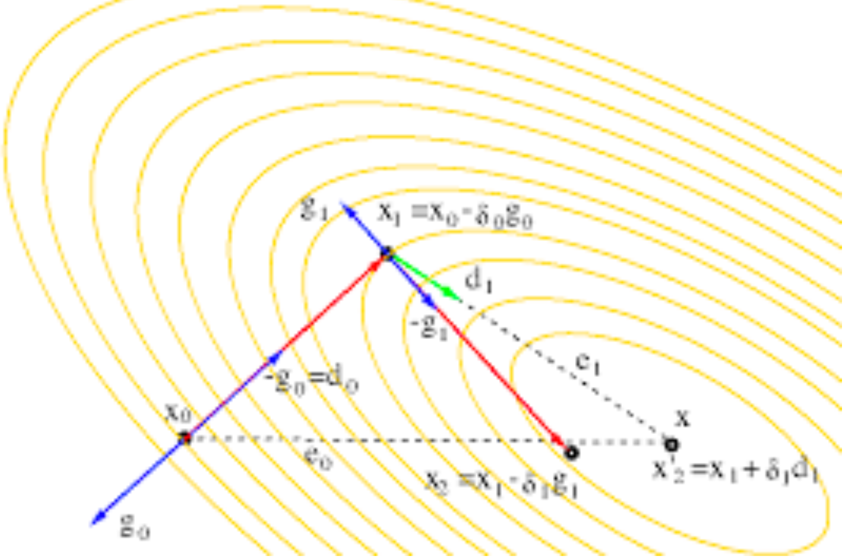
Randomization and the reality gap: how to transfer robotic policies from sim to real

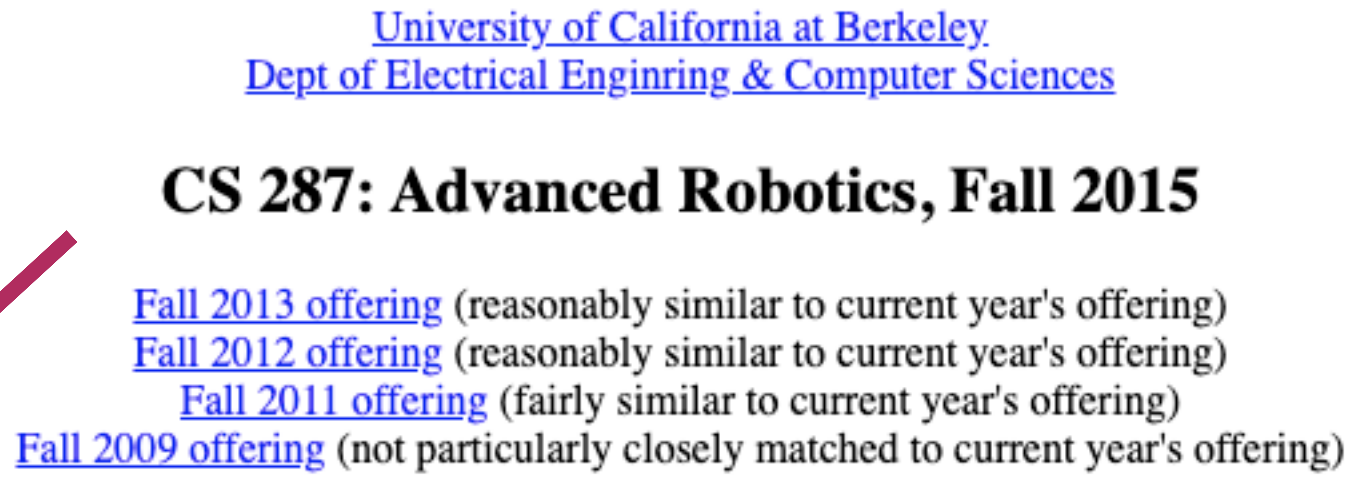
Josh Tobin
@josh_tobin_

About me

Pure math

Applied math @ Berkeley





[University of California at Berkeley](#)
[Dept of Electrical Enginring & Computer Sciences](#)

CS 287: Advanced Robotics, Fall 2015

[Fall 2013 offering](#) (reasonably similar to current year's offering)
[Fall 2012 offering](#) (reasonably similar to current year's offering)
[Fall 2011 offering](#) (fairly similar to current year's offering)
[Fall 2009 offering](#) (not particularly closely matched to current year's offering)

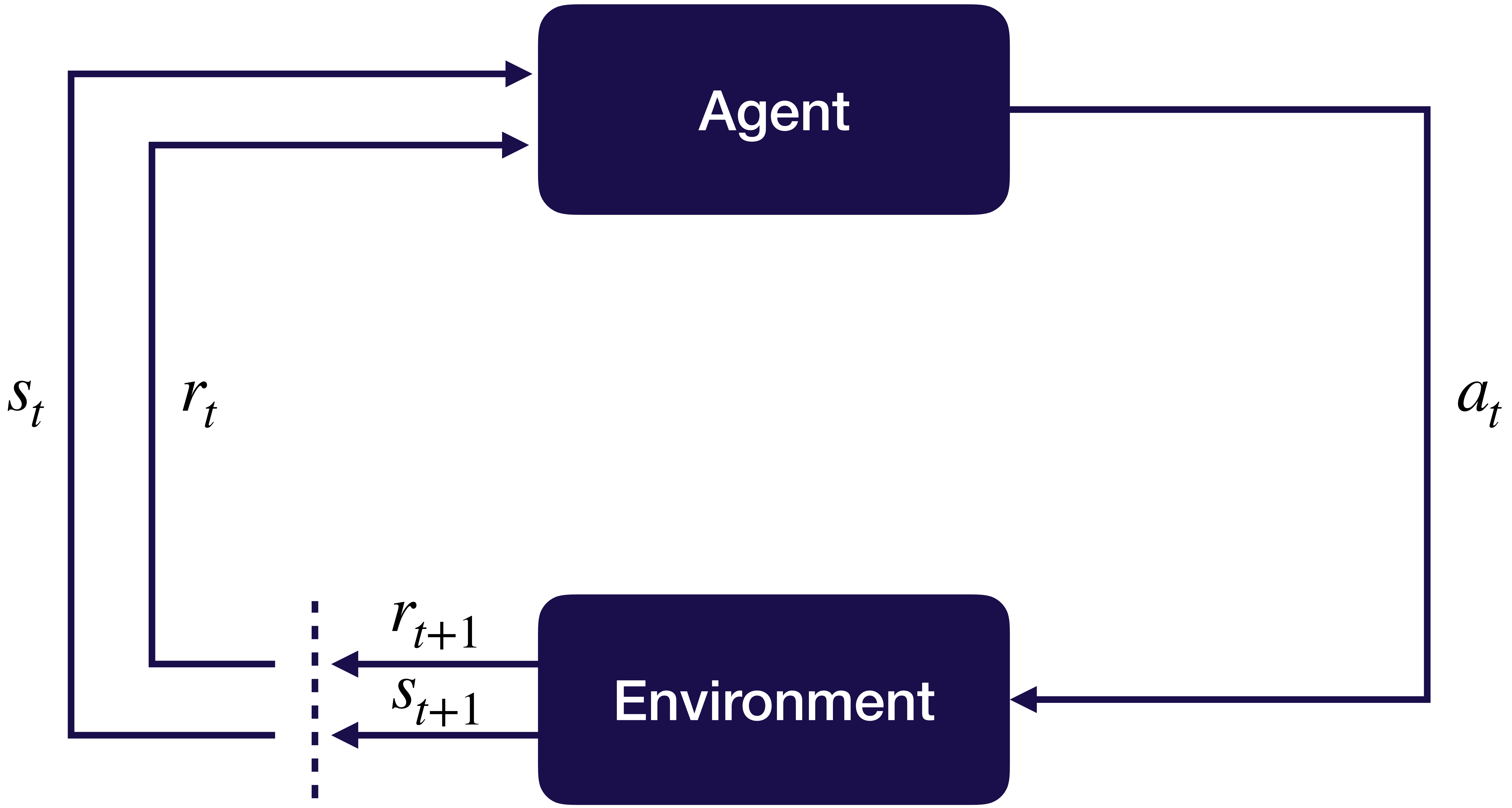
Management consulting

Deep Learning & Robotics at Berkeley / OpenAI

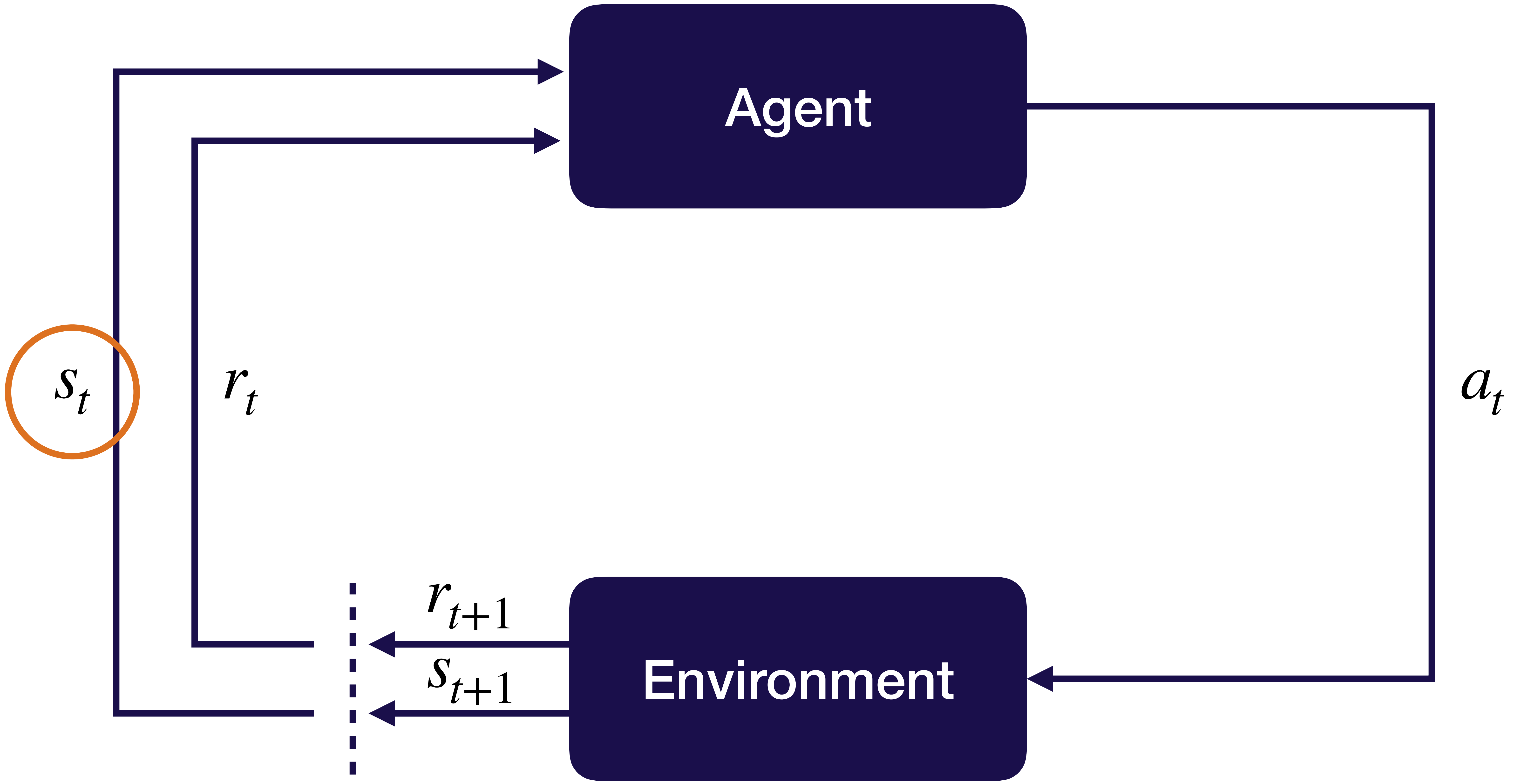
What I learned in CS287

Robotics is really hard

The real world isn't an MDP

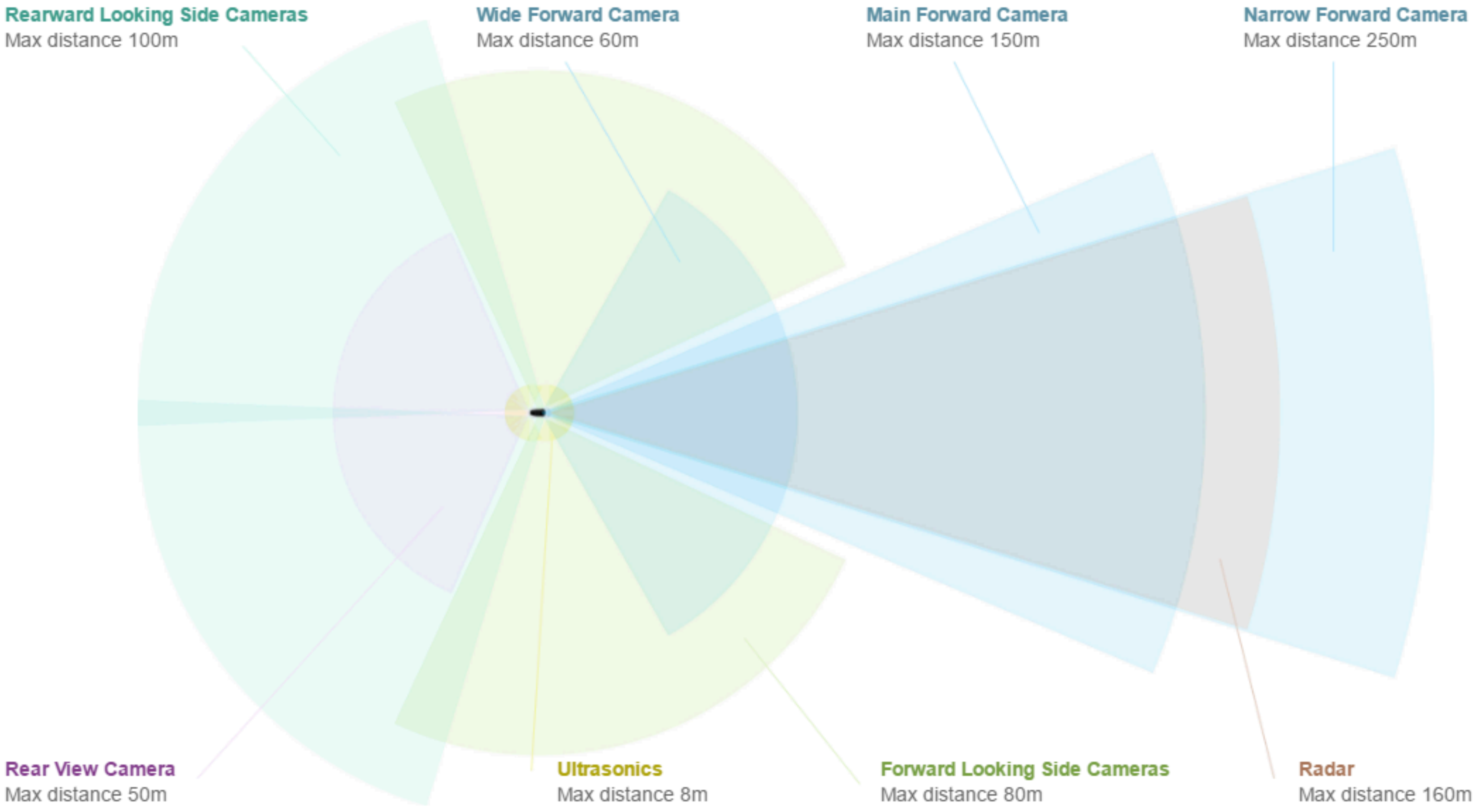


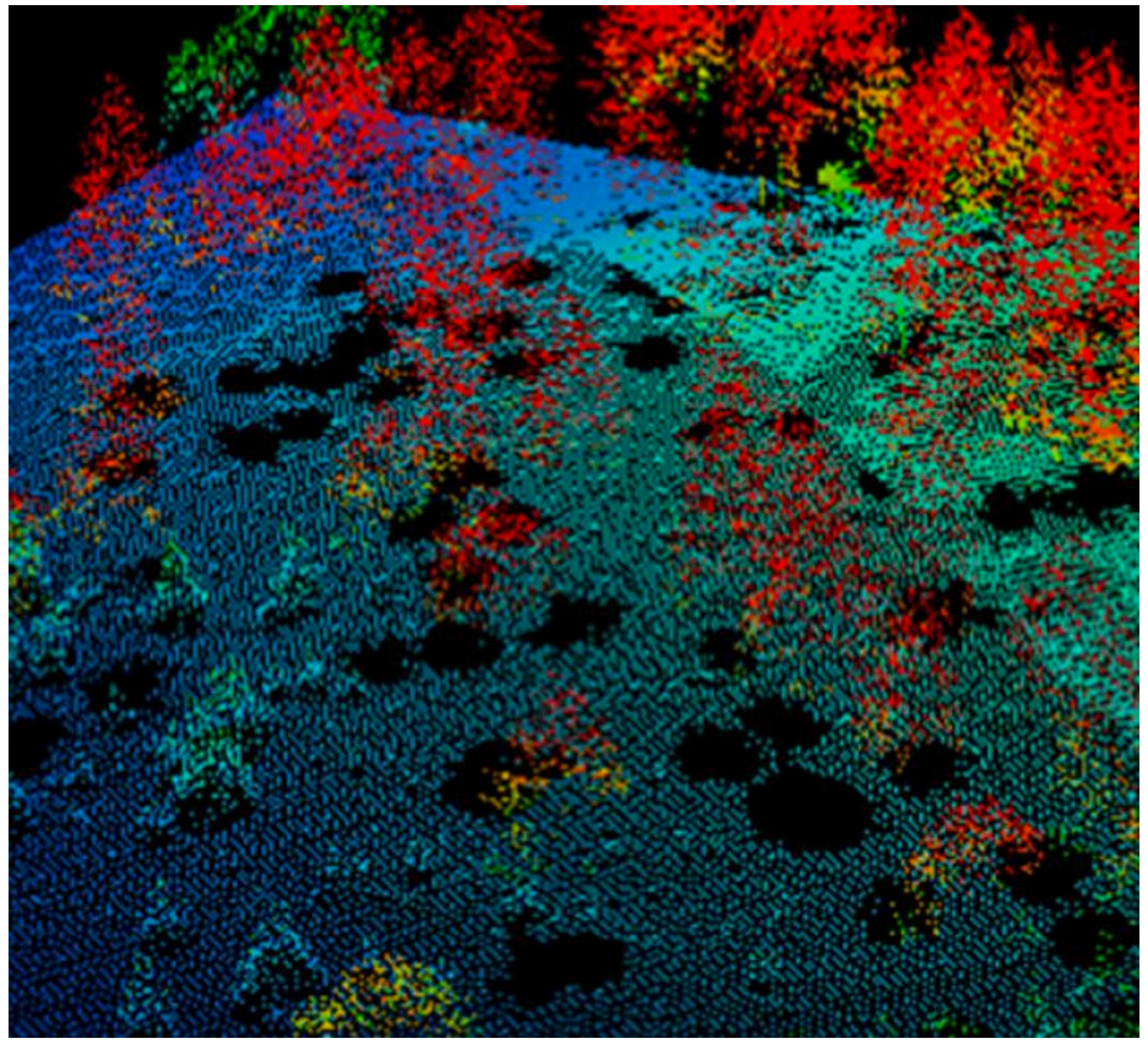
The real world isn't an MDP

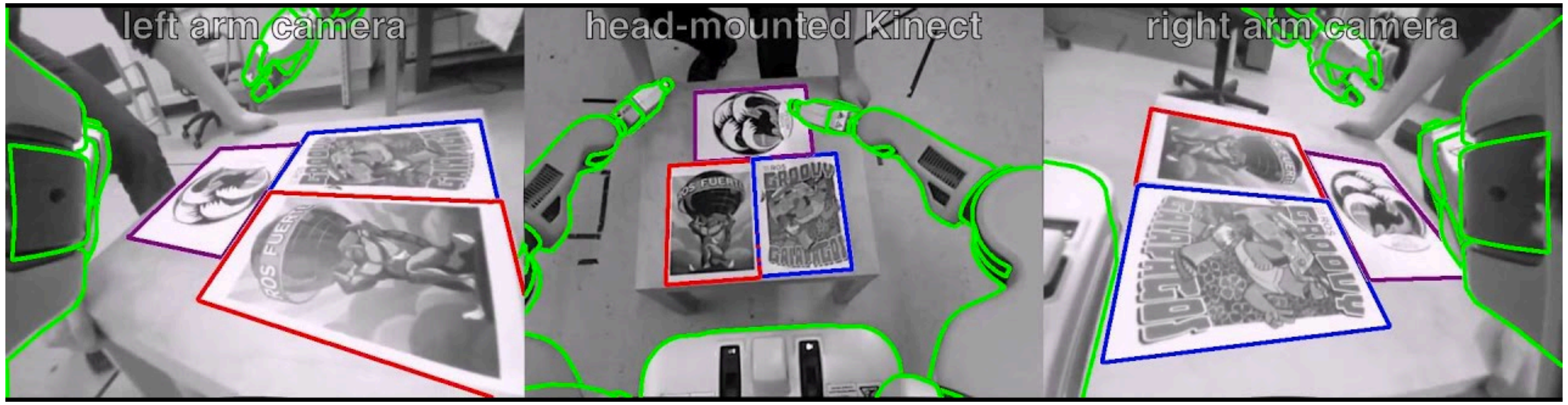


Real-world states are complex and ambiguous

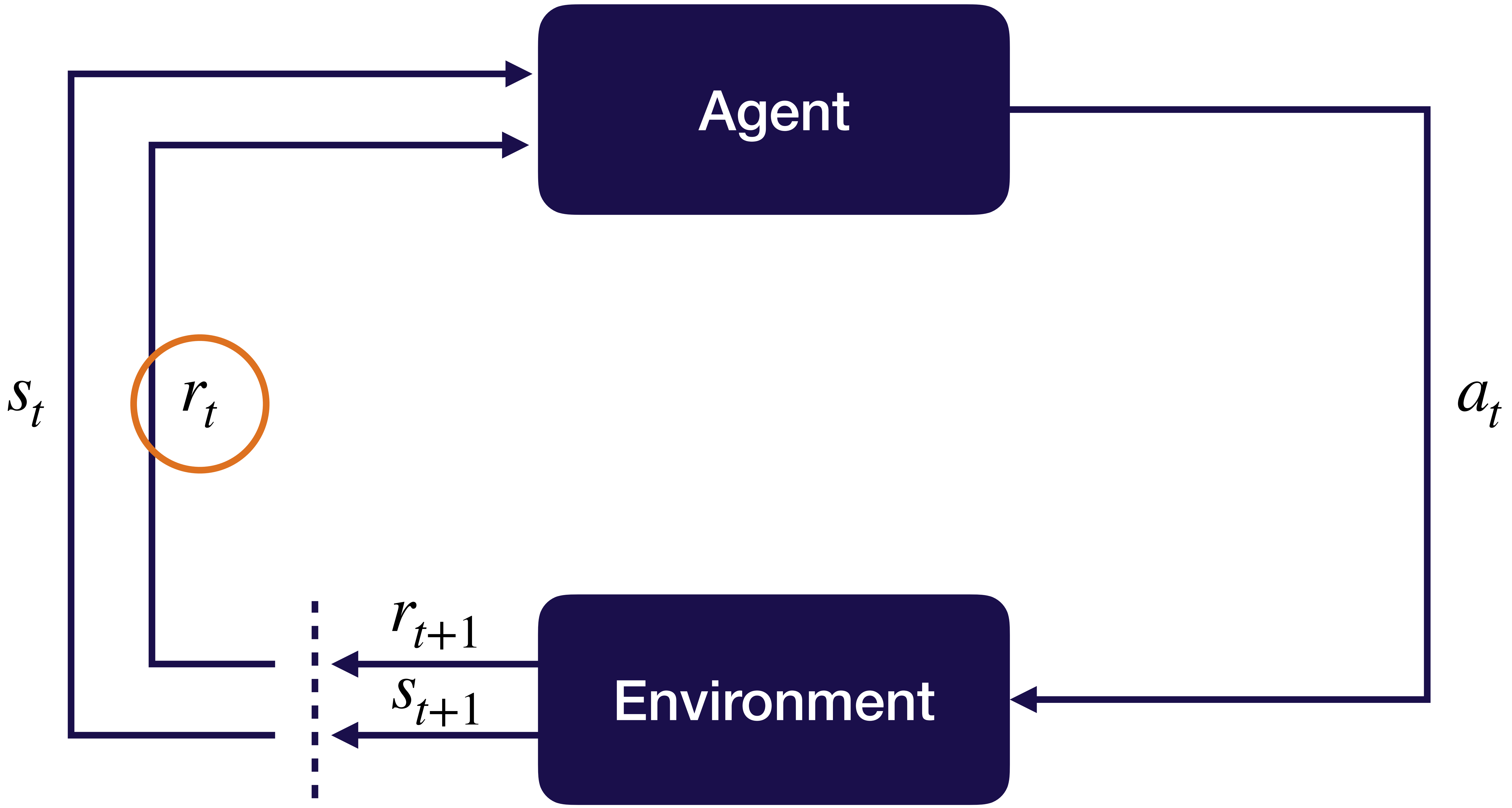
Observations are high-dimensional, multimodal, and noisy





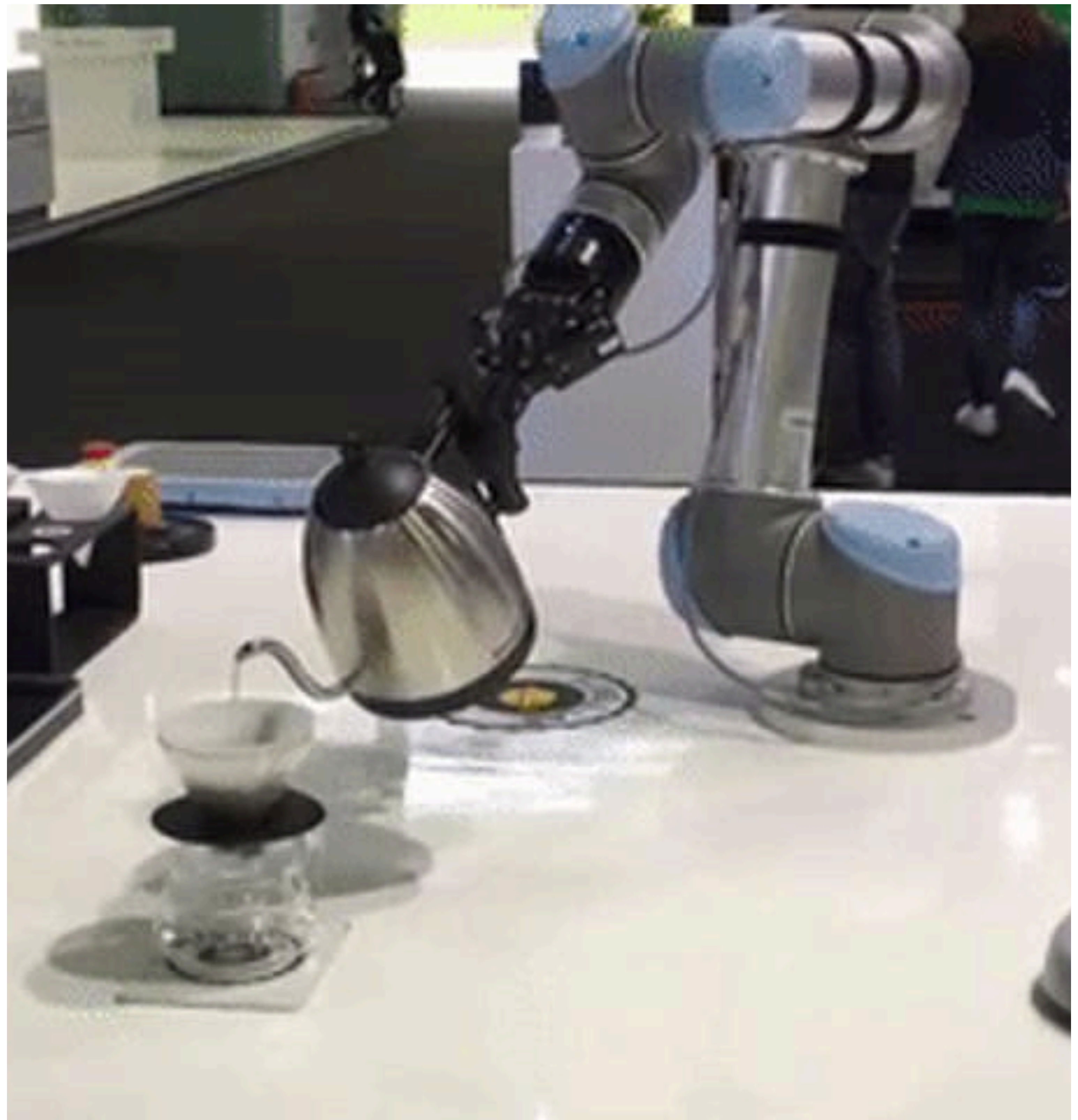


The real world isn't an MDP

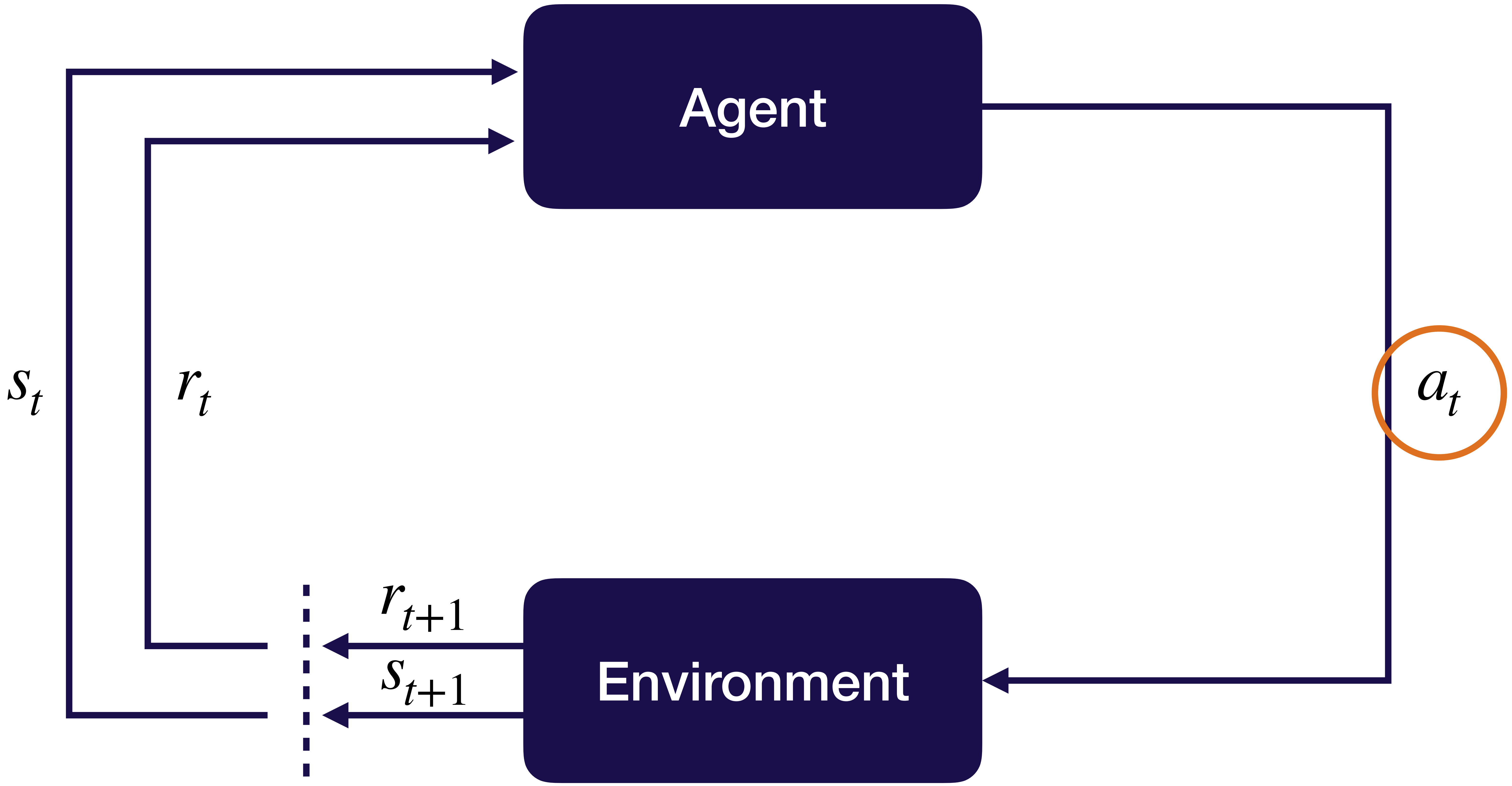


Where does reward come from in the real world?

- How do you specify a reward function for…
    - Folding a towel
    - Cooking dinner
    - Making a user happy
- How do you measure reward outside of the lab?



The real world isn't an MDP



Designing controllers is a hard engineering problem

- Requires good understanding of the system
- Doesn’t scale well to high-dimensional systems
- “Manipulation breaks all the rigorous/reliable approaches I know for control” - Russ Tedrake (MIT / Toyota Research Institute)
- Robots break and degrade all the time

Can deep learning + RL make robotics easy?

Rather than understanding your environment, simply collect a lot of experience and let the algorithm handle the rest

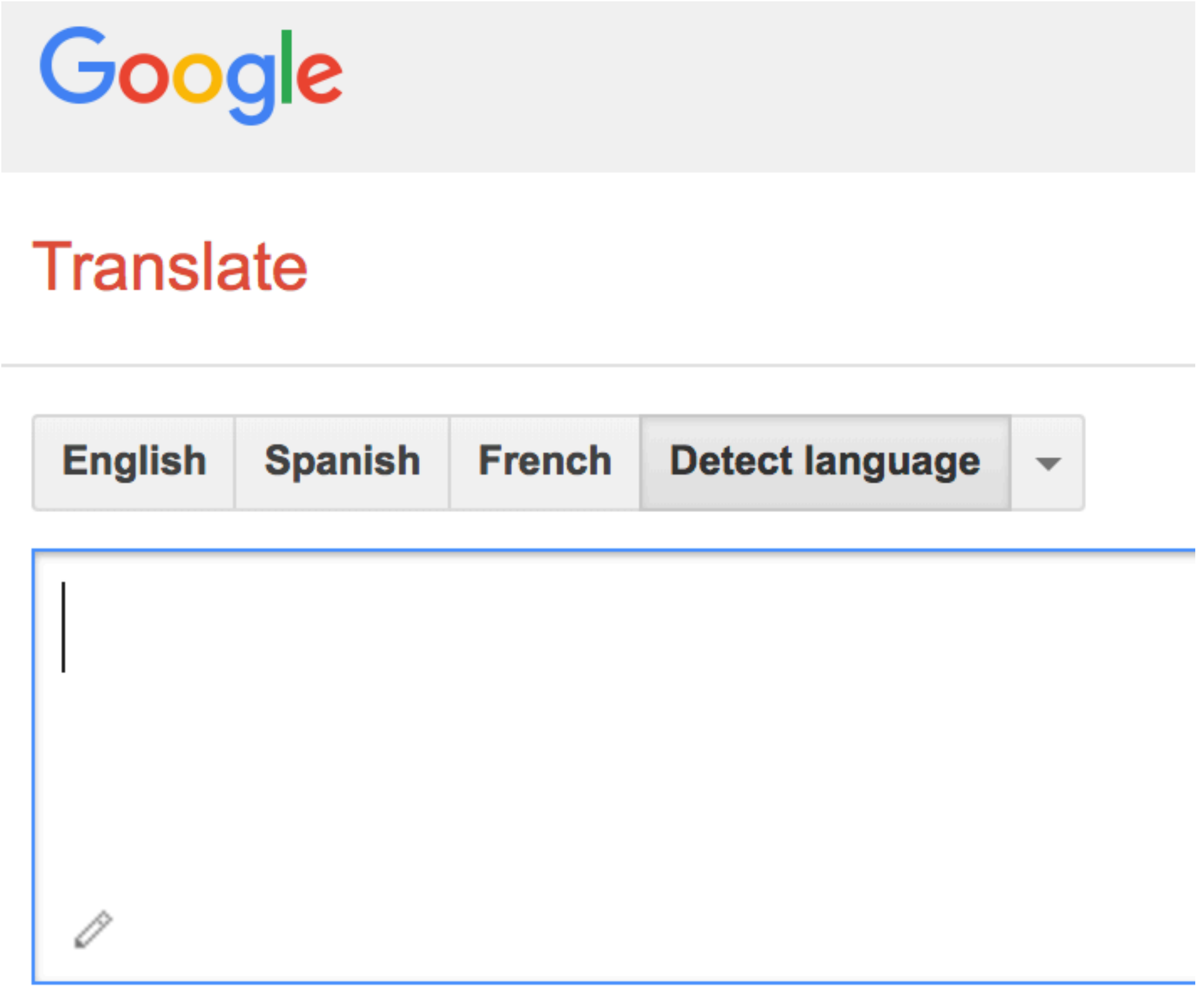
The core challenge of applying deep learning to robotics

Deep learning is data hungry…

ImageNet

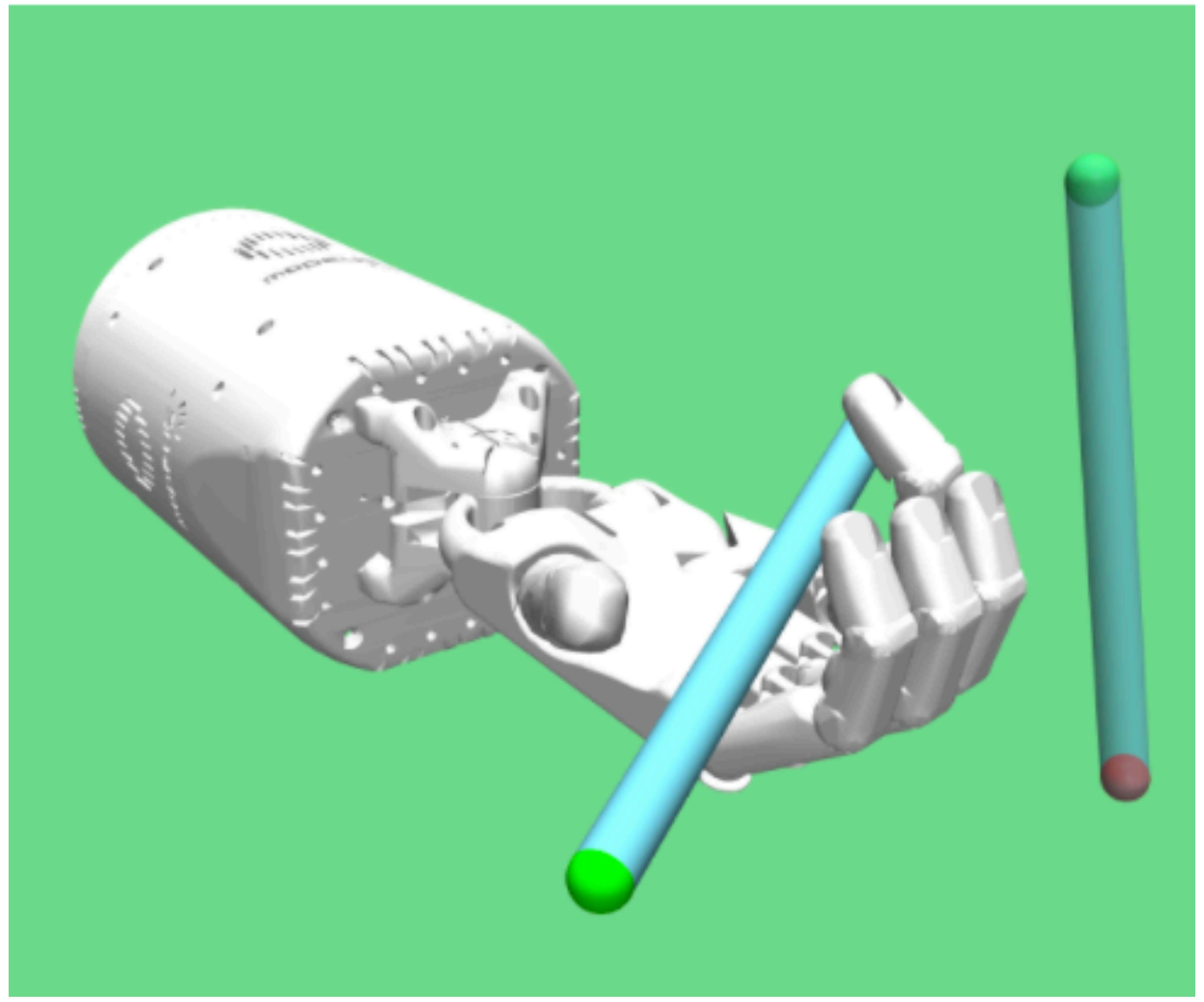
1.2M labeled images

Machine Translation



36M sentence pairs (WMT En->Fr)
"Several orders of magnitude more" (production data)

