UC San Diego

Learning Dexterous In-Hand Manipulation

Presenter: **SIDDARTH** Meenakshi Sundaram

21 May 2020

Outline

- Introduction/Challenges
- The Task
- Method
  - Training in Simulation (Transferable Simulations)
  - Learning Control Policy
  - State Estimation from Vision
- Results
- Limitations/Future Works
- Conclusion

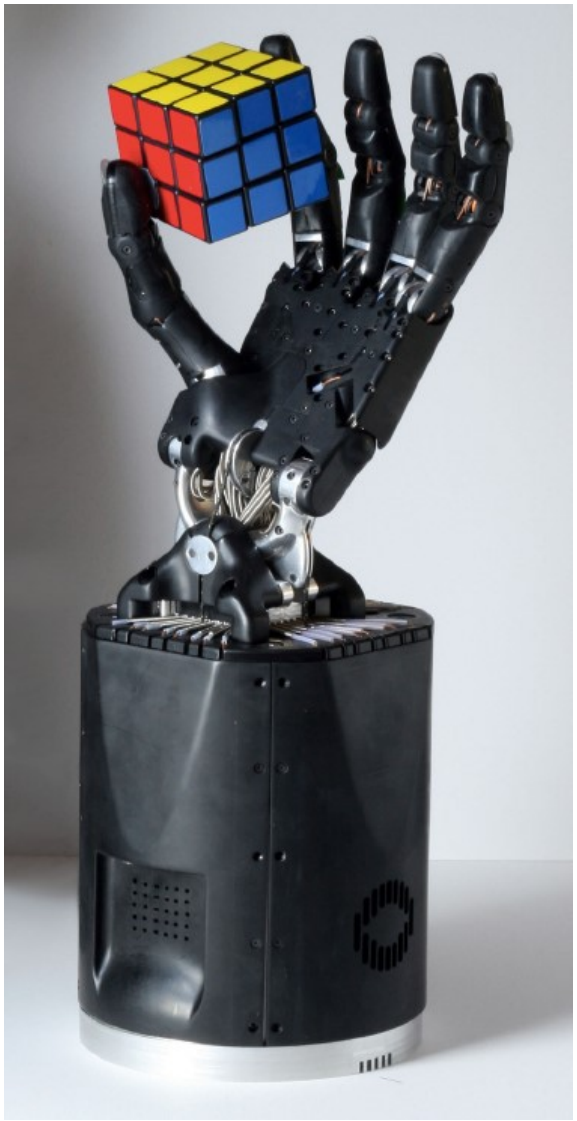
Introduction/Challenges

- Dexterous manipulation of objects is a fundamental everyday task for humans. But still challenging for autonomous robots

- Robots are typically designed for specific tasks in constrained settings and are largely unable to utilize complex end-effectors

- People are able to perform a wide range of dexterous manipulation tasks in a diverse set of environments

Introduction/Challenges



SETUP

- The Shadow Dexterous Hand is an example of a robotic hand designed for human-level dexterity.
- It has five fingers with a total of 24 degrees of freedom.
- 40 tendons controlled by 20 DC motors in the base of the hand

Introduction/Challenges

- NOT adopted widely. Can be attributed to the difficulty of controlling systems of such complexity.

- Some prior methods have shown promising in-hand manipulation results in simulation but do not attempt to transfer to a real world robot

- Hence, the task is to demonstrate methods to train control policies that perform in-hand manipulation and deploy them on a physical robot.

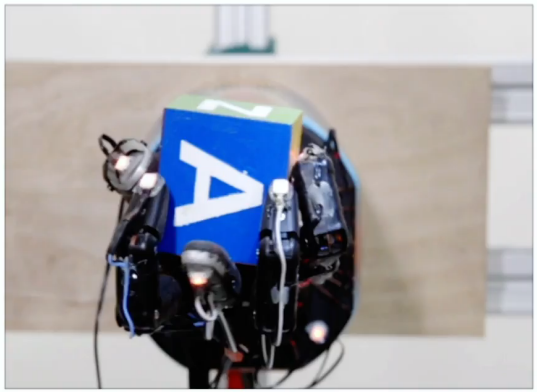
The Task

- We place an object such as a block or a prism in the palm of the hand and ask the system to reposition it into a different orientation;

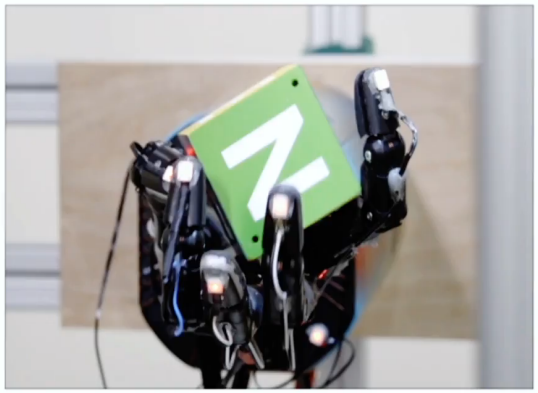
- For example, rotating the block to put a new face on top, or finger-gaiting to change the side of the robot without changing the top or the bottom of the block.

- The network observes only the coordinates of the fingertips and the images from three regular RGB cameras.

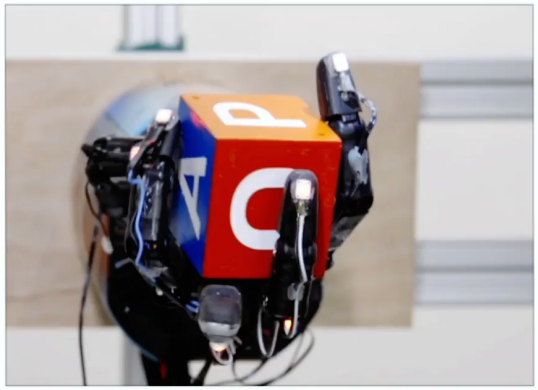
The Task



FINGER PIVOTING



SLIDING



FINGER GAITING

The Task
(Re-orienting an hand)

PROBLEMS TO BE SOLVED:

- **Working in the real world.**
- **High-dimensional control.**
- **Noisy and partial observations.**
- **Manipulating more than one object**

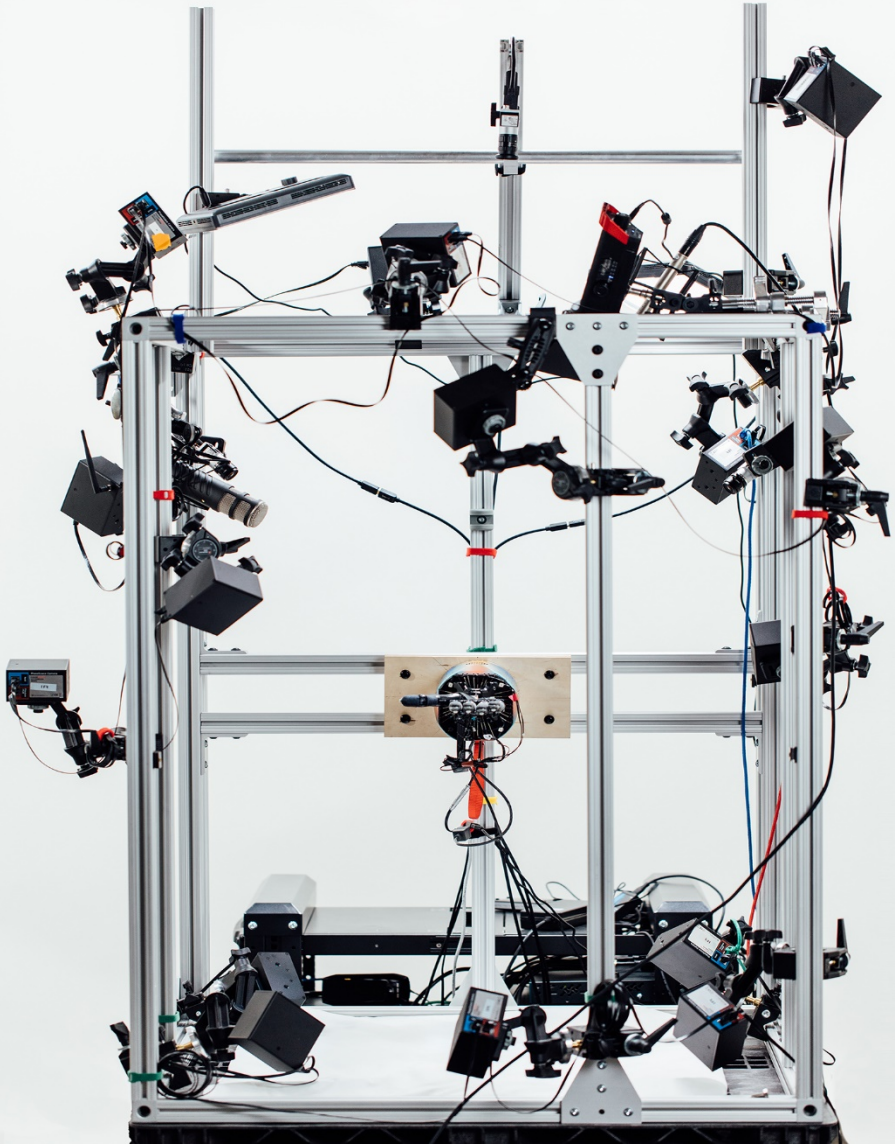
Method – [TRAINING IN SIMULATION]

