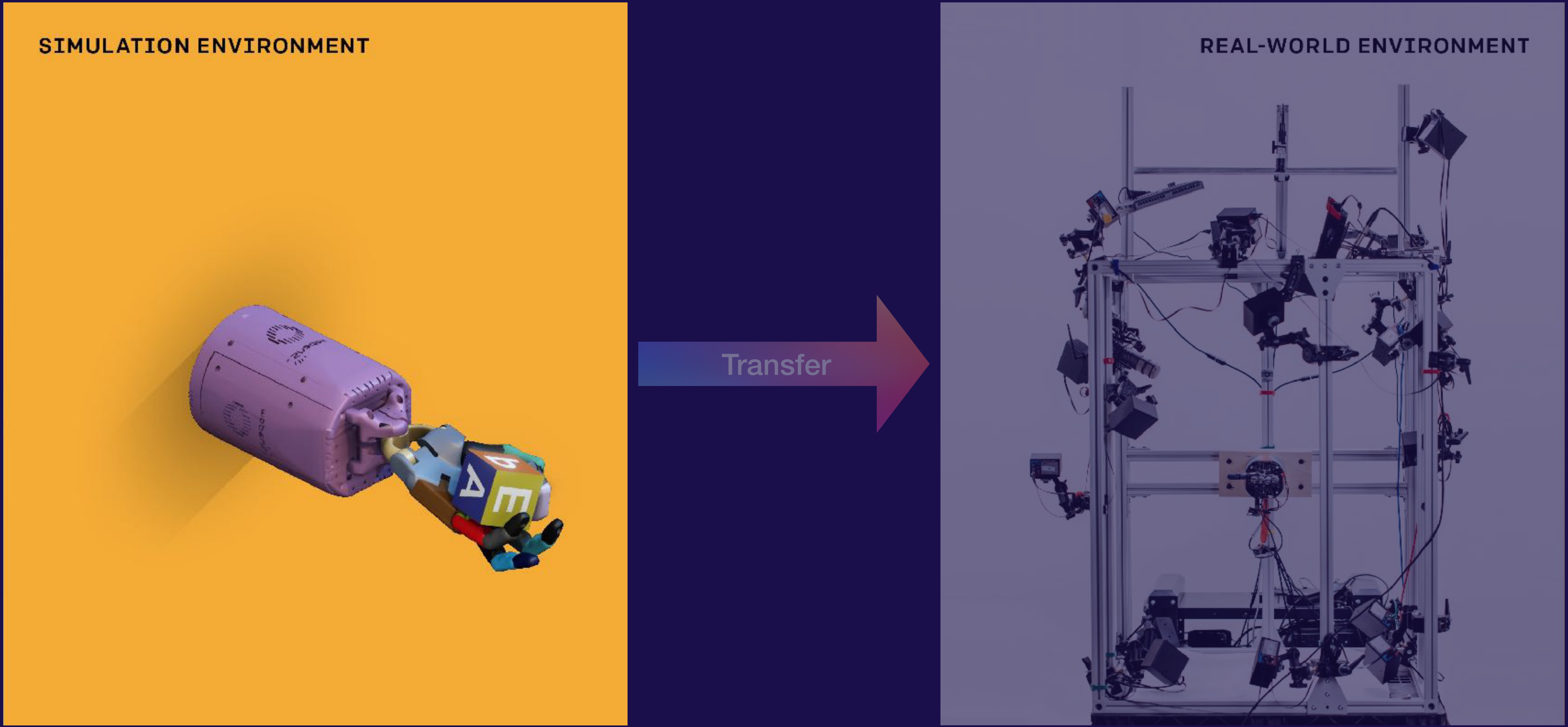
- On drop: -20

Policy Architecture

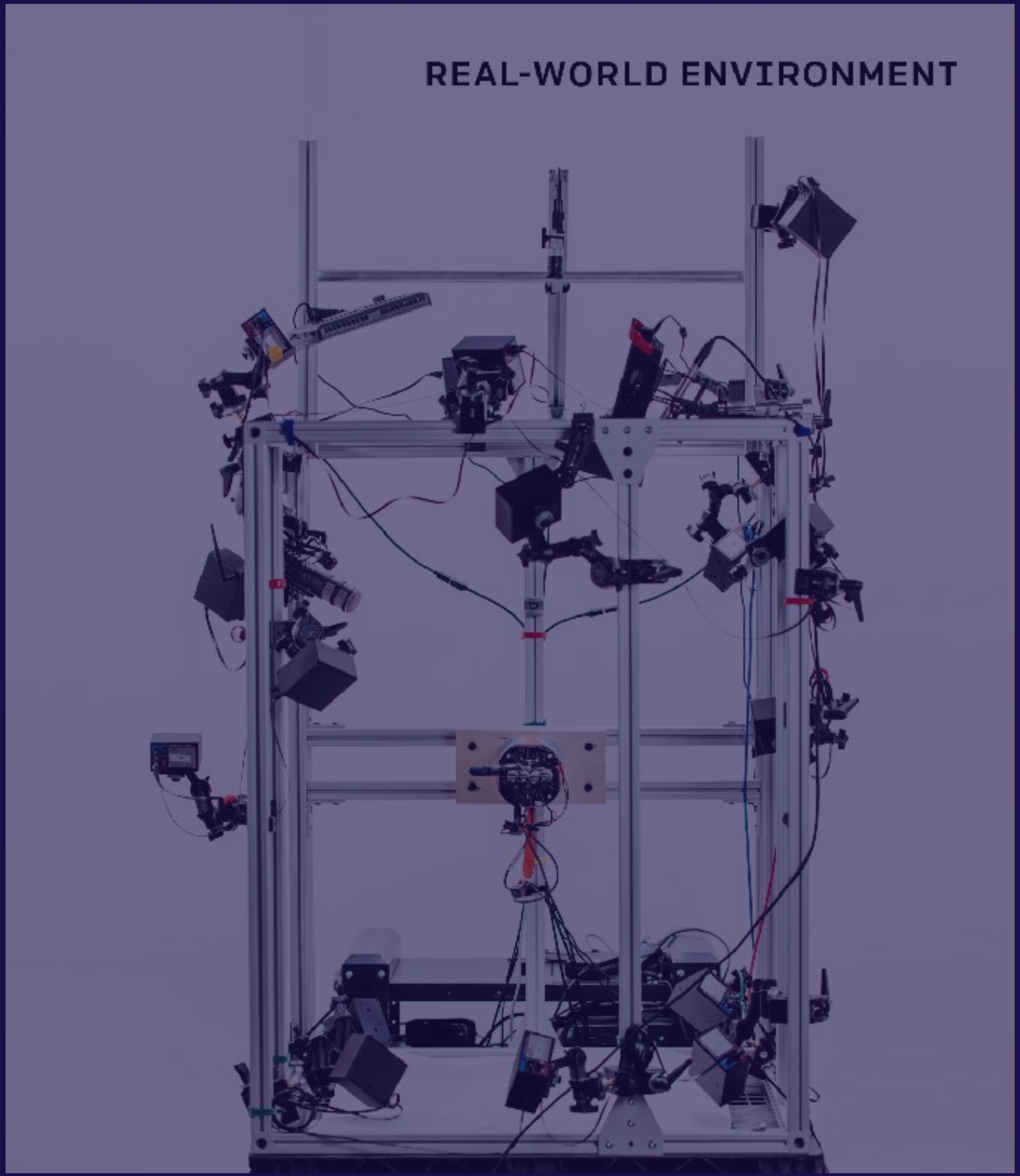


Distributed Training with Rapid



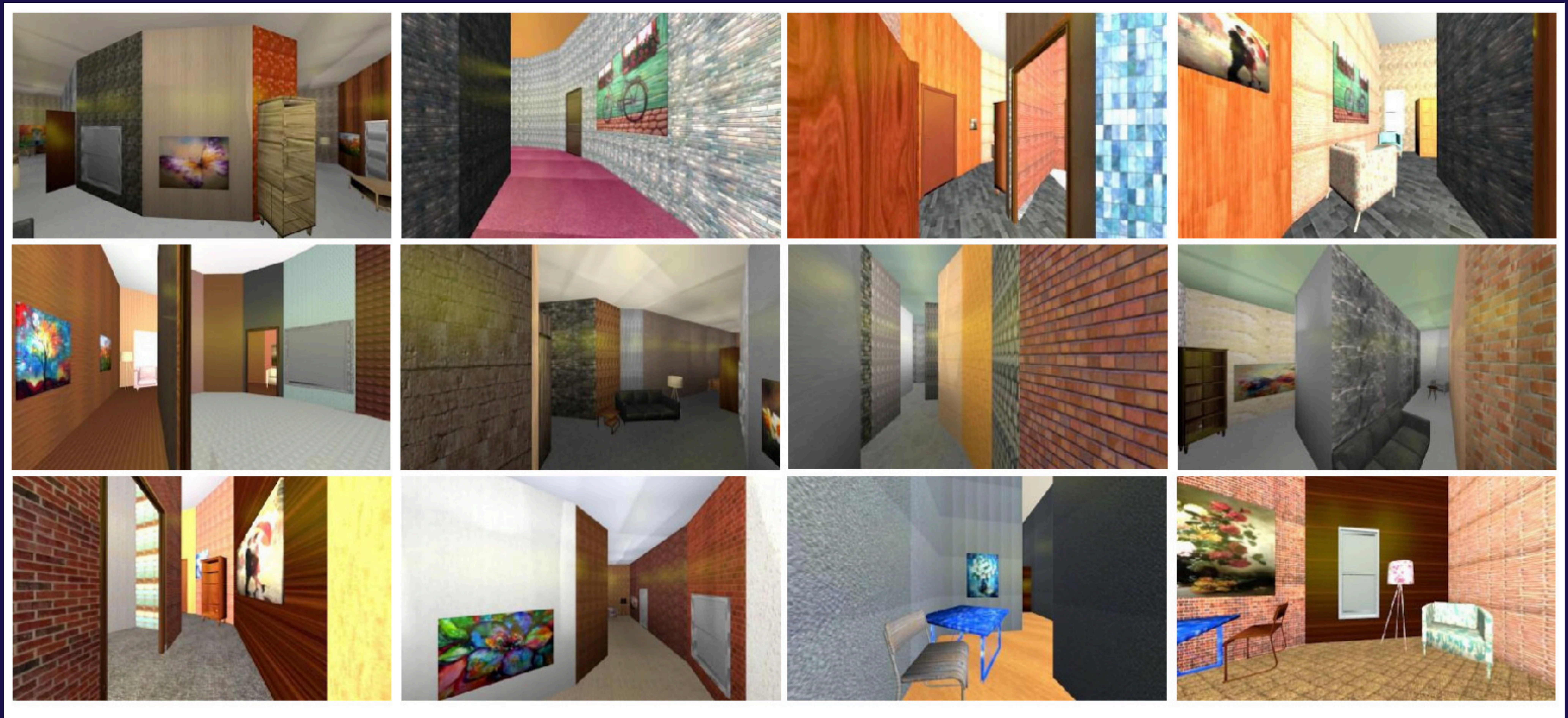