DeepRL

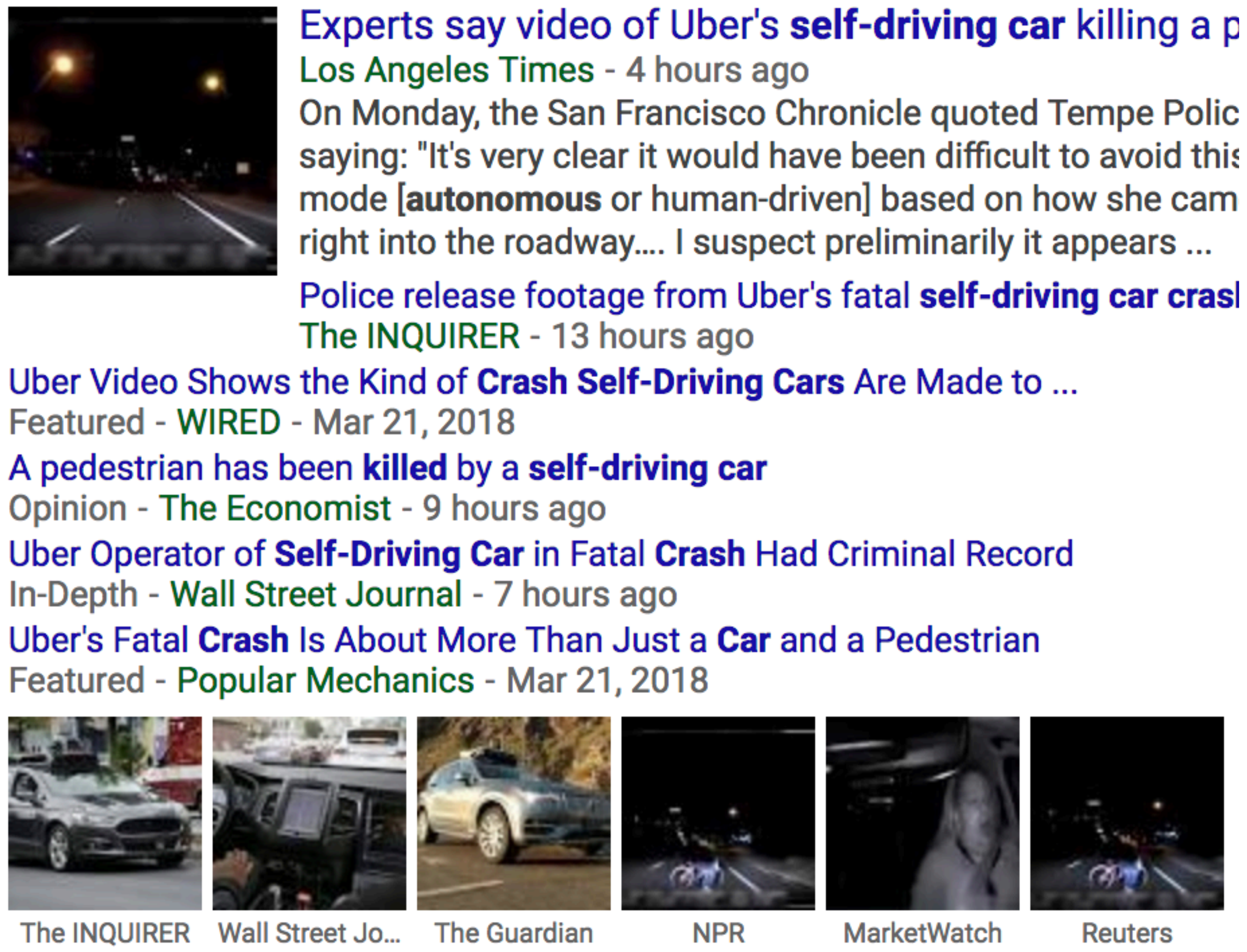


38M timesteps

… But robotic data is expensive

Robot cost

Safety



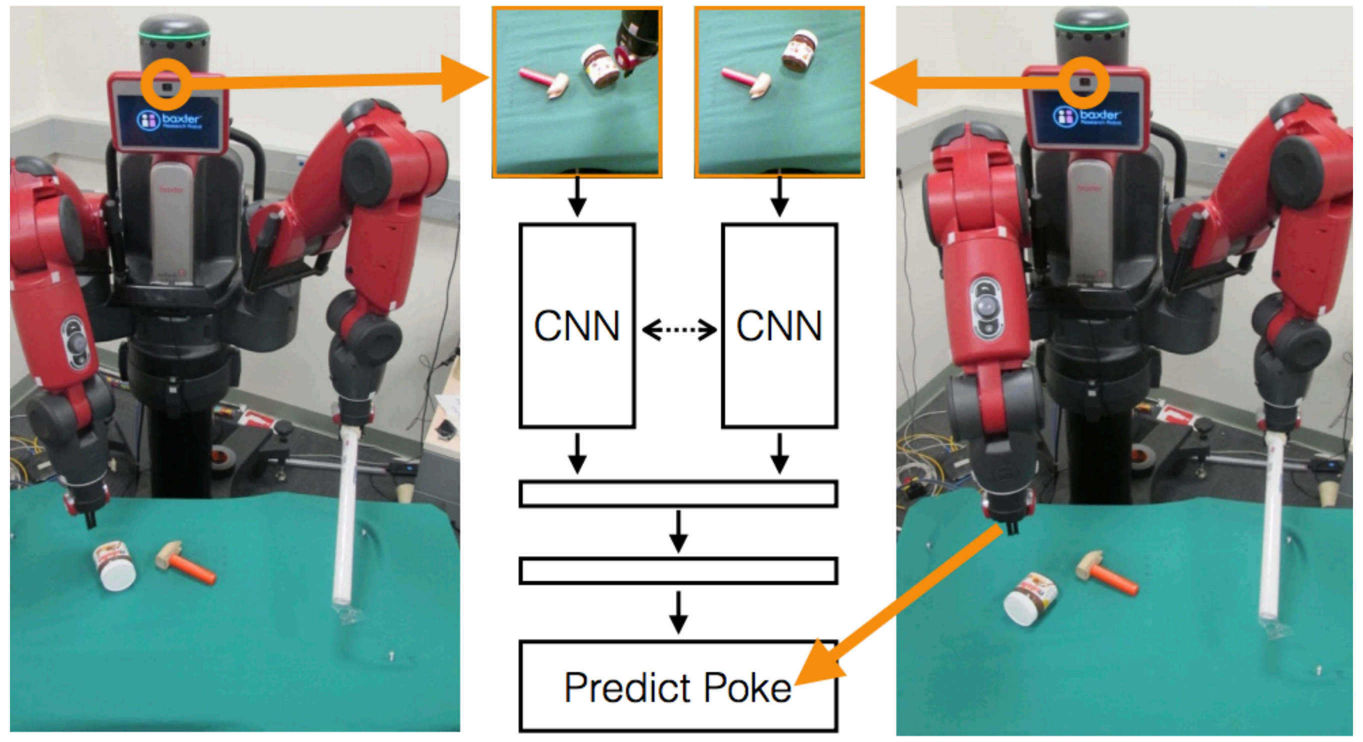
Labeling

How can we get around the data availability problem in robotics?

Large-scale robotic data collection

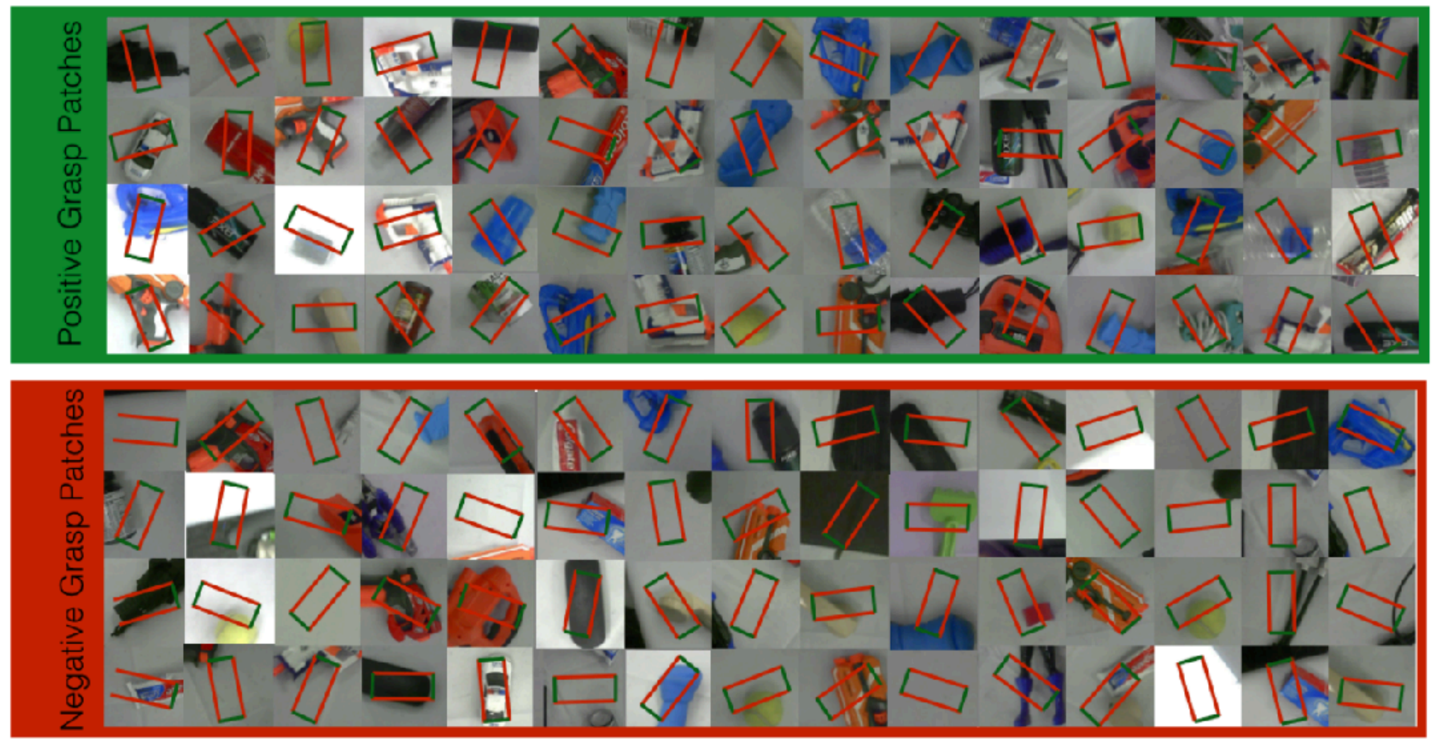
3,000 hours

Learning Hand-Eye Coordination with Deep Learning and Large Scale Data Collection [Levine, Pastor, Krizhevsky, Quillen, 2016]



400 hours

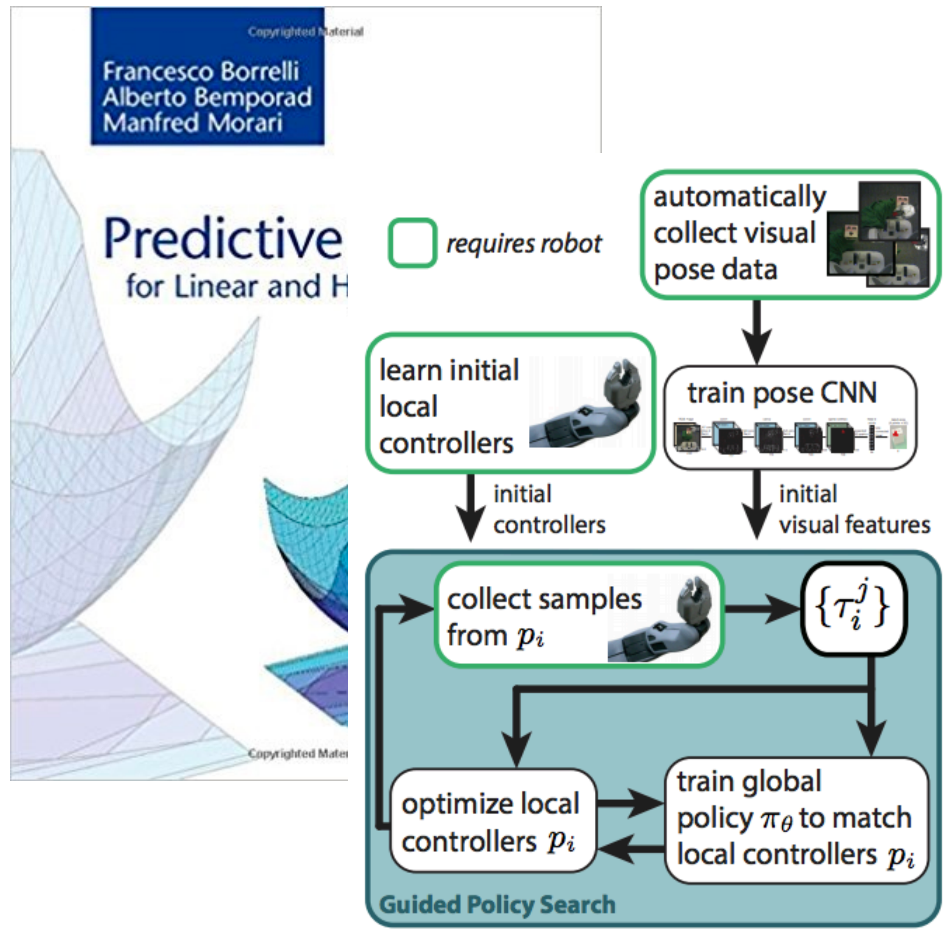
Learning to Poke by Poking: Experiential Learning of Intuitive Physics [Agarwal, Nair, Abbeel, Malik, Levine, 2016]

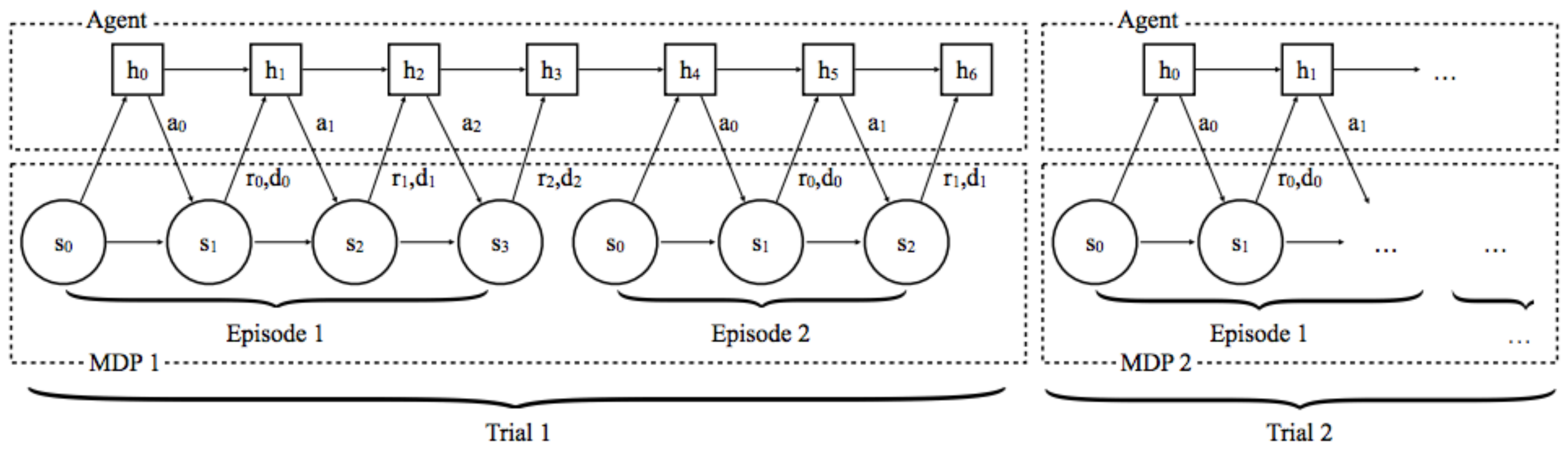


700 hours

Supervising Self-Supervison: Learning to Grasp from 50K Tries [Pinto, Gupta, ICRA 2016]

Efficient reinforcement learning





Model-based

Predictive Control
[Borrelli, Bemporad, Morari, 2017]
End-to-End Training of Deep Visuomotor Policies
[Levine*, Finn*, Darrell, Abbeel, 2016]

Meta-learning

Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks
[Finn, Abbeel, Levine, 2017]
RL2: Fast Reinforcement Learning Via Slow Reinforcement Learning
[Duan, Schulman, Chen, Bartlett, Sutskever, Abbeel, 2016]

Learning from Demonstrations

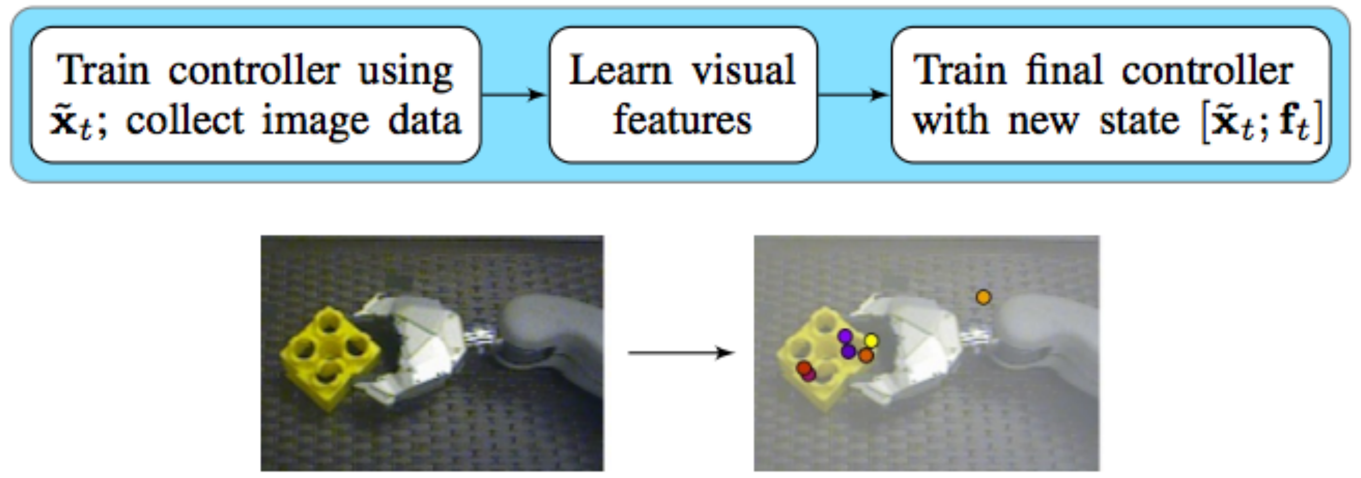
Deep Object-Centric Representations for Generalizable Robot Learning [Devin, Abbeel, Darrell, Levine, 2017]
Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation [Zhang, McCarthy, Jow, Lee, Chen, Goldberg, Abbeel, 2017]
One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning [Yu*, Finn*, Xie, Dasari, Zhang, Abbeel, Levine, 2018]

Unsupervised robotic learning

Augment with self-supervised tasks

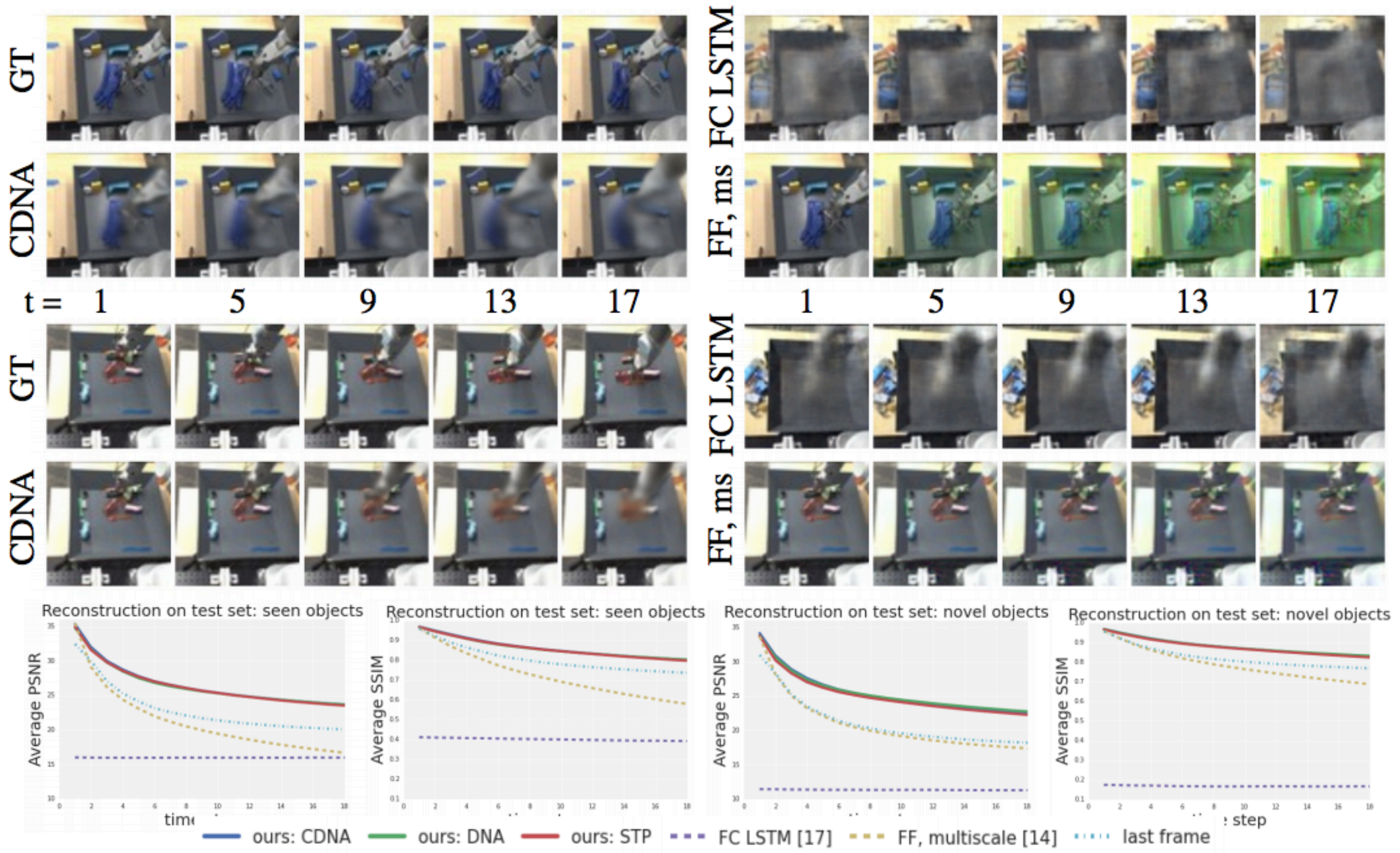
Hindsight Experience Replay [Andrychowicz, Wolski, Ray, Schneider, Fong, Welinder, McGrew, Tobin, Abbeel, Zaremba, 2017]
Loss is its own Reward: Self-Supervision for Reinforcement Learning [Shelhamer, Mahmoudich, Argus, Darrell, 2017]

Learn a feature space



Deep Spatial Autoencoders for Visuomotor Learning [Finn, Tan, Duan, Darrell, Levine, Abbeel 2016]

Learn a model



Unsupervised Learning for Physical Interaction through Video Prediction [Finn, Goodfellow, Levine, 2016]

Is simulated data the answer?

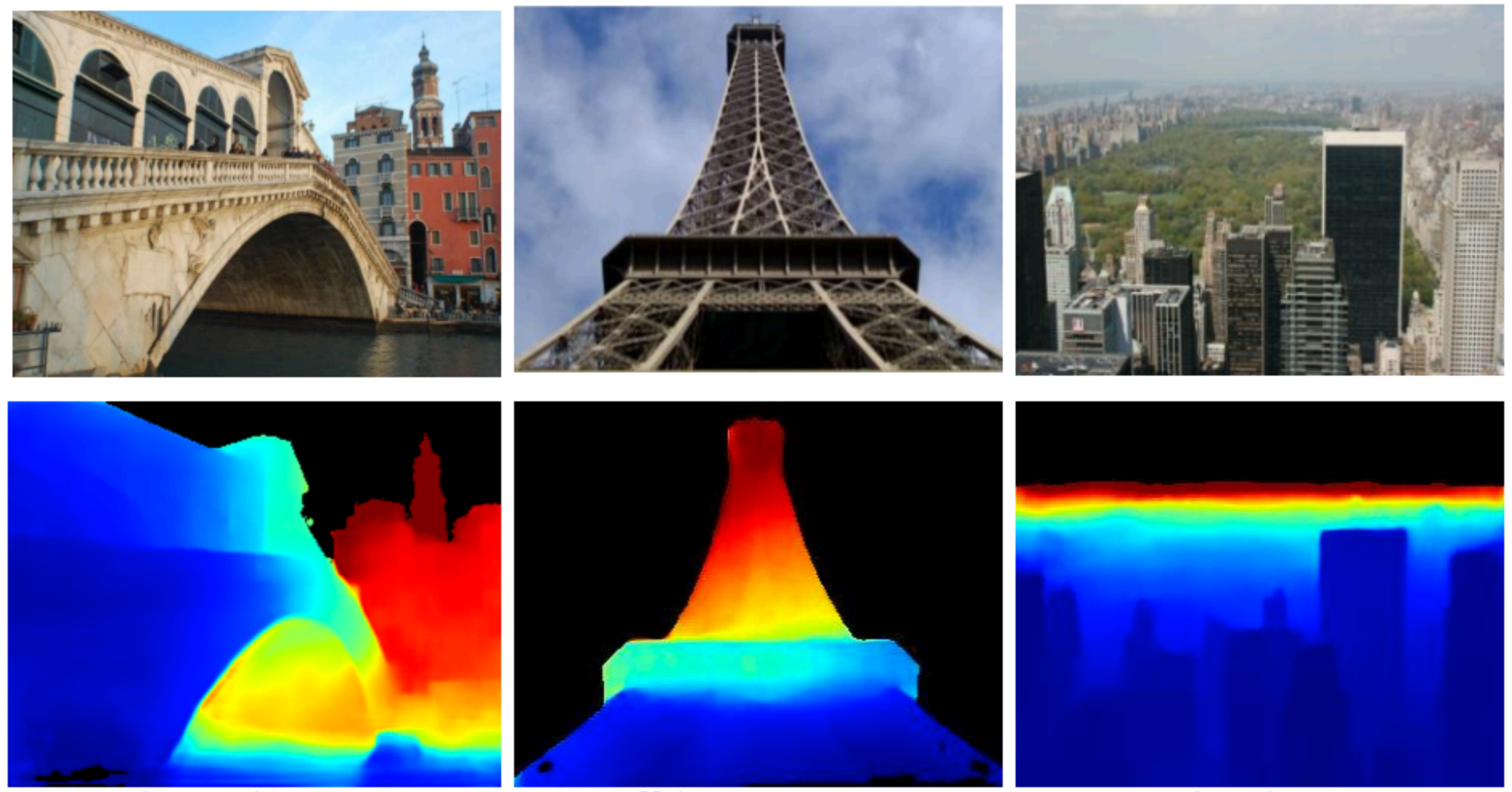
Advantages of simulated data

- Cheap, fast, and scalable
- Safe
- Labeled
- Not beholden to real-world probability distributions

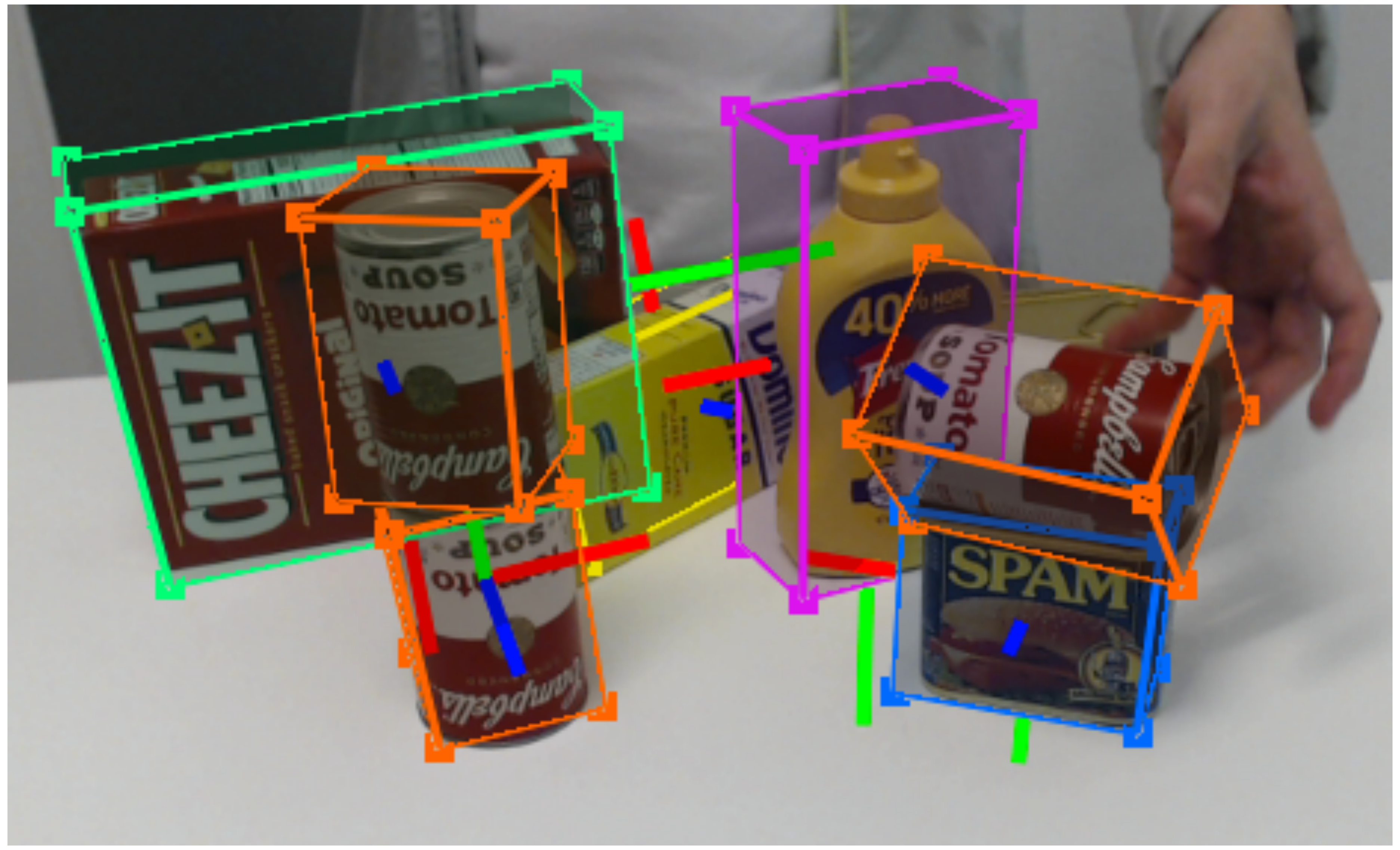
Advantages of simulated data

- Cheap, fast, and scalable
- Safe
- **Labeled**
- Not beholden to real-world probability distributions

Simulators give you rewards & labels for free



Rialto Bridge, Venice   Eiffel Tower, Paris   Central Park, NYC



Advantages of simulated data

- Cheap, fast, and scalable
- Safe
- Labeled
- **Not beholden to real-world probability distributions**

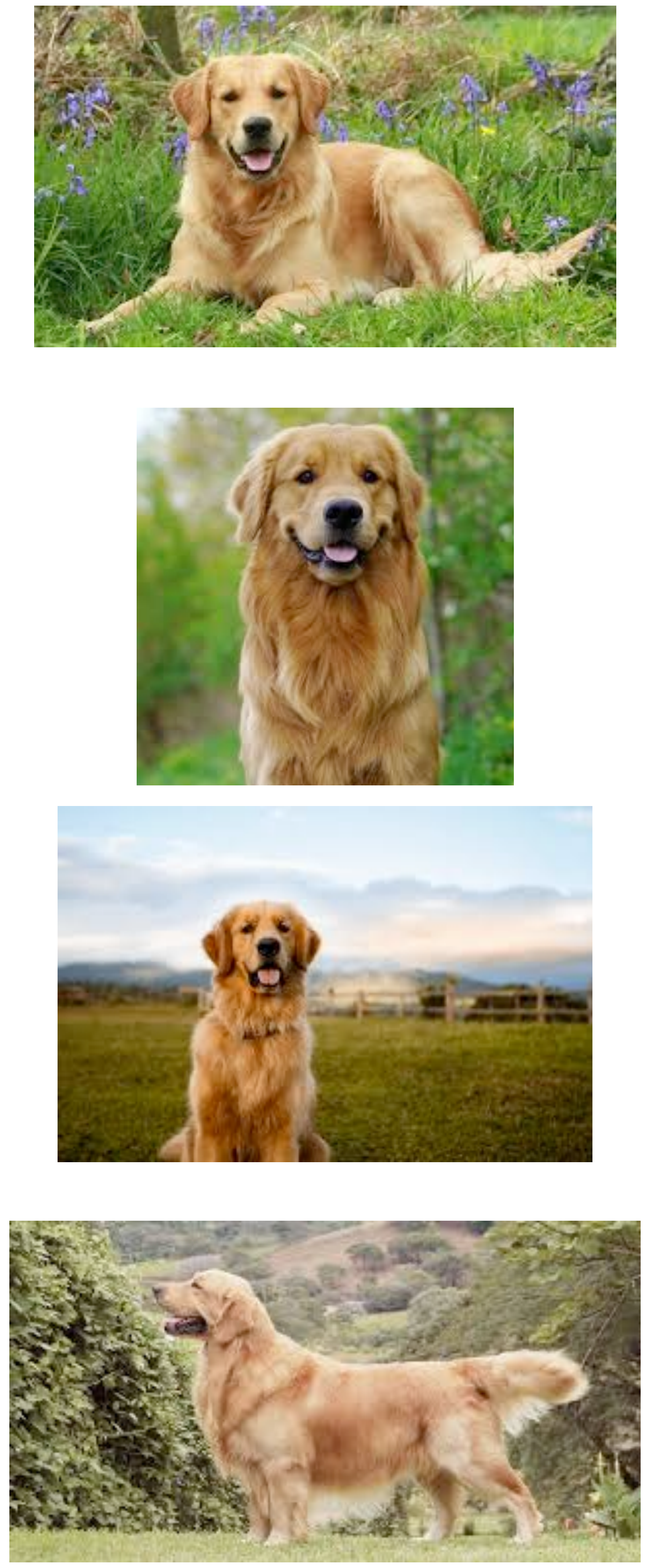
The edge case problem

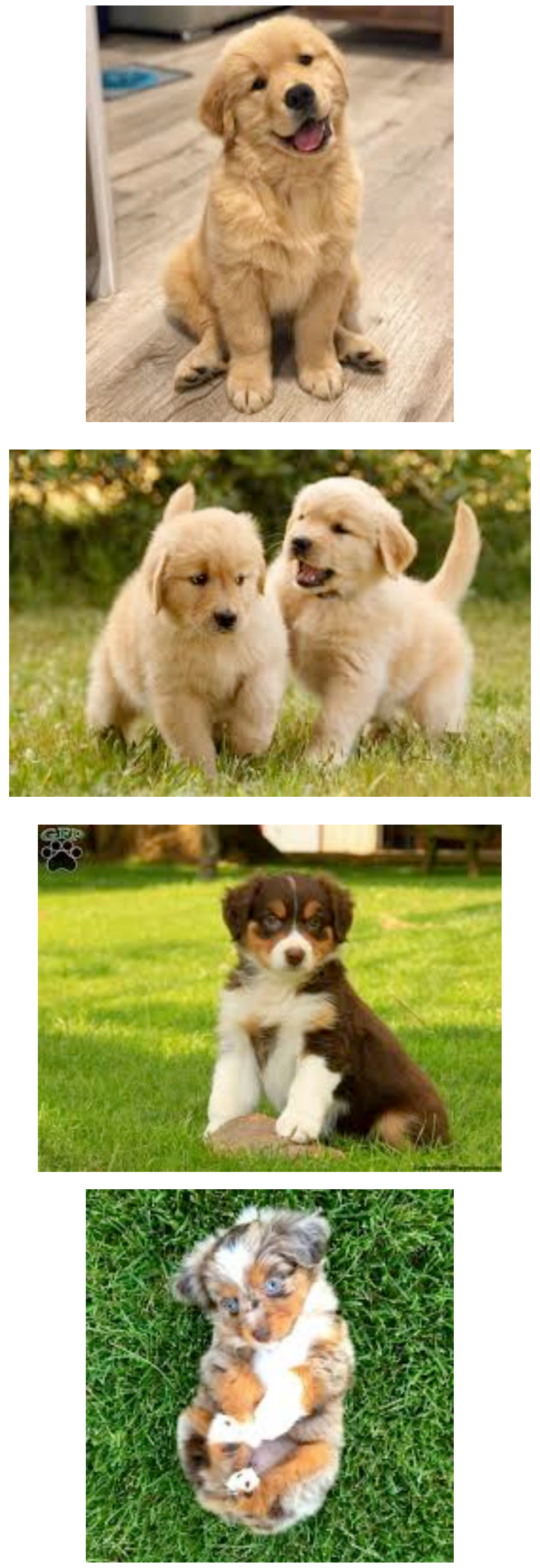
By definition, we have few if any training examples with rare events.
How can we make our systems robust to them?

Simulation for reducing bias?

Training data

Dogs	Puppies





- The model may classify all Australian shepherds as puppies
- Can we fix this by synthesizing adult Australian shepherds?

But does simulated data work?

"There is a real danger (in fact, a near certainty) that programs which work well on simulated robots will completely fail on real robots because of the differences in real world sensing and actuation - it is very hard to simulate the actual dynamics of the real world."

Artificial Life and Real Robots
[Rodney Brooks, 1992]

Outline of the rest of the talk

- Why is it so hard to use simulated training data?
- How can you use simulation without solving sim2real?
- Building a good simulation
- Domain adaptation
- Domain randomization
- What's next?

Outline of the rest of the talk

- **Why is it so hard to use simulated training data?**
- How can you use simulation without solving sim2real?
- Building a good simulation
- Domain adaptation
- Domain randomization
- What’s next?

Why is it so hard to use simulated training data?

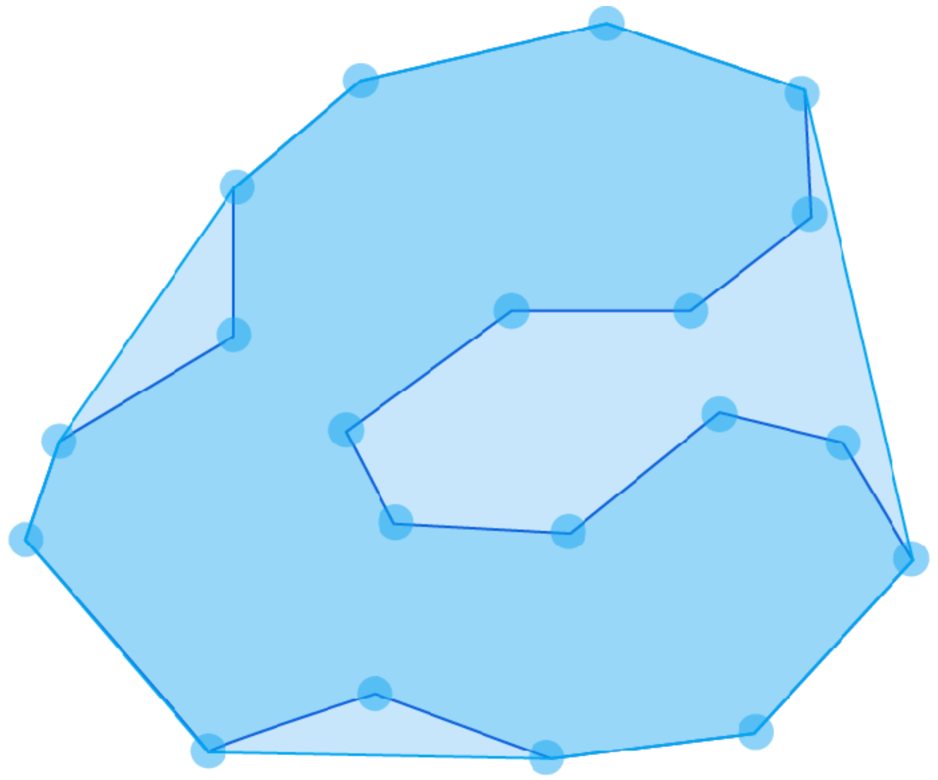
- It's hard to accurately & efficiently model sensors & physical systems
- Small modeling errors lead to large control errors

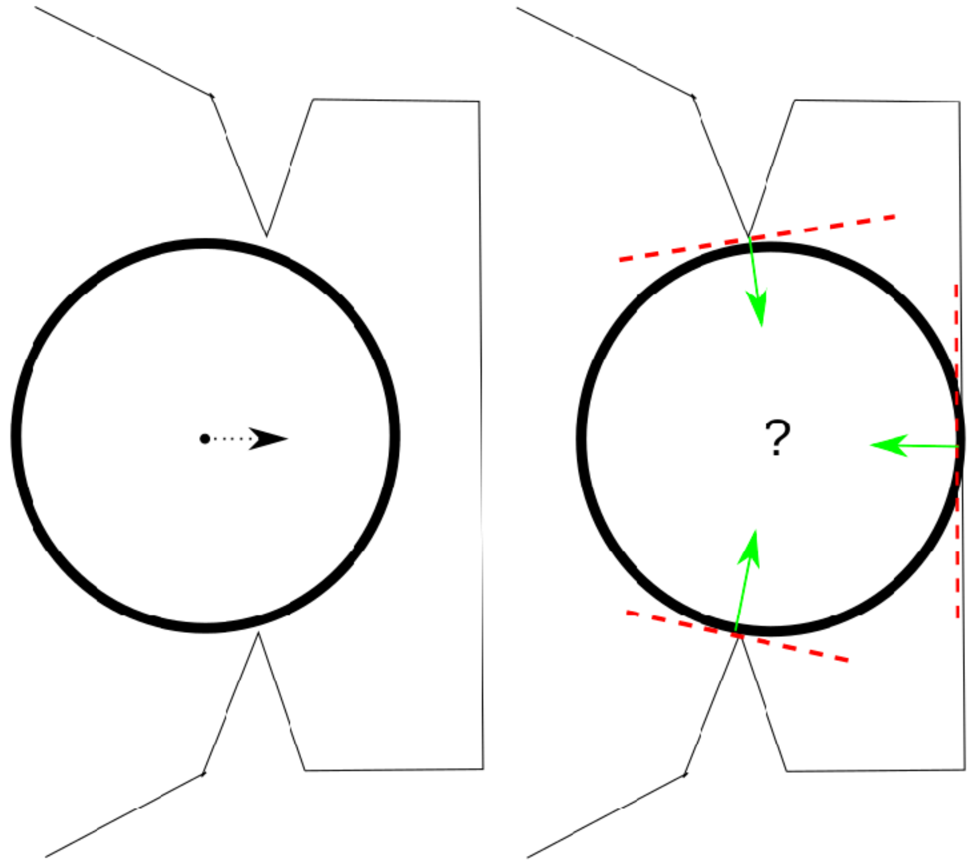
Why is it so hard to use simulated training data?