Training in Simulation Consists of

- Observation
- Randomization

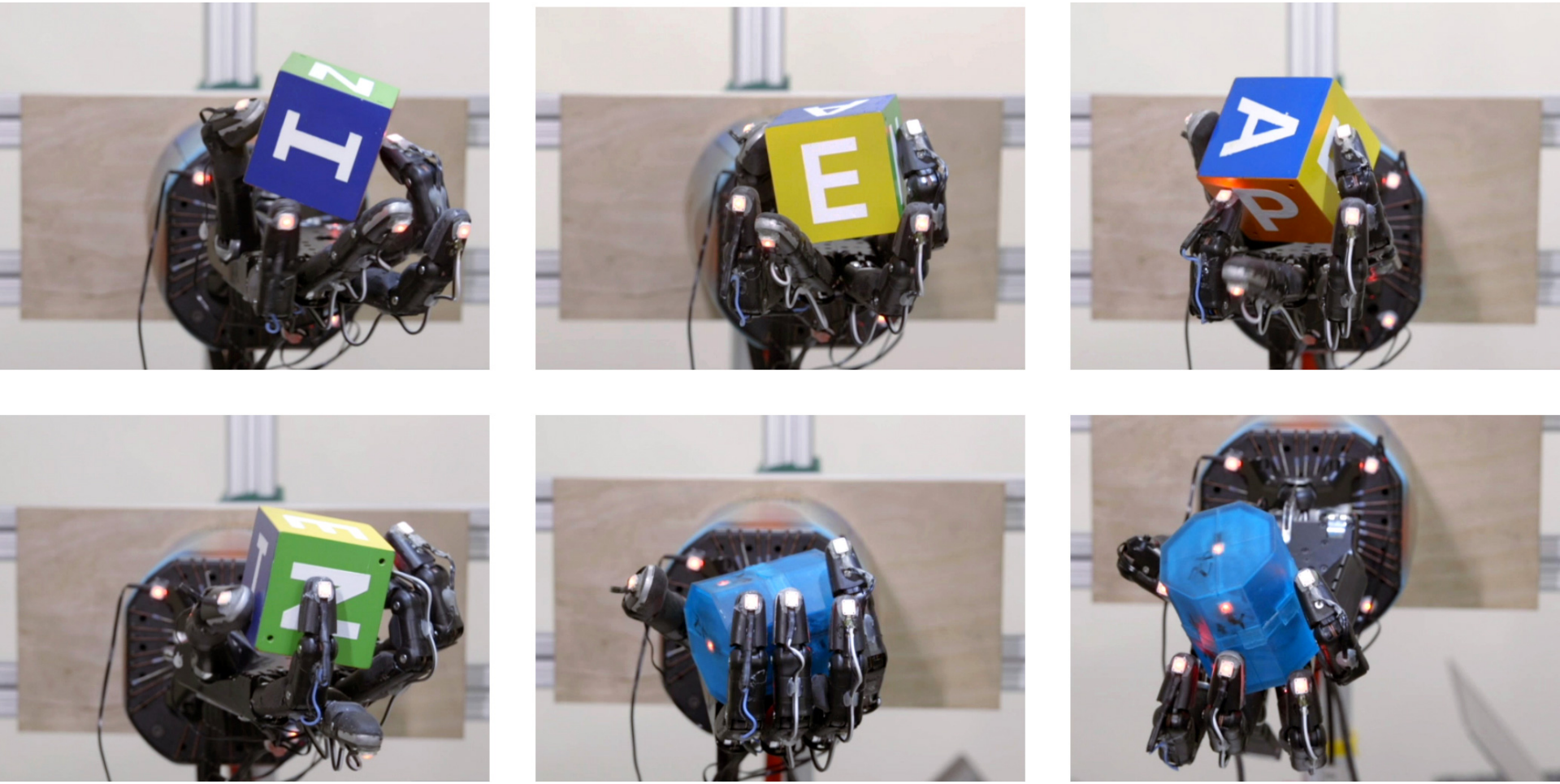
AIM is to collect as much experience as possible

Method - Part 1 - Observation



PhaseSpace motion tracking cameras, and Basler RGB cameras.

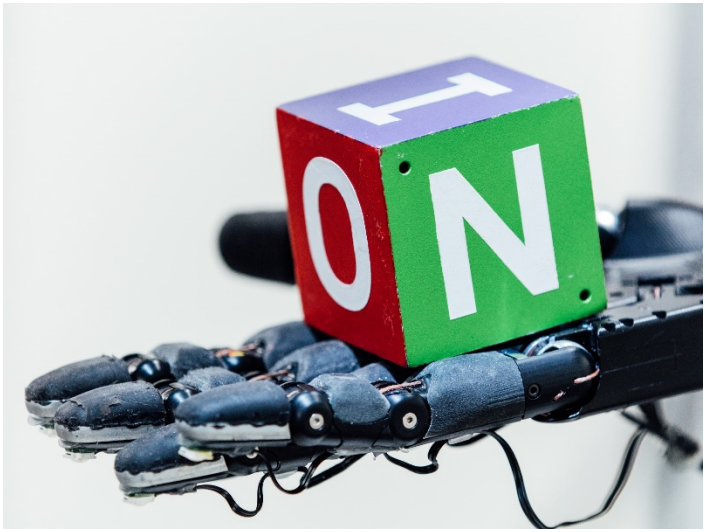
Method - Part 1 - Observation



PhaseSpace motion tracking cameras, and Basler RGB cameras.

Method - Part 1 - Observation

- We give the control policy observations of the of the object pose either from
  - PhaseSpace markers or
  - The vision based pose estimator.



- Although the Shadow Dexterous Hand contains a broad array of built-in sensors, we specifically avoided providing these as observations to the policy.

Method – part 2 - Randomization

- **Address sim2real gap**

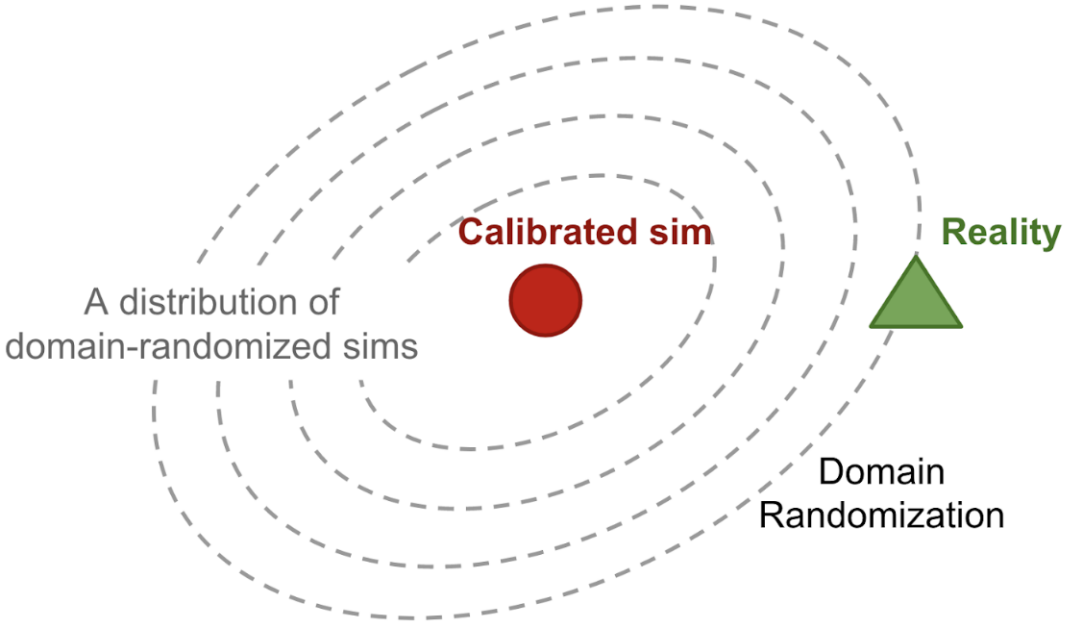
Method – part 2 - Randomization

- Address sim2real gap

- **Domain randomization:**
  - Observation noise:
  - Unmodeled effects:
  - Visual appearance randomizations
  - Physics randomization

Method – part 2 - Randomization

- **Domain randomization:** learns in a simulation which is designed to provide a variety of experiences rather than maximizing realism.