SIMULATION ENVIRONMENT
Transfer
REAL-WORLD ENVIRONMENT

Transfer



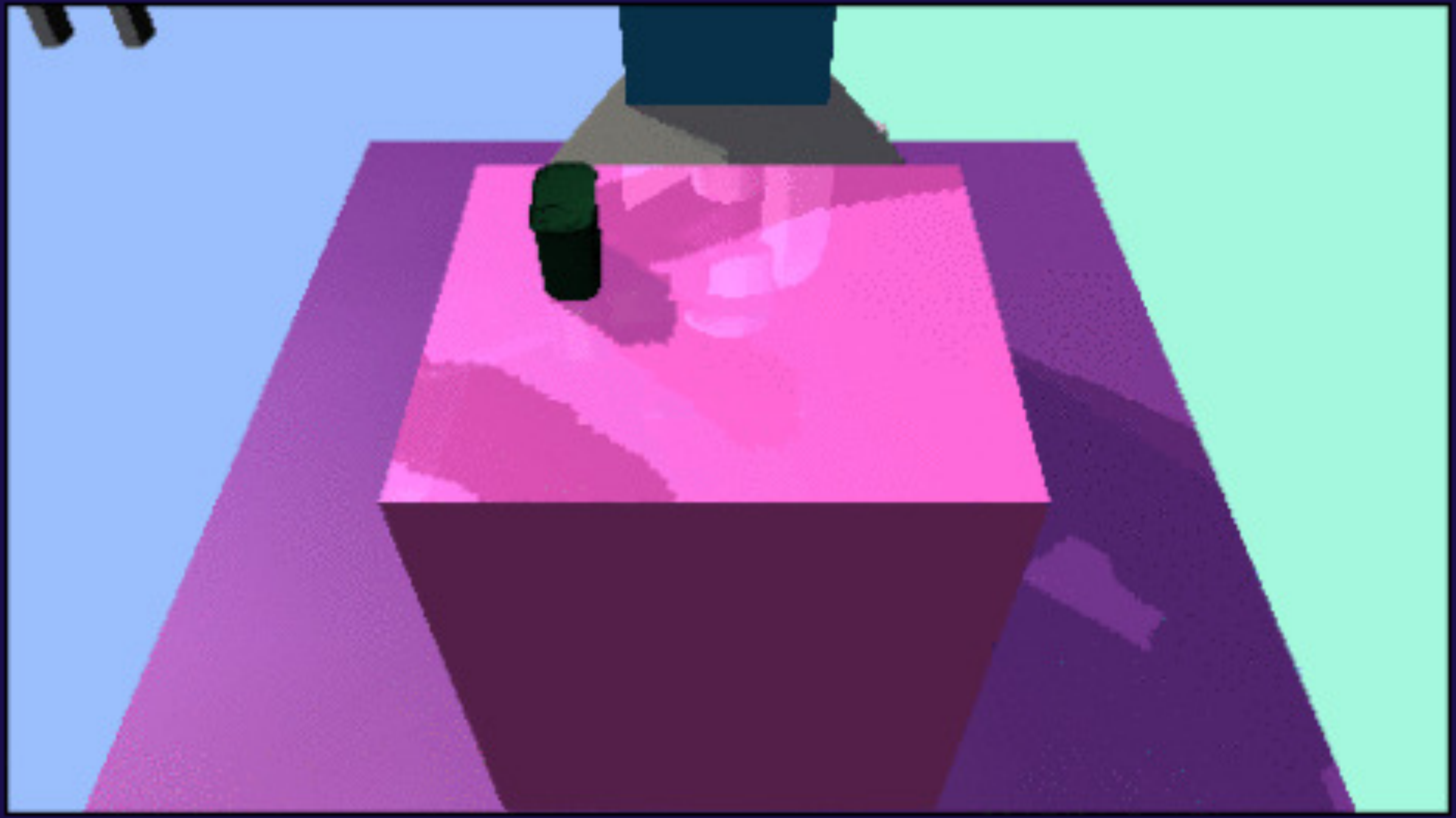
Domain Randomization

Domain Randomization (1)