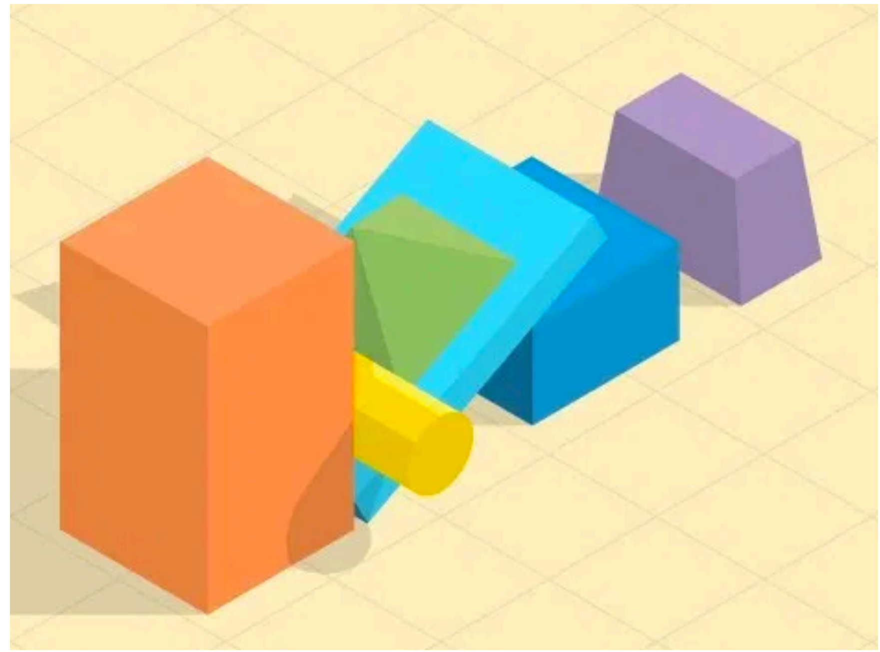
- **It's hard to accurately & efficiently model sensors & physical systems**
- Small modeling errors lead to large control errors

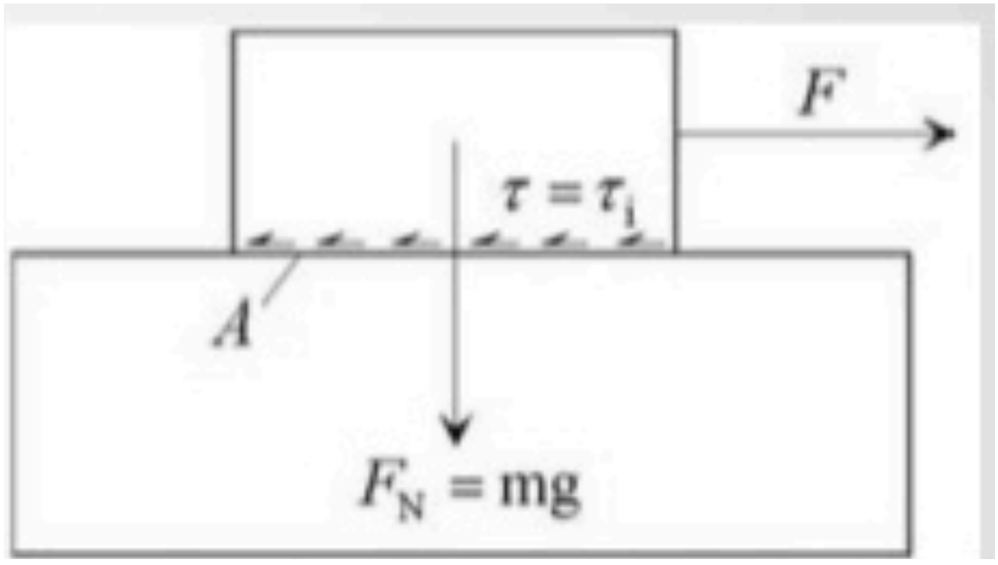
Physics simulators make big assumptions to run faster

Convex objects | Discrete time | Rigid bodies | Coulomb friction | Etc









Accurate models require getting your parameters right

- How to measure physical parameters like damping, inertia, friction that are not directly observable?

- The more accurate your model, the more of these to measure —> more data required

- (More on this later)

Photorealistic sensor simulation is expensive…

Jungle Book:
30M render hours
19 hours per frame
800 artist-years of effort

Jungle Book, 2016

Toward Understanding Stories From Videos
[Sanja Fidler, NIPS Deep Learning Workshop 2016]

… and not a solved problem

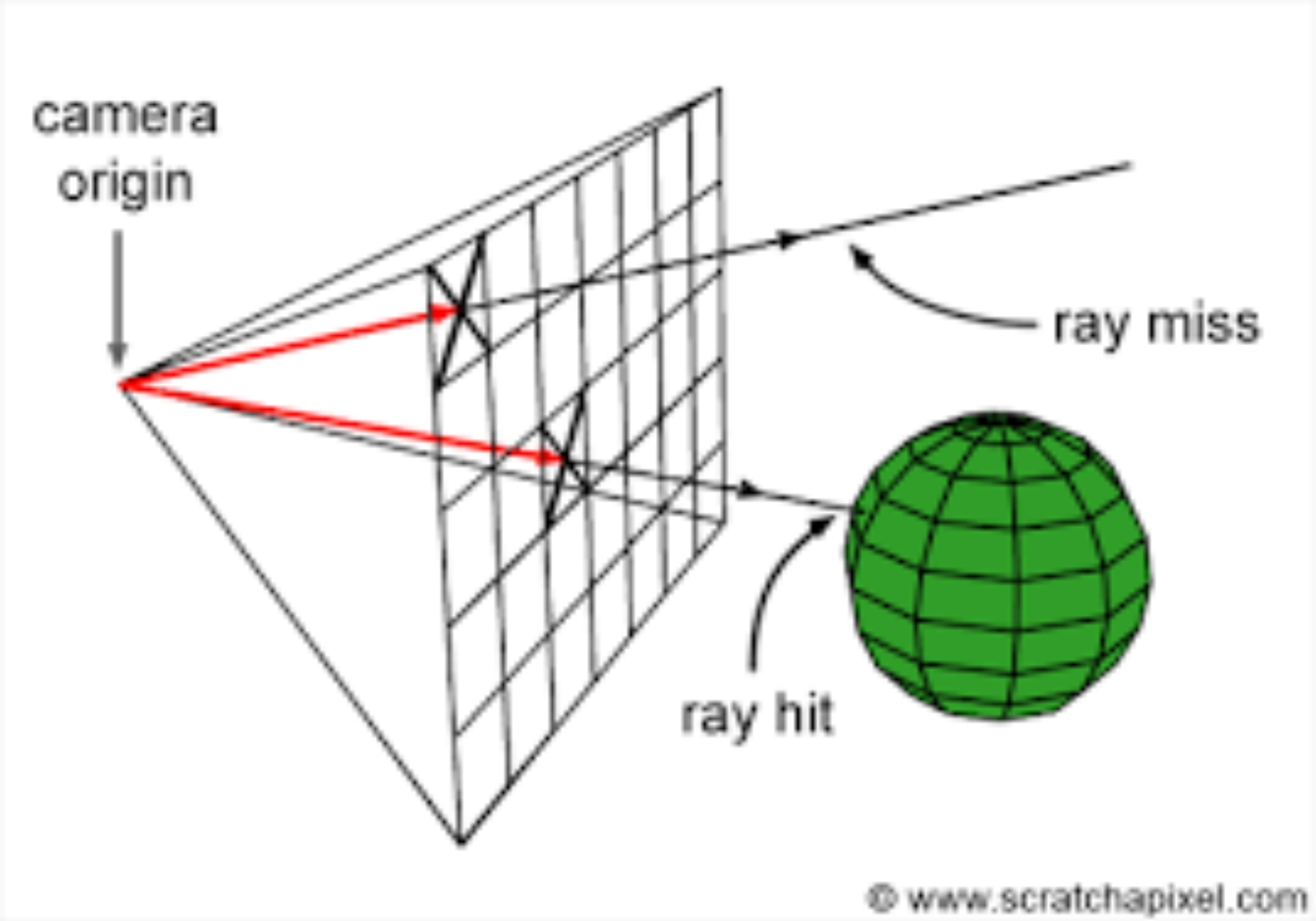
$$d = \frac{(Et \times c)}{2}$$

Where c = speed of light

LIDAR: fire millions of lasers per second and measure how long it takes for the beams to come back

An Introduction to LIDAR, Oliver Cameron, https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff

… and not a solved problem



A simple approach to simulating LIDAR: ray-tracing

An Overview of the Ray-Tracing Rendering Technique, https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-overview

… and not a solved problem

- Effect of weather? E.g., scattering of reflected signal, false-positive detection of rain droplets
- Effect of material properties?

Gusmão, Guilherme Ferreira, Carlos Roberto Hall Barbosa, and Alberto Barbosa Raposo. "A LiDAR system simulator based on raytracing, modeled with metrological parameters and environmental noise." Anais do XXI Symposium on Virtual and Augmented Reality. SBC, 2019.

Why is it so hard to use simulated training data?

- It’s hard to accurately & efficiently model sensors & physical systems
- **Small modeling errors lead to large control errors**

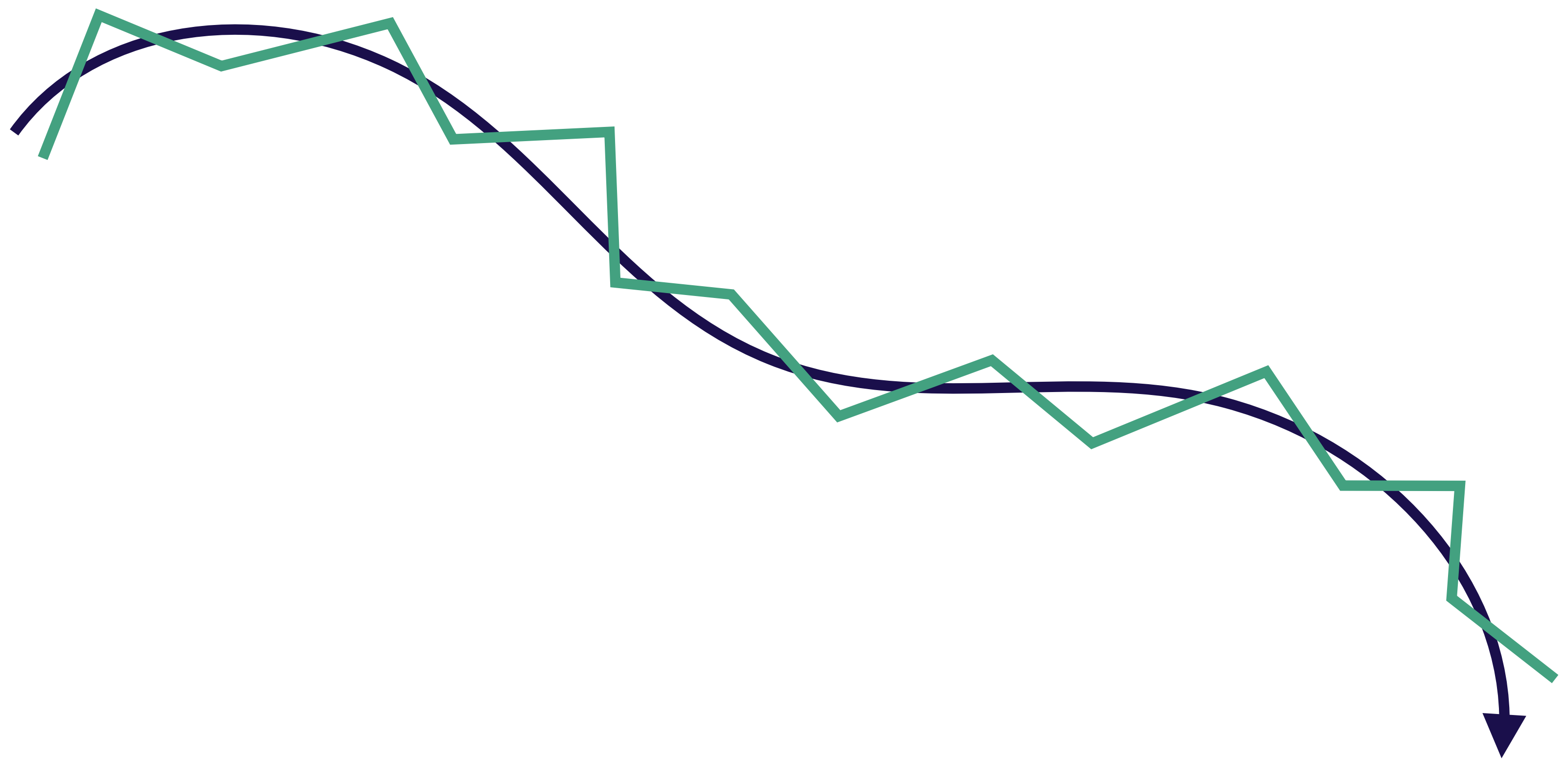
Neural nets overfit to tiny differences in data distribution

Virtual KITTI Dataset
Multi-object tracking accuracy:
Sim: 63.7%
Real: 78.1%

Virtual Worlds as Proxy for Multi-Object Tracking Analysis
[Gaidon*, Wang*, Cabon, Vig, 2016]

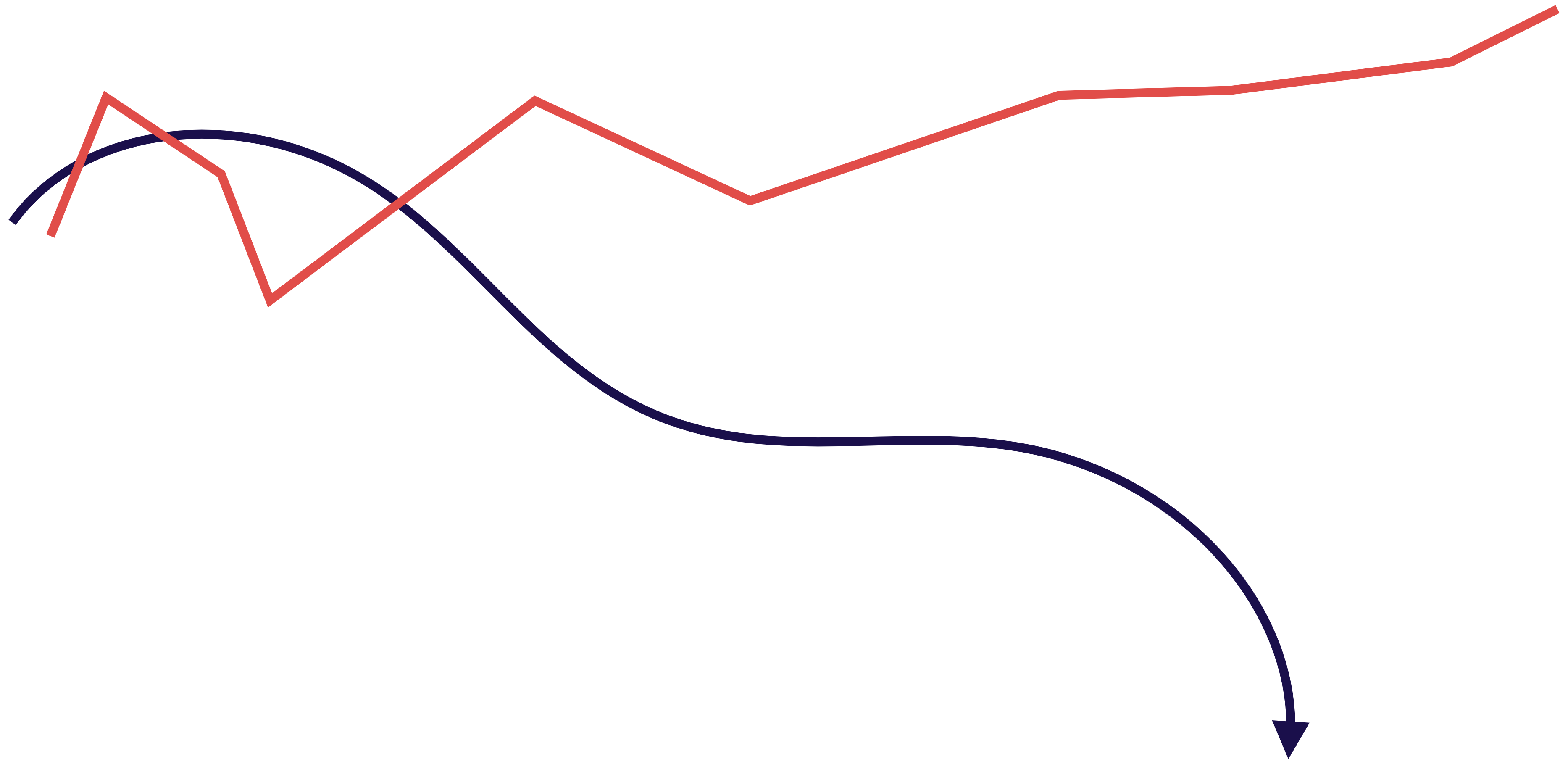
Errors compound

What we hope happens



Uncorrelated errors

What actually happens



Compounding errors

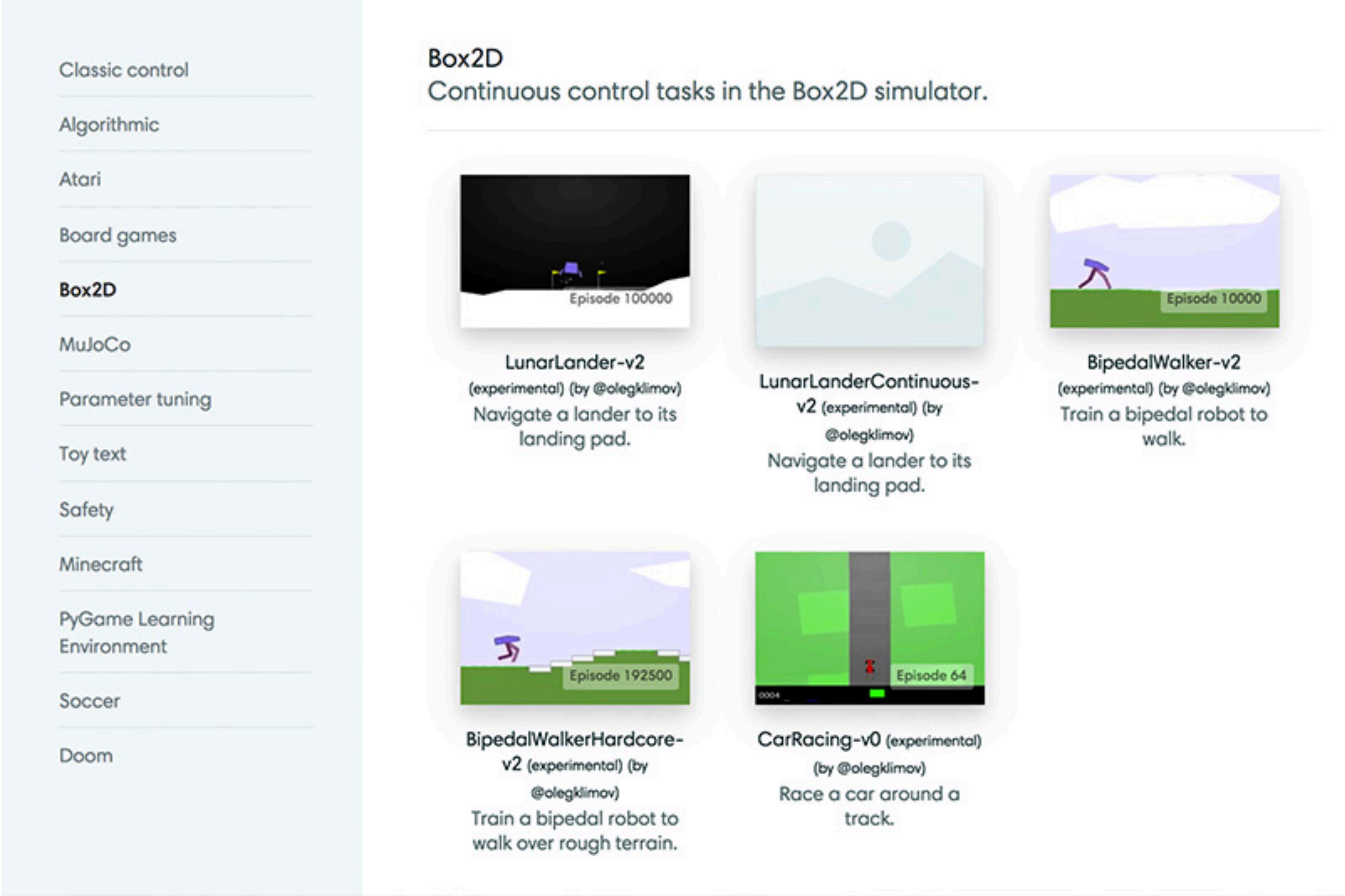
Outline of the rest of the talk

- Why is it so hard to use simulated training data?
- **How can you use simulation without solving sim2real?**
- Building a good simulation
- Domain adaptation
- Domain randomization
- What’s next?

Non sim2real uses of simulation

- Prototyping algorithms
- Debugging your implementation
- Prototyping systems
- Testing

Prototyping algorithms

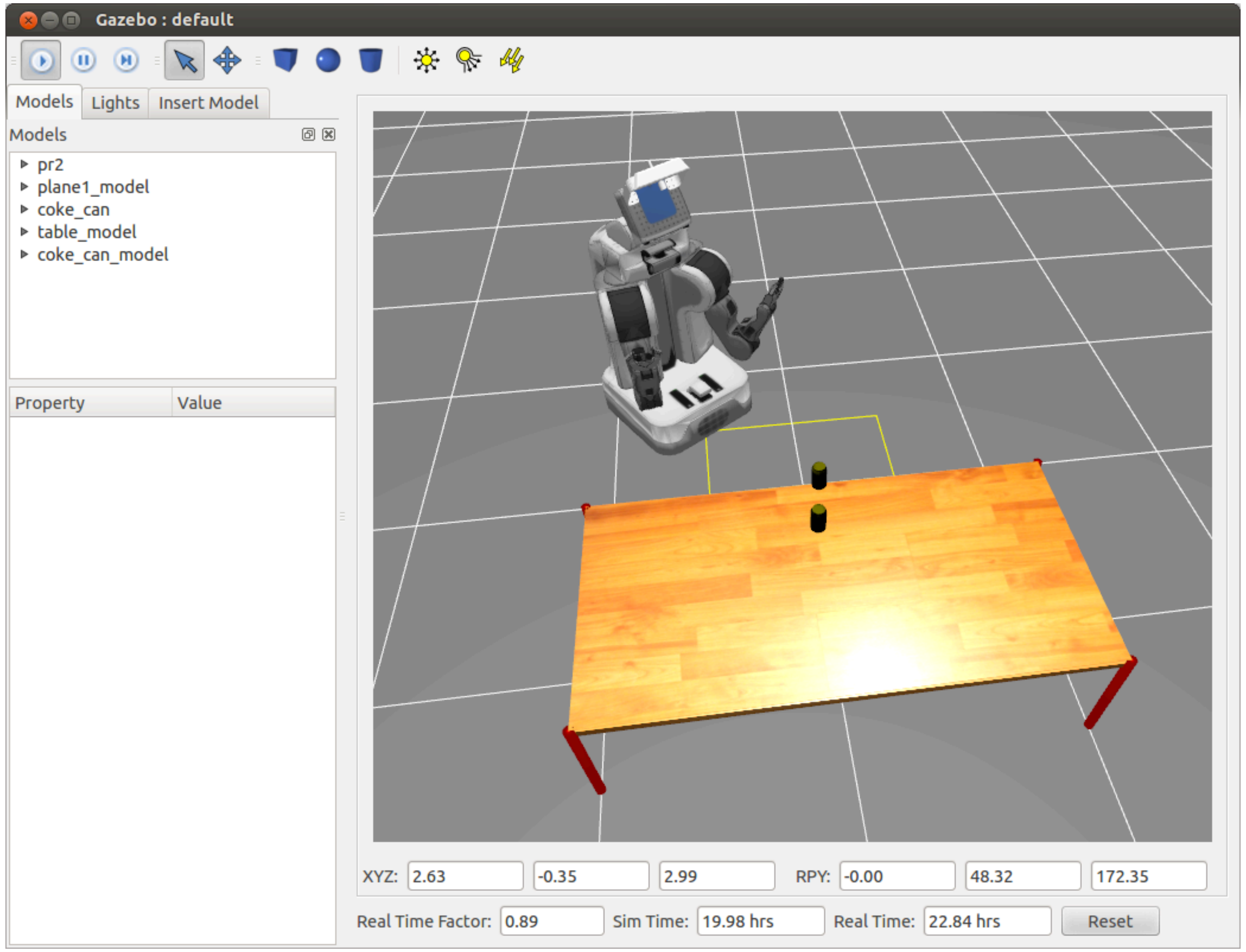


OpenAI Gym for reinforcement learning prototyping

Debugging your software

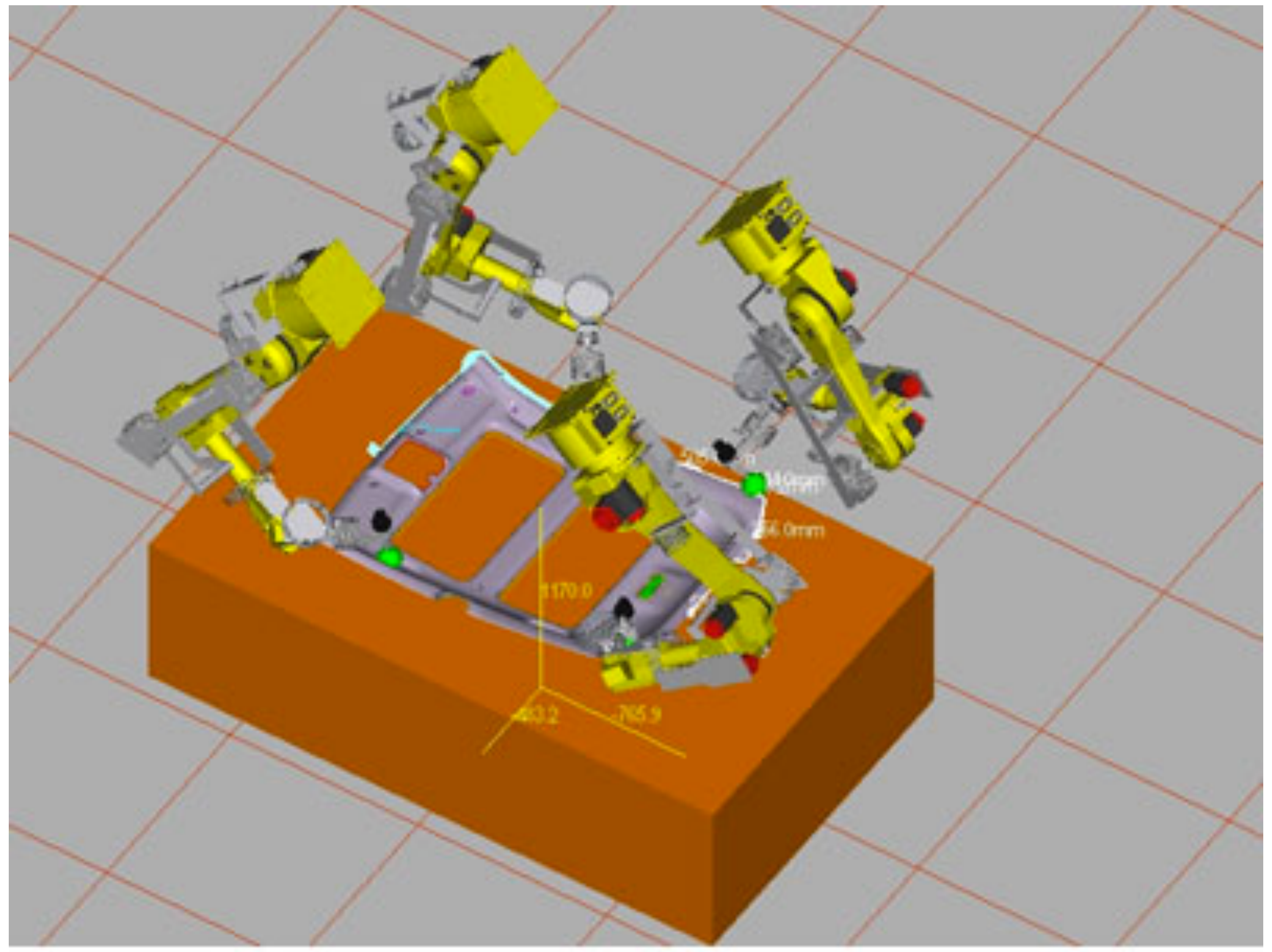
- Typically Gazebo / ROS

- Implement your entire stack and debug with the same software that will run on the robot, including realistic latency

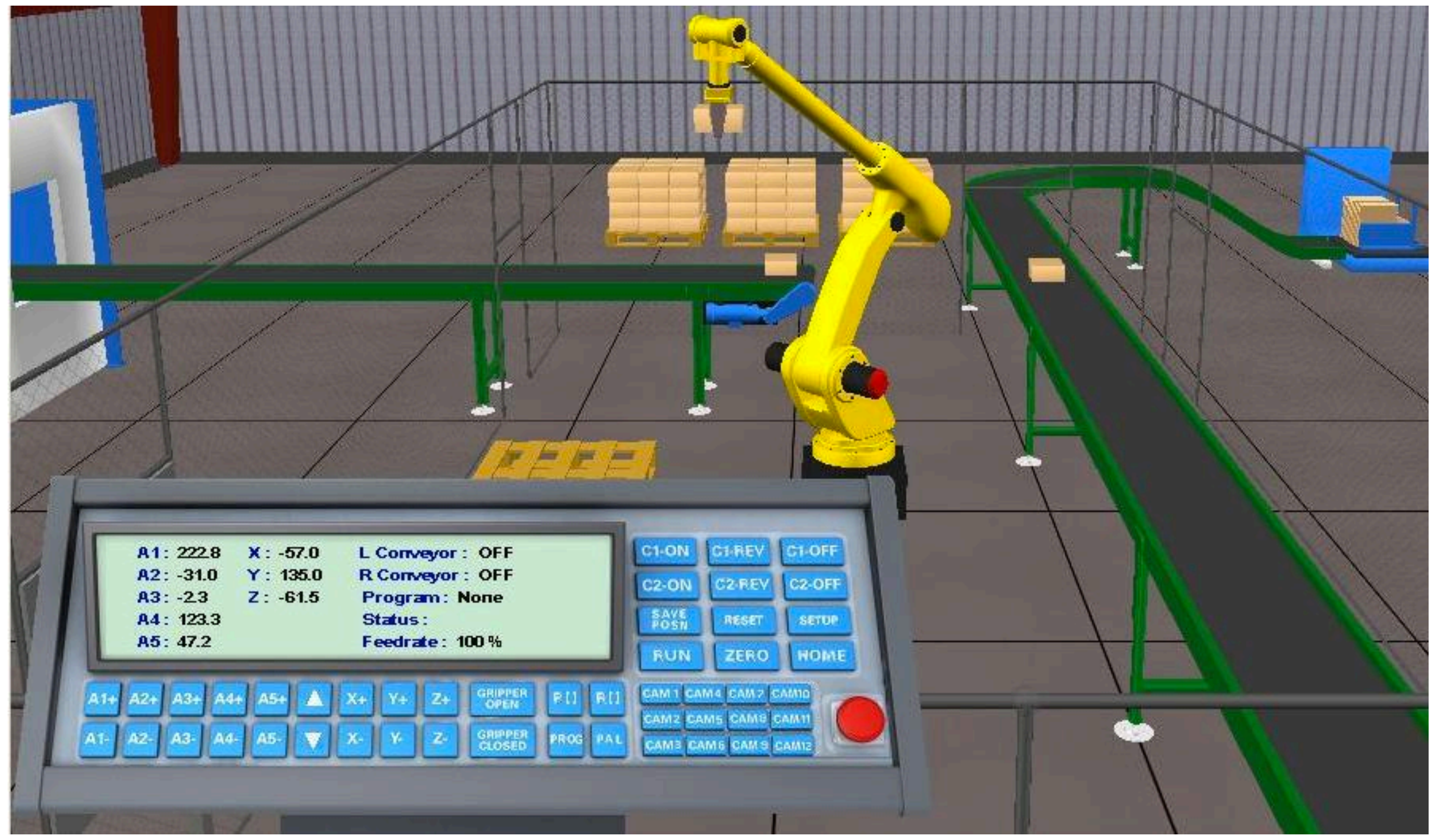


Prototyping systems

- Figure out what robot to use
- Verify the robot’s ability to solve the task
- Design the entire cell to see how it will fit into the broader workflow
- Estimate speed & make ROI calculations



Simulation of robotic headliner trimming, courtesy KMT Robotic Solutions Inc.

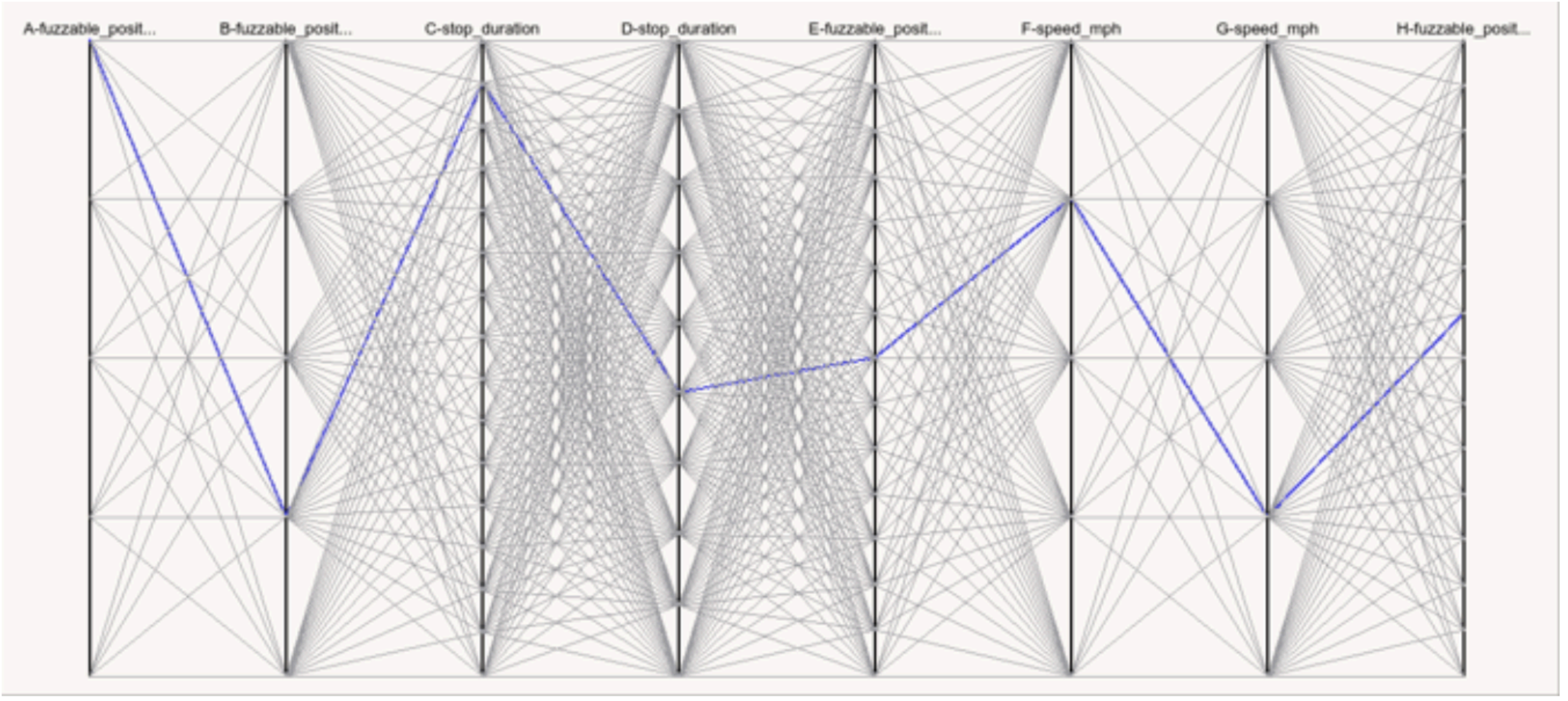


RoboLogix

Reliability testing / continuous integration

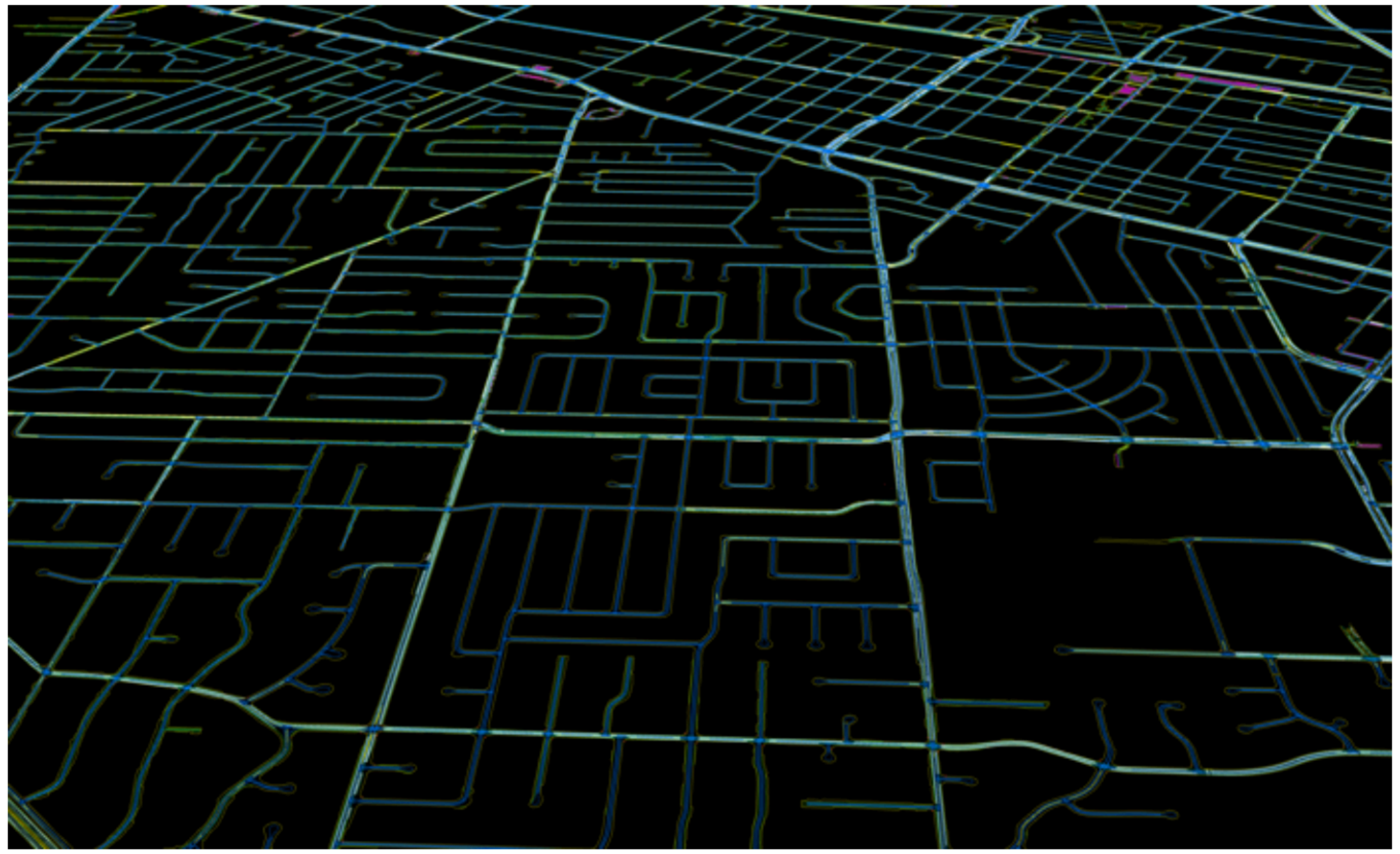
- Most straightforward way: run tests against your log data
- However, log data is:
  - Incomplete
  - Noisy
  - Partially observed
  - Static