Method – part 2 - Randomization

- **Domain randomization:** learns in a simulation which is designed to provide a variety of experiences rather than maximizing realism.

We randomize most of the aspects of the simulated environment in order to learn both a policy and a vision model that generalizes to reality

Method – part 2 - Randomization

- **Observation noise**: To better mimic the kind of noise we expect to experience in reality, we add noise to policy observations

- **Unmodeled effects:**

The physical robot experiences many effects that are not modeled by our simulation.

  - Eg. simulate marker occlusion by freezing its simulated position whenever it is close to another marker or the object.
  - We also handle imperfect actuation and other unmodeled dynamics.

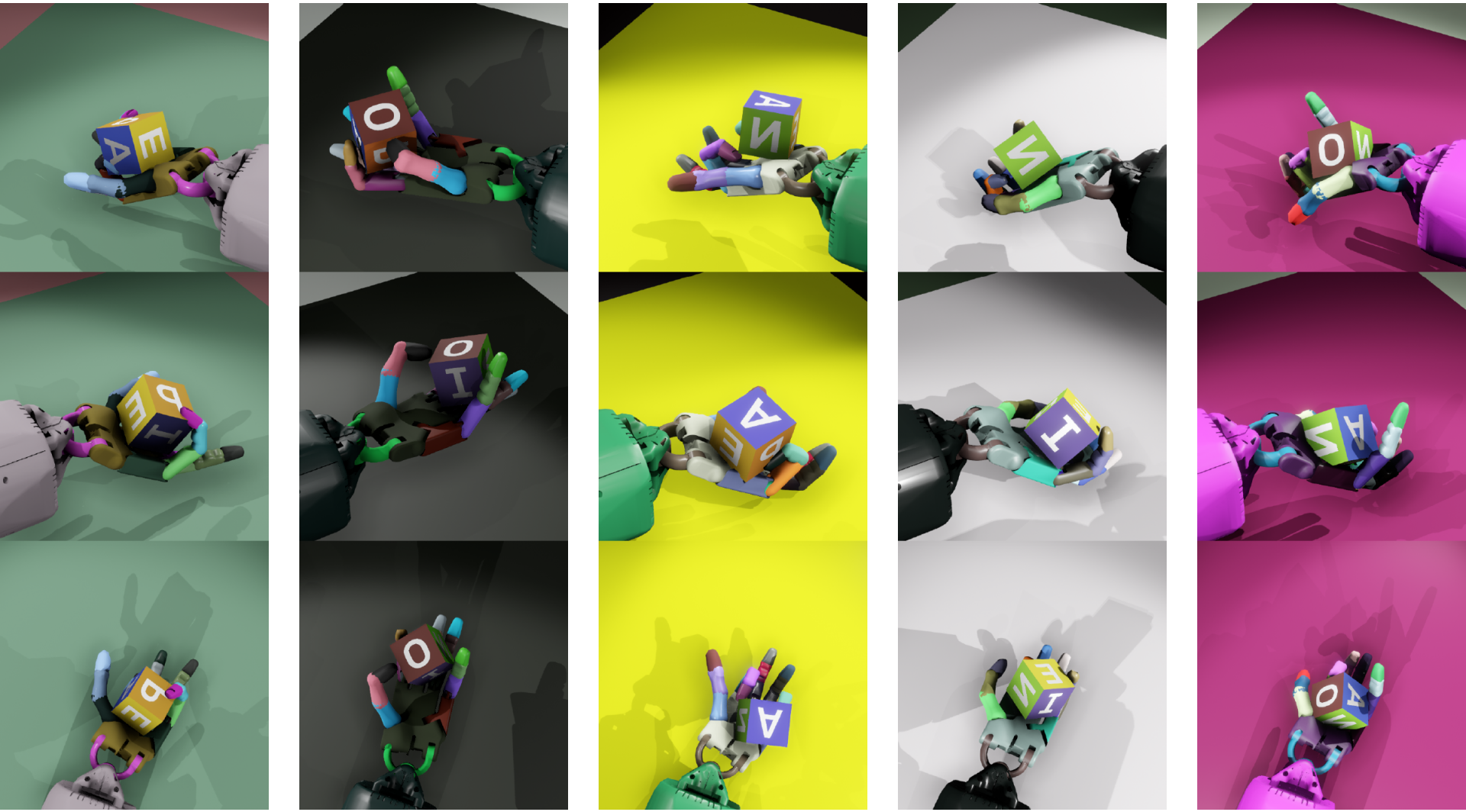
Method – part 2 - Randomization

- **<u>Visual appearance randomizations:</u>**

We randomize the following aspects of the rendered scene:

- Camera positions and intrinsics,
- lighting conditions,
- the pose of the hand and object,
- and the materials and textures for all objects in the scene.

Method – part 2 - Randomization

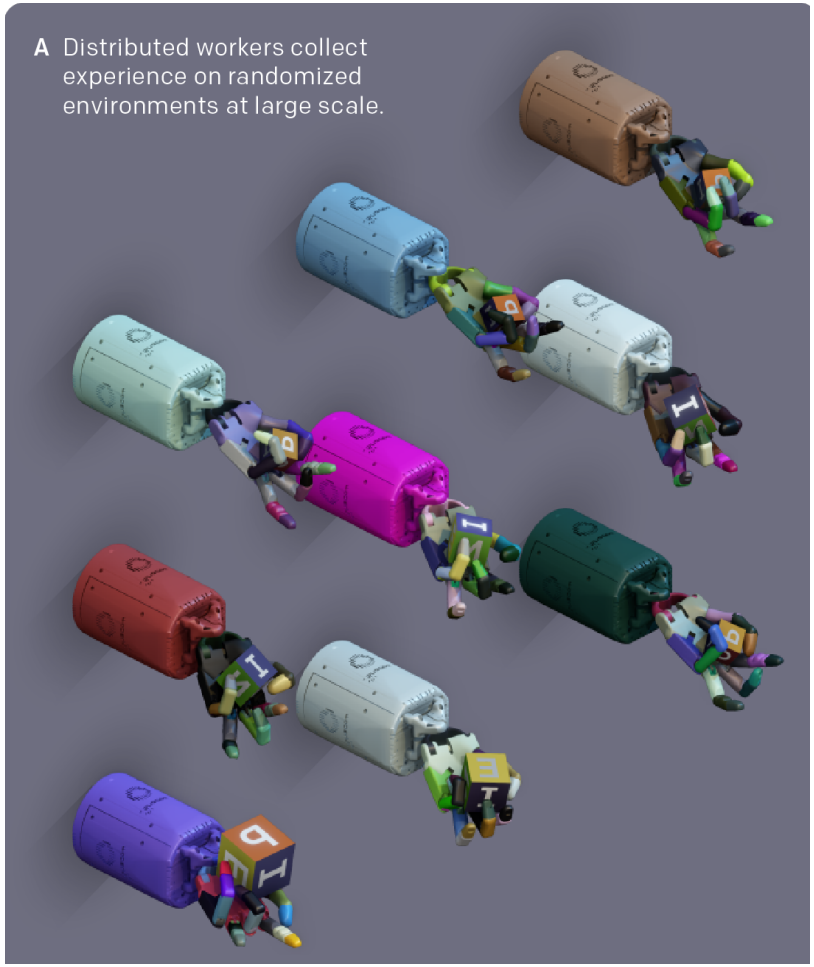


Method – part 2 - Randomization

- **Physics randomizations:**

Table 1: Ranges of physics parameter randomizations.