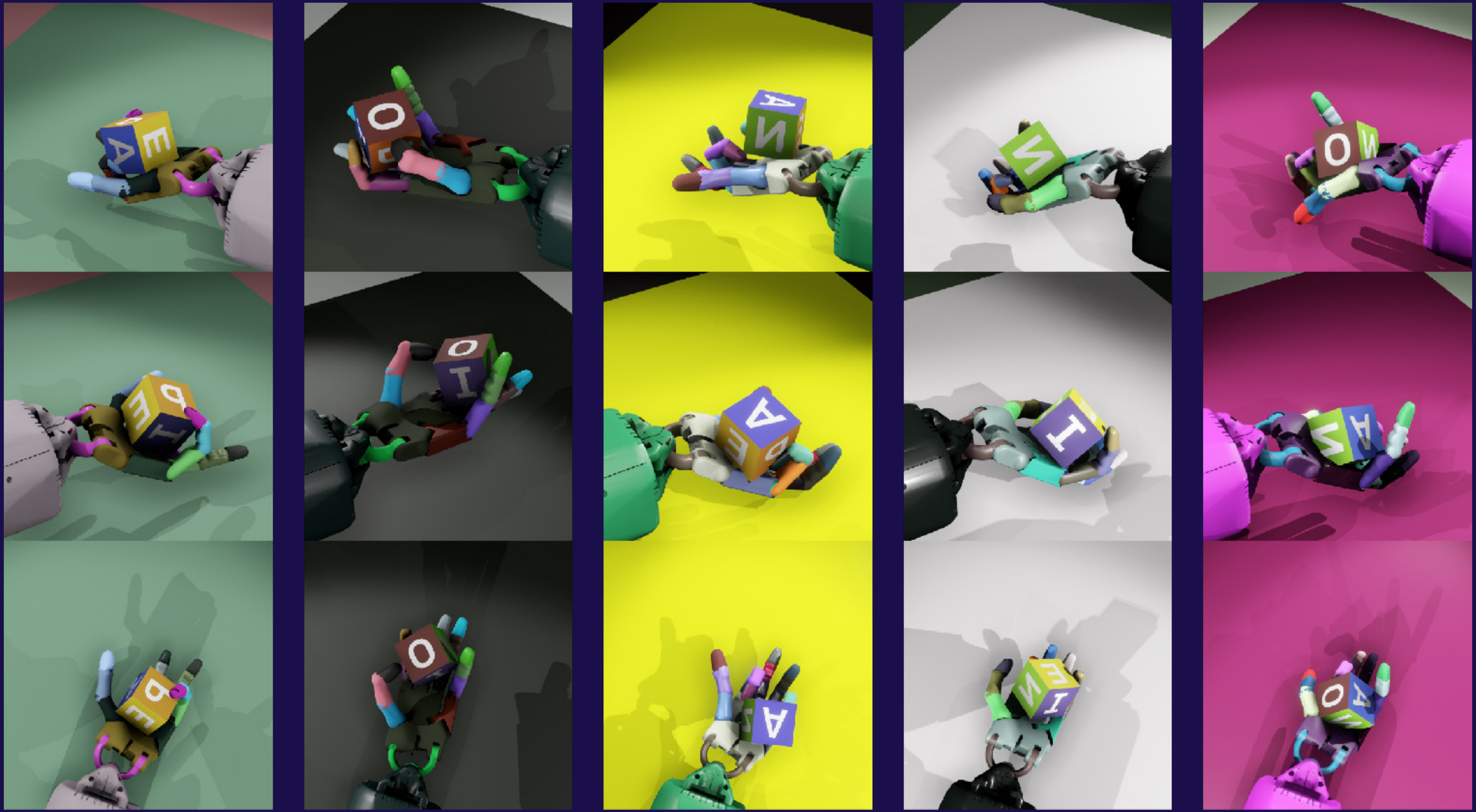
Sadeghi & Levine (2017)

Domain Randomization (2)

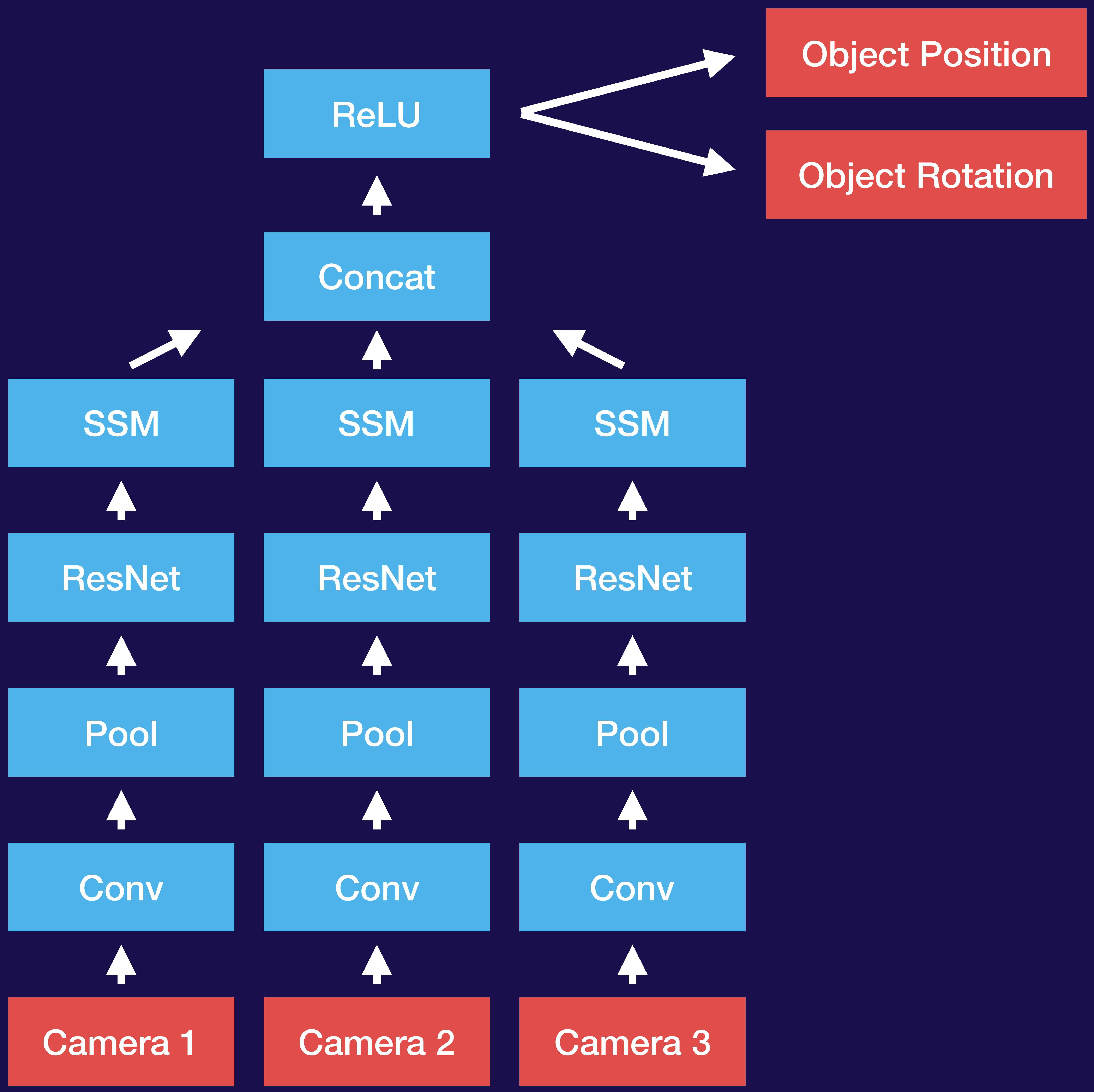


Tobin et al. (2017)