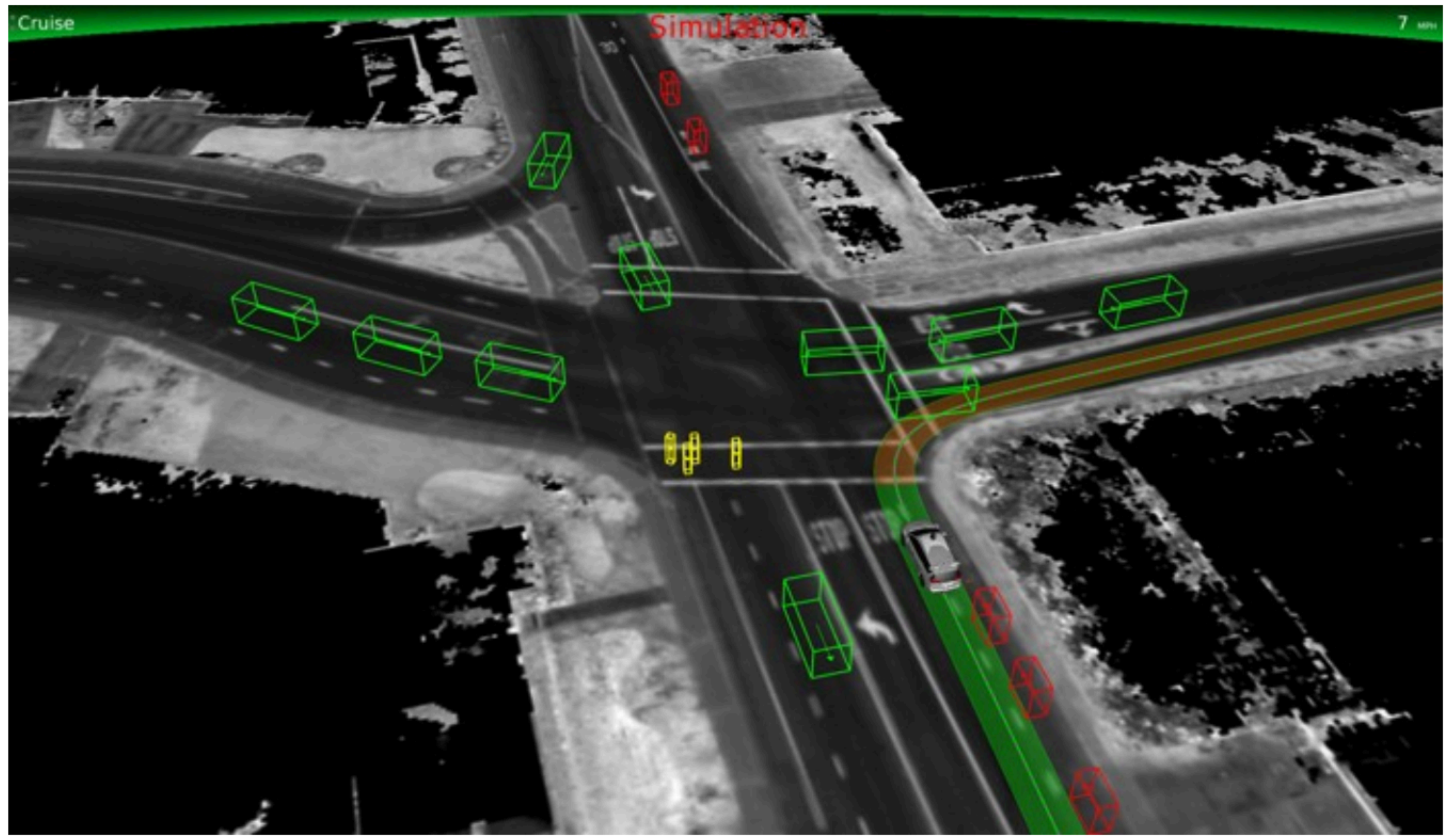
Simulation for testing autonomous vehicles



A Carcraft "fuzzing" chart (Waymo)

1M miles on the road

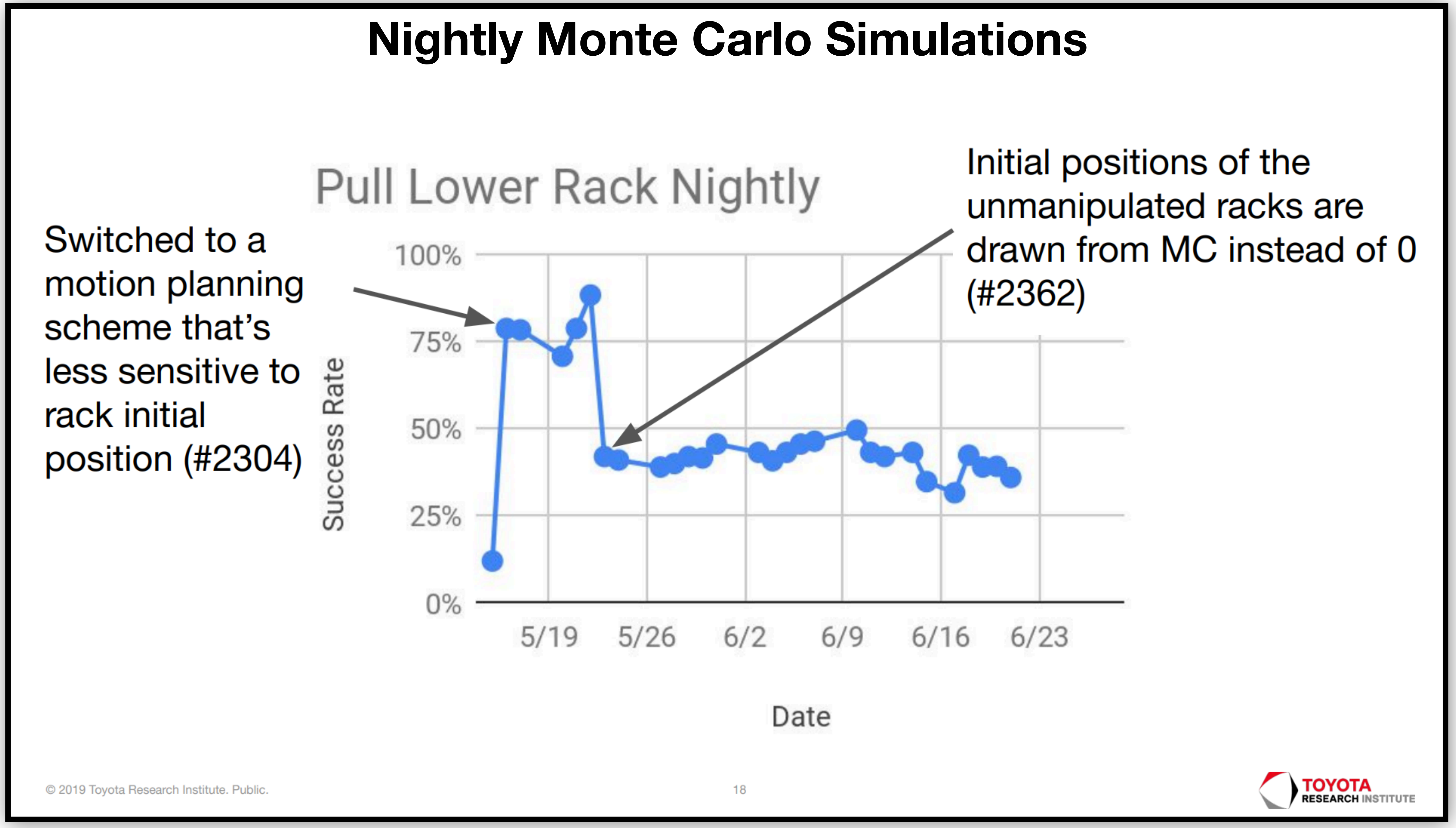




1B miles in simulation

https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/

“Simulation-First” robotic development



- Keys to success:
  - Make sim is harder than real
  - Rigorous about sources of randomness
  - Diagnose errors to find bugs
  - Good contact simulation

Sim2Real for the corner cases? Getting to robust manipulation.
Talk given at RSS Sim2real workshop, 2019
[Russ Tedrake, MIT and TRI]

Outline of the rest of the talk

- Why is it so hard to use simulated training data?
- How can you use simulation without solving sim2real?
- **Building a good simulation**
- Domain adaptation
- Domain randomization
- What's next?

The simulator design process

Design simulation model

Create scenarios

Collect data & improve simulation

The simulator design process

Design simulation model

Create scenarios

Collect data & improve simulation

Simulator design

- Pieter’s lecture from earlier in CS287
- In practice, most people pick Bullet / PyBullet or MuJoCo and use the models provided by the developer of their robot
- Also worth looking at:
  - Drake (https://drake.mit.edu/)
  - Gazebo (http://gazebosim.org/)

The simulator design process

Design simulation model

Create scenarios

Collect data & improve simulation

World design

- Where to get 3D models?
  - ShapeNet (https://www.shapenet.org/) - high volume [60K objects], low quality
  - YCB (http://www.ycbbenchmarks.com/), BigBird (http://rll.berkeley.edu/bigbird/) - high quality, low volume [10s of objects]
  - Dex-Net (https://berkeley.app.box.com/s/6mnb2bzi5zfa7qpwyn7uq5atb7vbztng). Combination of datasets. [10K objects]. Good compromise?
  - 3D model repositories (3D warehouse, Unity asset store): Millions of objects, but not free.
  - Procedural object generation

See more here: https://sim2realai.github.io/Synthetic-Datasets-of-Objects-Part-I/

World design

- How to place 3D models into the world?
  - Randomly
  - Randomly, according to physics
  - Procedurally
- Procedural content generation: http://pcgbook.com/ (mostly game design focused)

The simulator design process

Design simulation model

Create scenarios

Collect data & improve simulation

System ID

$$\min_{\eta} \sum_{a} D\left(\tau_{\eta}(a), \tau_{r}(a)\right)$$

η : simulation parameters

a : sequence of actions

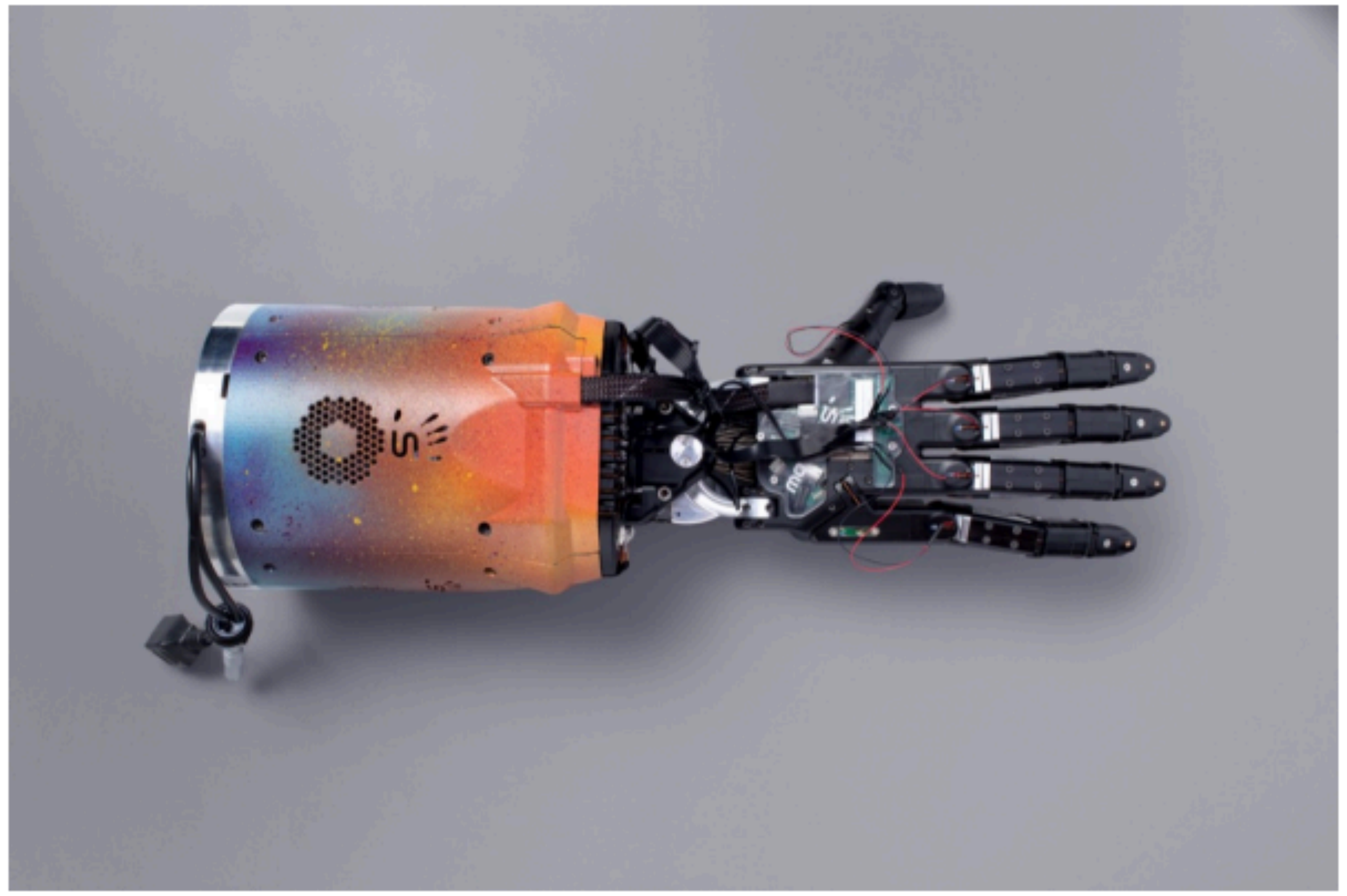
D : distance function

$\tau_{\eta}(a)$: trajectory from following a in this sim

$\tau_{r}(a)$: trajectory from following a in the real world

System ID: a case study

$$\min_{\eta} \sum_{a} D\left(\tau_\eta(a), \tau_r(a)\right)$$



- Trajectories a: extend each joint of each finger to limits separately, then oscillate each finger. ~3-4 mins of data
- Distance function D: L2 distance on the state after 1 second of actions
- Optimization algorithm: iterative coordinate descent
- Exclude changes to η that improve performance <0.1%

Outline of the rest of the talk

- Why is it so hard to use simulated training data?
- How can you use simulation without solving sim2real?
- Building a good simulation
- **Domain adaptation**
- Domain randomization
- What’s next?

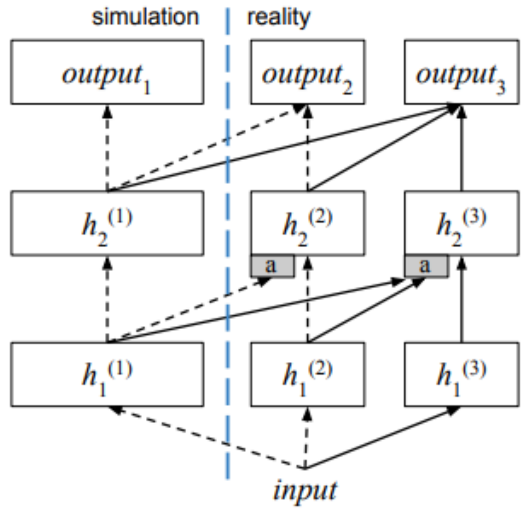
Domain Adaptation (in sim2real)

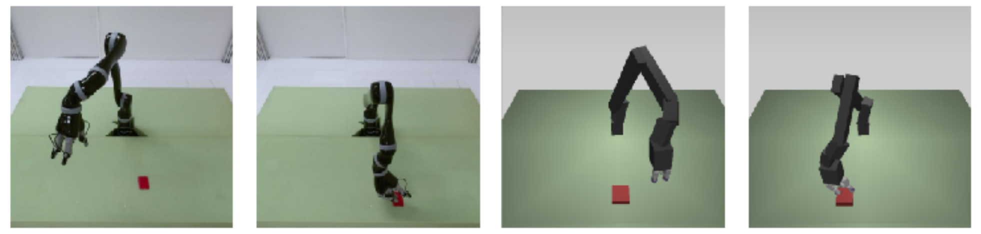
Supervised

Unsupervised / weakly supervised

Supervised Domain Adaptation

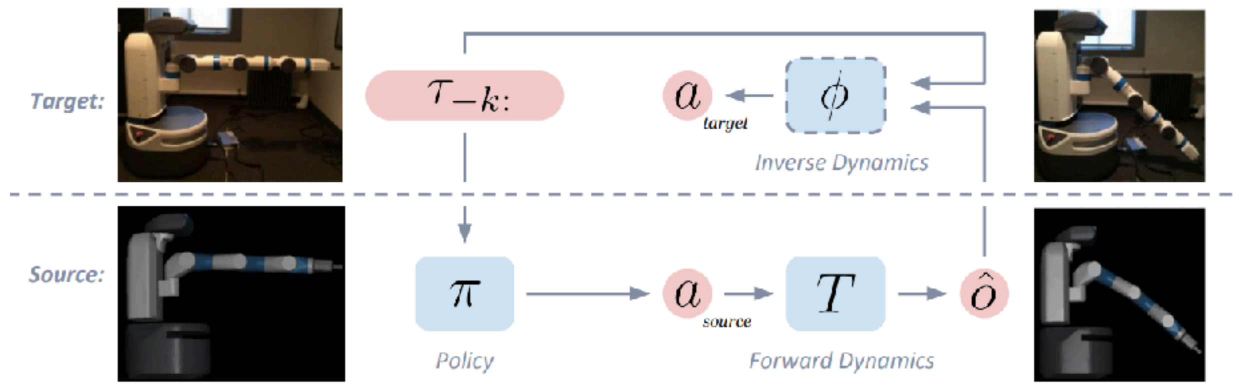
Progressive nets

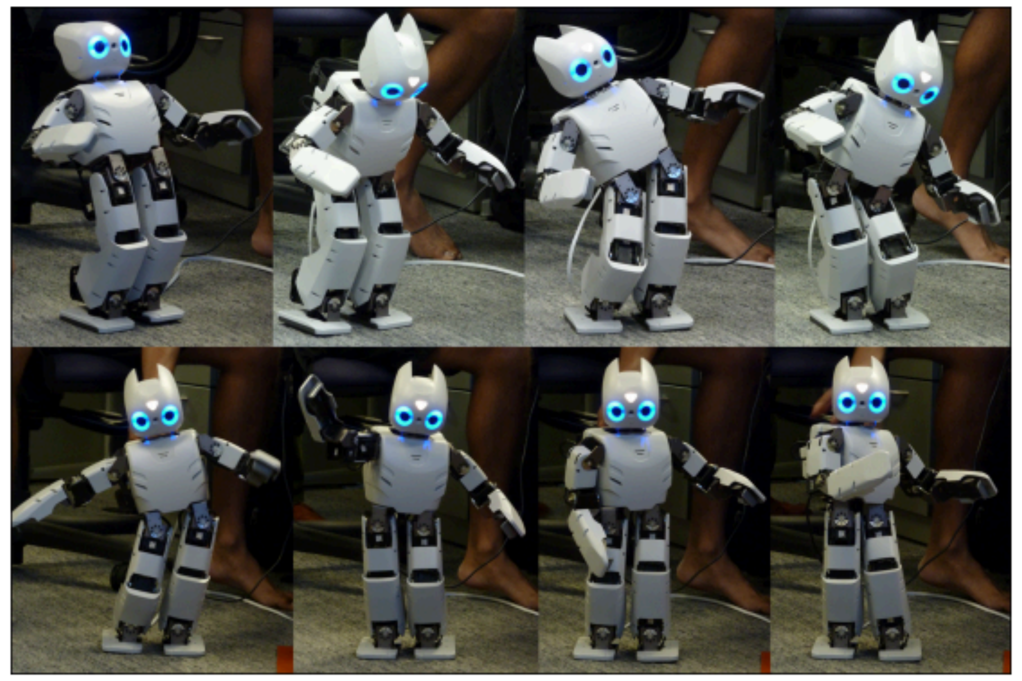




Sim-to-Real Robot Learning from Pixels with Progressive Nets
[Andrei Rusu et al., 2018]

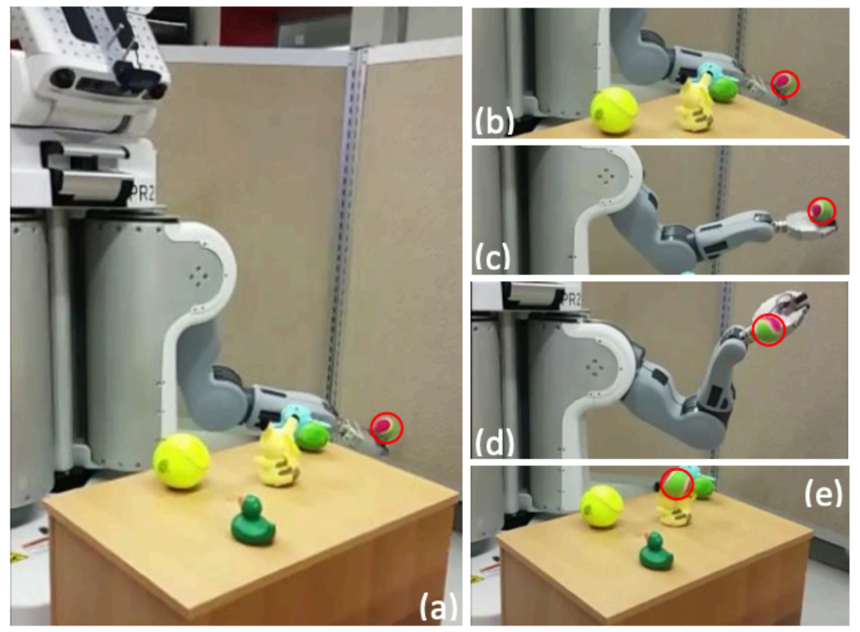
Learn inverse dynamics





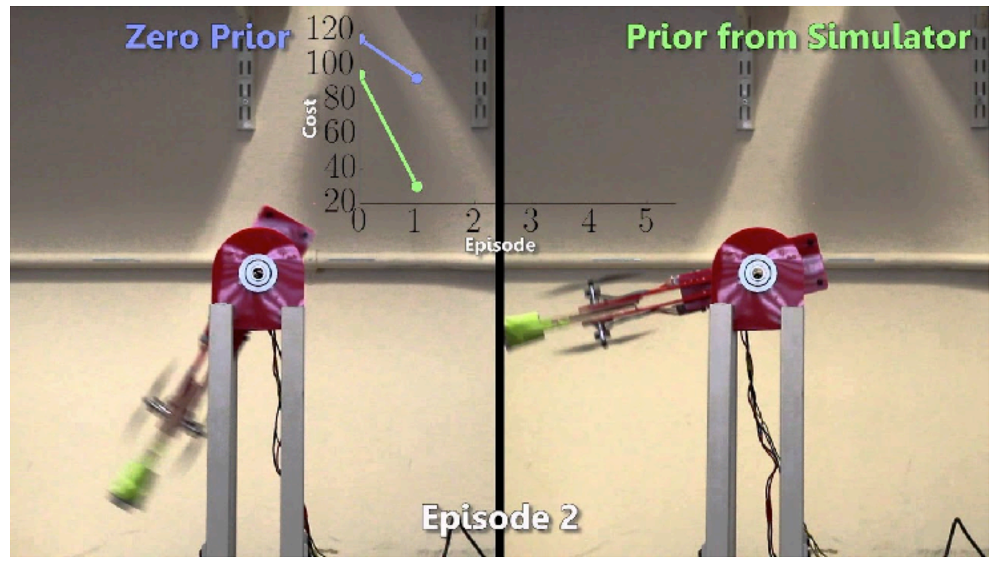
Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Models
[Paul Christiano et al, 2016]
Combining Model-Based Policy Search with Online Model Learning for Control of Physical Humanoids
[Igor Mordatch et al, 2016]

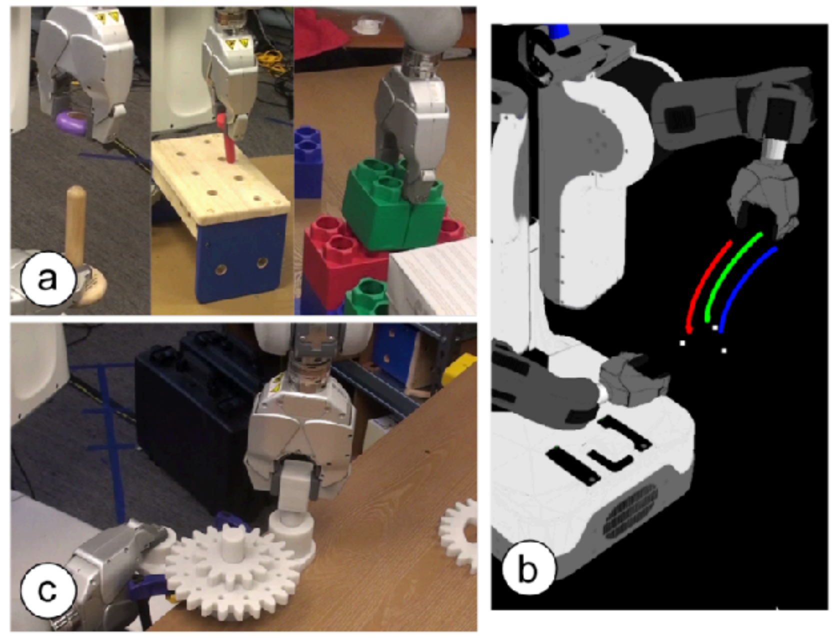
Sim to find low-dim search space



Learning Omnidirectional Path Following Using Dimensionality Reduction
[Zico Kolter & Andrew Ng, 2003]
Deep Predictive Policy Training using Reinforcement Learning
[Ali Ghadirzadeh et al, 2017]

Sim as Bayesian Prior

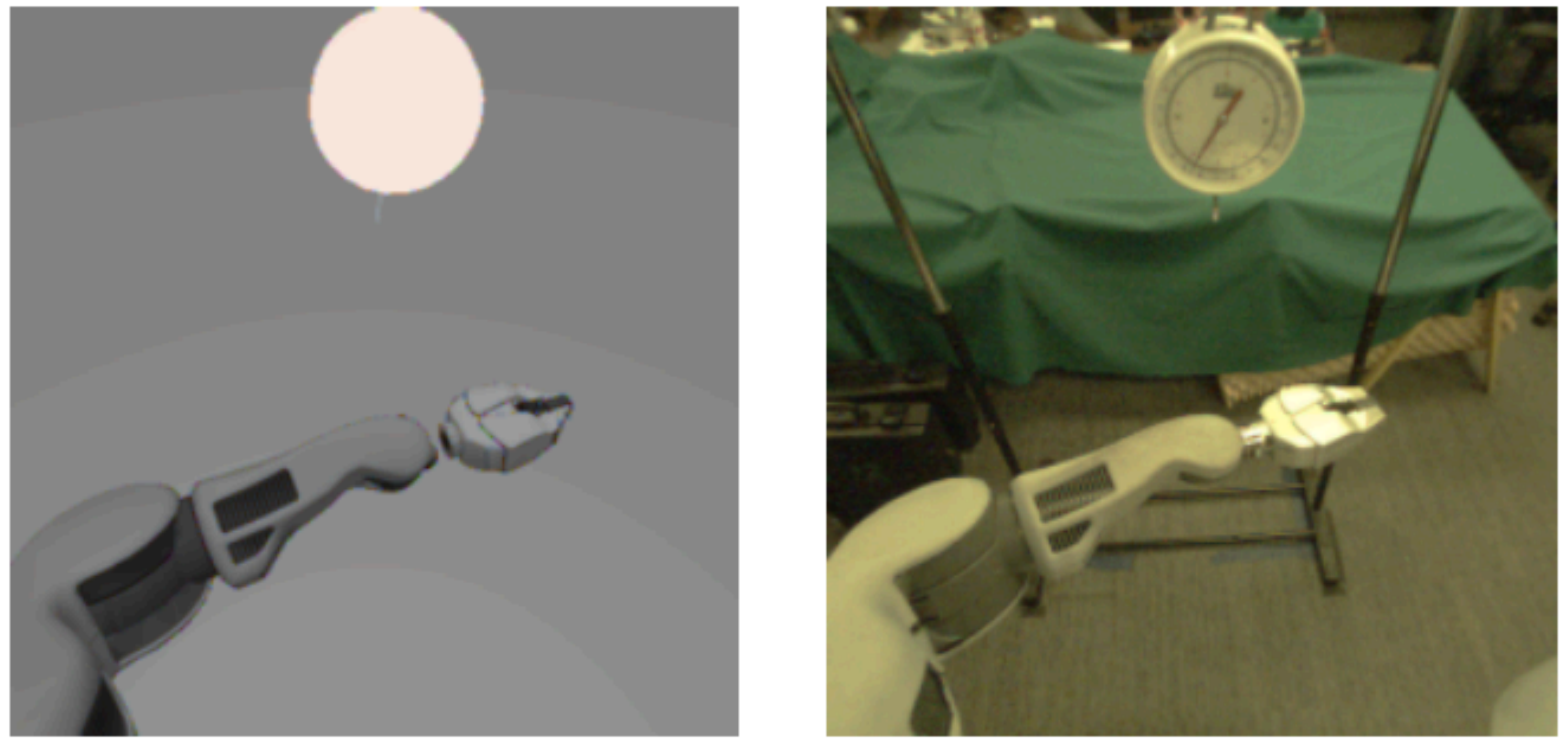




Efficient reinforcement learning for robots using informative simulated priors
[Mark Cutler & Jonathan How, 2015]
Bayesian Optimization with Automatic Prior Selection for Data-Efficient Direct Policy Search
[Remi Pautrat et al, 2018]
One-shot learning of manipulation skills with online dynamics adaptation and neural network priors
[Justin Fu et al, 2016]

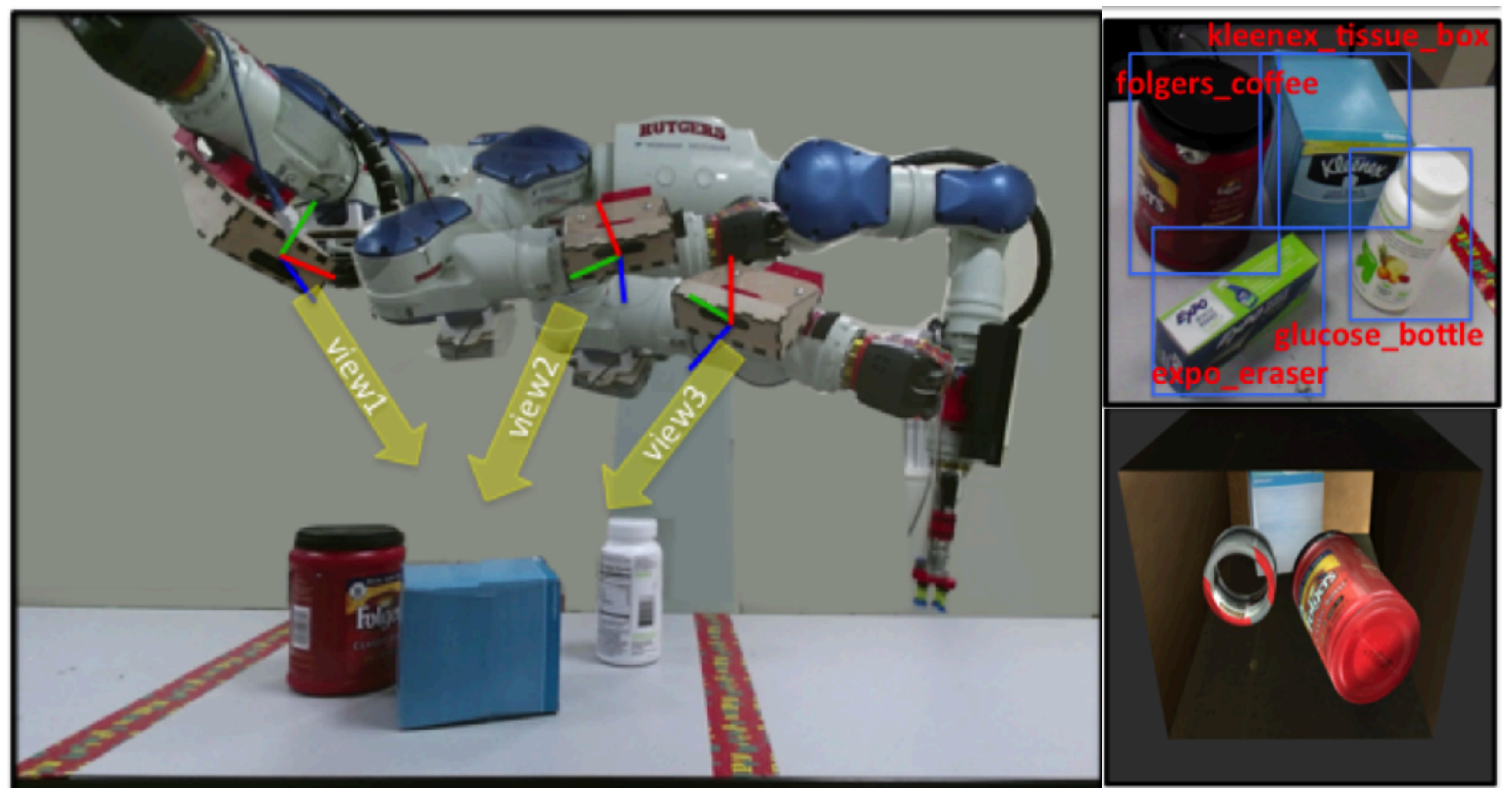
Less Supervised Domain Adaptation

Weakly Supervised



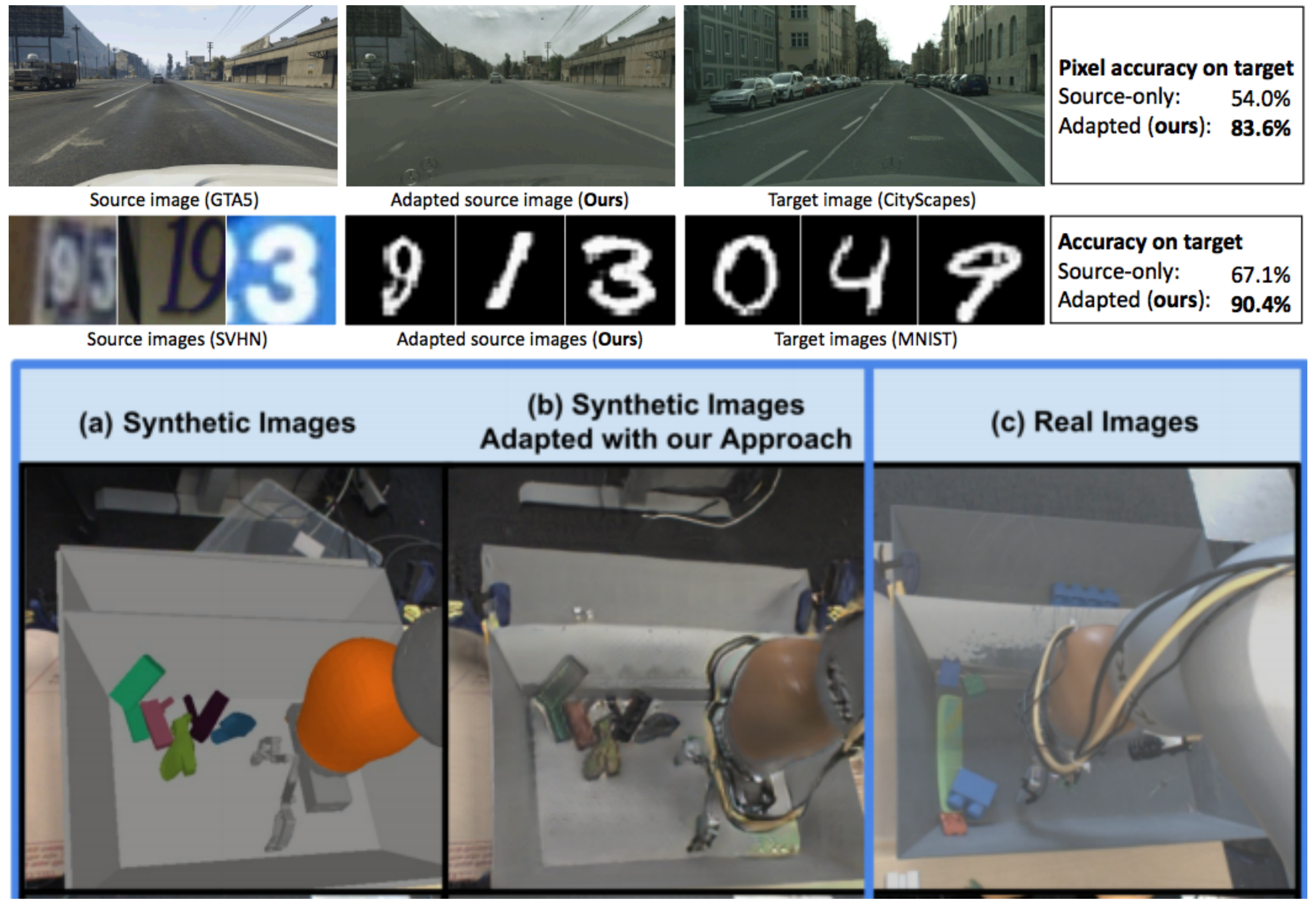
Adapting Deep Visuomotor Representations with Weak Pairwise Constraints
[Tzeng et al., 2016]

Self-Supervised



A Self-supervised Learning System for Object Detection using Physics Simulation and Multi-View Pose Estimation
[Mitash et al, 2017]

Unsupervised

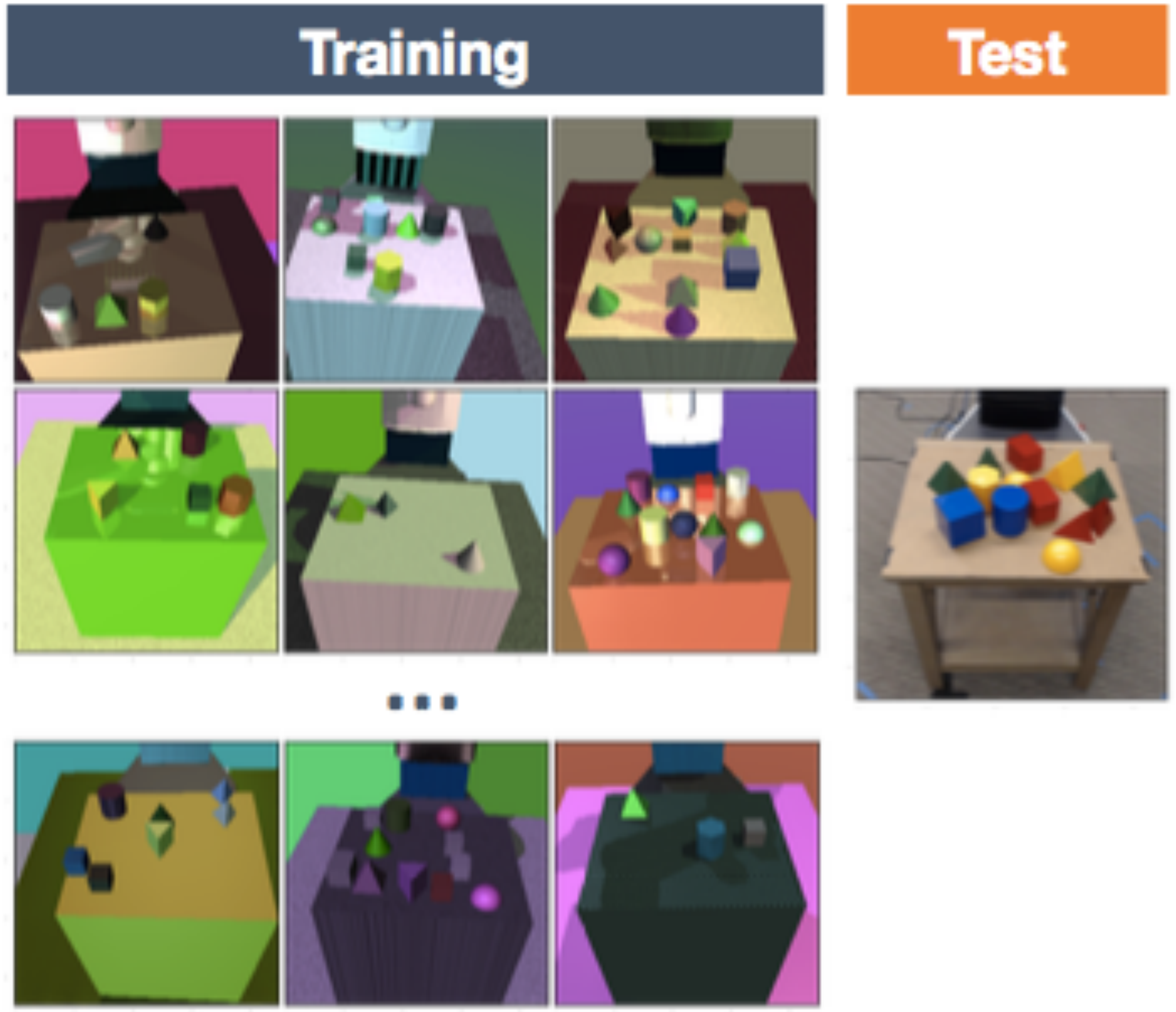


CyCADA
[Judy Hoffman et al, 2017]
Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping
[Bousmalis et al., 2017]

Outline of the rest of the talk

- Why is it so hard to use simulated training data?
- How can you use simulation without solving sim2real?
- Building a good simulation
- Domain adaptation
- **Domain randomization**
- What's next?