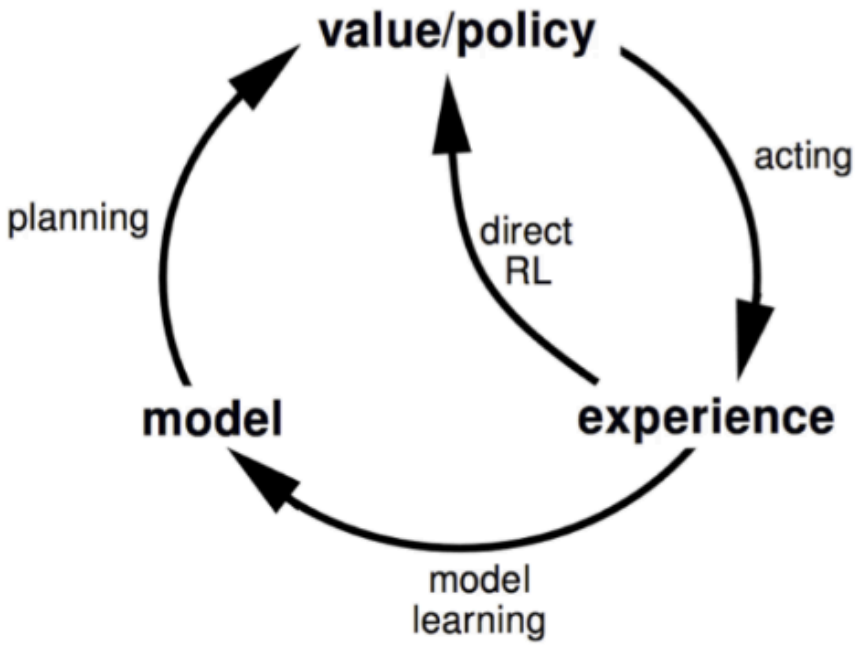
Parameter	Scaling factor range	Additive term range
object dimensions	uniform([0.95, 1.05])	
object and robot link masses	uniform([0.5, 1.5])	
surface friction coefficients	uniform([0.7, 1.3])	
robot joint damping coefficients	loguniform([0.3, 3.0])	
actuator force gains (P term)	loguniform([0.75, 1.5])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



After the completion of this step, Experience is collected.

Method – Learning Control Policy

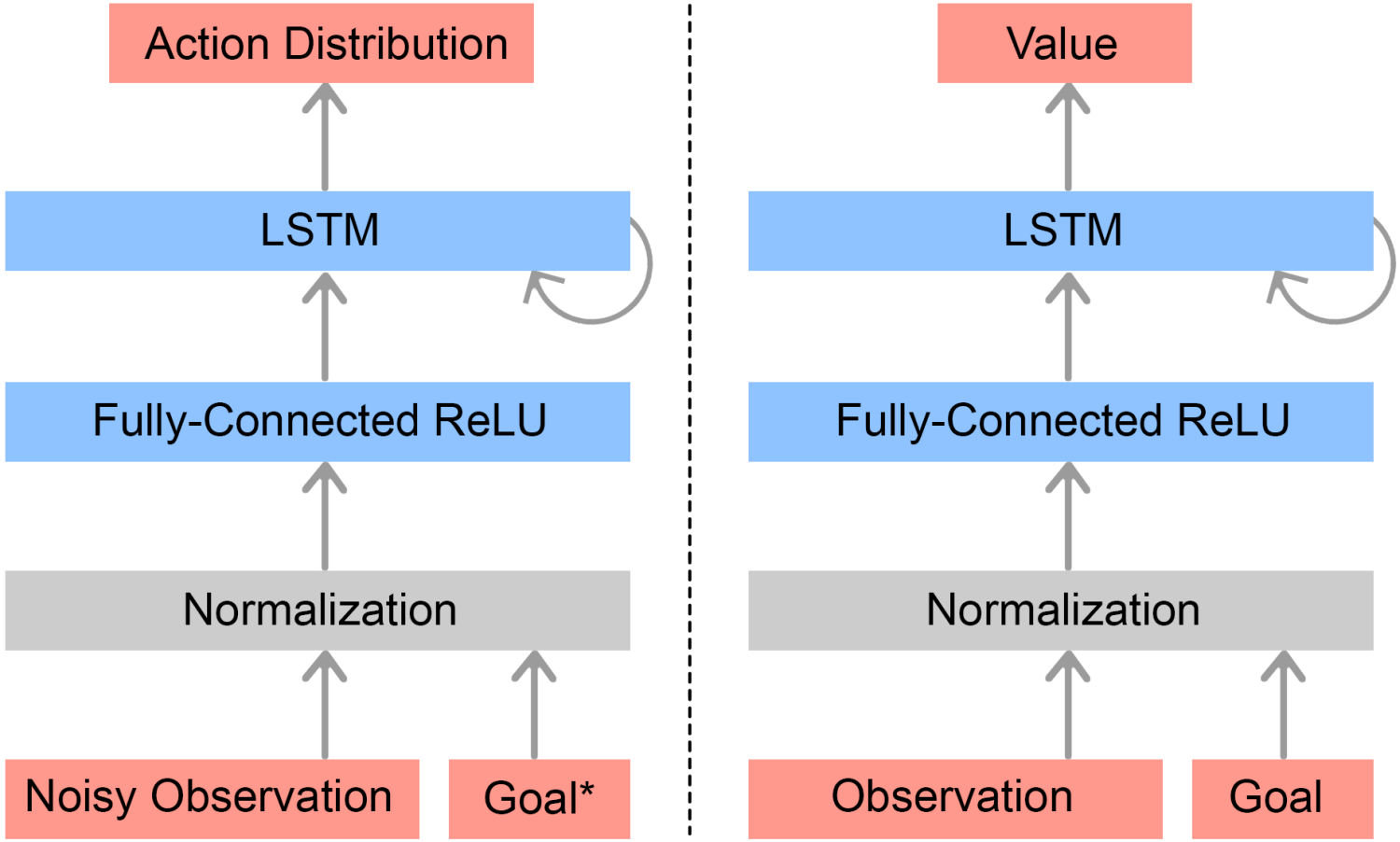
- Policy Architecture
- Actions and Rewards
- Distributed Training