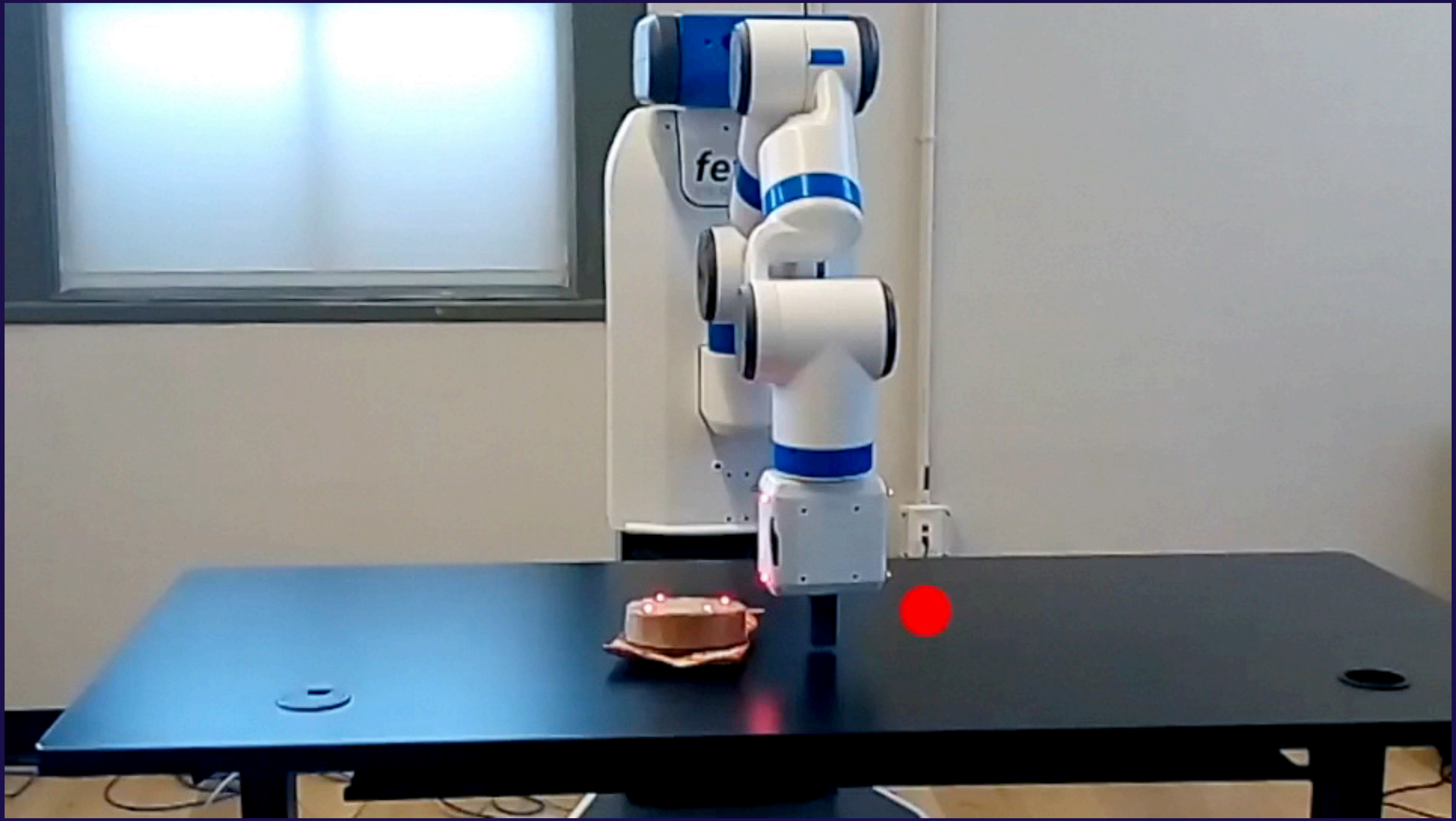
Vision Randomizations



Vision Architecture



Physics Randomizations (1)



Peng et al. (2018)

Physics Randomizations (2)

object dimensions

object and robot link masses

surface friction coefficients

robot joint damping coefficients

actuator force gains

joint limits

gravity vector

Full System Recap

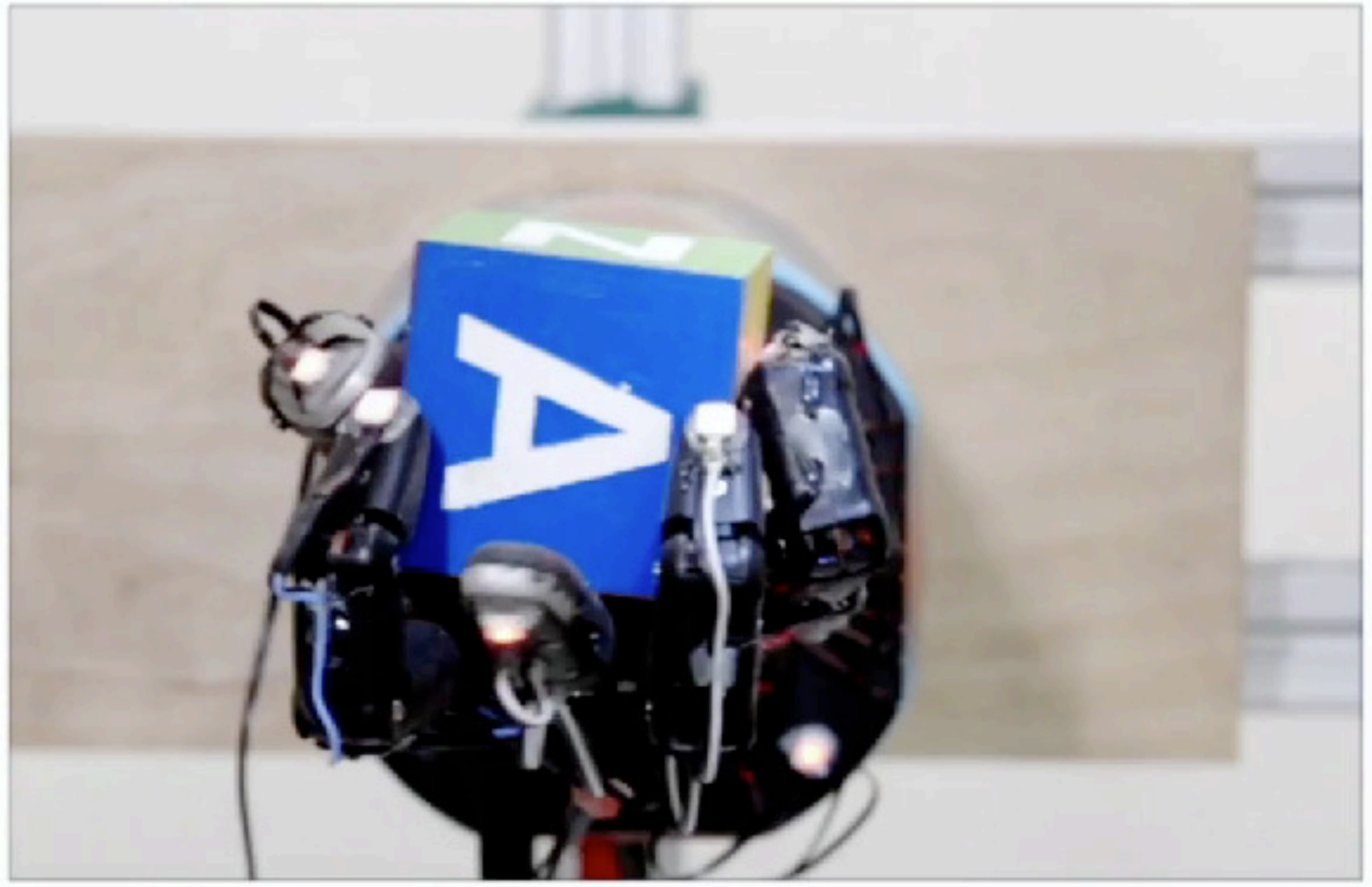
Train in Simulation
A Distributed workers collect experience on randomized environments at large scale.
B We train a control policy using reinforcement learning. It chooses the next action based on fingertip positions and the object pose.
Observed Robot States
LSTM
Actions
C We train a convolutional neural network to predict the object pose given three simulated camera images.
CONV
CONV
CONV
Object Pose