Domain Randomization



If the model sees enough simulated variation, the real world may look like just the next simulator

Domain Randomization Overview

A. **History**

B. Applications

C. Why does it work?

D. Tools

E. Challenges

F. Extensions

Radical Envelope of Noise Hypothesis

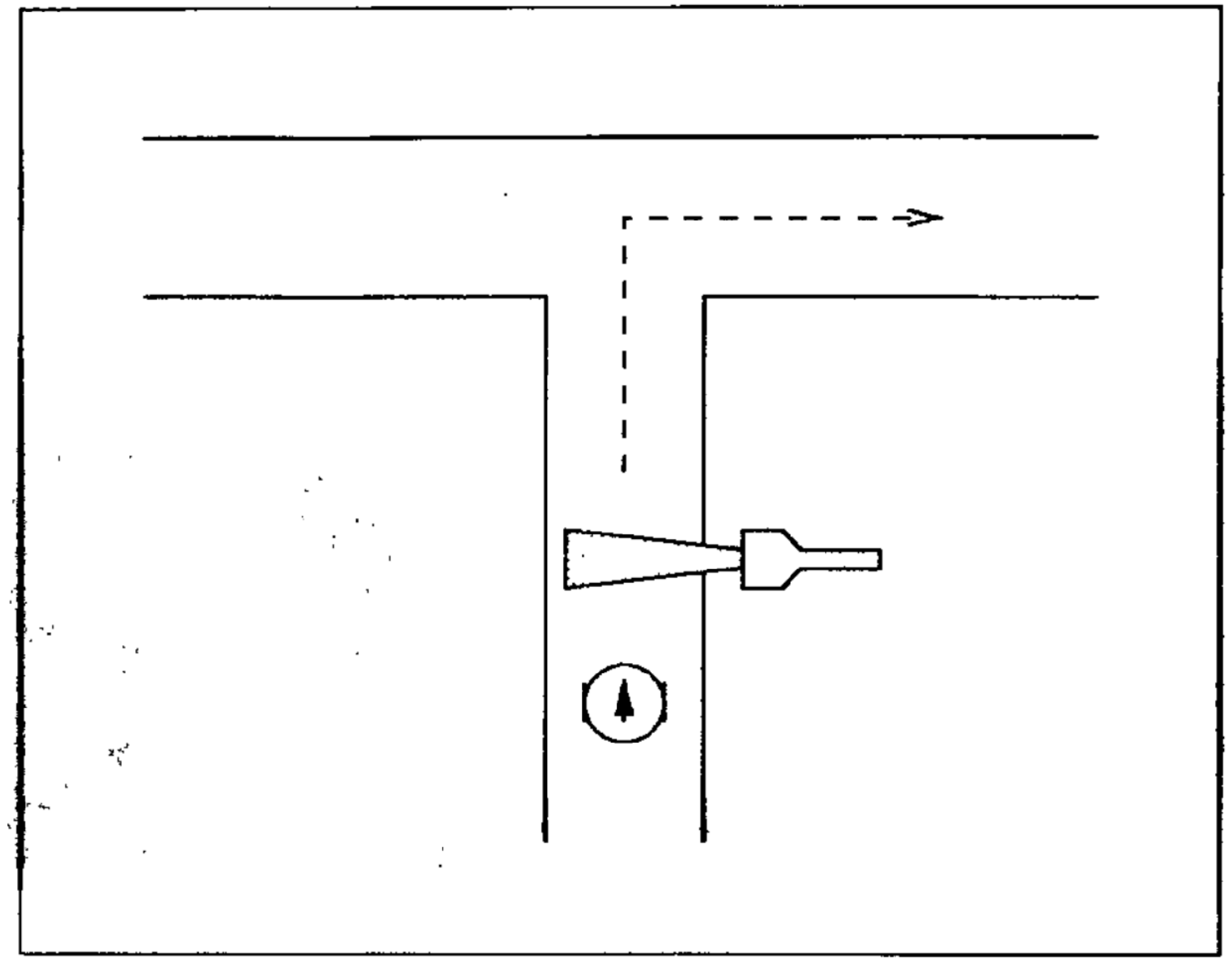
Create a “minimal simulation” consisting of:

1. Base Set

- Aspects of the simulator that are “sufficient to underlie the behavior we want”
- These will be measured and then randomized a bit for robustness

2. Implementation aspects

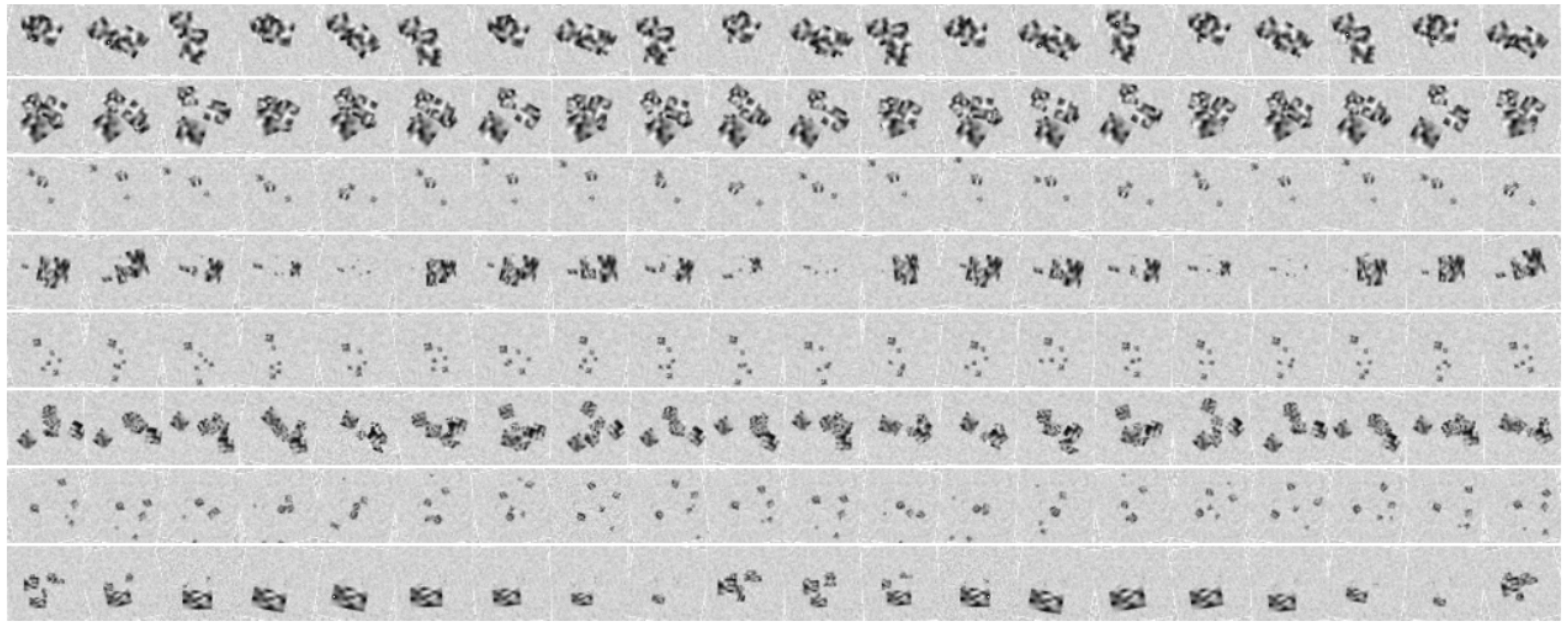
- All other aspects, which do not have a basis in reality in the simulator
- These will be randomized enough so successful controllers “ignore each implementation aspect entirely”



Evolutionary Robotics and the Radical Envelope of Noise Hypothesis
[Nick Jakobi, 1997]

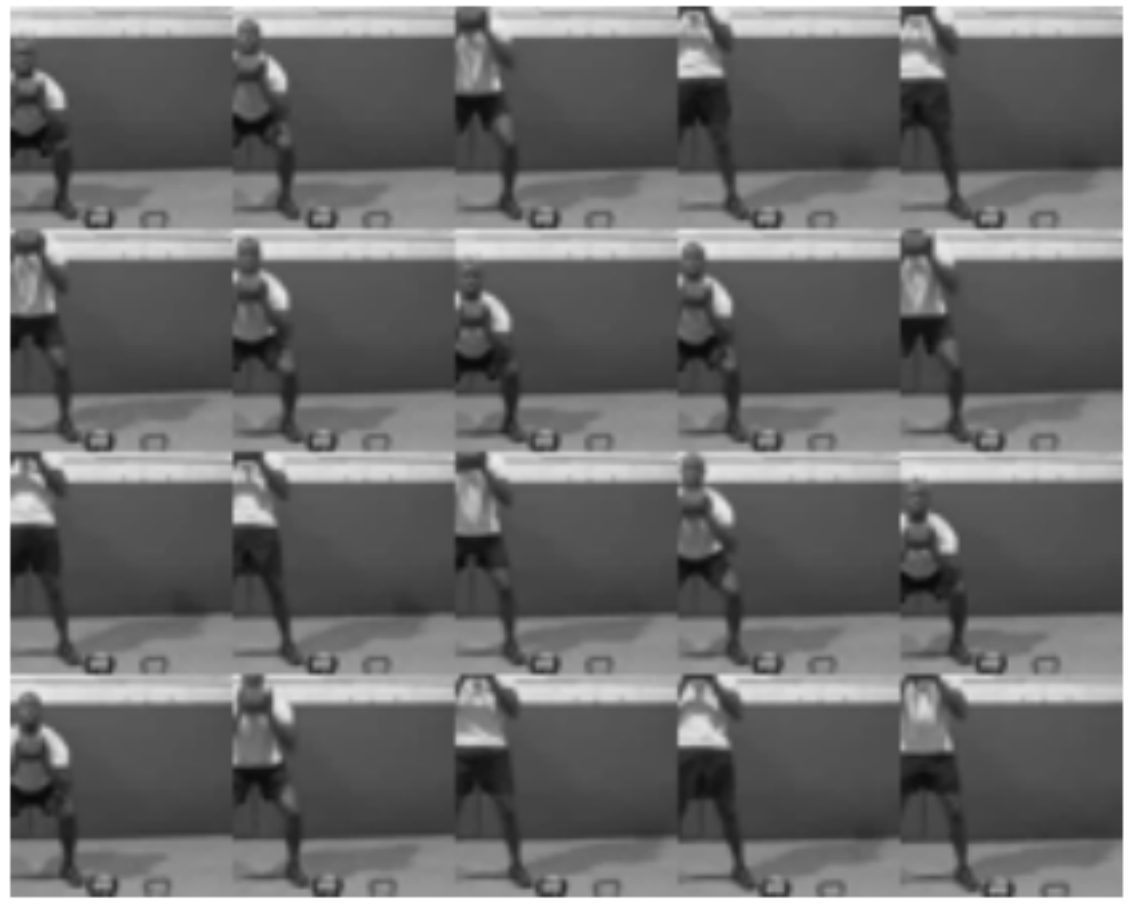
Live Repetition Counting

Training



Predict cycle length of periodic random noise

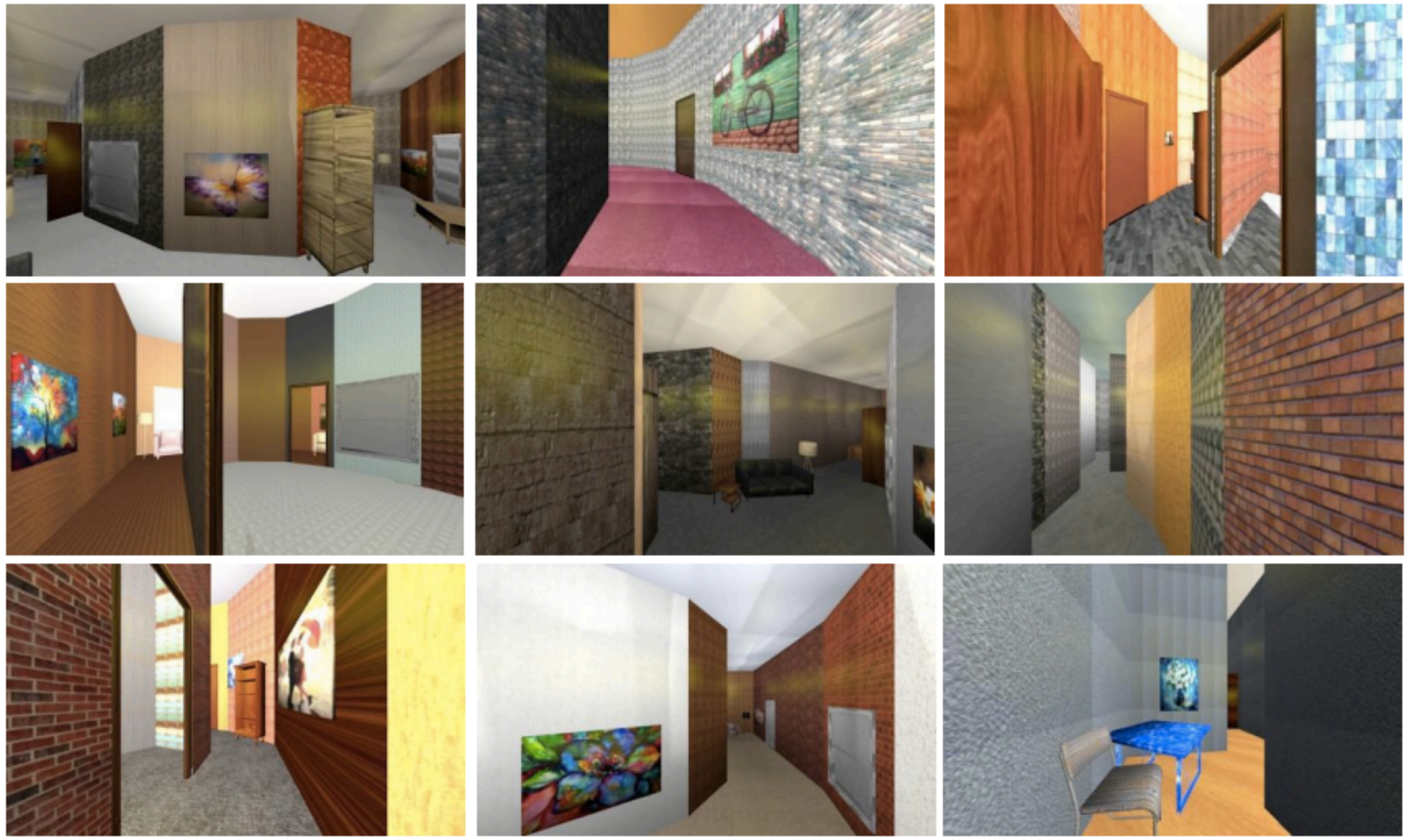
Test



Count repetitive behavior by integrating the predicted period

Live Repetition Counting
[Levy & Wolf, 2015]

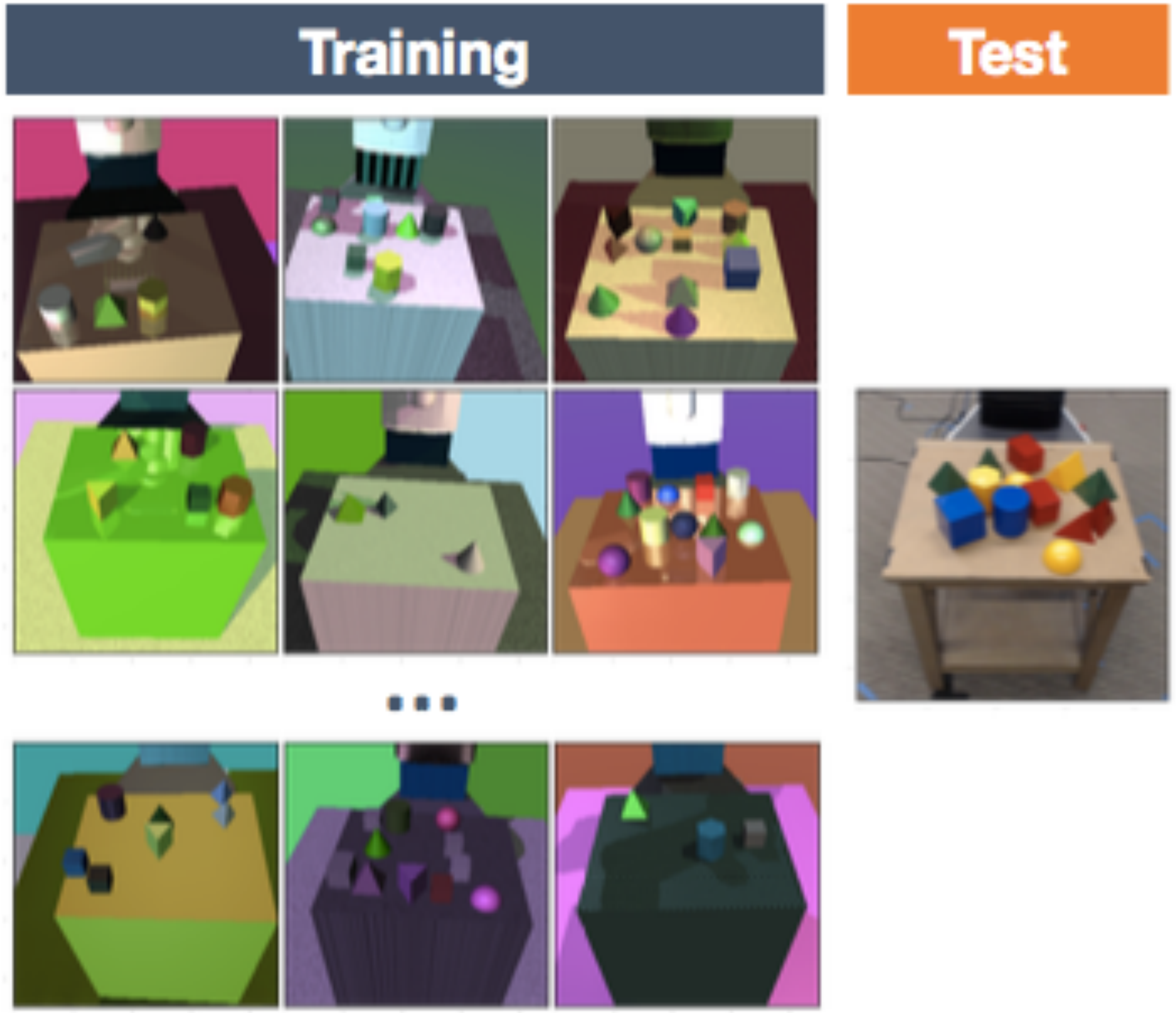
CAD² RL



- Quadcopter collision avoidance
- ~500 semi-realistic textures, 12 floorplans
- ~40-50% of 1000m trajectories are collision-free

(cad)^2 RL: Real Single-Image Flight Without a Single Real Image
[Sadeghi & Levine, 2016]

Domain Randomization for Transferring Neural Nets



- Unrealistic, highly randomized scenes with procedural textures
- No pretraining on real data
- Precise control (~1.5cm accuracy, enabling grasping)

Domain randomization for transferring deep neural networks from simulation to the real world.
[Josh Tobin et al, 2017]

Domain Randomization Overview

A. History

B. Applications

C. Why does it work?

D. Tools

E. Challenges

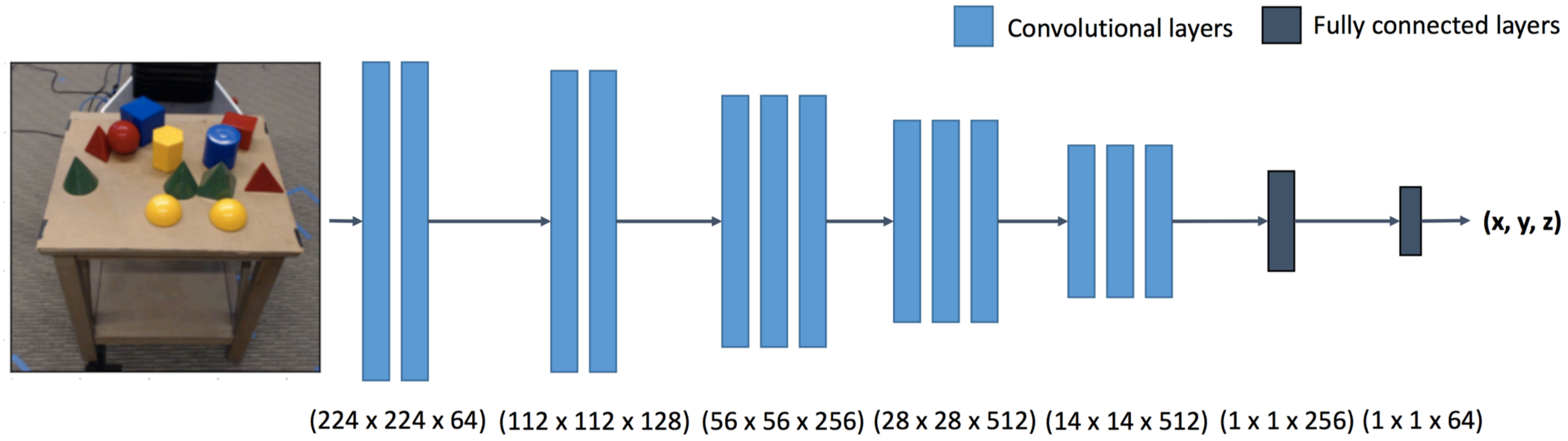
F. Extensions

Domain randomization for vision: pose estimation

- Each scene has a unique set of randomizations, including:
  - Texture & material properties of all objects, table, background, robot
  - Position of cameras (within a small range)
  - Lighting position, orientation, color, and specular properties
  - Distractor objects in the scene

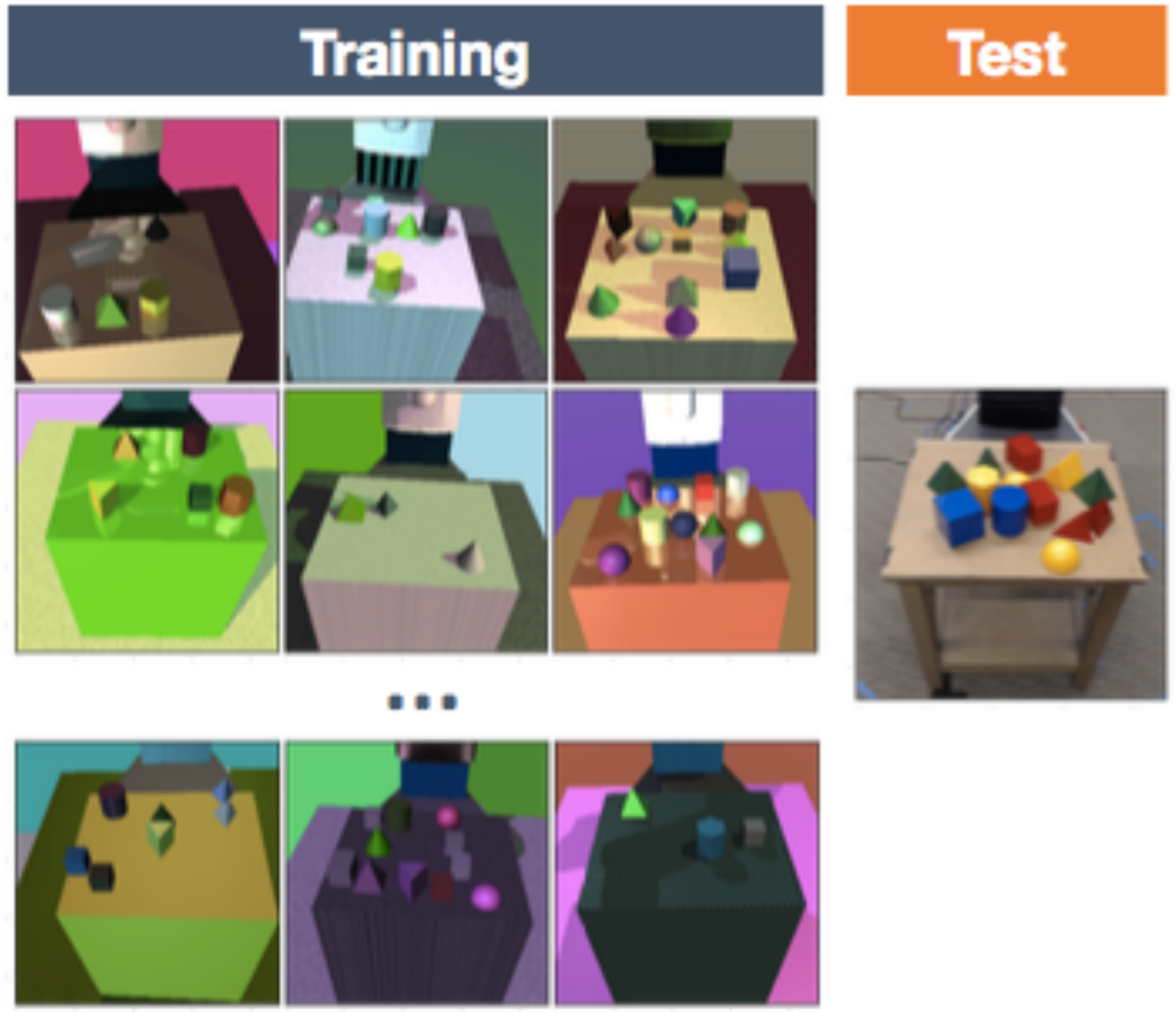
Domain randomization for transferring deep neural networks from simulation to the real world.
[Josh Tobin et al, 2017]

Neural network architecture



Domain randomization for transferring deep neural networks from simulation to the real world.
[Josh Tobin et al, 2017]

How well does it work?



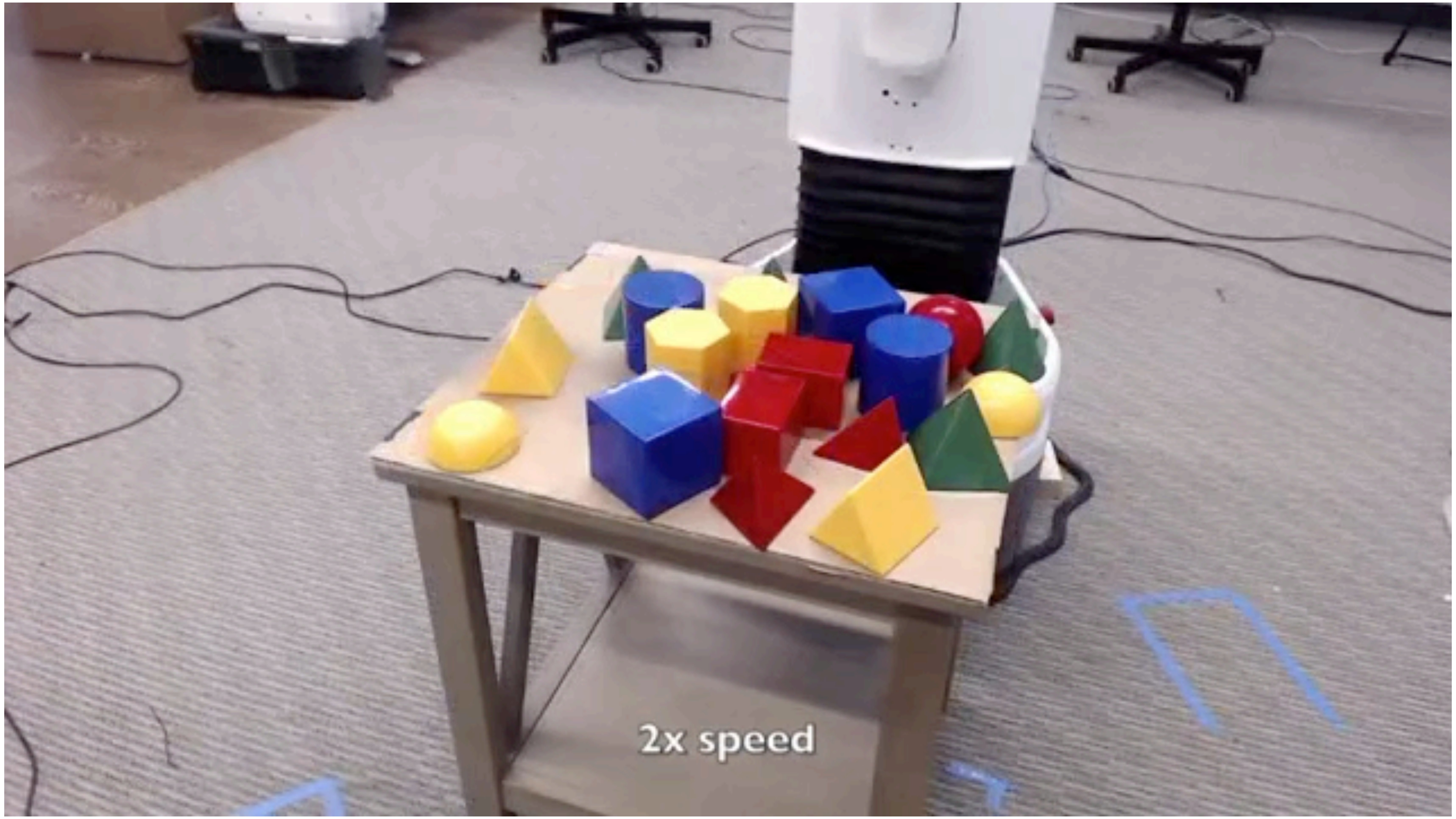
Ours	MOPED-1V [1]	[2]
1.5cm	1.45cm	0.46cm – 1.62cm*

[1] The MOPED framework: Object Recognition and Pose Estimation for Manipulation
[Collet, Martinez, Srinivasa, 2011]

[2] Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation
[Collet, Berenson, Srinivasa, Ferguson, 2009]

* Depending on the distance from the camera. 0.46cm at 70cm, 1.62cm at 90cm. Ours are between 70 and 105cm from the camera.

Grasping using a sim2real-trained pose estimator

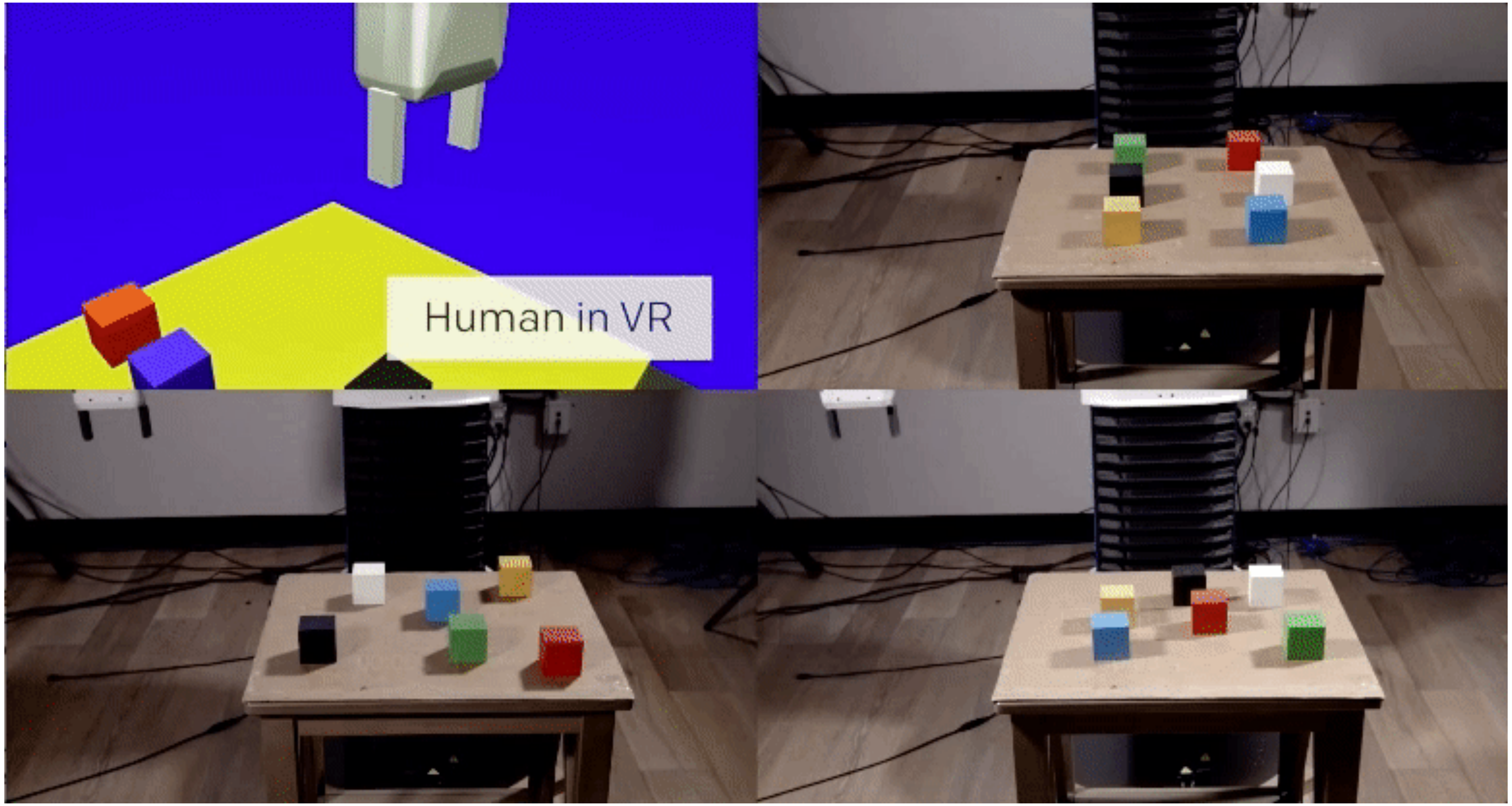


Domain randomization for transferring deep neural networks from simulation to the real world.
[Josh Tobin et al, 2017]

Grasping using a sim2real-trained pose estimator

Spam Detection in the Physical World
[Rachel Fong et al, 2017]

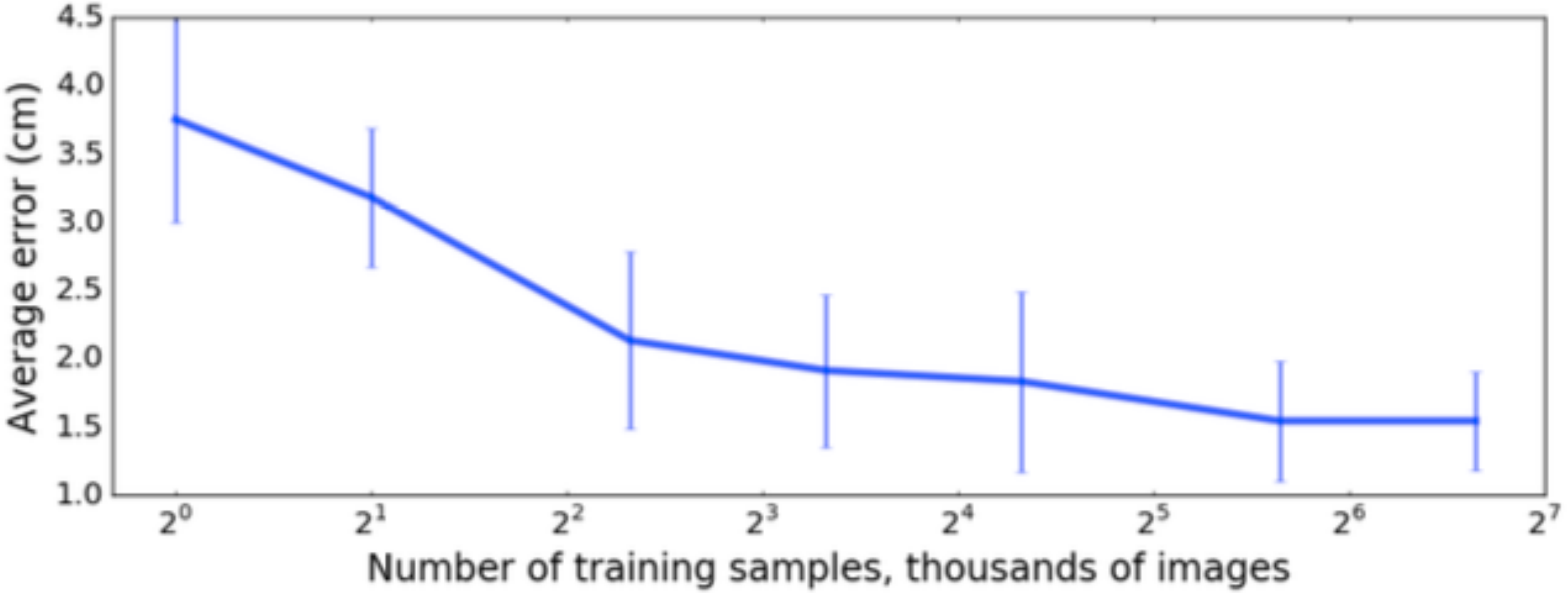
Block stacking using a sim2real trained pose estimator



Robots that Learn
[Peter Welinder et al, 2017]

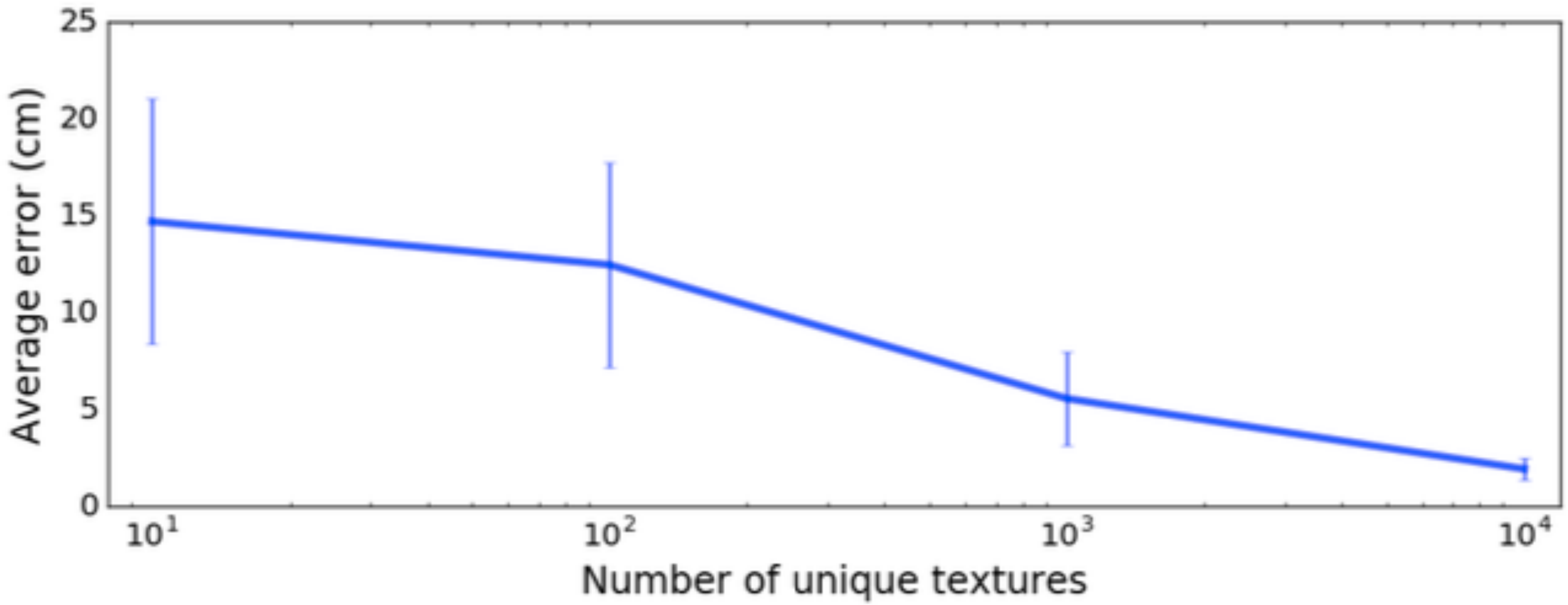
How does it work?