Method – Learning Control Policy

- **Policy Architecture:**

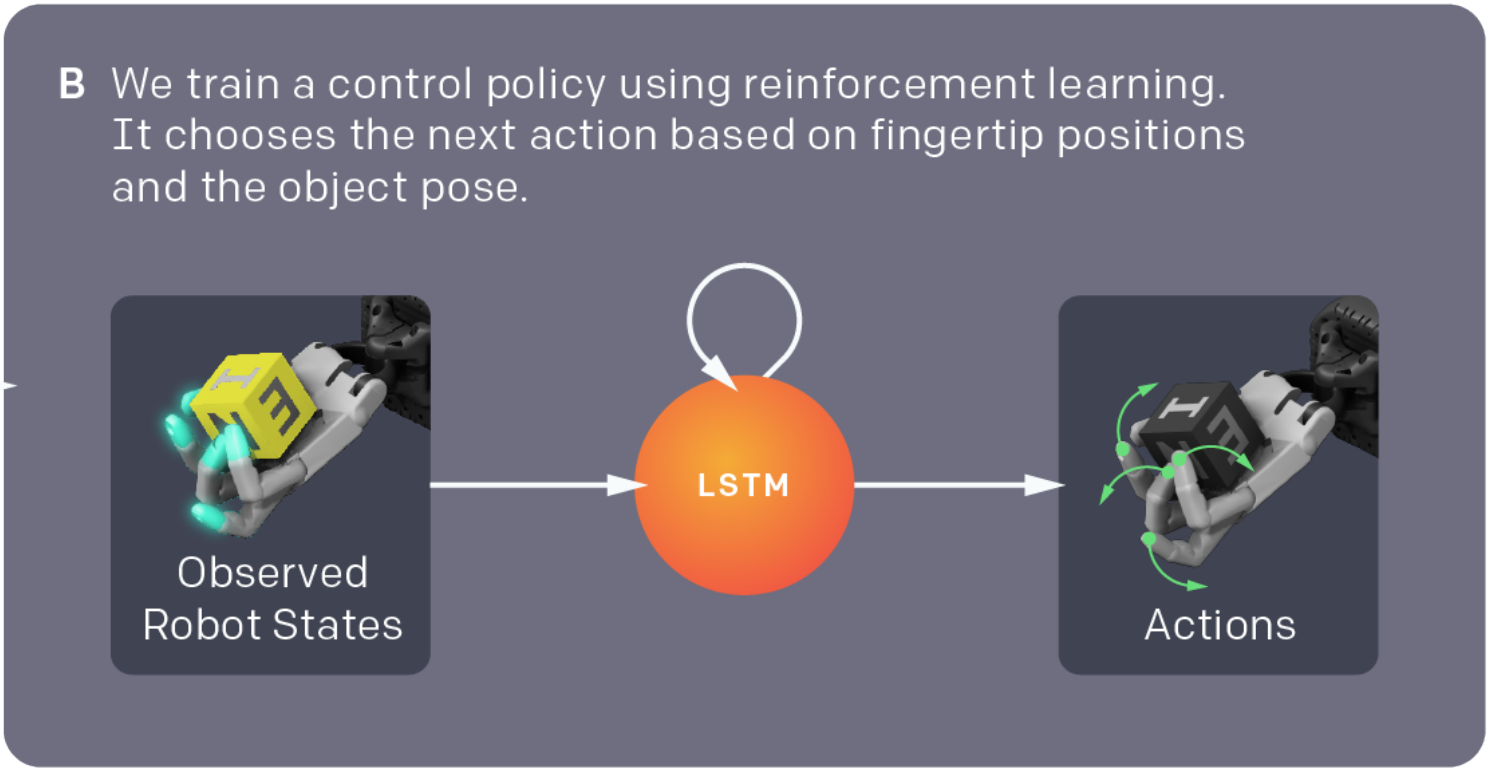
We use Proximal policy optimization:



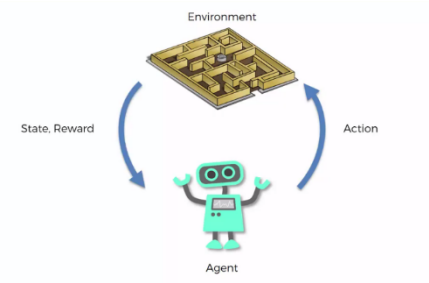
Method – Learning Control Policy

- **Policy Architecture – Why LSTM?**

The initial steps of interaction with the environment can reveal the weight of the object or how fast the index finger can move



Method – Learning Control Policy



- **Actions**

Policy actions correspond to desired joints angles relative to the current ones (e.g. rotate this joint by 10 degrees). While PPO can handle both continuous and discrete action spaces, it is noticeable that discrete action spaces work much better.

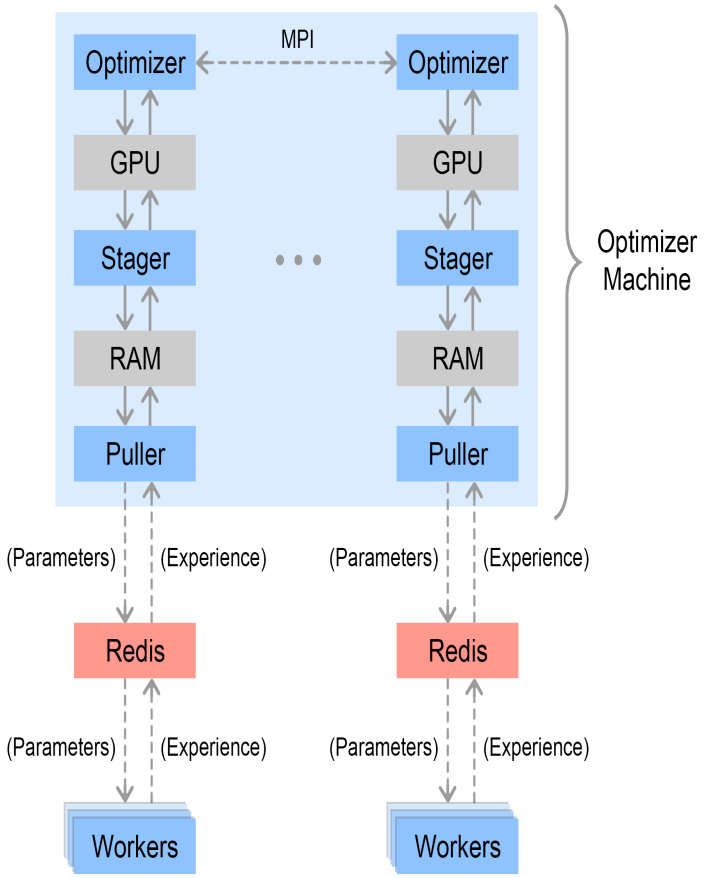
- **Rewards**

The reward given at timestep t is $r_t = d_t - d_{t+1}$ We give an additional reward of 5 whenever a goal is achieved and a reward of - 20 (a penalty) whenever the object is dropped

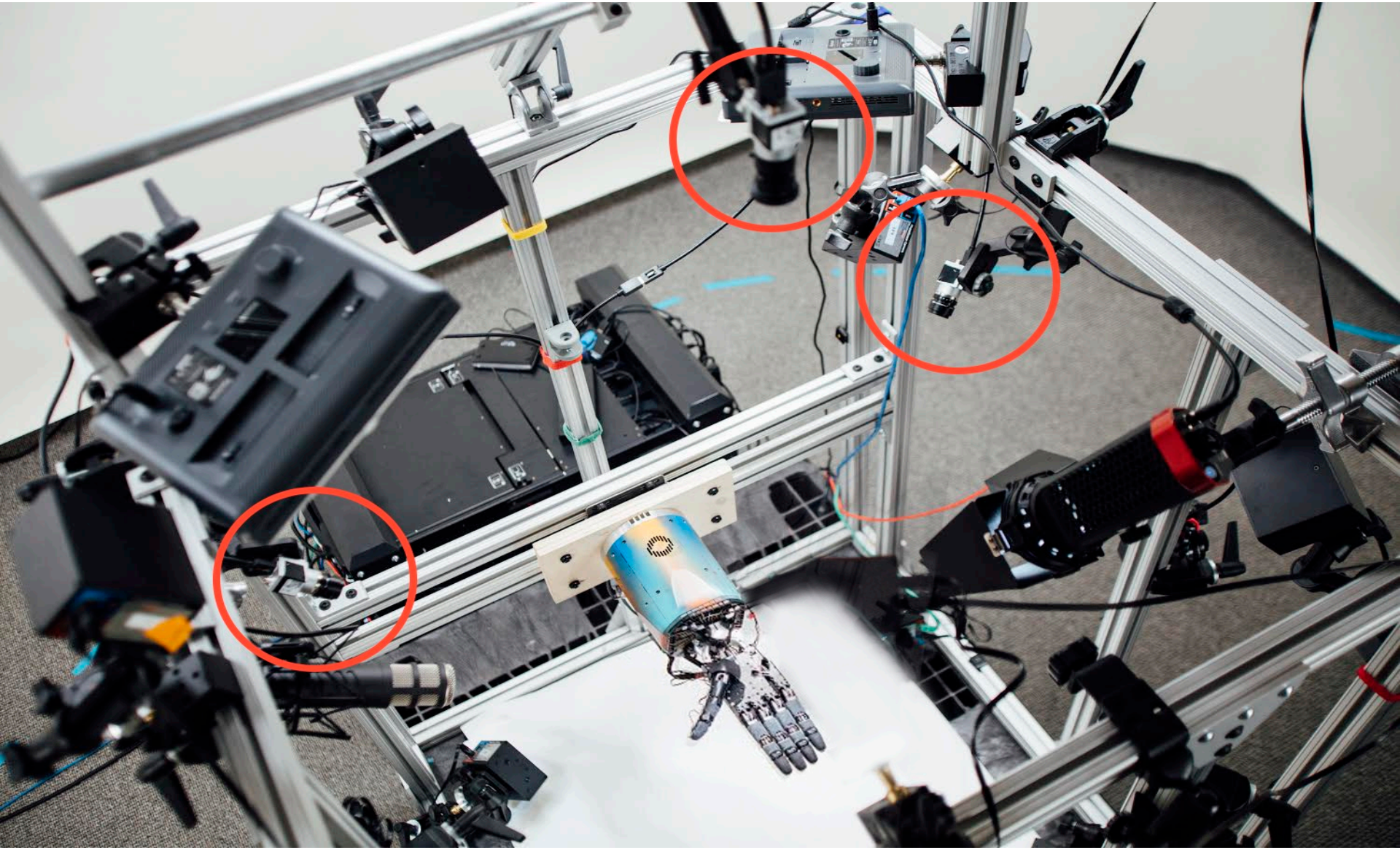
Method – Learning Control Policy

- **Distributed Training:**

We use the same distributed implementation of PPO that was used to train OpenAI Five without modifications.