Transfer to the Real World

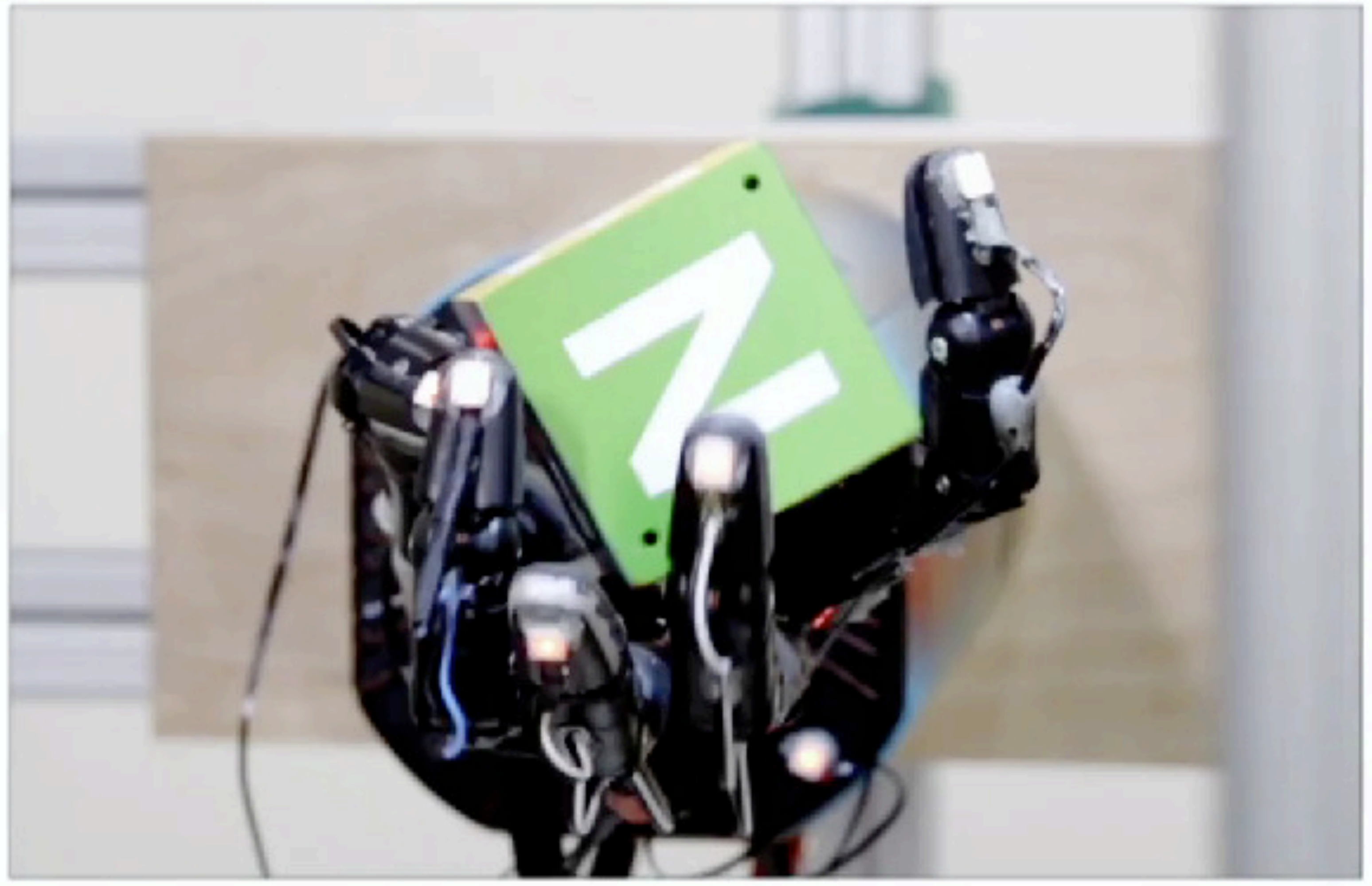
D We combine the pose estimation network and the control policy to transfer to the real world.