More data = better



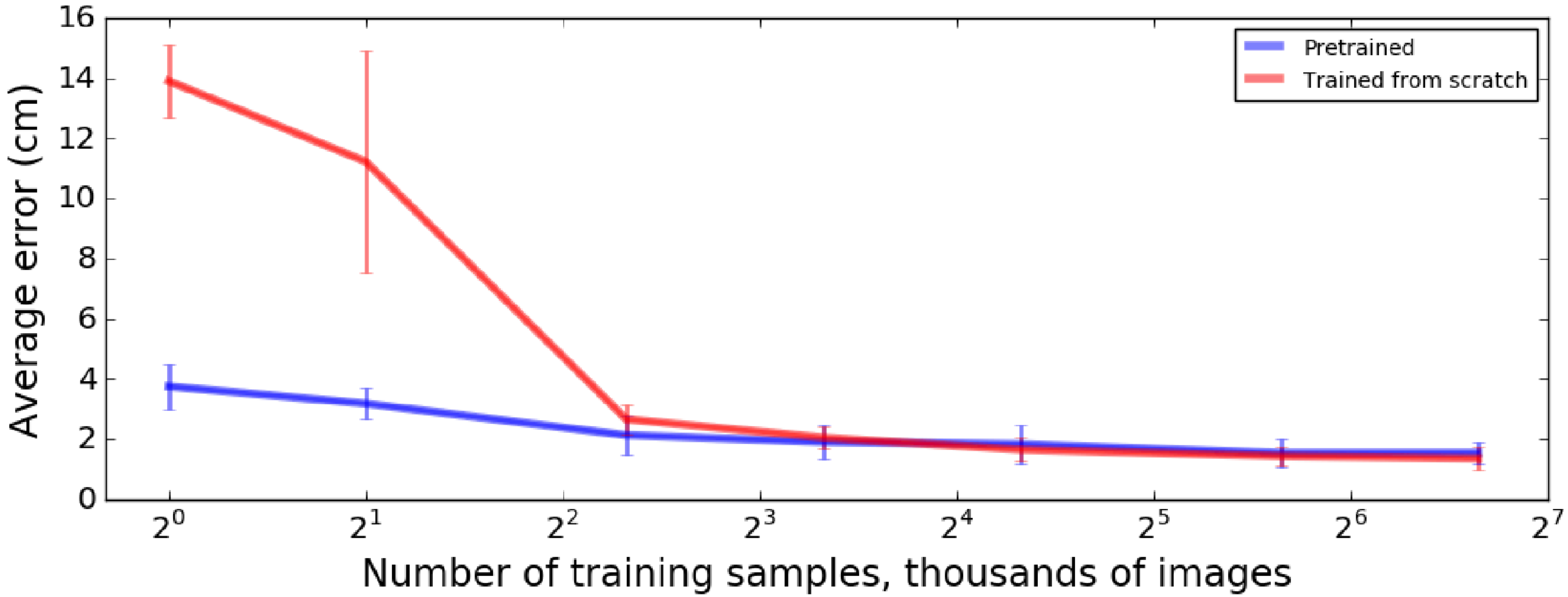
Domain randomization for transferring deep neural networks from simulation to the real world.
[Josh Tobin et al, 2017]

More textures = better



Domain randomization for transferring deep neural networks from simulation to the real world.
[Josh Tobin et al, 2017]

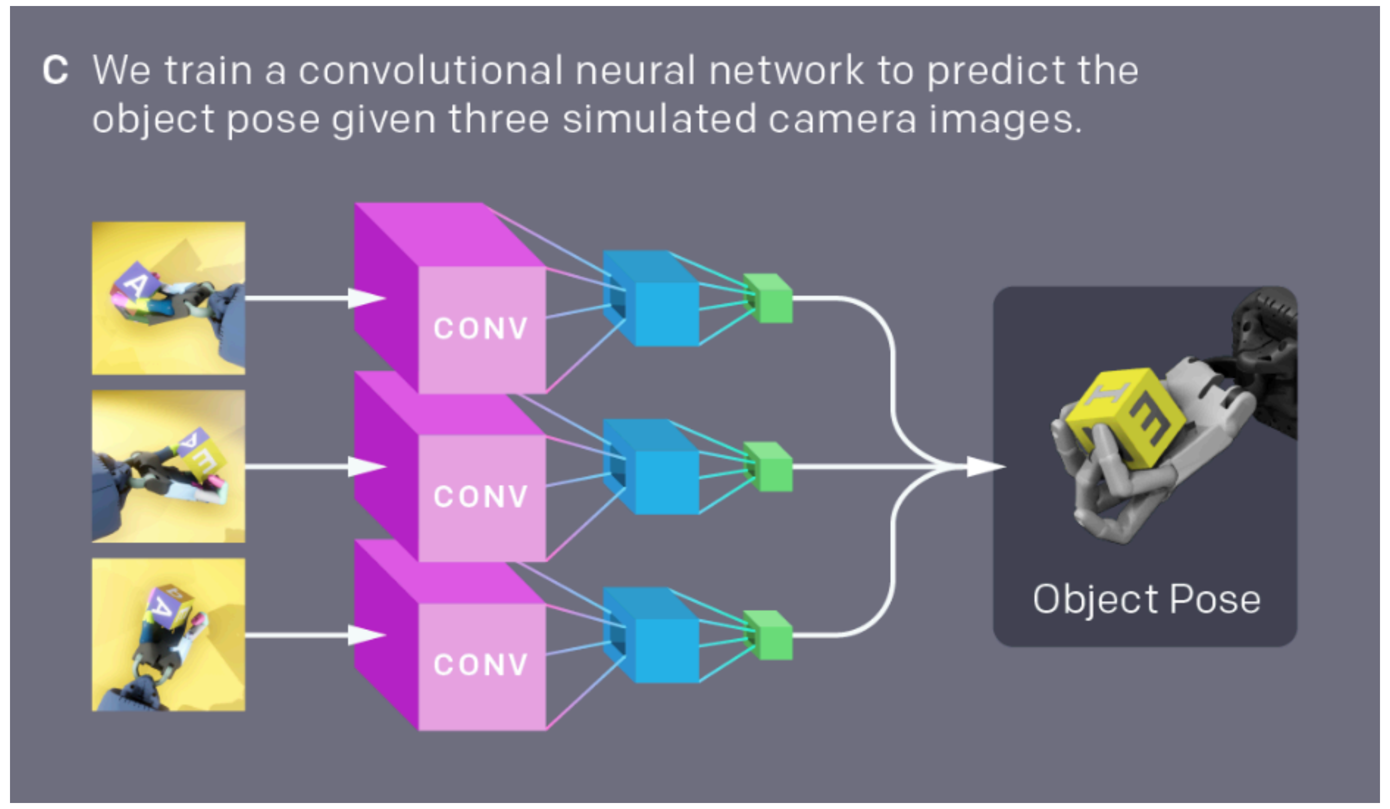
Pre-training is not necessary

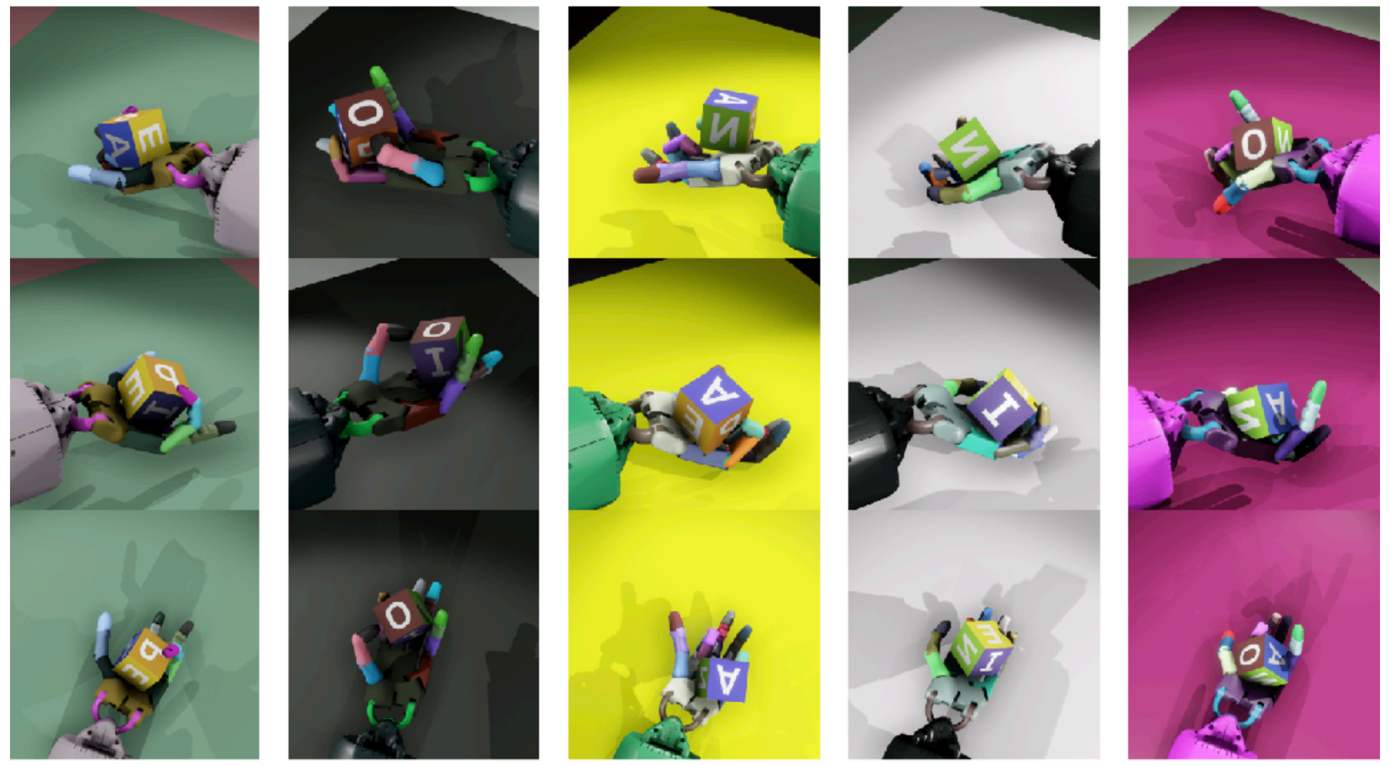


Domain randomization for transferring deep neural networks from simulation to the real world.
[Josh Tobin et al, 2017]

Other appearance randomization results

Full 6D pose



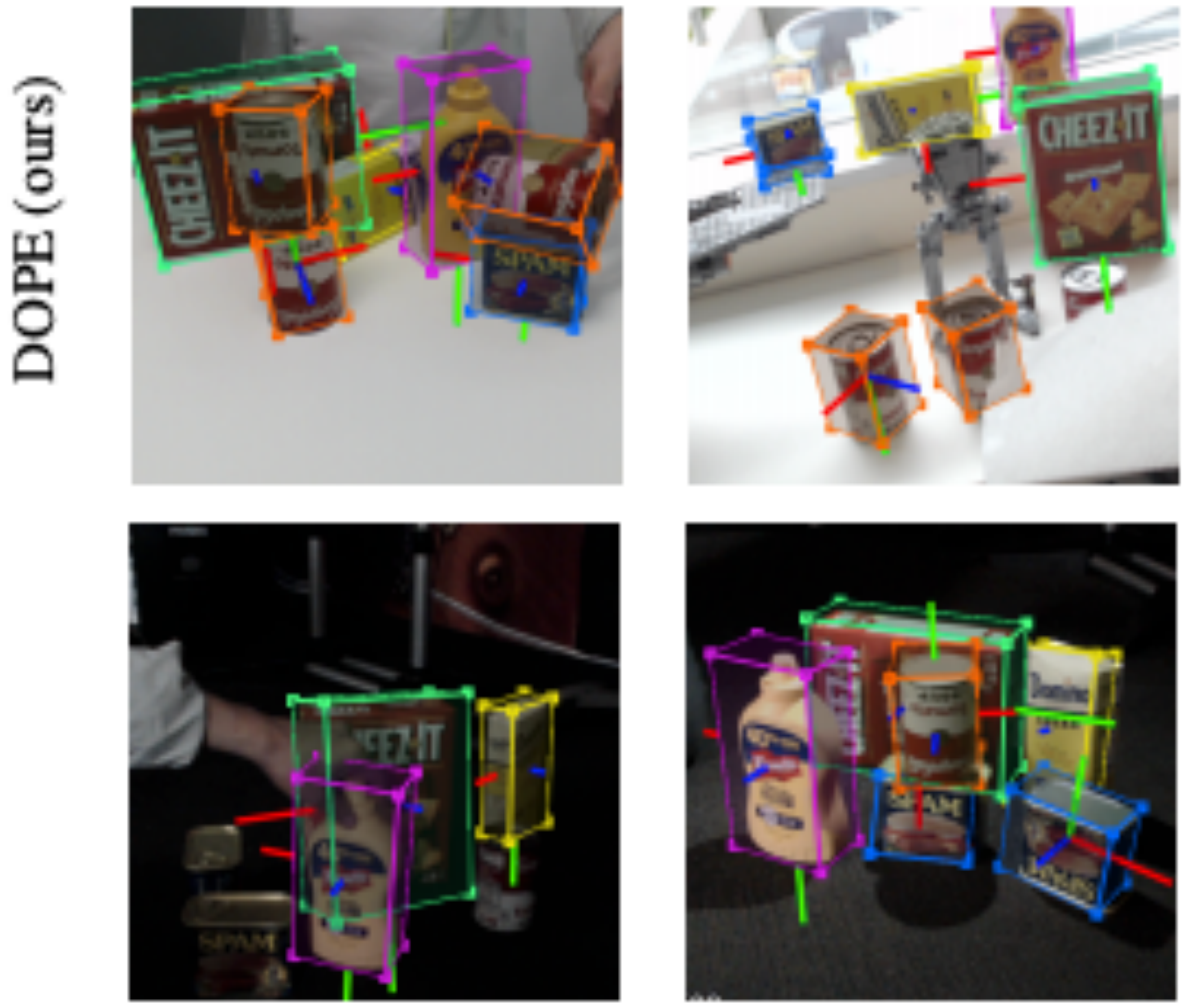


Learning dextrous in-hand manipulation
[OpenAI Robotics, 2018]

Challenging textures

Grasping virtual fish: a step toward robotic deep learning from demonstration in virtual reality
[Jonotan Dyrstad & John Reidar Mathiassen, 2017]

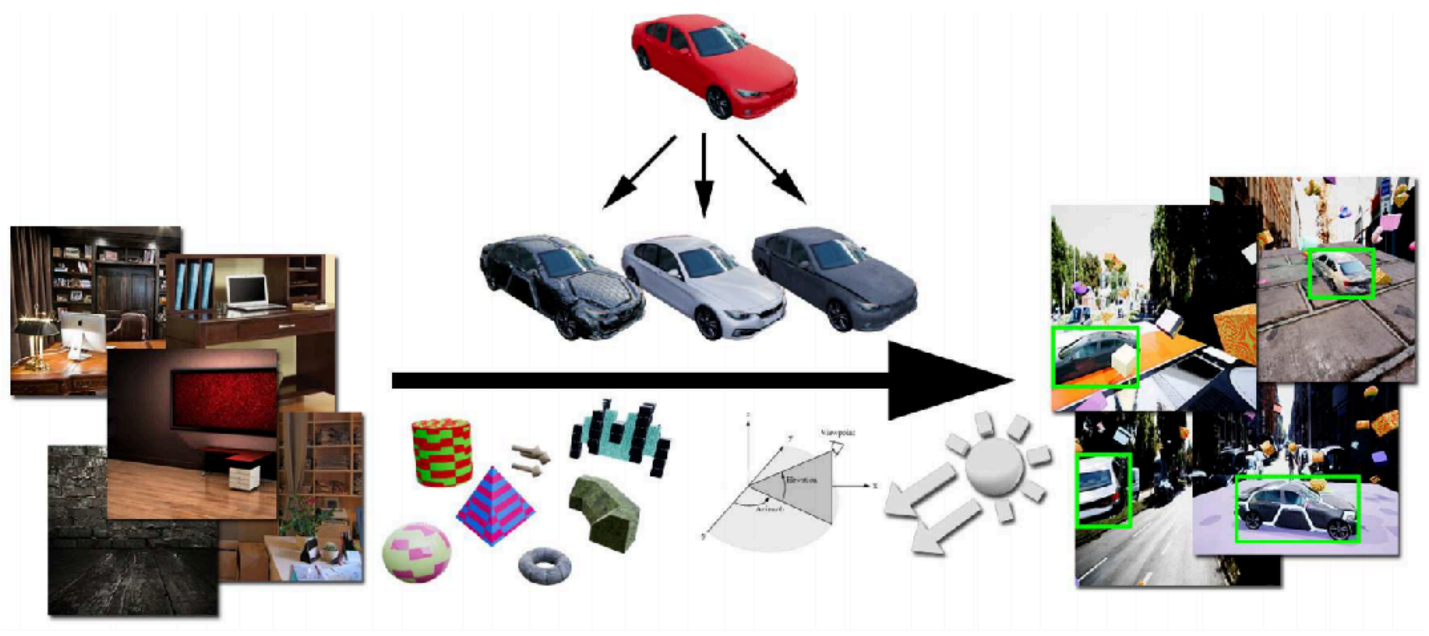
Wider range of objects



Deep object pose estimation for semantic robotic grasping of household objects
[Jonathan Tremblay et al, 2018]

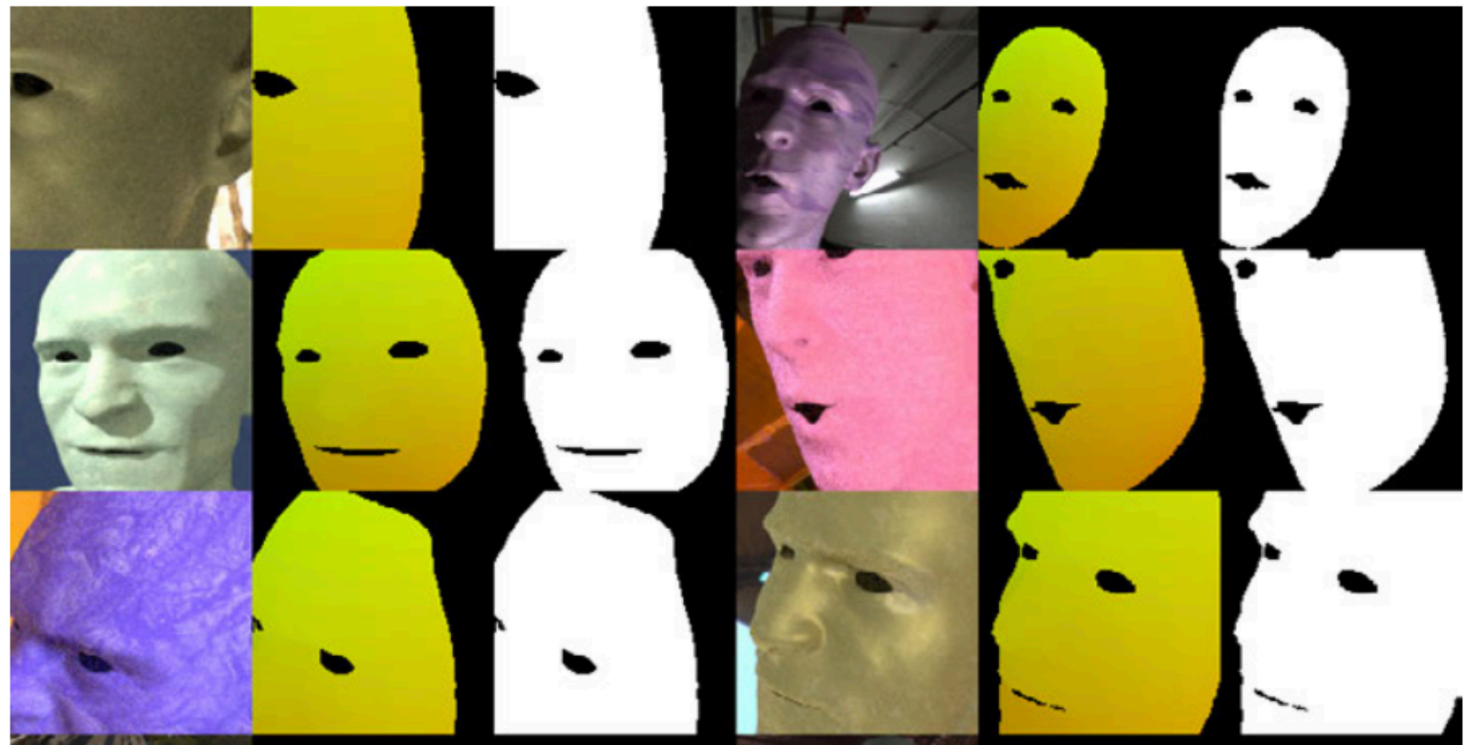
Other appearance randomization results

Object detection for autonomous vehicles



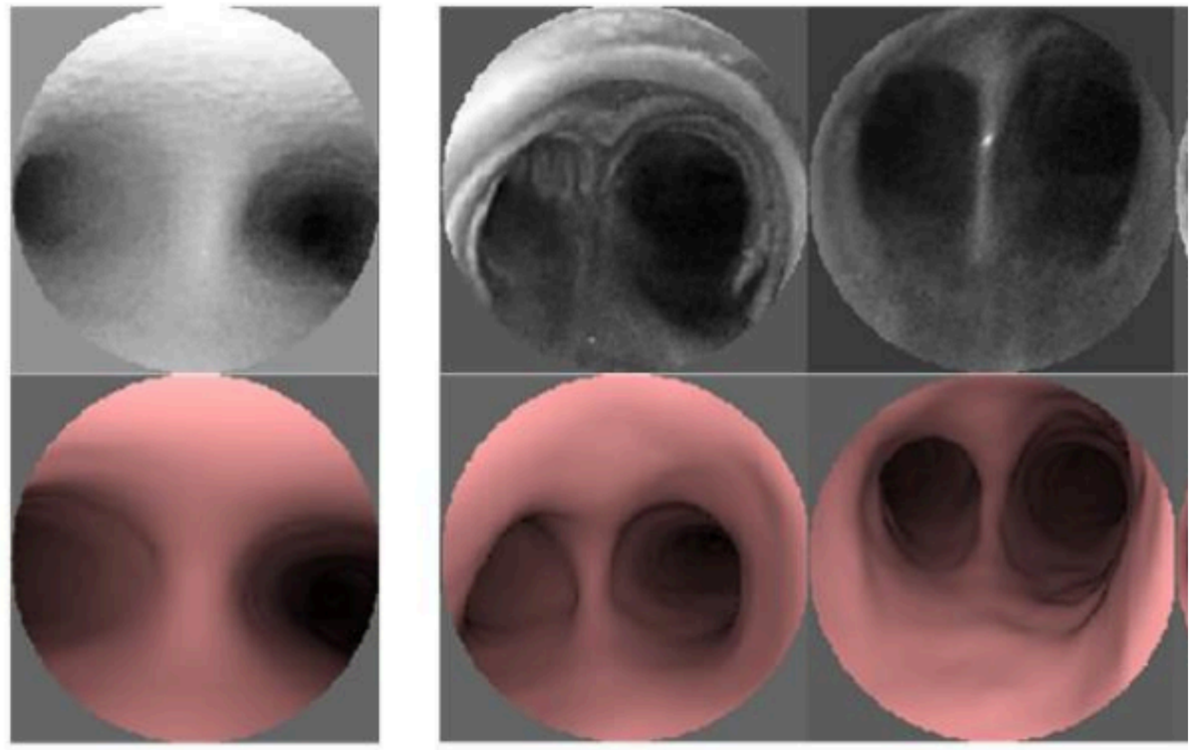
Jonathan Tremblay et al. “Training deep networks with synthetic data: Bridging the reality gap by domain randomization”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018, pp. 969-977.

Face tracking



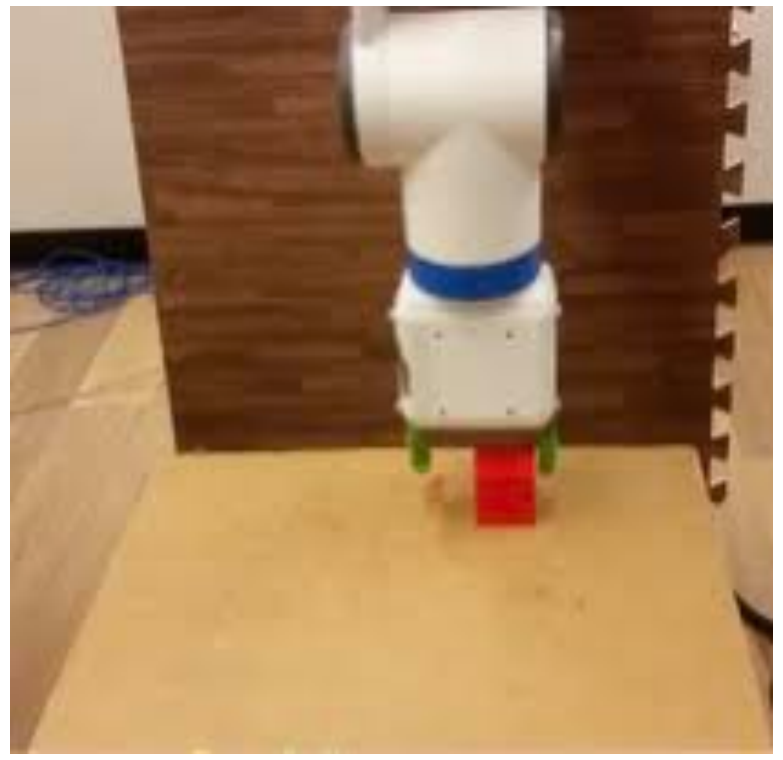
Mikko Ronkainen et al. “Dense tracking of human facial geometry-aware”. In: (2017).

In-lung localization



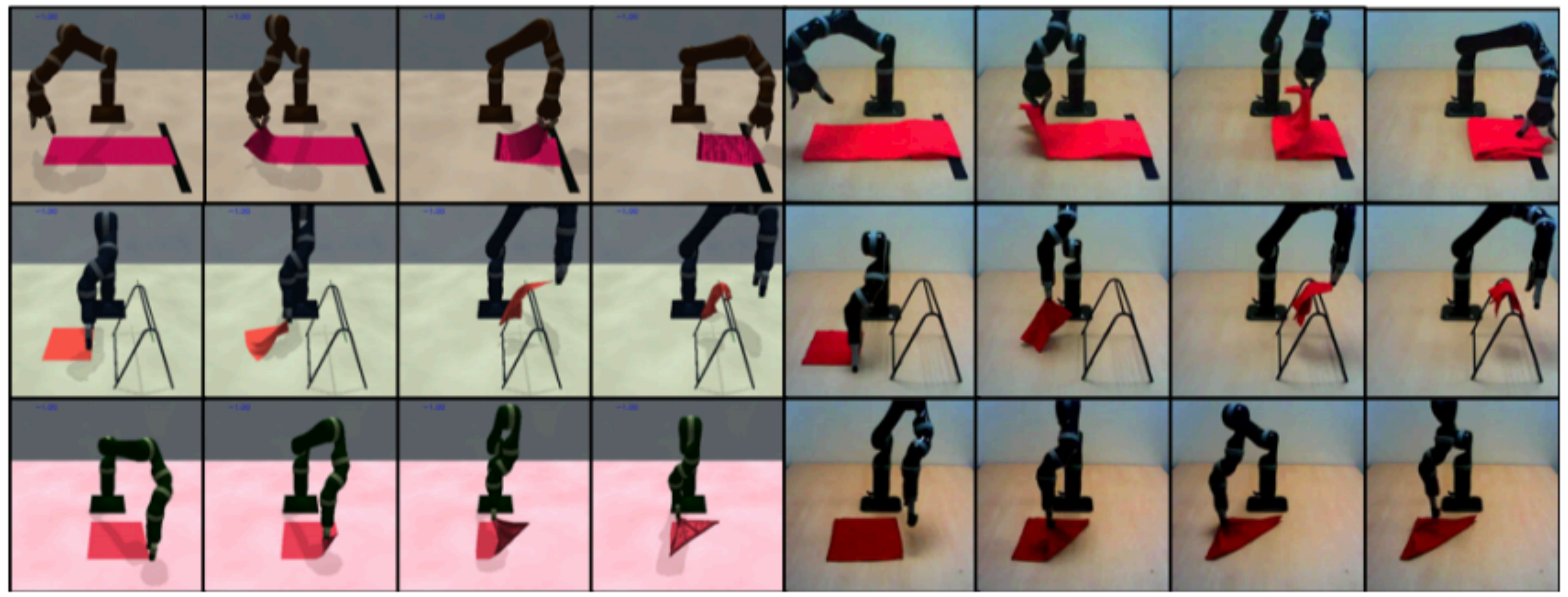
Sganga, Jake, et al. "Deep Learning for Localization in the Lung." arXiv preprint arXiv:1903.10554 (2019).

Visuomotor control



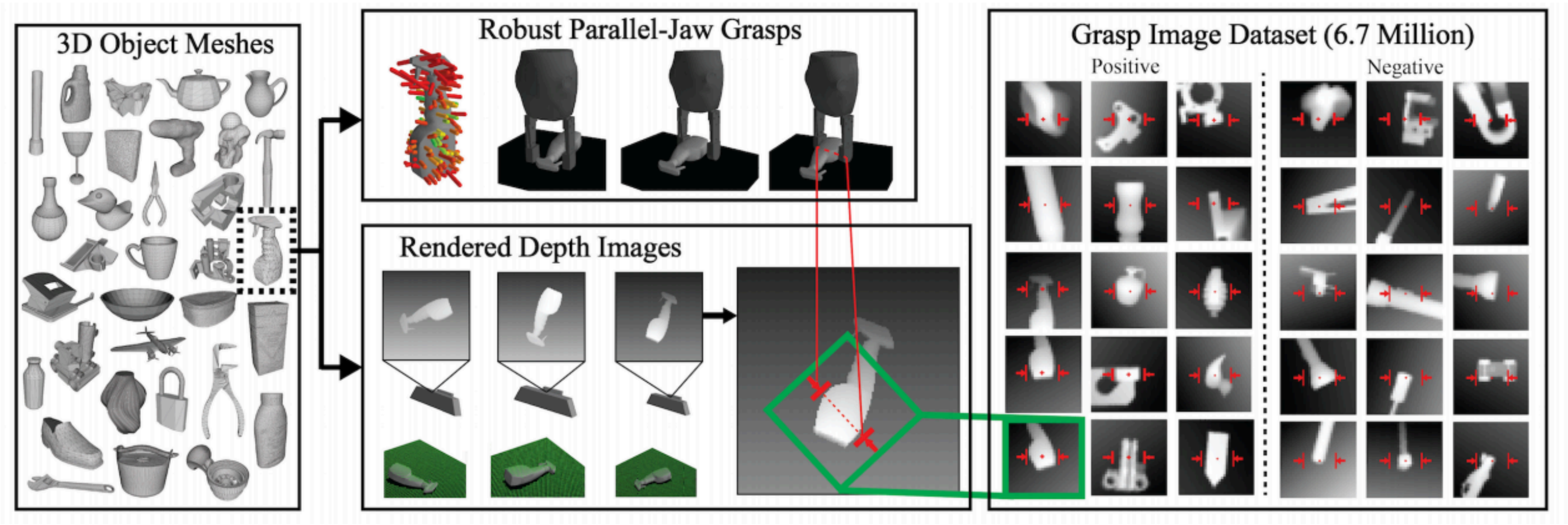
Lerrel Pinto et al. “Asymmetric actor critic for image-based robot learning”. In: arXiv preprint arXiv:1710.06542 (2017).

Cloth manipulation



Jan Matas, Stephen James, and Andrew J Davison. “Sim-to-real reinforcement learn- ing for deformable object manipulation”. In: arXiv preprint arXiv:1806.07851 (2018).

DR works for other sensors too

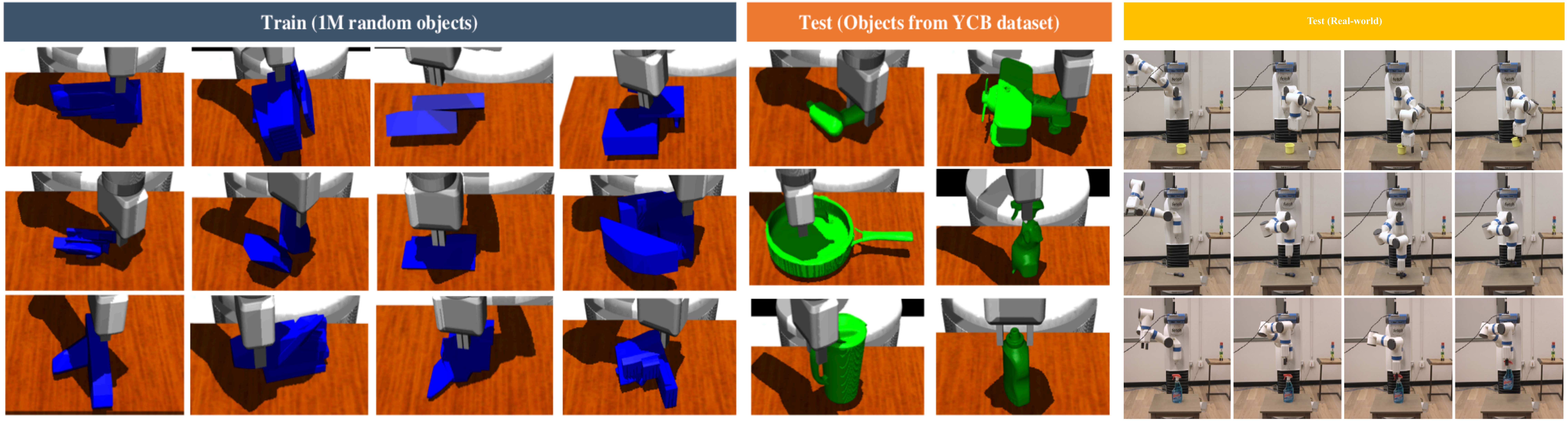


Dex-Net 2.0
[Jeffrey Mahler et al, 2017]

Common assumption in these results

- We have 3D models of all objects we want to track
- How to move on to arbitrary objects?

Scene / object randomziation

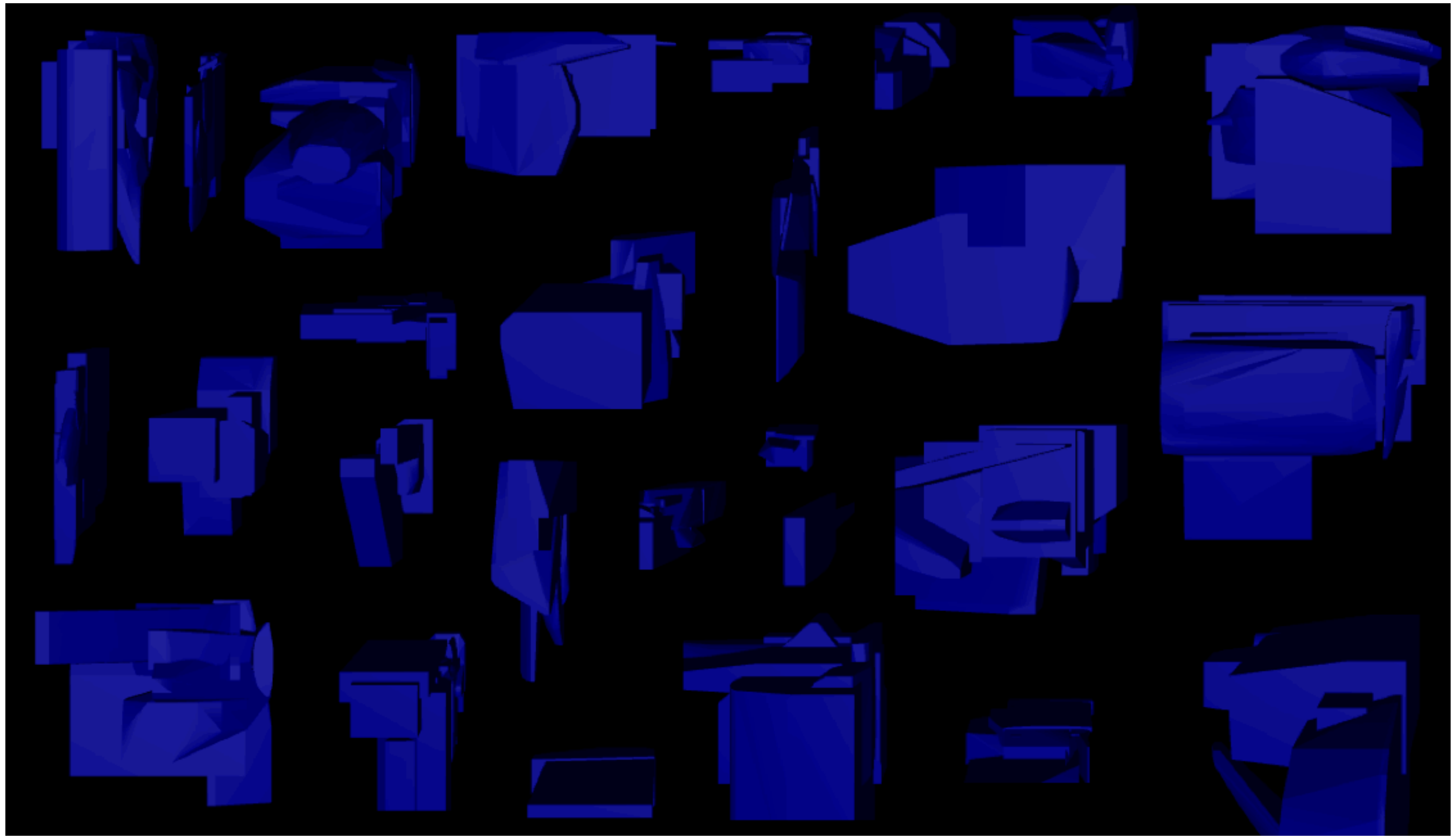


Hypothesis: If the model sees a wide enough range of (unrealistic) objects during training, at test time it will generalize to realistic objects

Domain randomization and generative models for robotic grasping
[Josh Tobin et al, 2018]

Data generation pipeline

- Create a primitive object set: take every ShapeNet object and decompose it into convex parts. Each part is a primitive
- For each object to generate:
  - Choose a number np of primitives from 1 - 15
  - Sample np primitives from the primitive object set
  - Scale the primitives so that their dimensions are between 1 and 15cm
  - Place them sequentially so that each intersects with the previous
  - Rescale the final object if it is too large



Domain randomization and generative models for robotic grasping
[Josh Tobin et al, 2018]

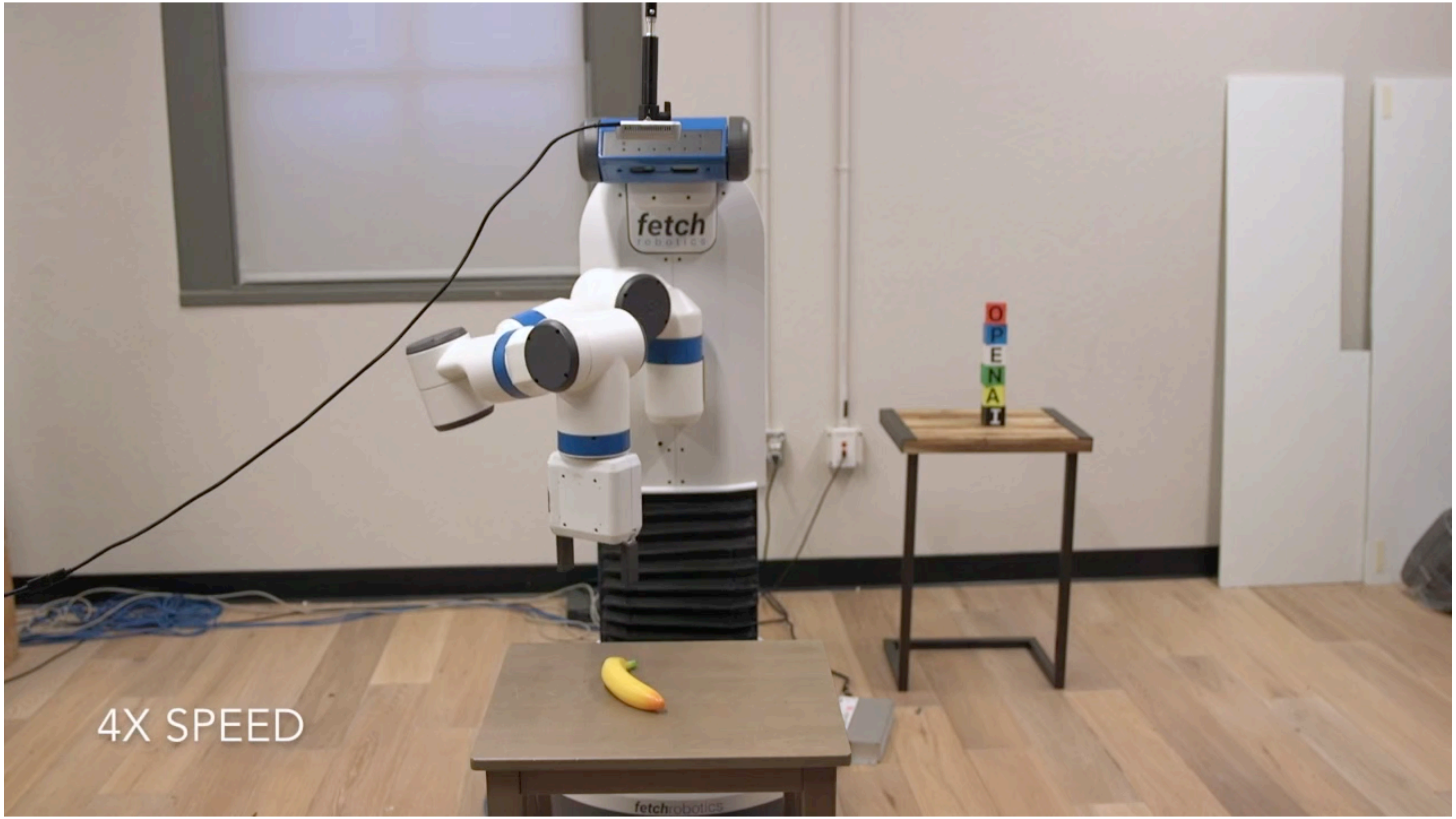
Random objects work as well as realistic

Training set	ShapeNet Train	ShapeNet Test	Random Train	Random Test	Ycb
ShapeNet-1M	0.91	**0.91**	0.72	0.71	**0.93**
Random-1M	0.91	0.89	**0.86**	**0.84**	0.92
ShapeNet-Random-1M	**0.92**	0.90	0.84	0.81	0.92

Training set	ShapeNet Train	ShapeNet Test	Random Train	Random Test	Ycb
Full Algorithm	**0.91**	**0.89**	**0.86**	**0.84**	**0.92**
Autoregressive-Only	0.89	0.86	0.80	0.76	0.89
Random	0.22	0.21	0.10	0.11	0.26
Centroid	0.30	0.25	0.10	0.12	0.54

Domain randomization and generative models for robotic grasping
[Josh Tobin et al, 2018]

Application of our method



Domain randomization and generative models for robotic grasping
[Josh Tobin et al, 2018]

Assumption

- Dynamics are relatively consistent between sim and real
- What if not?