State Estimation from Vision



3-camera setup for vision-based state estimation.

State Estimation from Vision

The policy that we describe in the previous section takes the object's position as input and requires a motion capture system for tracking the object on the physical robot.

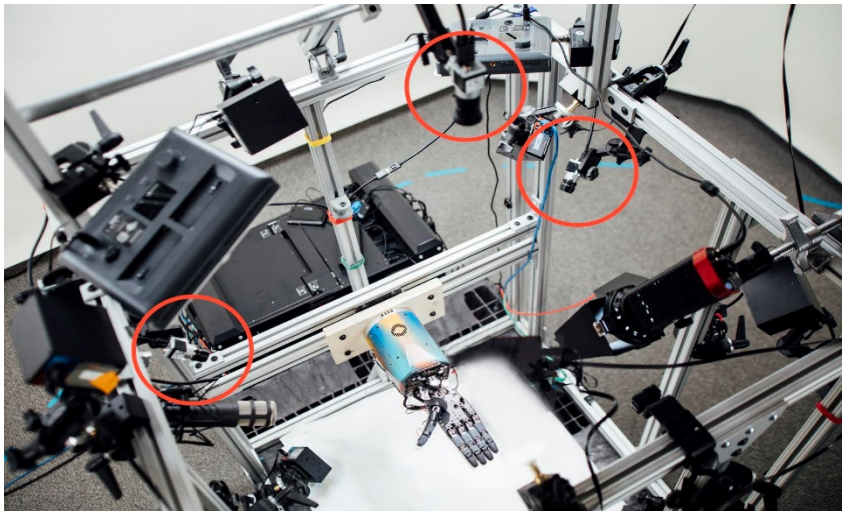
Why do we need State Estimation from Vision now?

So, for the State Estimation from Vision, we need to now understand the

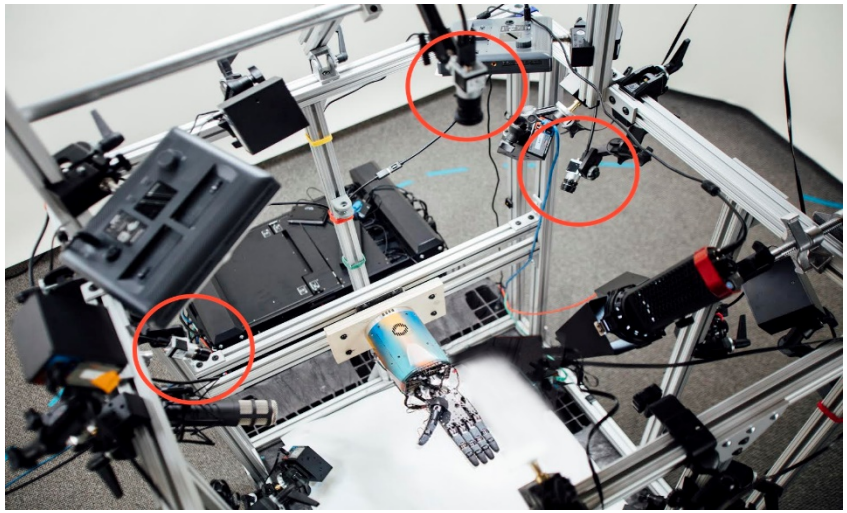
- Model Architecture

AND

- Training



State Estimation from Vision

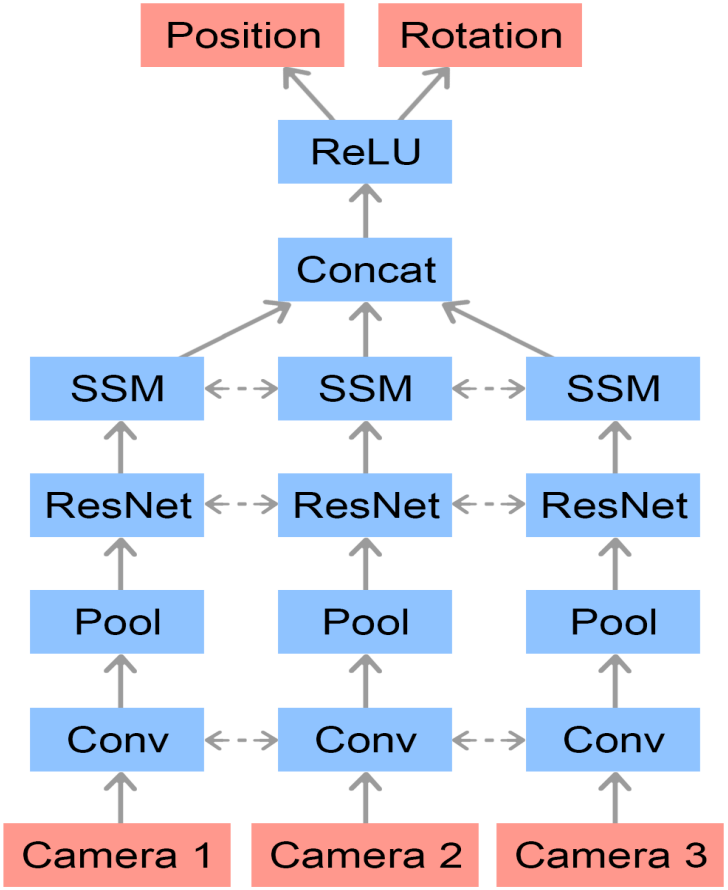


Setup:

We use three RGB cameras mounted with differing viewpoints of the scene. The recorded images are passed through a convolutional neural network, which is depicted in Figure next slide.

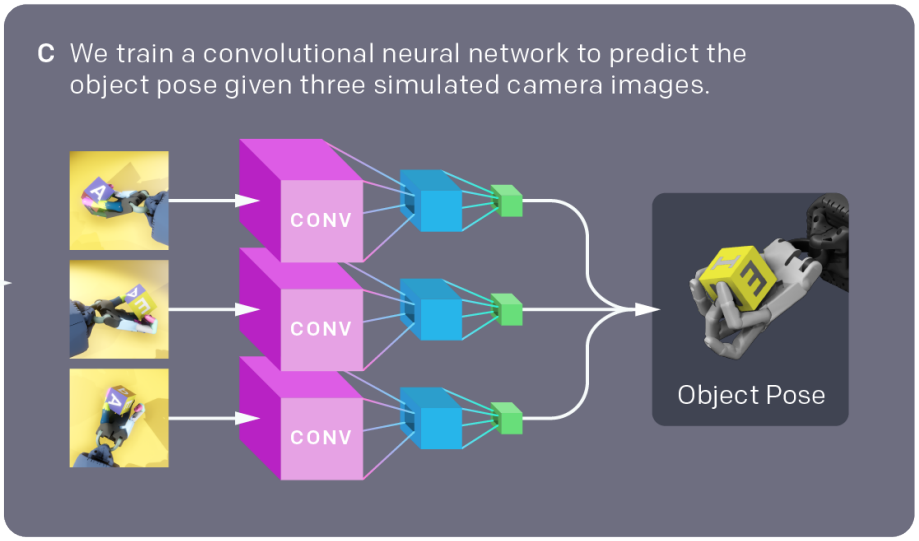
State Estimation from Vision

- **Model Architecture:** The network predicts both the position and the orientation of the object.