Results

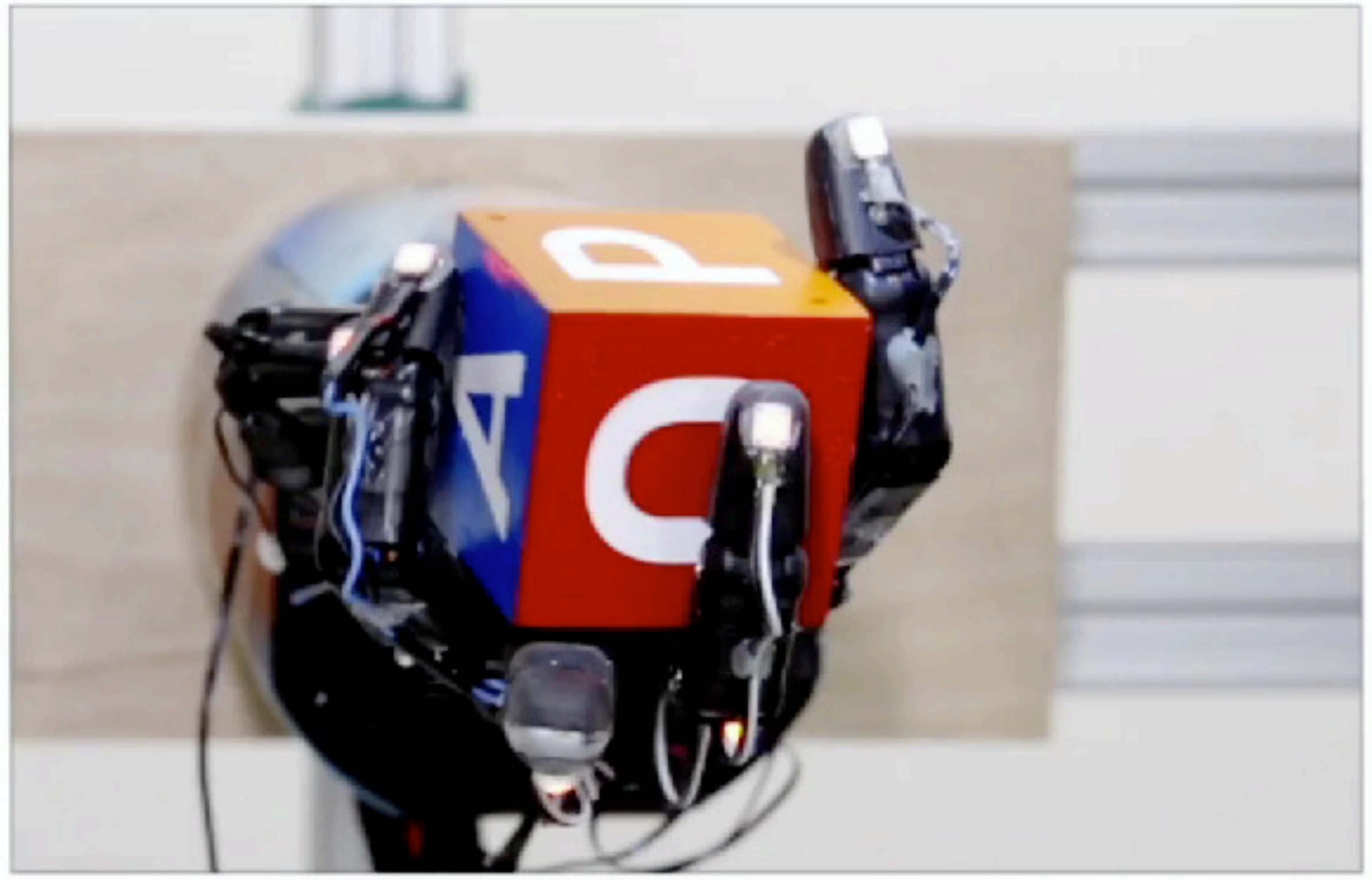
Learned Strategies



FINGER PIVOTING

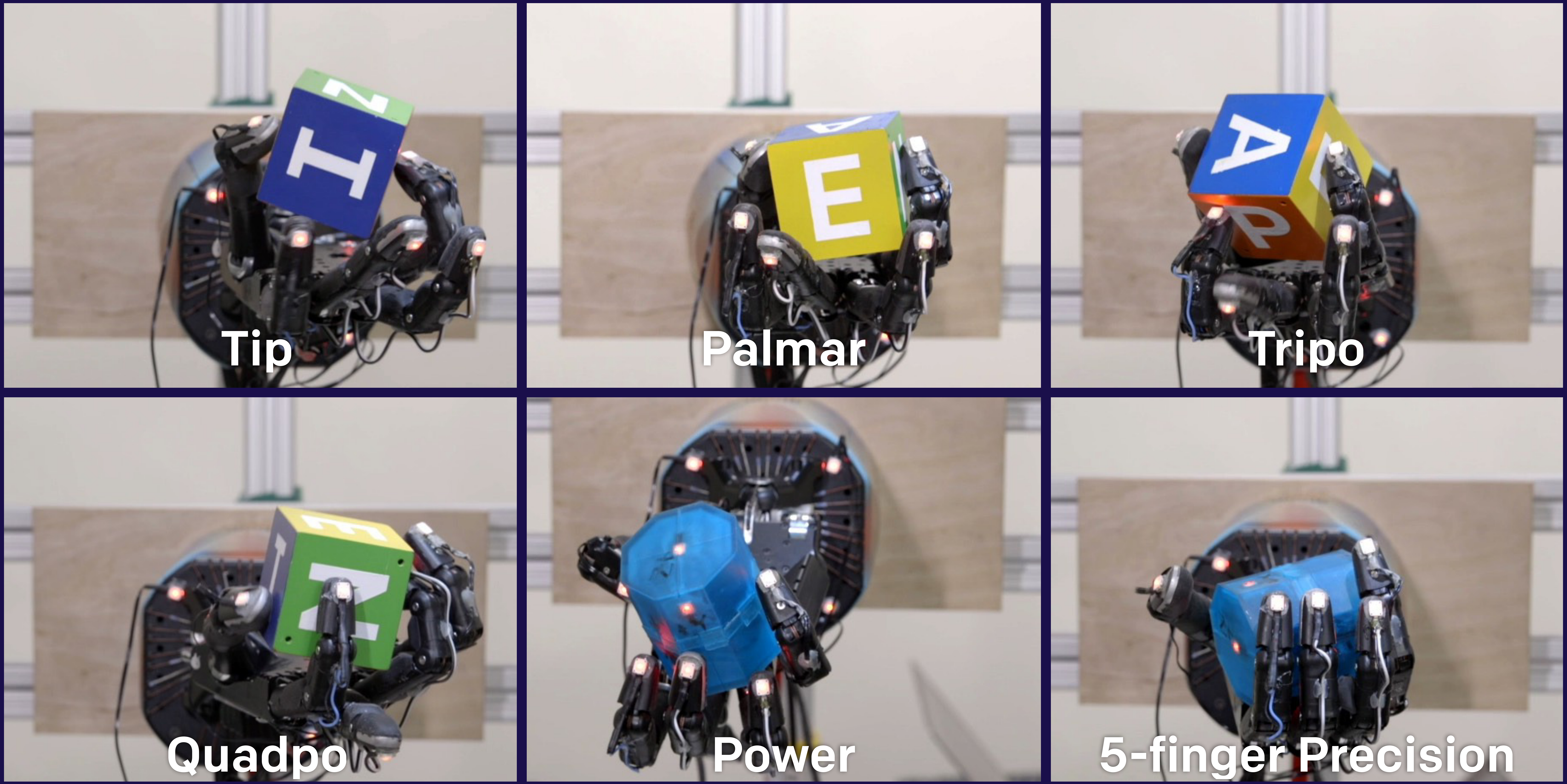


SLIDING



FINGER GAITING

Learned Grasps



Quantitative Results

Table 3: The number of successful consecutive rotations in simulation and on the physical robot. All policies were trained on environments with all randomizations enabled. We performed 100 trials in simulation and 10 trails per policy on the physical robot. Each trial terminates when the object is dropped, 50 rotations are achieved or a timeout is reached. For physical trials, results were taken at different times on the physical robot.

Simulated task	**Mean**	**Median**	**Individual trials (sorted)**
Block (state)	43.4 ± 13.8	50	-
Block (state, locked wrist)	44.2 ± 13.4	50	-
Block (vision)	30.0 ± 10.3	33	-
Octagonal prism (state)	29.0 ± 19.7	30	-
Physical task			
Block (state)	18.8 ± 17.1	13	50, 41, 29, 27, 14, 12, 6, 4, 4, 1
Block (state, locked wrist)	26.4 ± 13.4	28.5	50, 43, 32, 29, 29, 28, 19, 13, 12, 9
Block (vision)	15.2 ± 14.3	11.5	46, 28, 26, 15, 13, 10, 8, 3, 2, 1
Octagonal prism (state)	7.8 ± 7.8	5	27, 15, 8, 8, 5, 5, 4, 3, 2, 1

Quantitative Results

Table 3: The number of successful consecutive rotations in simulation and on the physical robot. All policies were trained on environments with all randomizations enabled. We performed 100 trials in simulation and 10 trails per policy on the physical robot. Each trial terminates when the object is dropped, 50 rotations are achieved or a timeout is reached. For physical trials, results were taken at different times on the physical robot.