DR for dynamics

- Standard RL: train a feedforward neural network policy in a single best environment
- Instead: train a **recurrent** network
- For each rollout, sample a different set of physics parameters

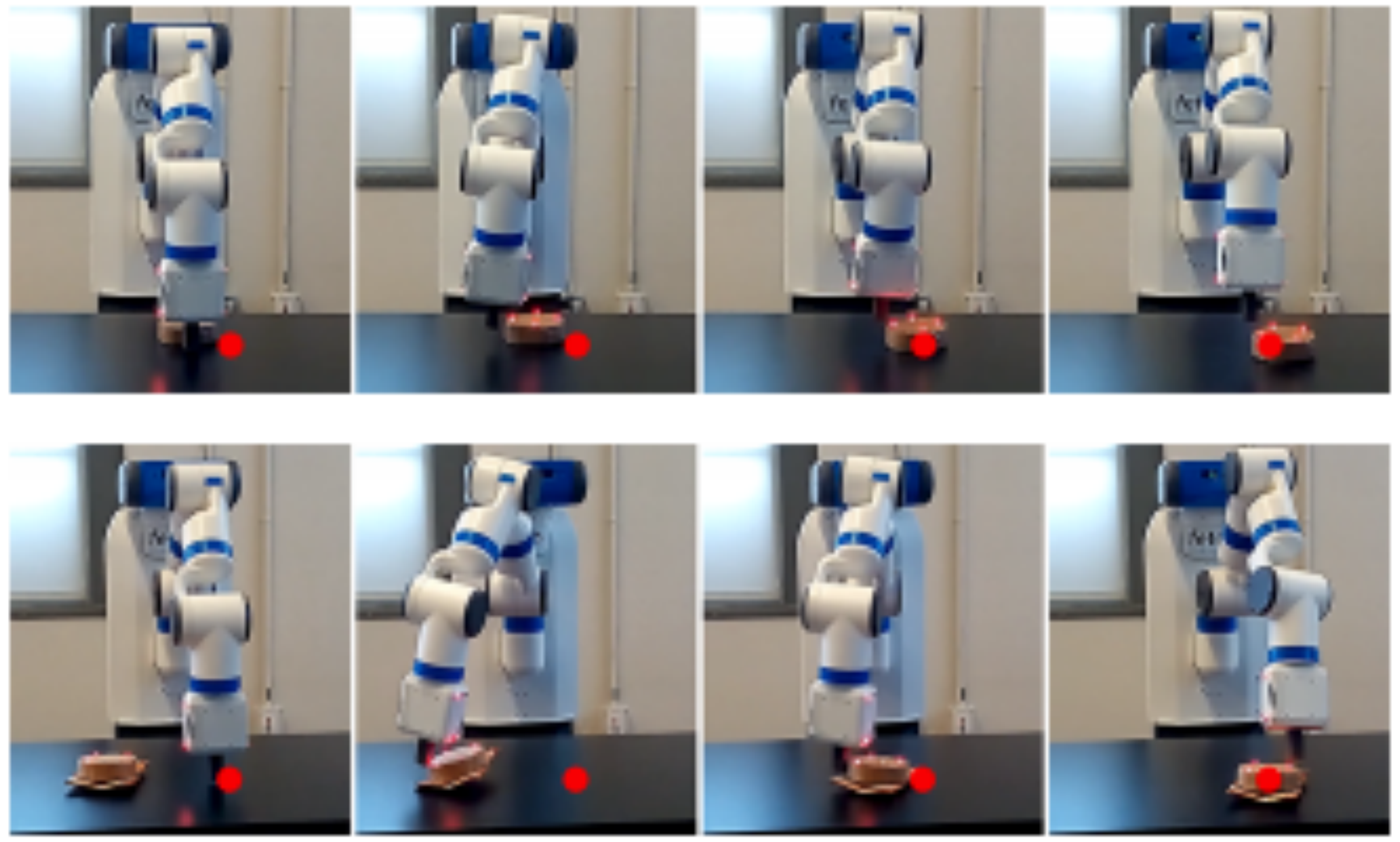


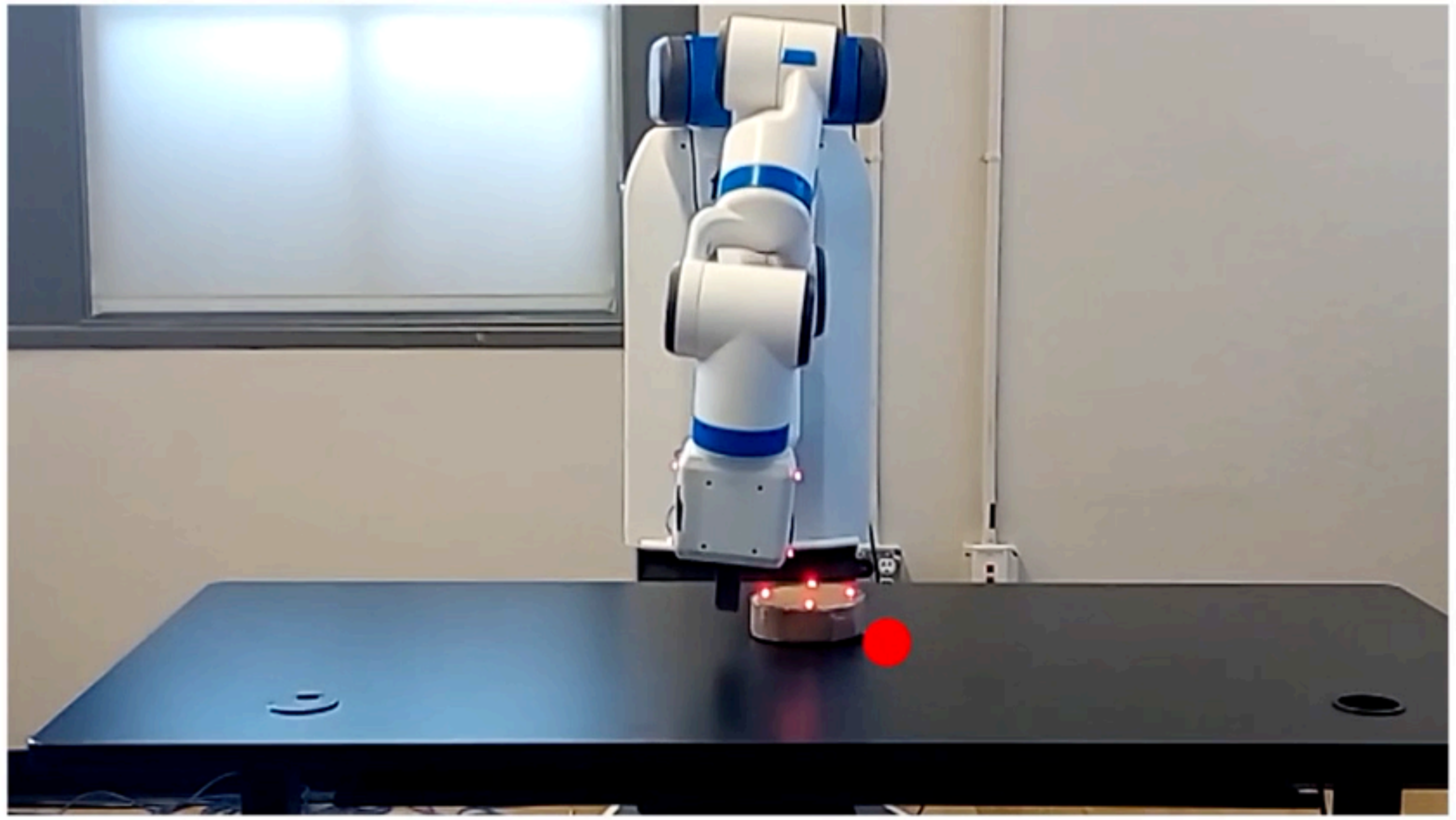
Fig. 3. LSTM policy deployed on the Fetch arm. **Bottom:** The contact dynamics of the puck was modified by attaching a packet of chips to the bottom.

Parameter	**Range**
Link Mass	$[0.25, 4] \times$ default mass of each link
Joint Damping	$[0.2, 20] \times$ default damping of each joint
Puck Mass	$[0.1, 0.4] kg$
Puck Friction	$[0.1, 5]$
Puck Damping	$[0.01, 0.2] Ns/m$
Table Height	$[0.73, 0.77] m$
Controller Gains	$[0.5, 2] \times$ default gains
Action Timestep λ	$[125, 1000] s^{-1}$

Sim-to-real transfer of robotic control with dynamics randomization
[Peng et al, 2018]

DR for dynamics

Results (real)

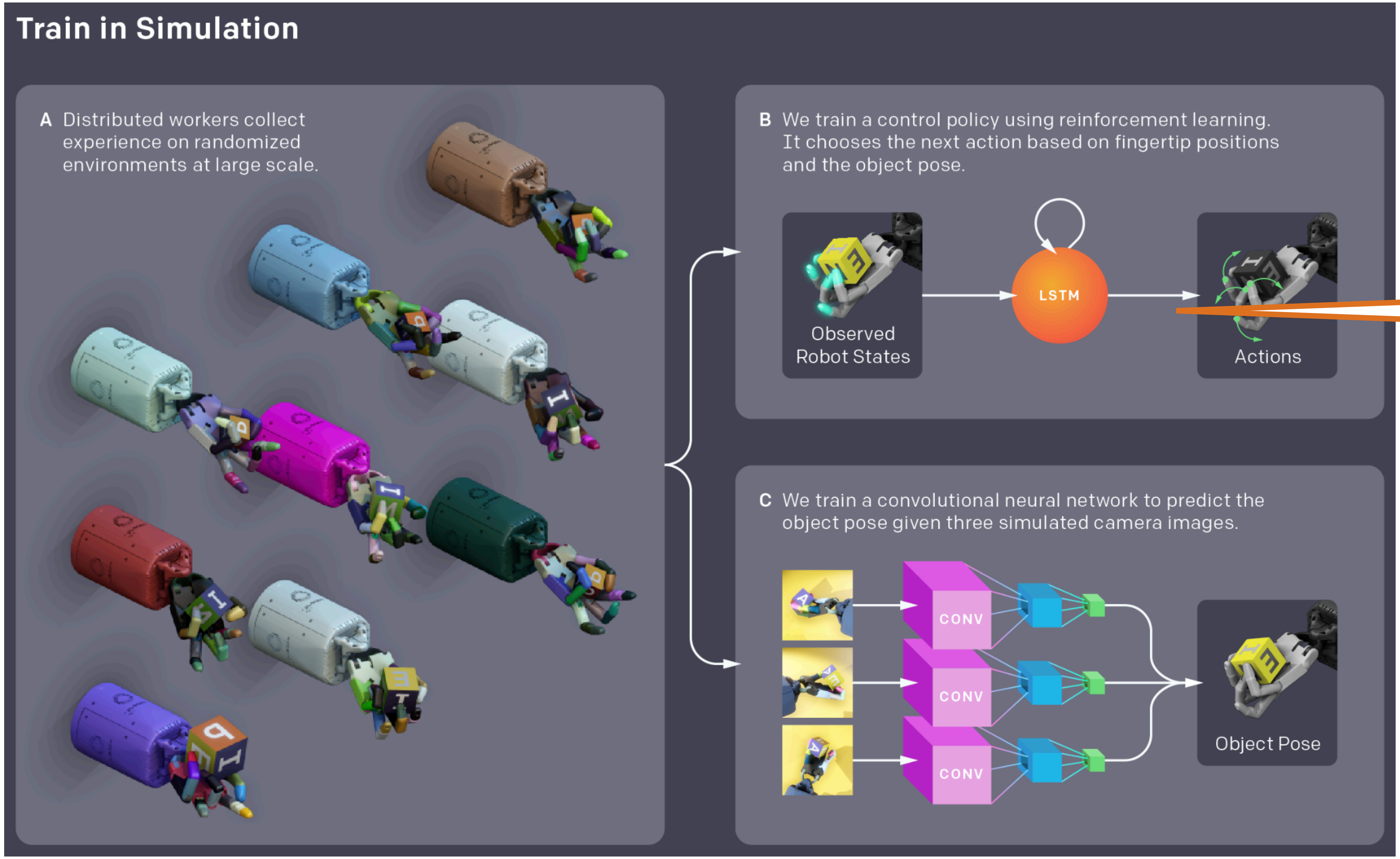


Sim-to-real transfer of robotic control with dynamics randomization
[Peng et al, 2018]

In-hand manipulation

Learning dextrous in-hand manipulation
[OpenAI Robotics, 2018]

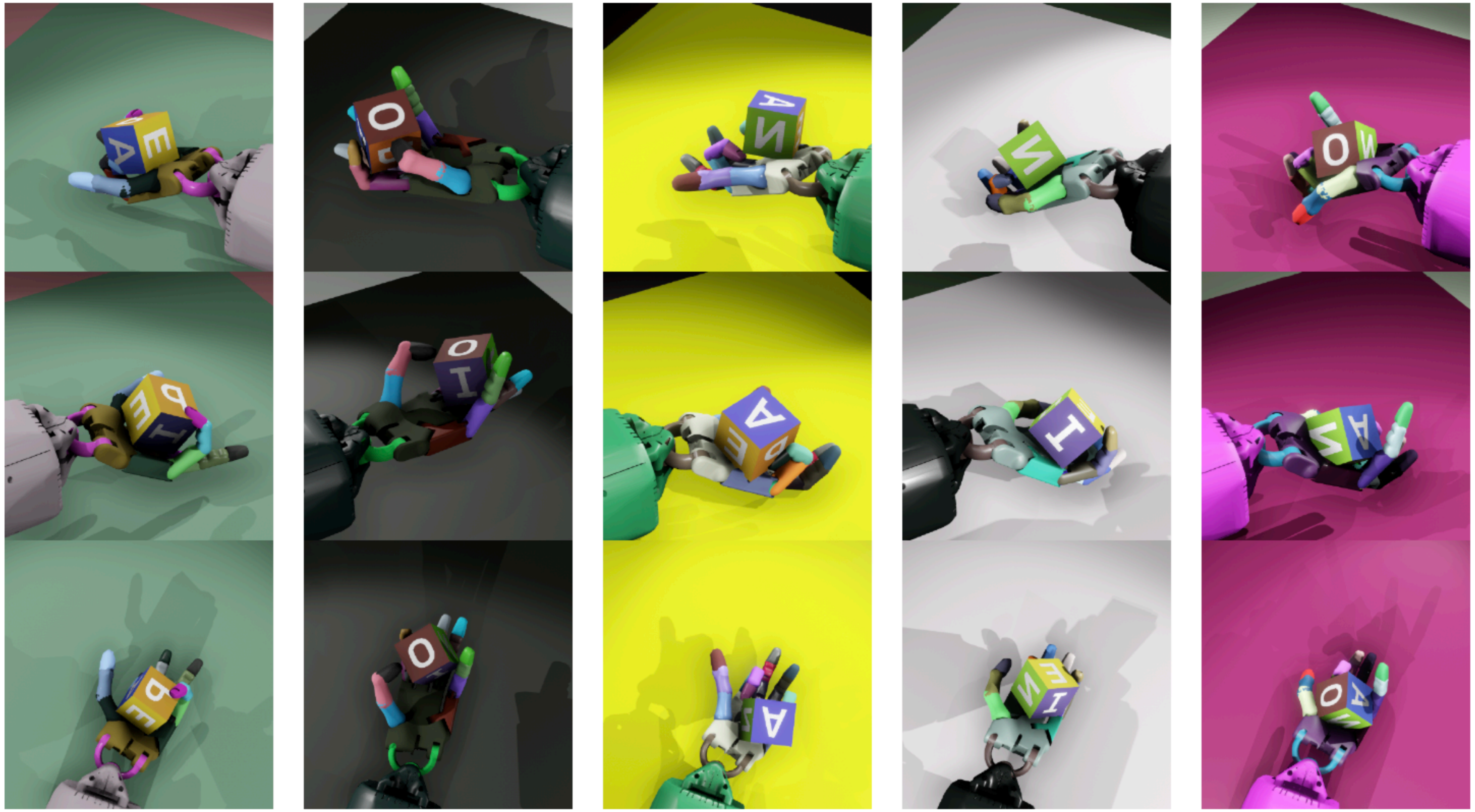
How does it work?



Trained using PPO

Learning dextrous in-hand manipulation
[OpenAI Robotics, 2018]

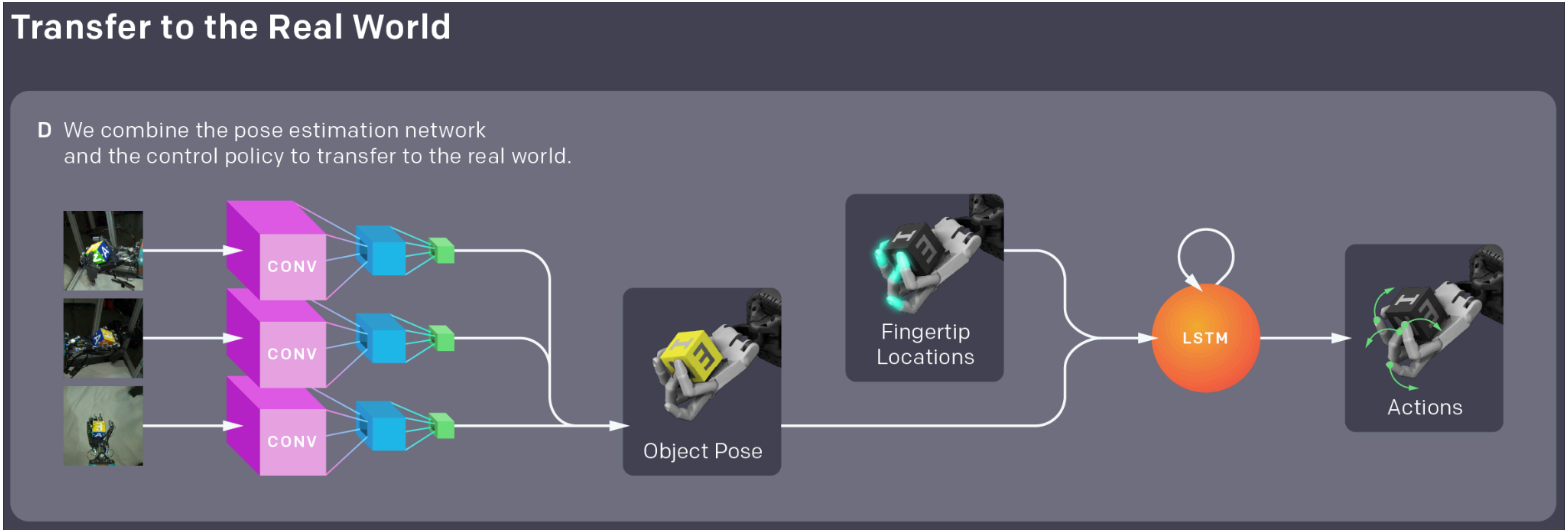
How does it work?



Example training images used for learning to estimate the pose of the block.

Learning dextrous in-hand manipulation
[OpenAI Robotics, 2018]

How does it work?



Learning dextrous in-hand manipulation
[OpenAI Robotics, 2018]

What is randomized?

- Physical parameters
- Correlated and uncorrelated noise applied to policy inputs
- Sensor dropout
- Physics discretization timestep
- Backlash
- Random forces applied to the object
- Visual appearance

What is randomized?

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)		$\mathcal{N}(0, 0.4)$ $\mathrm{m/s^2}$

What is randomized?

Table 9: Vision randomizations.

Randomization type	Range
number of cameras	3
camera position	$\pm$ 1.5 mm
camera rotation	0–3° around a random axis
camera field of view	$\pm$ 1°
robot material colors	RGB
robot material metallic level	5%–25%[17]
robot material glossiness level	0%–100%[17]
object material hue	calibrated hue $\pm$ 1%
object material saturation	calibrated saturation $\pm$ 15%
object material value	calibrated value $\pm$ 15%
object metallic level	5%–15%[17]
object glossiness level	5%–15%[17]
number of lights	4–6
light position	uniform over upper half-sphere
light relative intensity	1–5
total light intensity	0–15[17]
image contrast adjustment	50%–150%
additive per-pixel Gaussian noise	$\pm$ 10%

Domain Randomization Overview

A. History

B. Applications

C. Why does it work?

D. Tools

E. Challenges

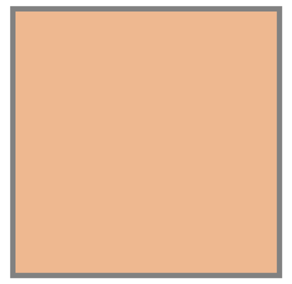
F. Extensions

Why does domain randomization work?

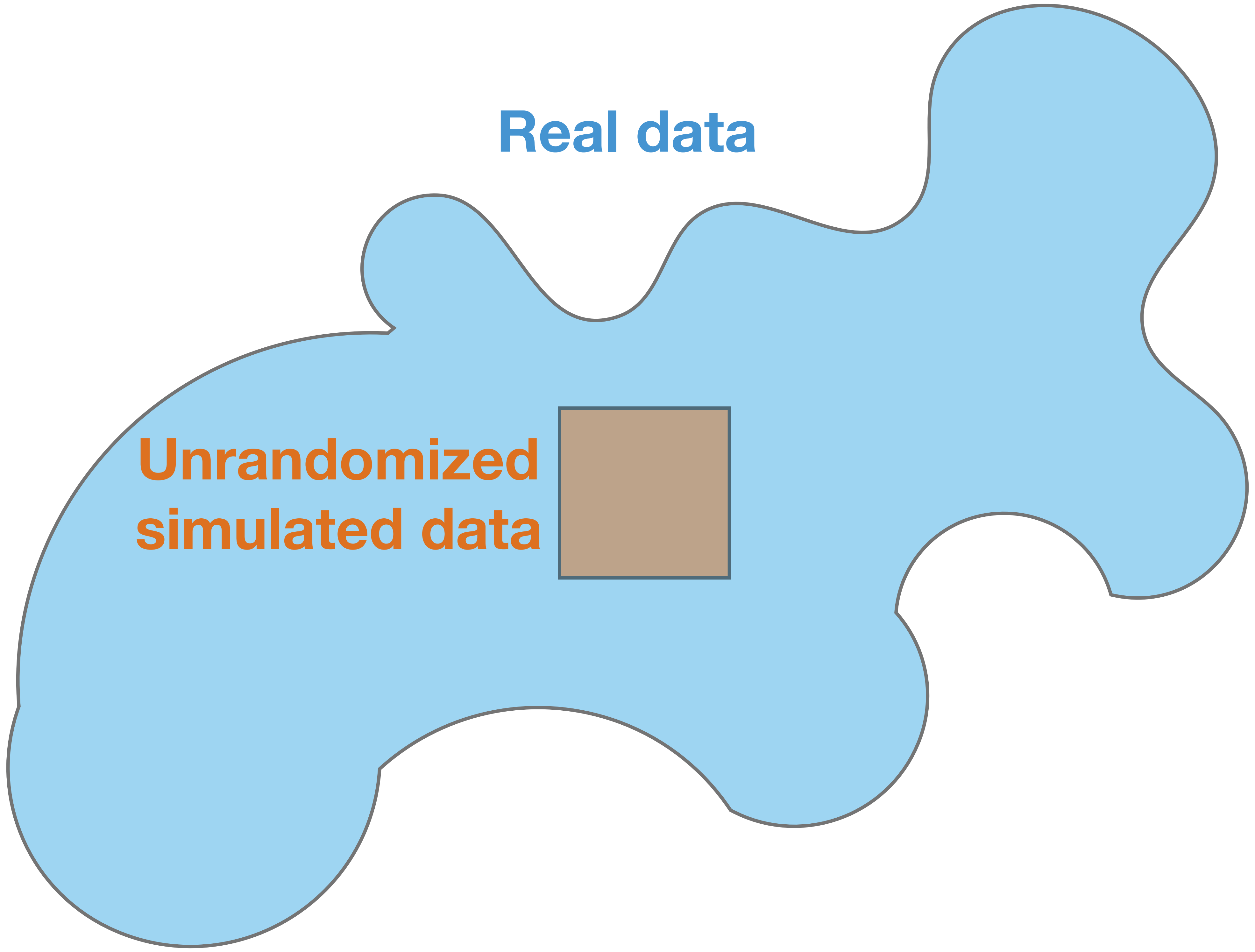
- Intuition 1: training data comes from a **covering distribution**
- Intuition 2: DR is a way of specifying to the model **what to ignore**
- Intuition 3: DR is **meta-learning**

Domain randomization as a covering distribution

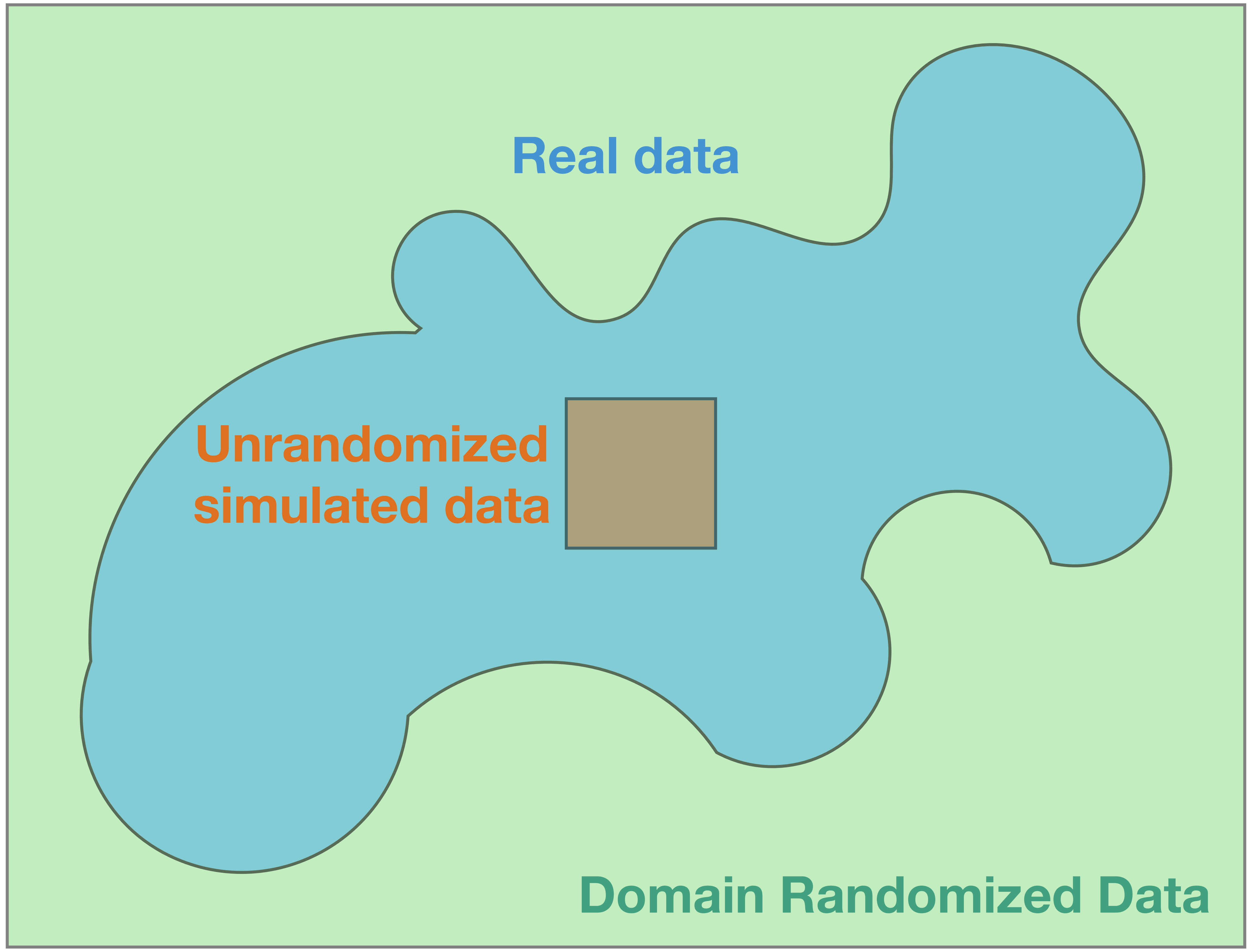
Unrandomized
simulated data



Domain randomization as a covering distribution



Domain randomization as a covering distribution



Problems with this intuition

- In a high-dimensional space, we would need a massive amount of data to truly cover the real data distribution
- Some real-world effects are not modeled in our simulator, e.g., camera distortion, gear backlash

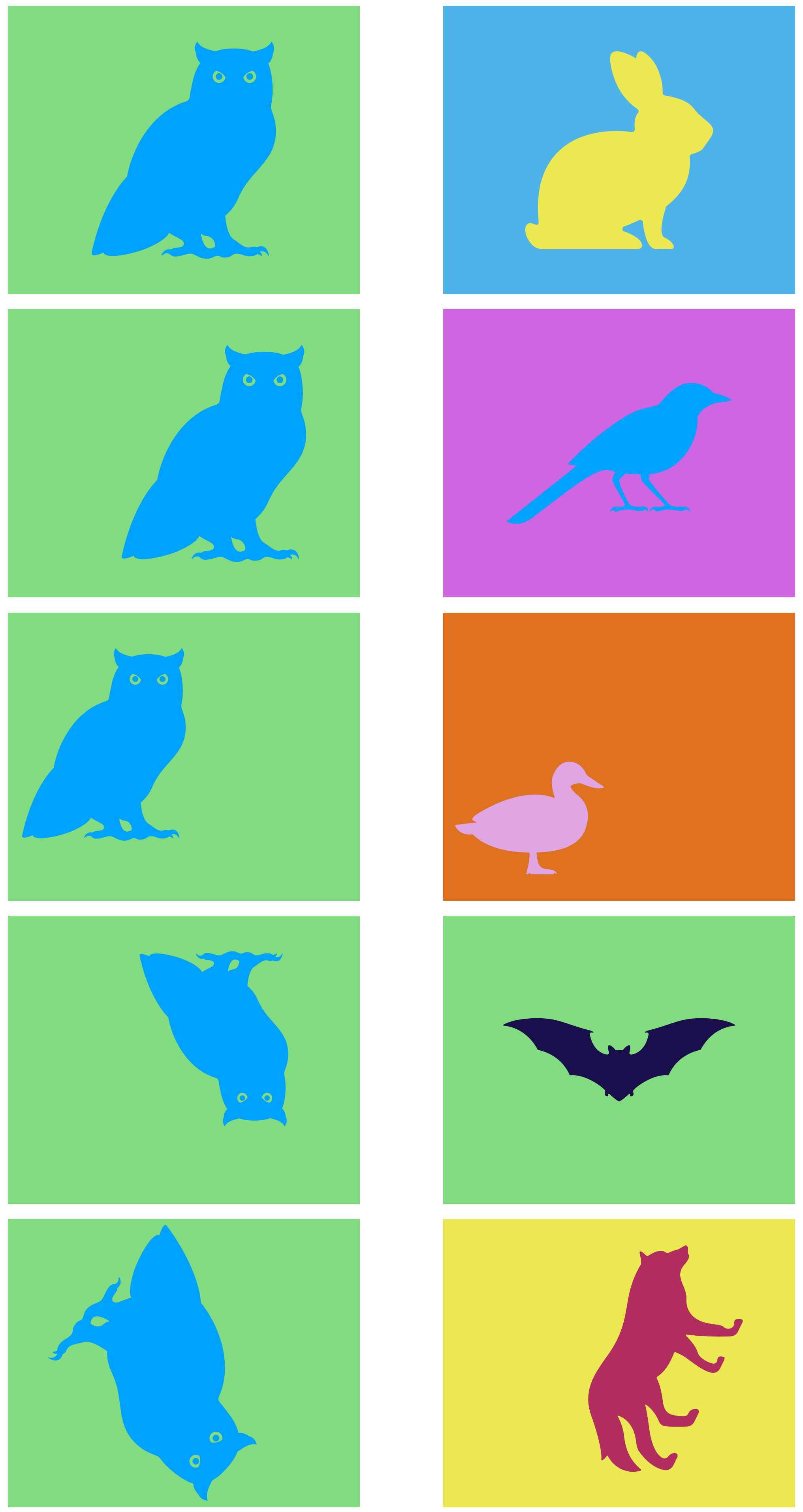
Why does domain randomization work?

- Intuition 1: training data comes from a **covering distribution**
- Intuition 2: DR is a way of specifying to the model **what to ignore**
- Intuition 3: DR is **meta-learning**

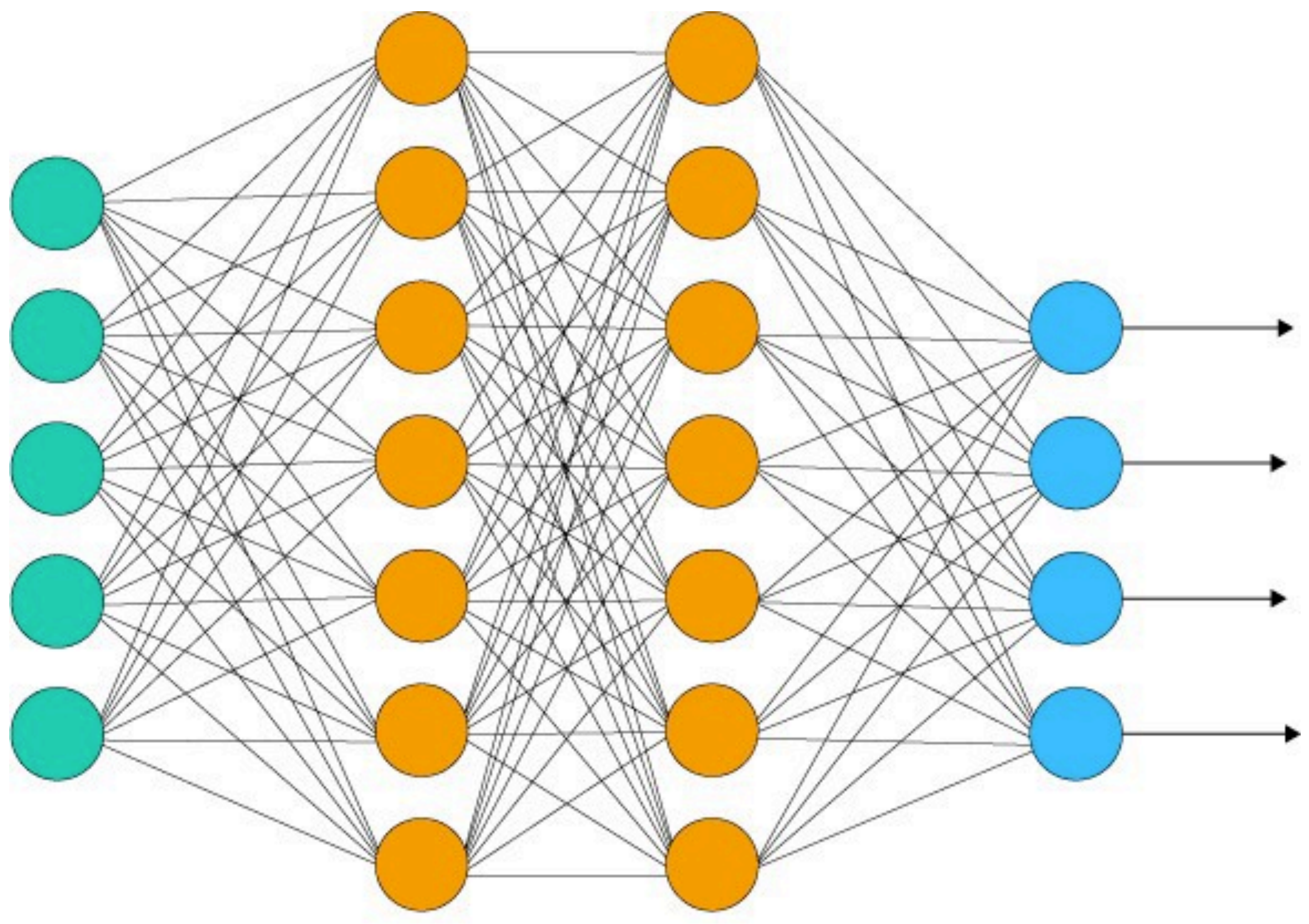
DR tells the model what to ignore

Data

Label 0	Label 1



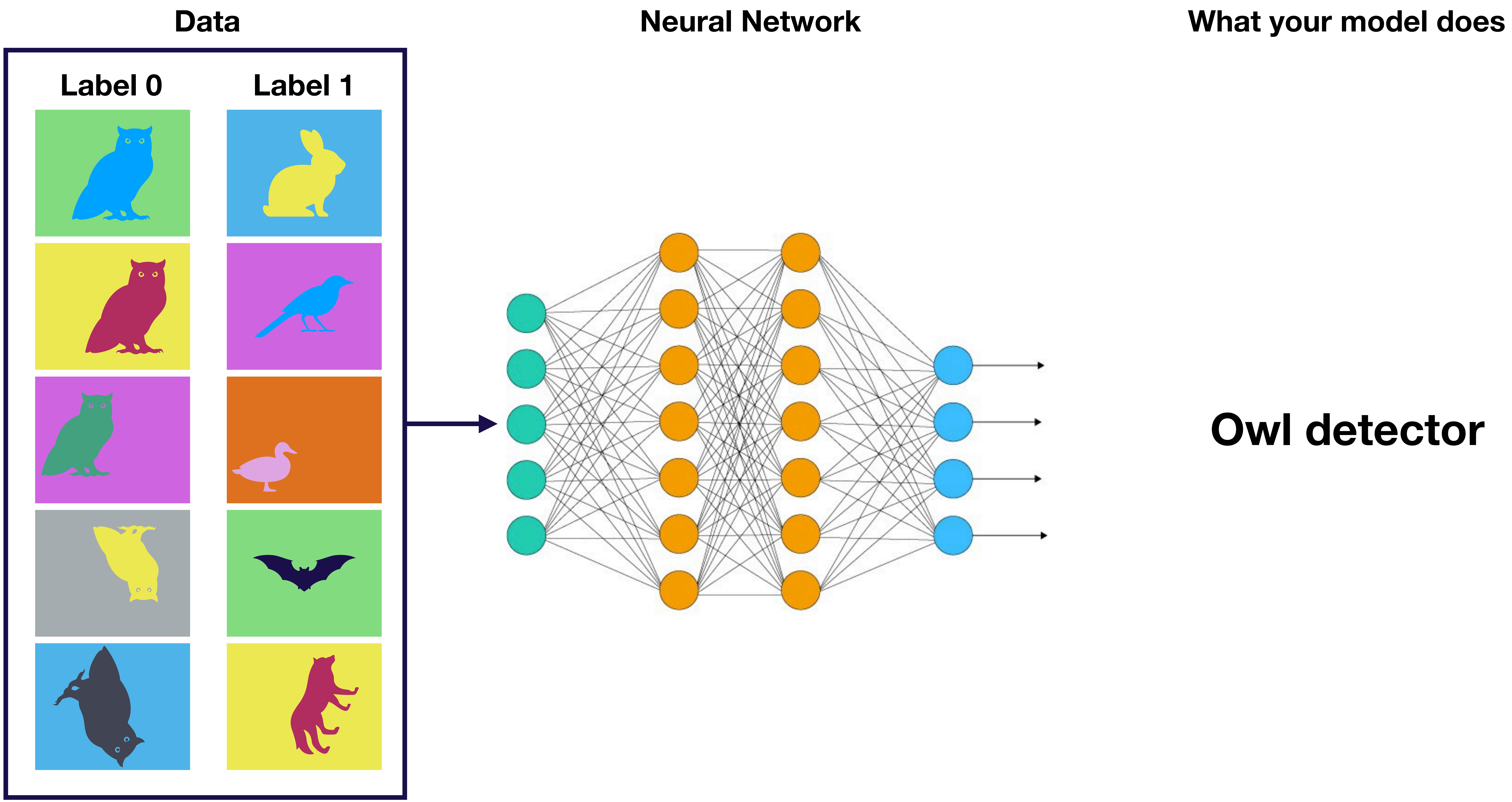
Neural Network



What your model does

Blue owl on green background detector

DR tells the model what to ignore



Why does domain randomization work?