State Estimation from Vision

- **Training**

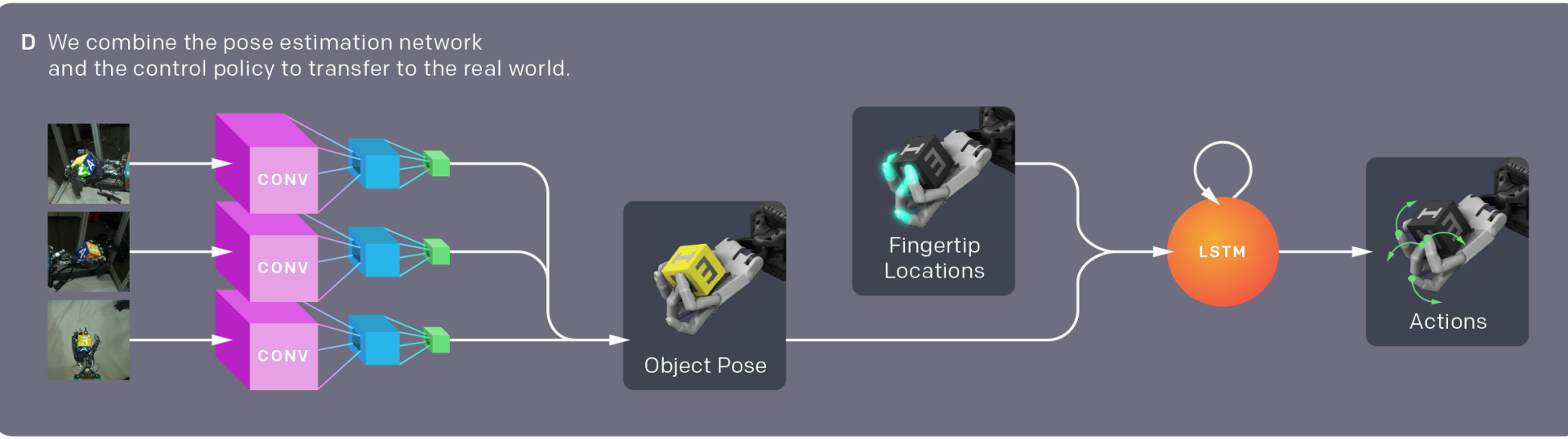


1. We run the trained policy in the simulator until we gather one million states.

2. We then train the vision network by minimizing the mean squared error between the normalized prediction and the ground truth with mini-batch gradient descent.

3. We augment the data by modifying the object pose.

Results



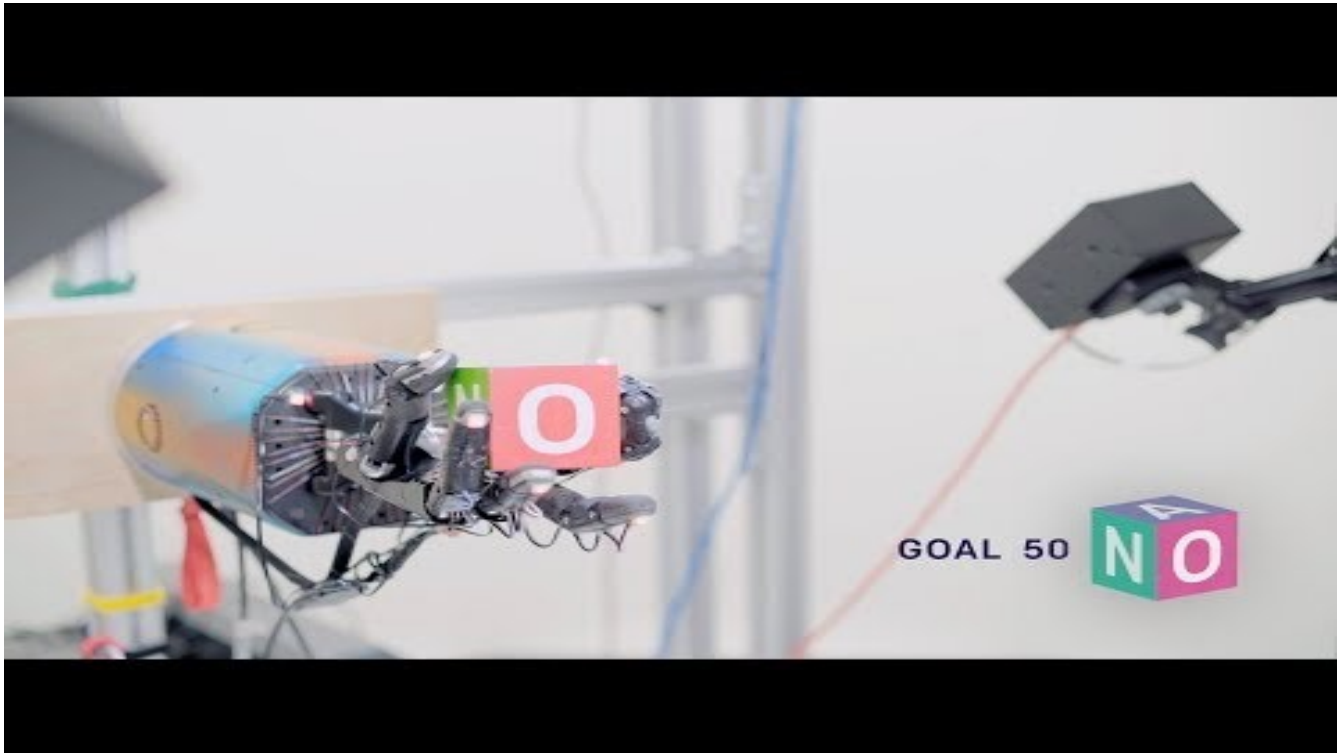
I. Quantitative Results

II. Qualitative Results

Results

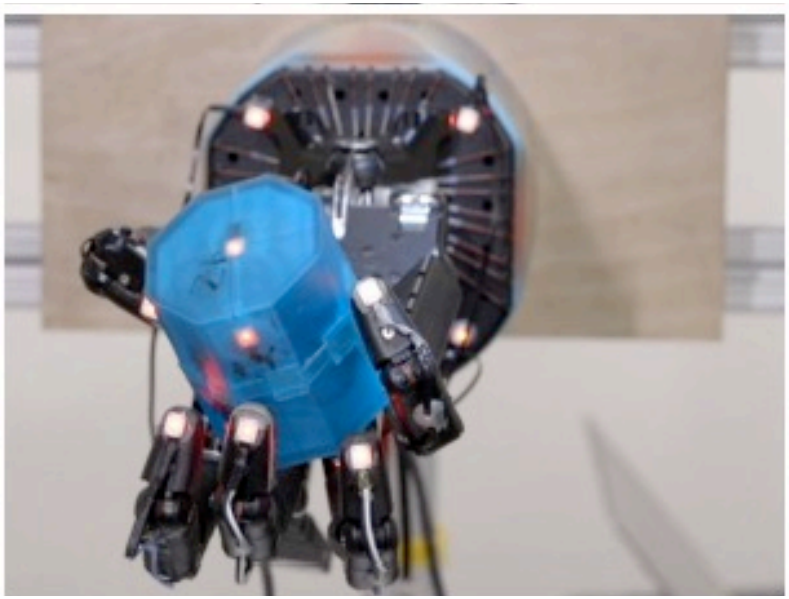
- **Quantitative Results:**

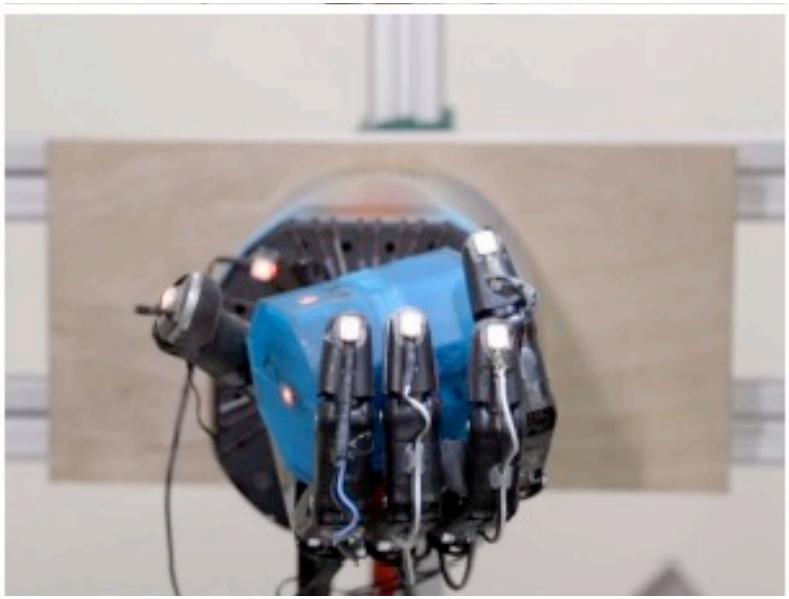
The number of consecutive successful rotations until the object is either dropped, a goal has not been achieved within 80 seconds, or until 50 rotations are achieved is measured.