Simulated task	**Mean**	**Median**	**Individual trials (sorted)**
Block (state)	43.4 ± 13.8	50	-
Block (state, locked wrist)	44.2 ± 13.4	50	-
Block (vision)	30.0 ± 10.3	33	-
Octagonal prism (state)	29.0 ± 19.7	30	-
Physical task			
Block (state)	18.8 ± 17.1	13	50, 41, 29, 27, 14, 12, 6, 4, 4, 1
Block (state, locked wrist)	26.4 ± 13.4	28.5	50, 43, 32, 29, 29, 28, 19, 13, 12, 9
Block (vision)	15.2 ± 14.3	11.5	46, 28, 26, 15, 13, 10, 8, 3, 2, 1
Octagonal prism (state)	7.8 ± 7.8	5	27, 15, 8, 8, 5, 5, 4, 3, 2, 1

Quantitative Results

Table 3: The number of successful consecutive rotations in simulation and on the physical robot. All policies were trained on environments with all randomizations enabled. We performed 100 trials in simulation and 10 trails per policy on the physical robot. Each trial terminates when the object is dropped, 50 rotations are achieved or a timeout is reached. For physical trials, results were taken at different times on the physical robot.

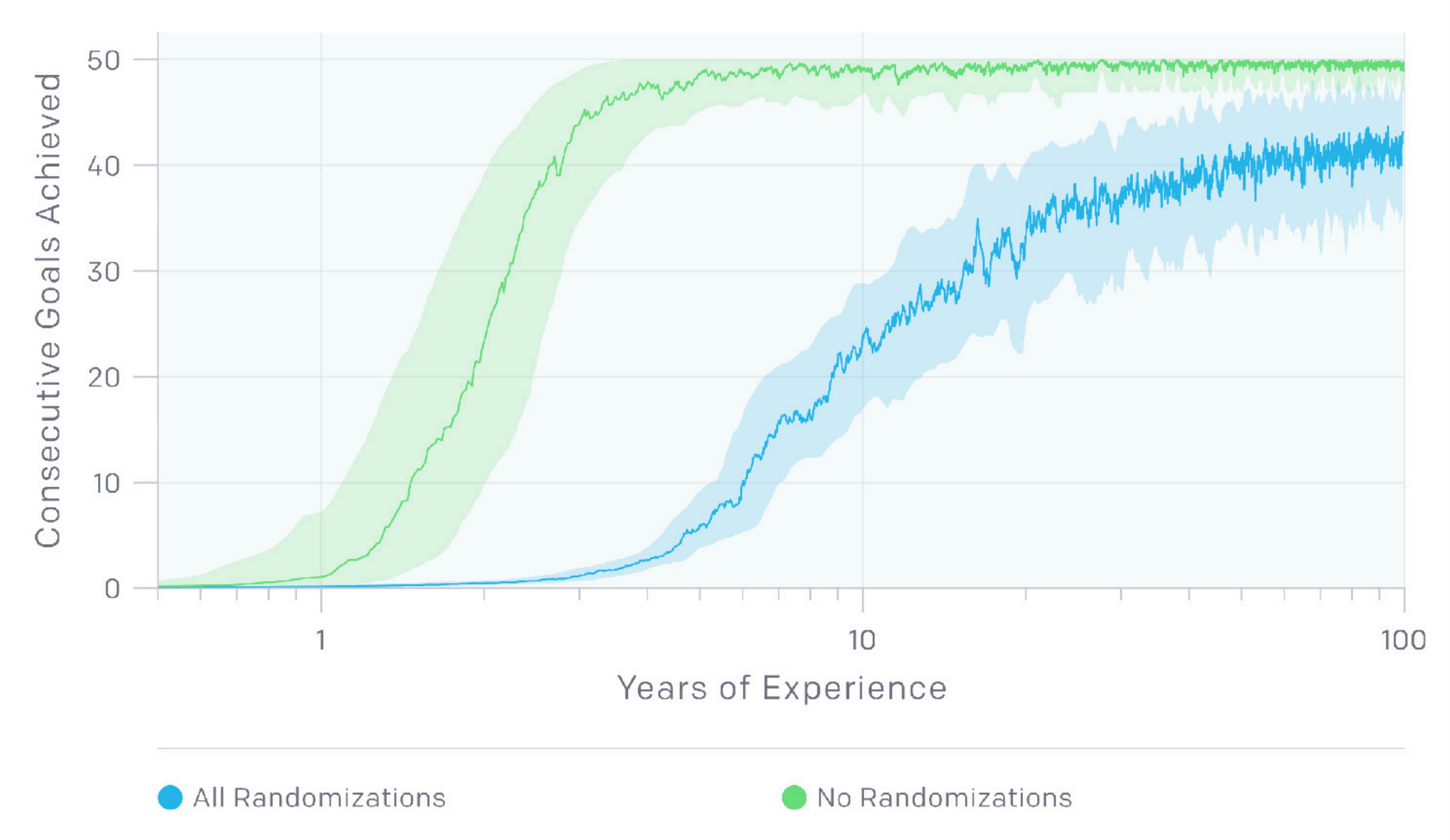
Simulated task	Mean	Median	Individual trials (sorted)
Block (state)	43.4 ± 13.8	50	-
Block (state, locked wrist)	44.2 ± 13.4	50	-
Block (vision)	30.0 ± 10.3	33	-
Octagonal prism (state)	29.0 ± 19.7	30	-
Physical task			
Block (state)	18.8 ± 17.1	13	50, 41, 29, 27, 14, 12, 6, 4, 4, 1
Block (state, locked wrist)	26.4 ± 13.4	28.5	50, 43, 32, 29, 29, 28, 19, 13, 12, 9
Block (vision)	15.2 ± 14.3	11.5	46, 28, 26, 15, 13, 10, 8, 3, 2, 1
Octagonal prism (state)	7.8 ± 7.8	5	27, 15, 8, 8, 5, 5, 4, 3, 2, 1

Ablation of Randomizations

Table 4: The number of successful consecutive rotations on the physical robot of 5 policies trained separately in environments with different randomizations held out. The first 5 rows use PhaseSpace for object pose estimation and were run on the same robot at the same time. Trials for each row were interleaved in case the state of the robot changed during the trials. The last two rows were measured at a different time from the first 5 and used the vision model to estimate the object pose.

Training environment	**Mean**	**Median**	**Individual trials (sorted)**
All randomizations (state)	18.8 ± 17.1	13	50, 41, 29, 27, 14, 12, 6, 4, 4, 1
No randomizations (state)	1.1 ± 1.9	0	6, 2, 2, 1, 0, 0, 0, 0, 0, 0
No observation noise (state)	15.1 ± 14.5	8.5	45, 35, 23, 11, 9, 8, 7, 6, 6, 1
No physics randomizations (state)	3.5 ± 2.5	2	7, 7, 7, 3, 2, 2, 2, 2, 2, 1
No unmodeled effects (state)	3.5 ± 4.8	2	16, 7, 3, 3, 2, 2, 1, 1, 0, 0
All randomizations (vision)	15.2 ± 14.3	11.5	46, 28, 26, 15, 13, 10, 8, 3, 2, 1
No observation noise (vision)	5.9 ± 6.6	3.5	20, 12, 11, 6, 5, 2, 2, 1, 0, 0

Training Time



Effect of Memory

Table 5: The number of successful consecutive rotations on the physical robot of 3 policies with different network architectures trained on an environment with all randomizations. Results for each row were collected at different times on the physical robot.

Network architecture	**Mean**	**Median**	**Individual trials (sorted)**
LSTM policy / LSTM value (state)	18.8 ± 17.1	13	50, 41, 29, 27, 14, 12, 6, 4, 4, 1
FF policy / LSTM value (state)	4.7 ± 4.1	3.5	15, 7, 6, 5, 4, 3, 3, 2, 2, 0
FF policy / FF value (state)	4.6 ± 4.3	3	15, 8, 6, 5, 3, 3, 2, 2, 2, 0

Surprises

- Tactile sensing is not necessary to manipulate real-world objects
- Randomizations developed for one object generalize to others with similar properties
- With physical robots, good systems engineering is as important as good algorithms

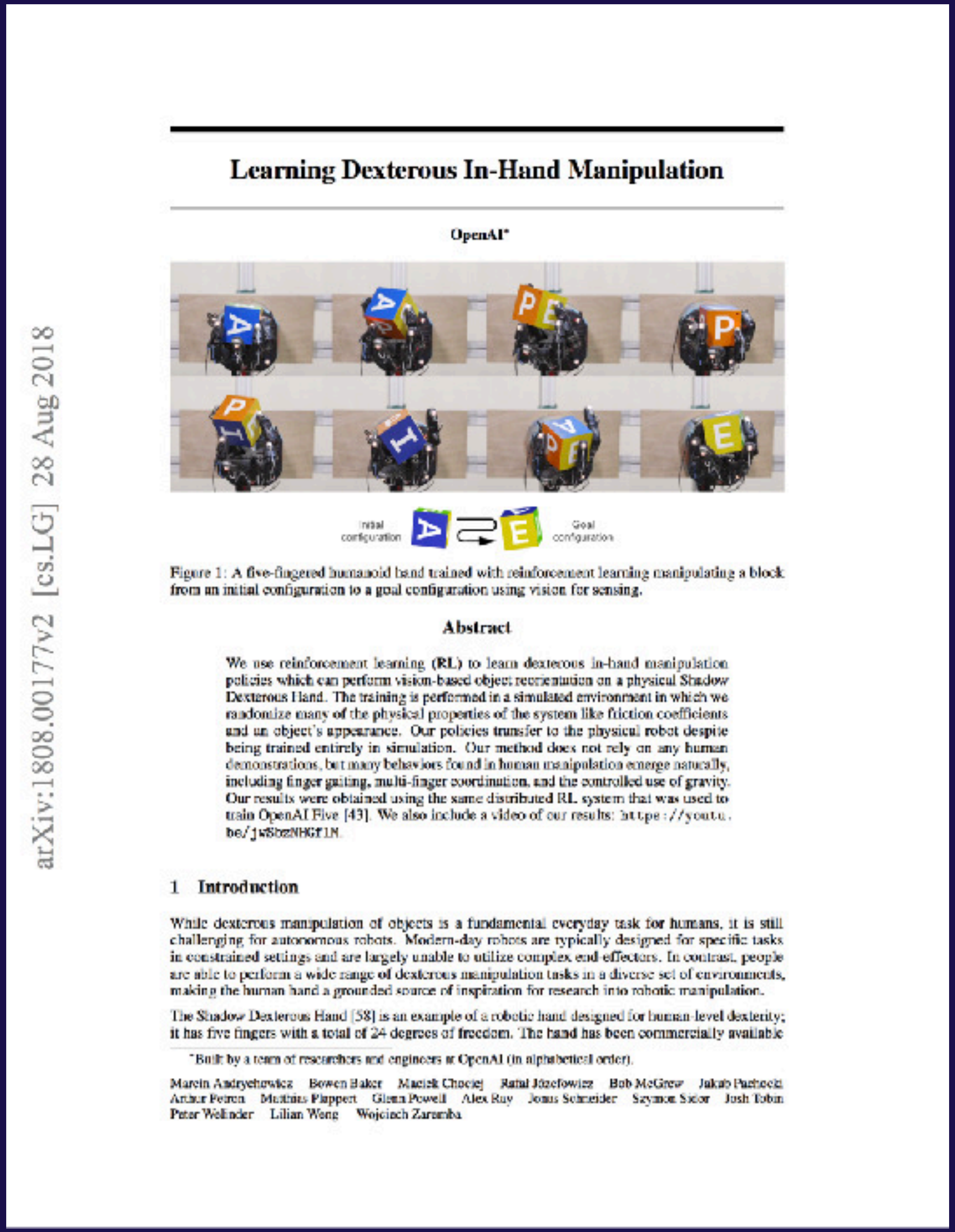
Challenges Ahead

- Less manual tuning in domain randomization
- Make use of demonstrations
- Faster learning in real world

Blog Post

https://blog.openai.com/learning-dexterity/

Pre-print



arXiv:1808.00177v2 [cs.LG] 28 Aug 2018

Learning Dexterous In-Hand Manipulation

OpenAI*

initial configuration

Goal configuration

Figure 1: A five-fingered humanoid hand trained with reinforcement learning manipulating a block from an initial configuration to a goal configuration using vision for sensing.

Abstract

We use reinforcement learning (RL) to learn dexterous in-hand manipulation policies which can perform vision-based object reorientation on a physical Shadow Dexterous Hand. The training is performed in a simulated environment in which we randomize many of the physical properties of the system like friction coefficients and an object's appearance. Our policies transfer to the physical robot despite being trained entirely in simulation. Our method does not rely on any human demonstrations, but many behaviors found in human manipulation emerge naturally, including finger gaiting, multi-finger coordination, and the controlled use of gravity. Our results were obtained using the same distributed RL system that was used to train OpenAI Five [43]. We also include a video of our results: https://youtu.be/jwSbzNHGflM.

1 Introduction

While dexterous manipulation of objects is a fundamental everyday task for humans, it is still challenging for autonomous robots. Modern-day robots are typically designed for specific tasks in constrained settings and are largely unable to utilize complex end-effectors. In contrast, people are able to perform a wide range of dexterous manipulation tasks in a diverse set of environments, making the human hand a grounded source of inspiration for research into robotic manipulation.

The Shadow Dexterous Hand [58] is an example of a robotic hand designed for human-level dexterity; it has five fingers with a total of 24 degrees of freedom. The hand has been commercially available

*Built by a team of researchers and engineers at OpenAI (in alphabetical order).

Marcin Andrychowicz Bowen Baker Maciek Chociej Rafal Józefowicz Bob McGrew Jakub Pachocki Arthur Petron Matthias Plappert Glenn Powell Alex Ray Jonas Schneider Szymon Sidor Josh Tobin Peter Welinder Lilian Weng Wojciech Zaremba

https://arxiv.org/pdf/1808.00177.pdf

Thank You

Visit openai.com for more information.

FOLLOW @OPENAI ON TWITTER

References

- Levine et al. "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection." The International Journal of Robotics Research37.4-5 (2018): 421-436.
- Peng et al. "Sim-to-real transfer of robotic control with dynamics randomization." arXiv preprint arXiv:1710.06537 (2017).
- Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world." Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on. IEEE, 2017.
- Rajeswaran et al. "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations." arXiv preprint arXiv: 1709.10087 (2017).
- Kumar, Vikash, Emanuel Todorov, and Sergey Levine. "Optimal control with learned local models: Application to dexterous manipulation." Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016.
- Schulman et al. "Proximal policy optimization algorithms." arXiv preprint arXiv: 1707.06347 (2017).
- Sadeghi & Levine. "CAD2RL: Real single-image flight without a single real image." arXiv preprint arXiv:1611.04201 (2016).