Other Experimentation

We also evaluate the performance on a second type of object, an octagonal prism.





To do so, we fine-tuned a trained block rotation control policy with the octagonal prism as the target object instead of the block.

What would be the Performance comparison?

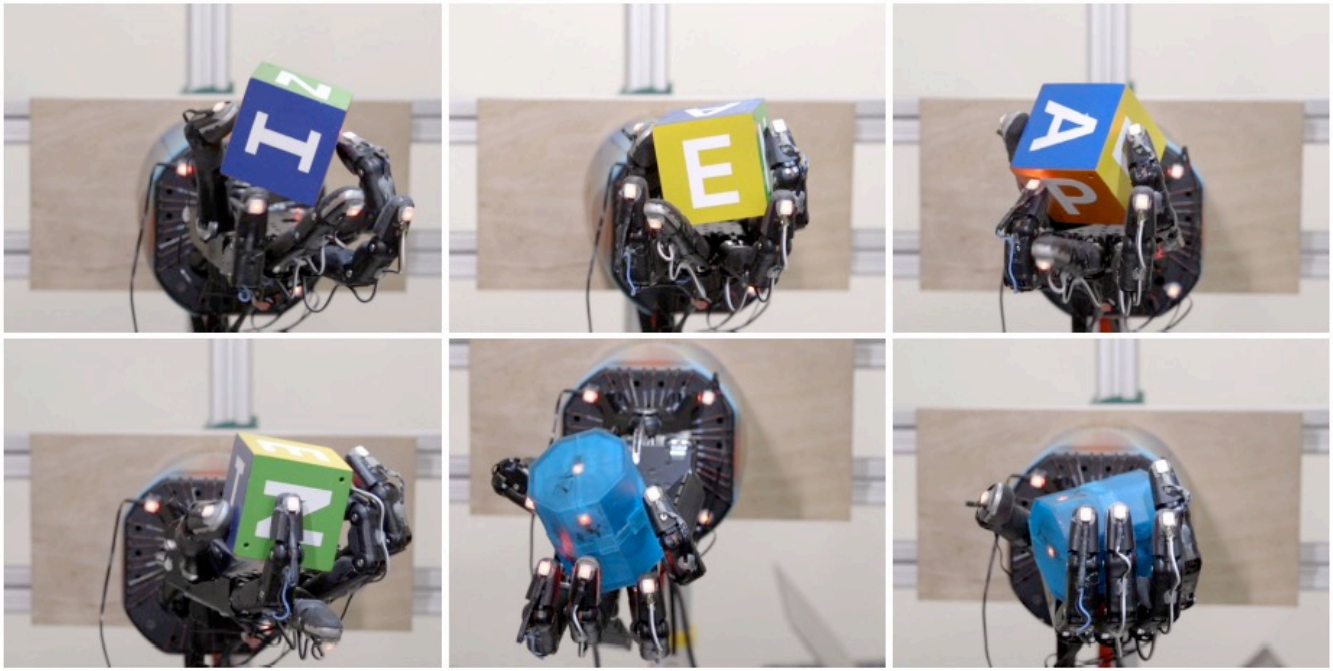
Results

- **Quantitative Results:**

Simulated task	Mean	Median
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
Block (state, locked wrist)	26.4 ± 13.4	28.5
Block (vision)	15.2 ± 14.3	11.5
Octagonal prism (state)	7.8 ± 7.8	5

Results

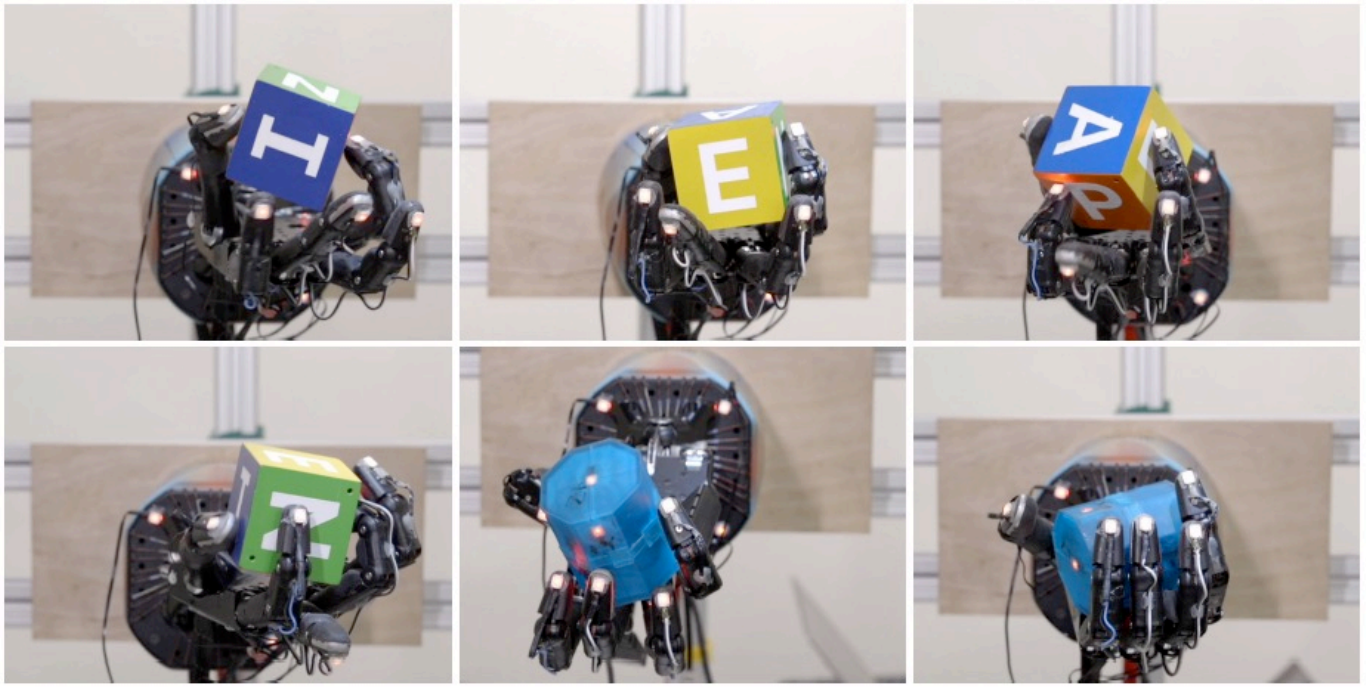
- **Qualitative Results**



- Many of the grasps seen in humans are found, see in the figure above.
- We do not use any human demonstrations

Results

- **Qualitative Results**



- For precision grasps, our policy tends to use the little finger instead of the index or middle finger. This means that our system can rediscover grasps found in humans.

Surprising Observations

- Tactile sensing is not necessary to manipulate real-world objects.
- Randomizations developed for one object generalize to others with similar properties.
- Using real data to train our vision policies didn't make a difference
- Decreasing reaction time did not improve performance.

Limitations

- Even though randomizations and calibration narrow the reality gap, it still exists and performance on the real system is worse than in simulation.

Simulated task	Mean	Median
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
Block (state, locked wrist)	26.4 ± 13.4	28.5
Block (vision)	15.2 ± 14.3	11.5
Octagonal prism (state)	7.8 ± 7.8	5

Future Works

- This suggests that further tuning is necessary and that the introduction of additional randomization could improve transfer to the physical system.

- It would also be interesting to train a unified policy that can handle multiple objects.

CONCLUSION

1. Demonstrated that in-hand manipulation skills learned with RL in a simulator can achieve an unprecedented level of dexterity on a physical five-fingered hand

2. Contemporary deep RL algorithms can be applied to solving complex real-world robotics problems which are beyond the reach of existing non-learning-based approaches.

THANK YOU!

Questions?