- Intuition 1: training data comes from a **covering distribution**
- Intuition 2: DR is a way of specifying to the model **what to ignore**
- Intuition 3: DR is **meta-learning**

Meta-learning

- Standard ML task:

$$\theta^* = \arg\min_{\theta} \mathscr{L}_{\theta}(D)$$

- Meta-learning task:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{D \sim p(D)} \left[\mathscr{L}_{\theta}(D) \right]$$

Meta-learning

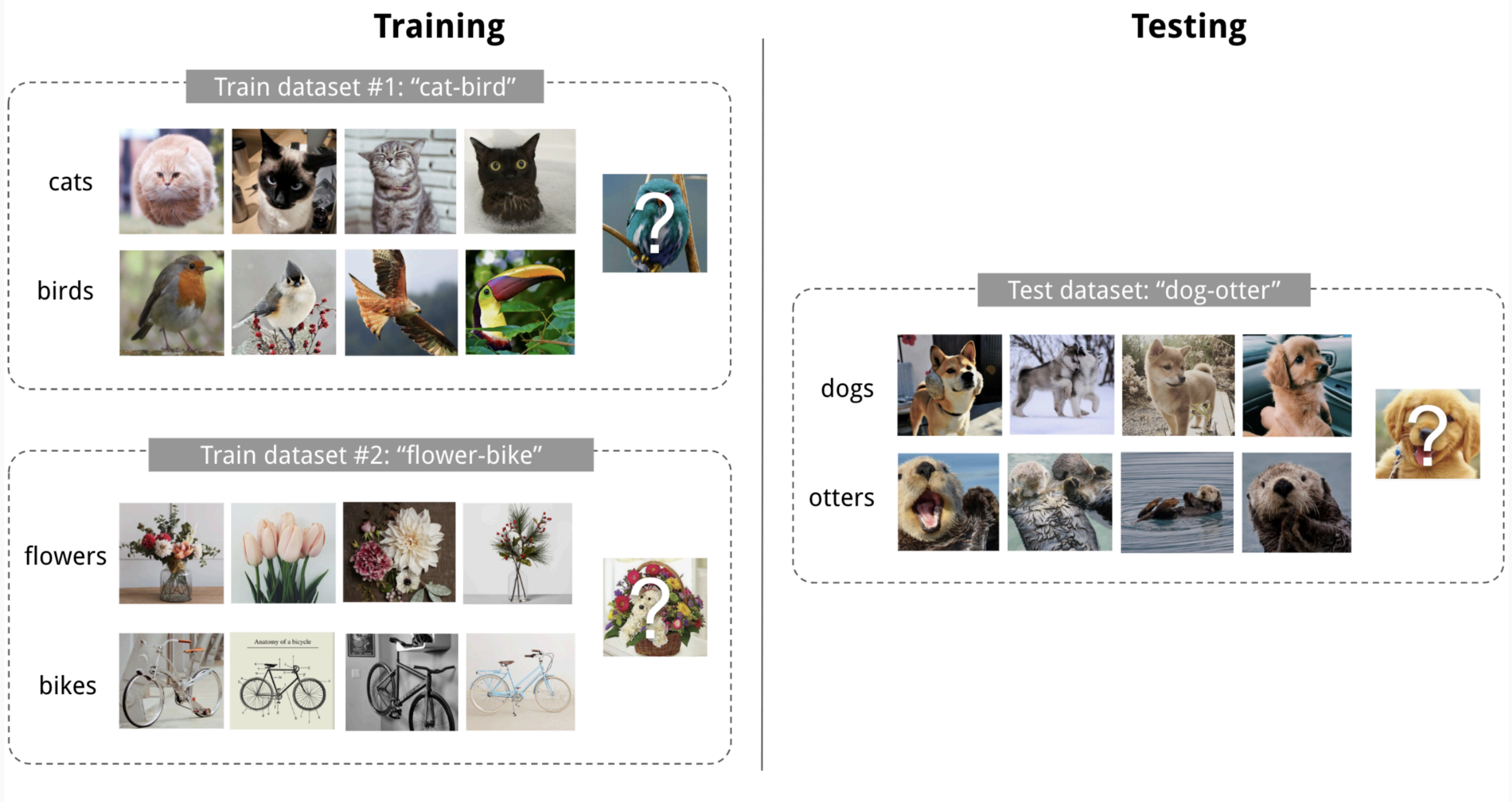


Fig. 1. An example of 4-shot 2-class image classification. (Image thumbnails are from Pinterest)

https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html

Meta-RL: RL^2

- A **task** consists of one or more rollouts in a given environment
- Reset the hidden state of the recurrent policy between each task
- During a task, the hidden state allows the policy to adapt to the environment *(Fast RL)*
- The ability to adapt to new environments is learned by the RL algorithm across all environments *(Slow RL)*

RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning
[Duan et al, 2017]

Meta-RL: RL^2

$$\max_{\theta} \mathbb{E}_M \mathbb{E}_{\tau_M^{(k)}} \left[\sum_{k=1}^{K} R(\tau_M^{(k)}) \mid \mathrm{RLagent}_{\theta} \right]$$

M : sample environment

$\tau_M^{(k)}$: k'th episode in environment M

RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning
[Duan et al, 2017]

Meta-RL: RL^2

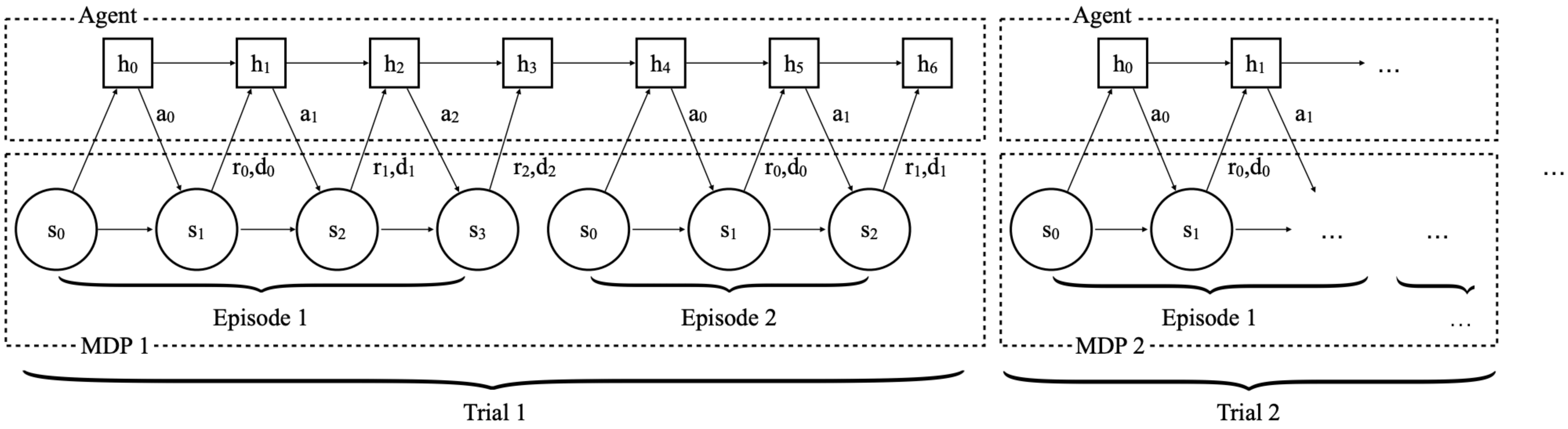


Figure 1: Procedure of agent-environment interaction

RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning

[Duan et al, 2017]

Training an RL^2 agent

$$\max_{\theta} \sum_{M \in M_{\text{train}}} \mathbb{E}_{\tau_M^{(k)}} \left[\sum_{k=1}^{K} R(\tau_M^{(k)}) \mid \text{RLagent}_{\theta} \right]$$

M : sample environment

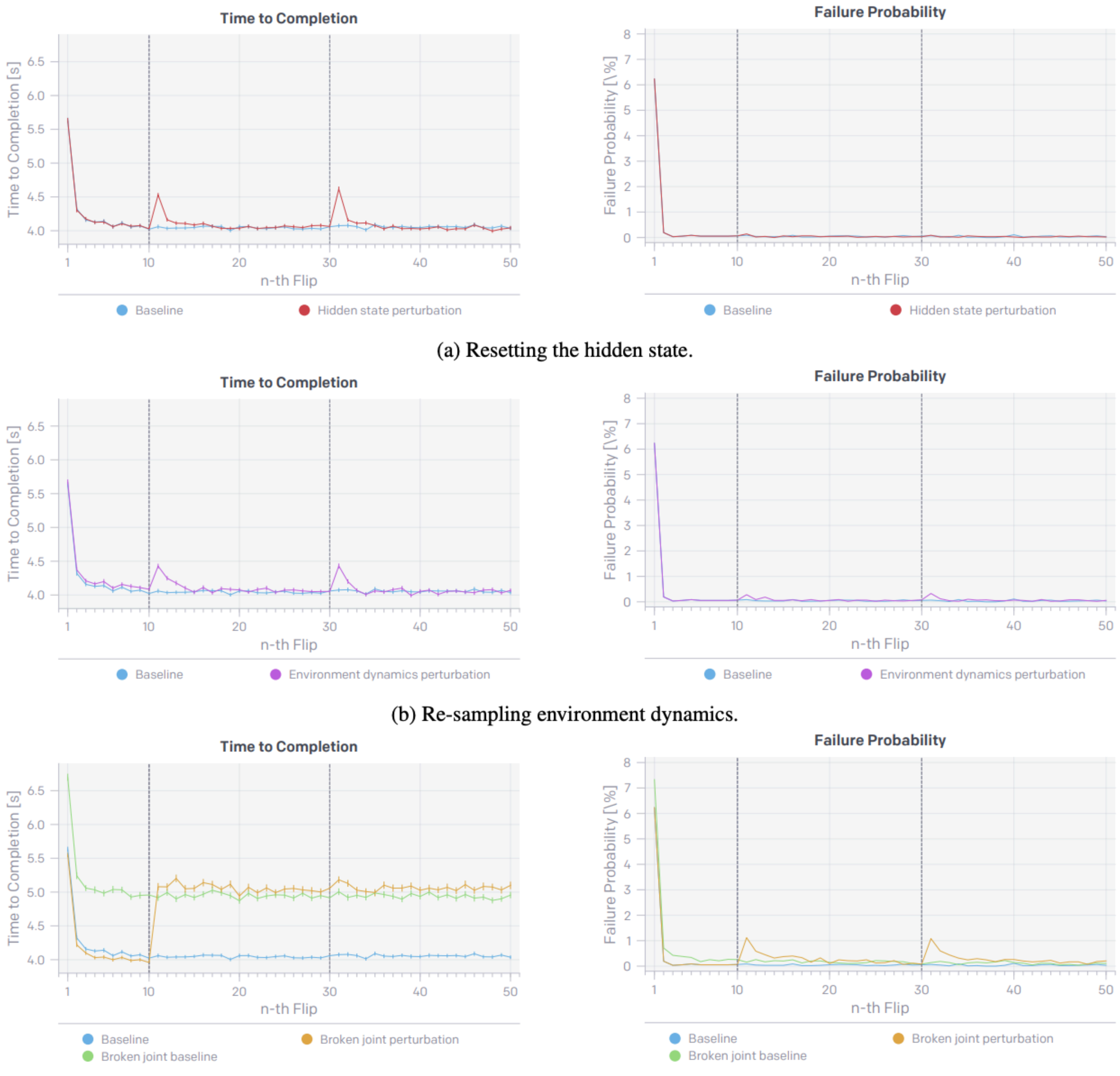
$\tau_M^{(k)}$: k'th episode in environment M

RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning
[Duan et al, 2017]

Domain randomization (dynamics) as meta-learning

- Each set of physics parameters corresponds to an **environment**
- **One** rollout in that environment is a **task**
- During the rollout, the **recurrent state** of the policy allows the policy to **adapt to new physics**

Does a DR-trained policy really adapt?



(a) Resetting the hidden state.

(b) Re-sampling environment dynamics.

(c) Breaking a random joint.

Domain Randomization Overview

A. History

B. Applications

C. Why does it work?

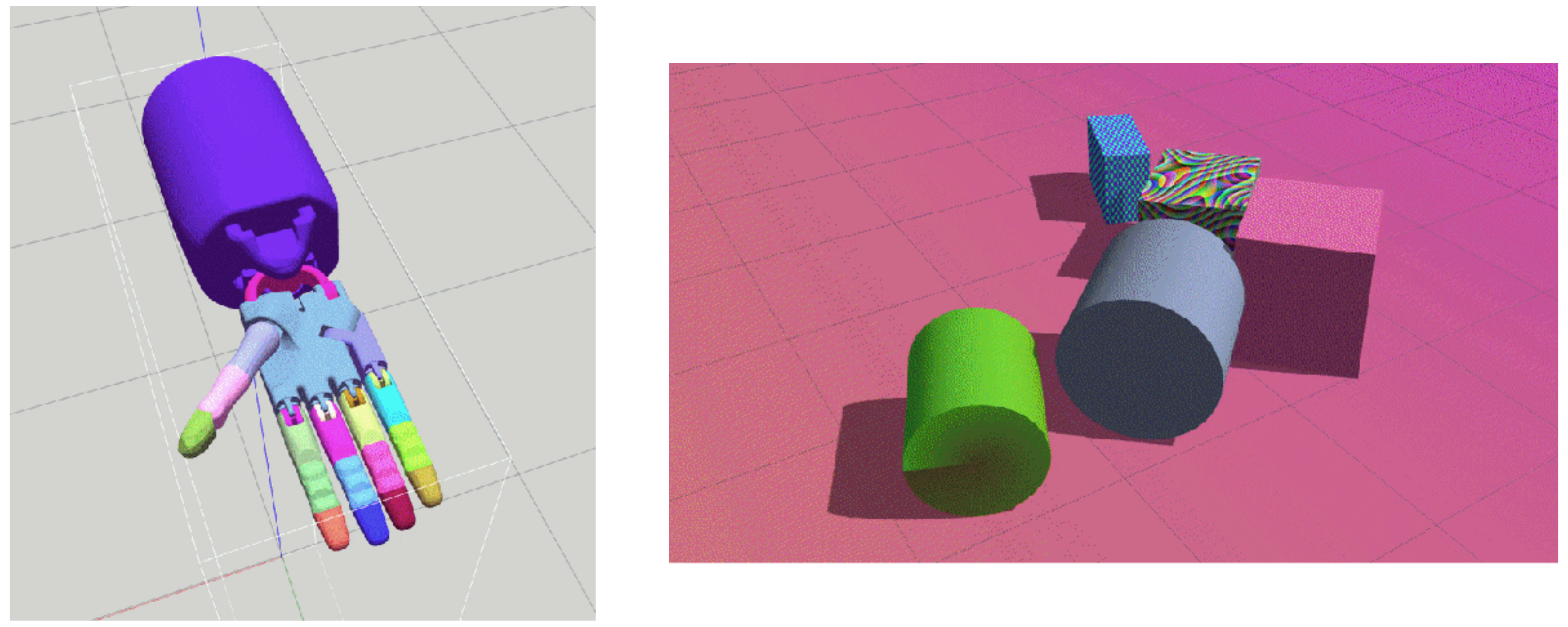
D. Tools

E. Challenges

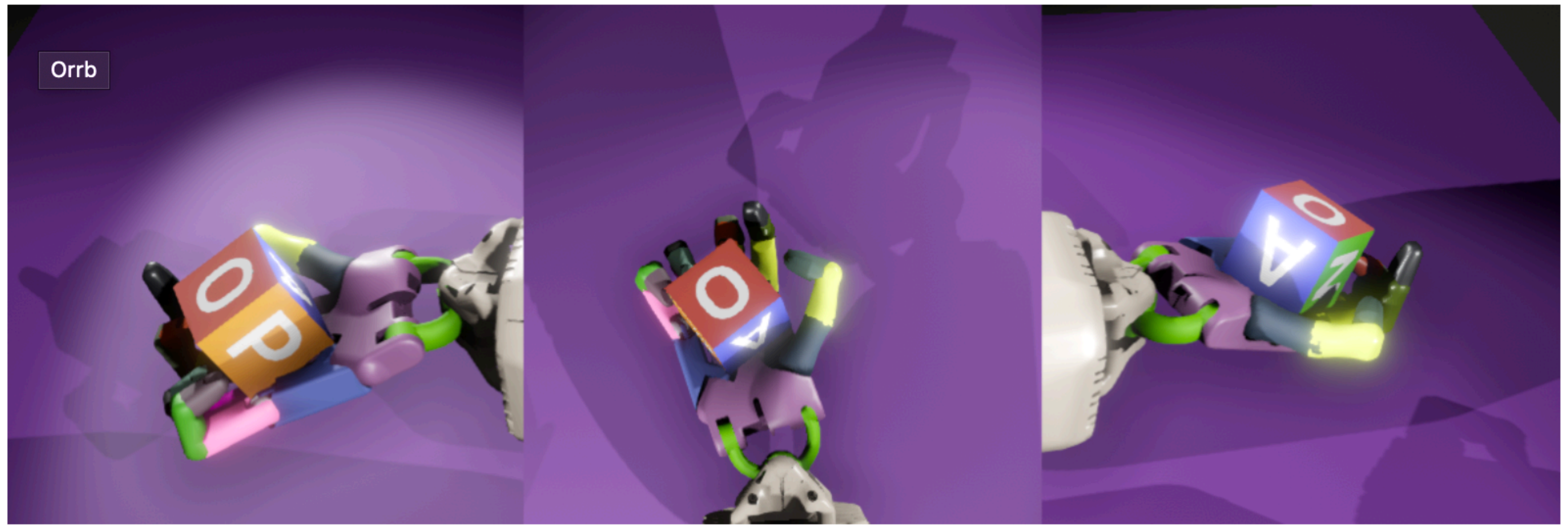
F. Extensions

Tools for performing domain randomization

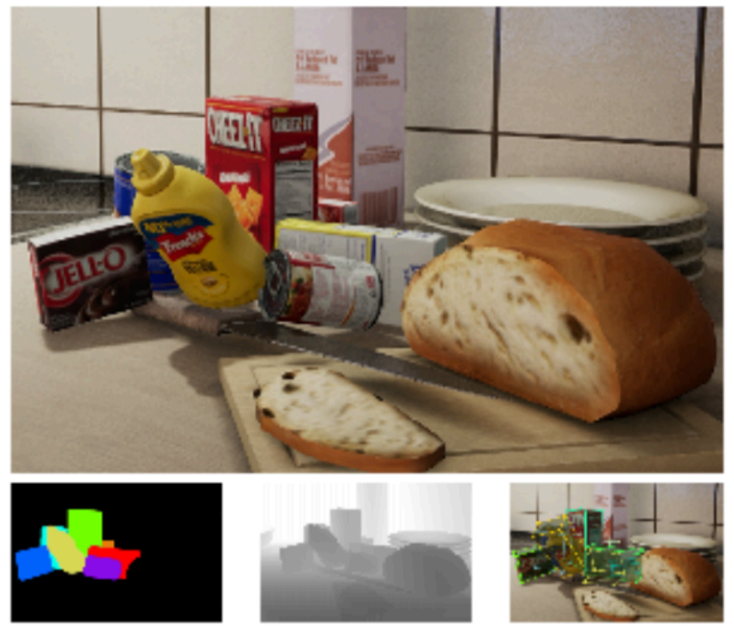
Gazebo Awesome Plugins
github.com/jsbruglie/gap



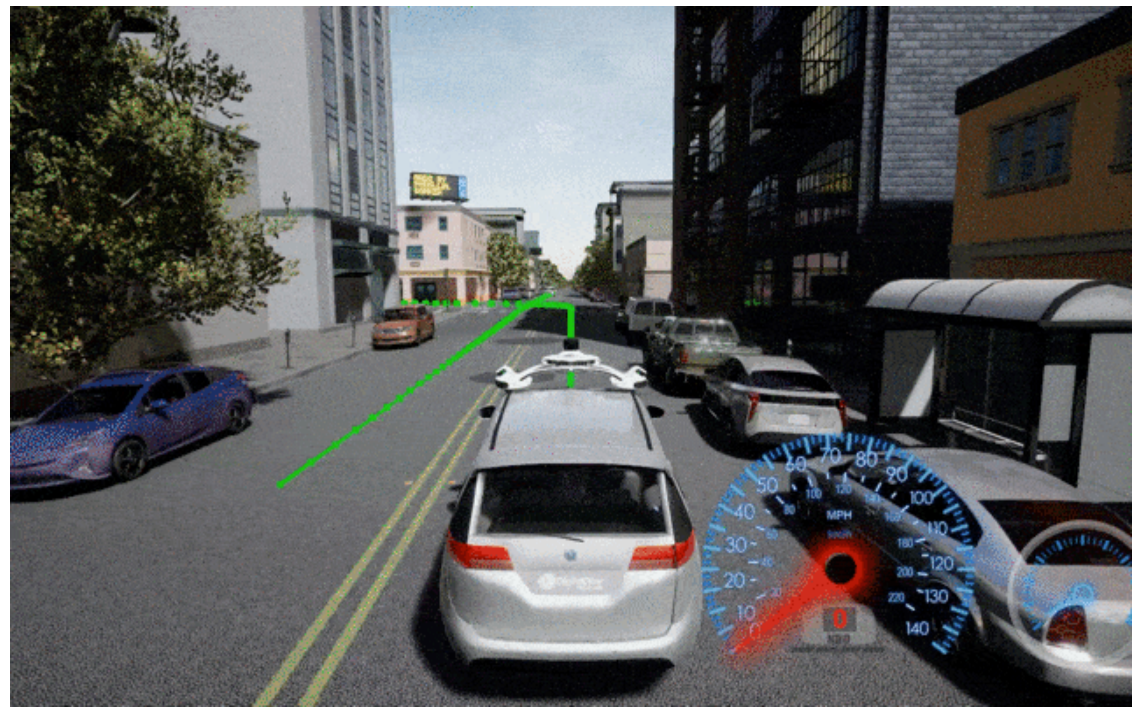
ORRB (Unity)
github.com/openai/orrb



NDDS (Unreal)
github.com/NVIDIA/Dataset_Synthesizer



DeepDrive
github.com/deepdrive/deepdrive



Domain Randomization Overview

A. History

B. Applications

C. Why does it work?

D. Tools

E. Challenges

F. Extensions

How does domain randomization work in practice?

- Build a simulated world
- Calibrate it to the environment
- Design randomizations to “cover” real-world variability
- Train a model and evaluate in real
- Examine failure modes and add randomization

How does domain randomization work in practice?

- Build a simulated world
- Calibrate it to the environment
- Design randomizations to “cover” real-world variability
- Train a model and evaluate in real
- Examine failure modes and add randomization

Manual process: 3D modeling, designing procedural pipelines

How does domain randomization work in practice?

- Build a simulated world
- Calibrate it to the environment
- Design randomizations to “cover” real-world variability
- Train a model and evaluate in real
- Examine failure modes and add randomization

System ID

How does domain randomization work in practice?

- Build a simulated world
- Calibrate it to the environment
- Design randomizations to “cover” real-world variability
- Train a model and evaluate in real
- Examine failure modes and add ran

What to randomize?
Judgement + trial & error

How does domain randomization work in practice?

- Build a simulated world
- Calibrate it to the environment
- Design randomizations to “cover” real-world variability
- Train a model and evaluate in real
- Examine failure modes and add ran

How much to randomize? As much as possible

How does domain randomization work in practice?

- Build a simulated world
- Calibrate it to the environment
- Design randomizations to “cover” re
- Train a model and evaluate in real
- Examine failure modes and add randomization

How to figure out what randomizations to add?

Challenges in applying domain randomization

- Building simulations is manual and time consuming
- Deciding what parameters to randomize requires judgment
- Randomizing parameters as much as possible may not be optimal

Domain Randomization Overview

A. History

B. Applications

C. Why does it work?

D. Tools

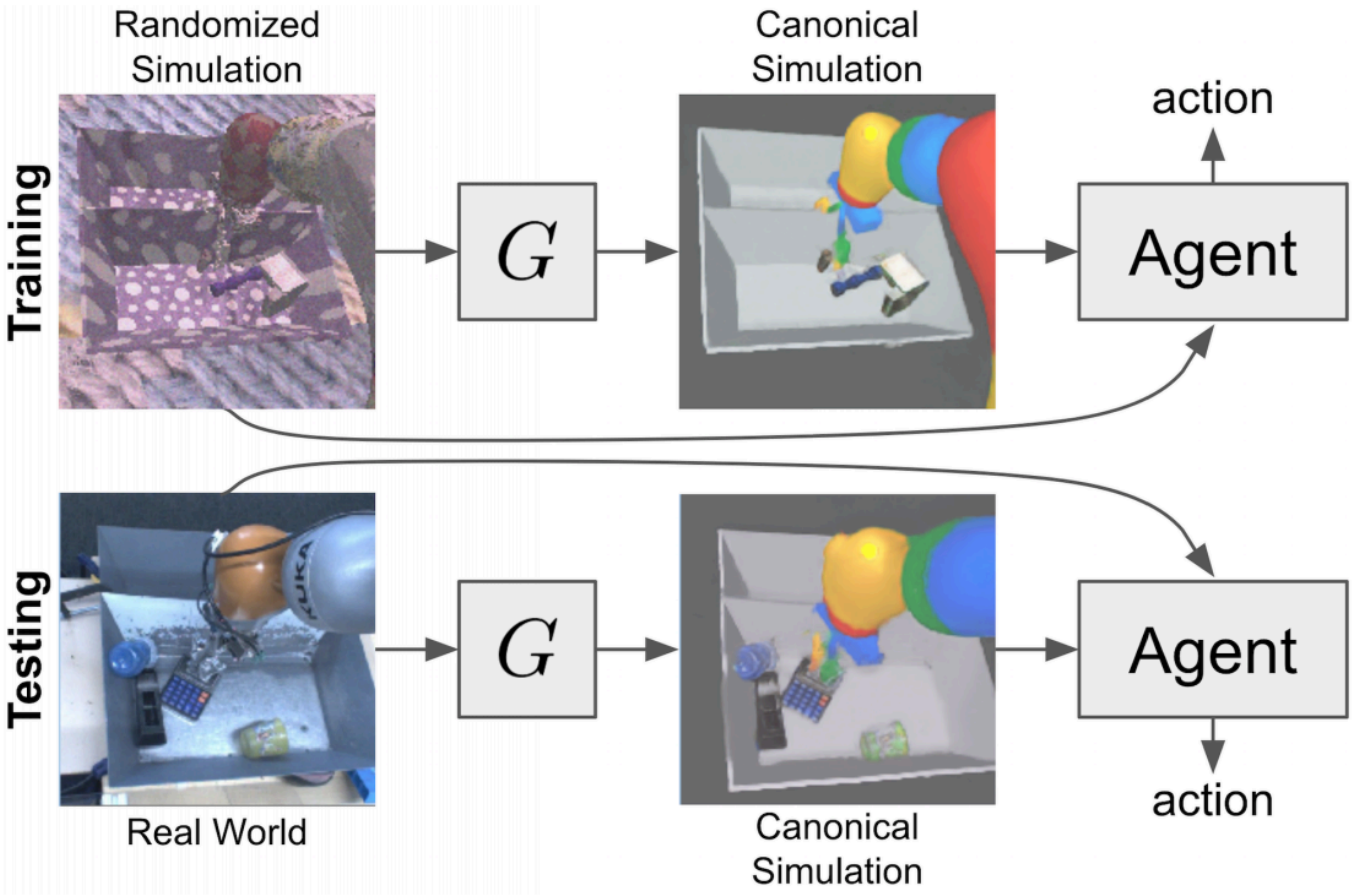
E. Challenges

F. Extensions

Extensions of domain randomization

- **Network architectures better suited to transfer**
- Match the simulator to real data
- Use real data to improve task performance
- Automatically surface hard cases in the simulator
- Allow the model to perform well on a wider range of simulations

Randomized-to-Canonical Adaptation Networks



Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks
[Stephen James et al, 2019]

Randomized-to-Canonical Adaptation Networks

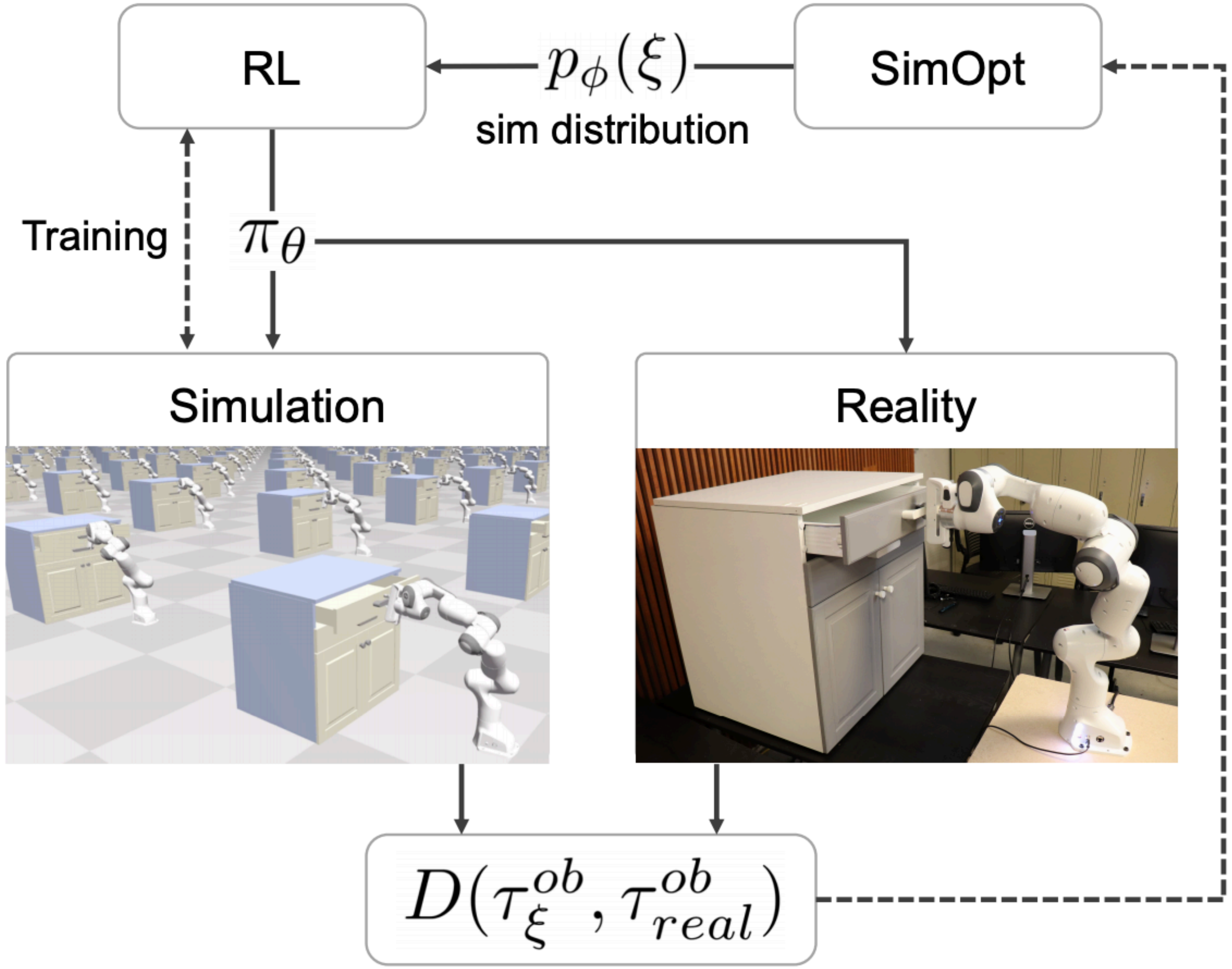
QT-Opt Data Source	Offline Real Grasps	Performance In Sim	Performance In Real	Online Real Grasps	Performance In Real
Real	580,000	-	87%	+5,000 +28,000	85% 96%
Canonical Sim	0	99%	21%	+5,000	30%
Mild Randomization	0	98%	37%	+5,000	85%
Medium Randomization	0	98%	35%	+5,000	77%
Heavy Randomization	0	98%	33%	+5,000 +28,000	85% 92%
RCAN	**0**	99%	**70%**	+5,000 +28,000	**91%** **94%**

Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks
[Stephen James et al, 2019]

Extensions of domain randomization

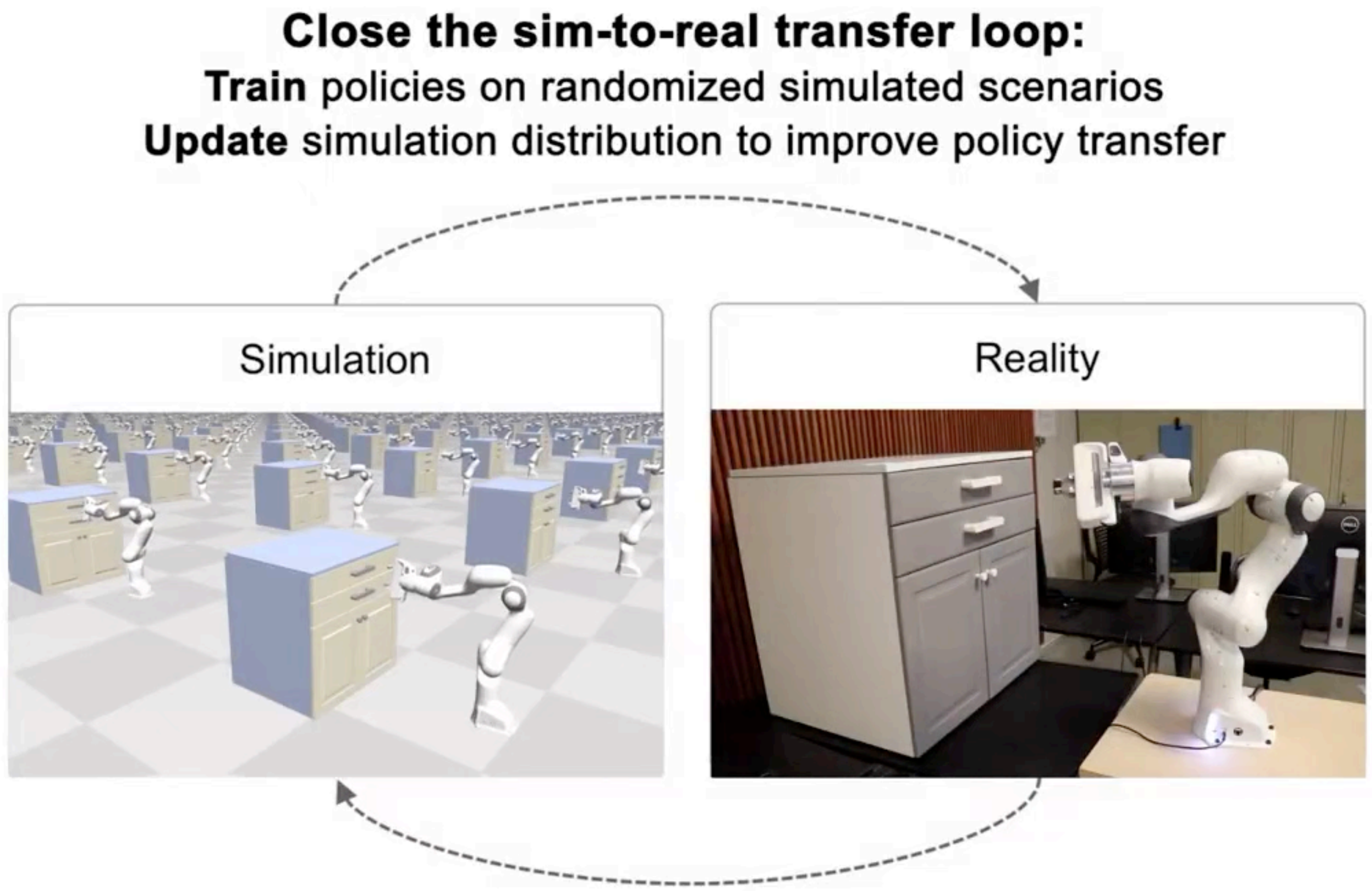
- Network architectures better suited to transfer
- **Match the simulator to real data**
- Use real data to improve task performance
- Automatically surface hard cases in the simulator
- Allow the model to perform well on a wider range of simulations

SimOpt



Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience
[Yevgen Cehbotar et al, 2019]

SimOpt



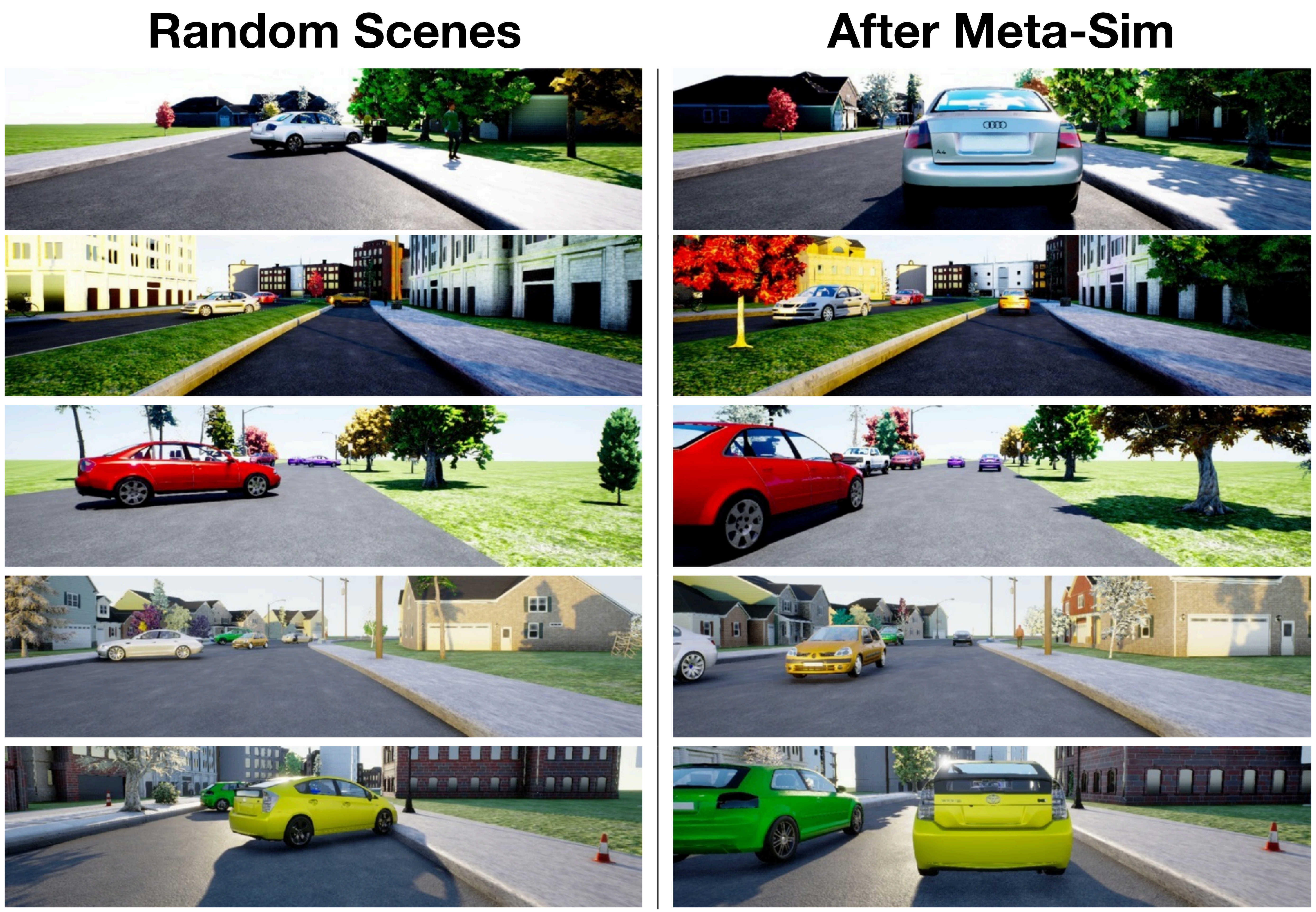
Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience
[Yevgen Cehbotar et al, 2019]

Meta-Sim

- Generating realistic randomization distributions is hard

- You end up with scenes like the left

- **Goal:** use some real data to make the scenes realistic



Meta-Sim: Learning to Generate Structured Datasets
[Amlan Kar et al, 2019]

Generating complex scenes: scene graph

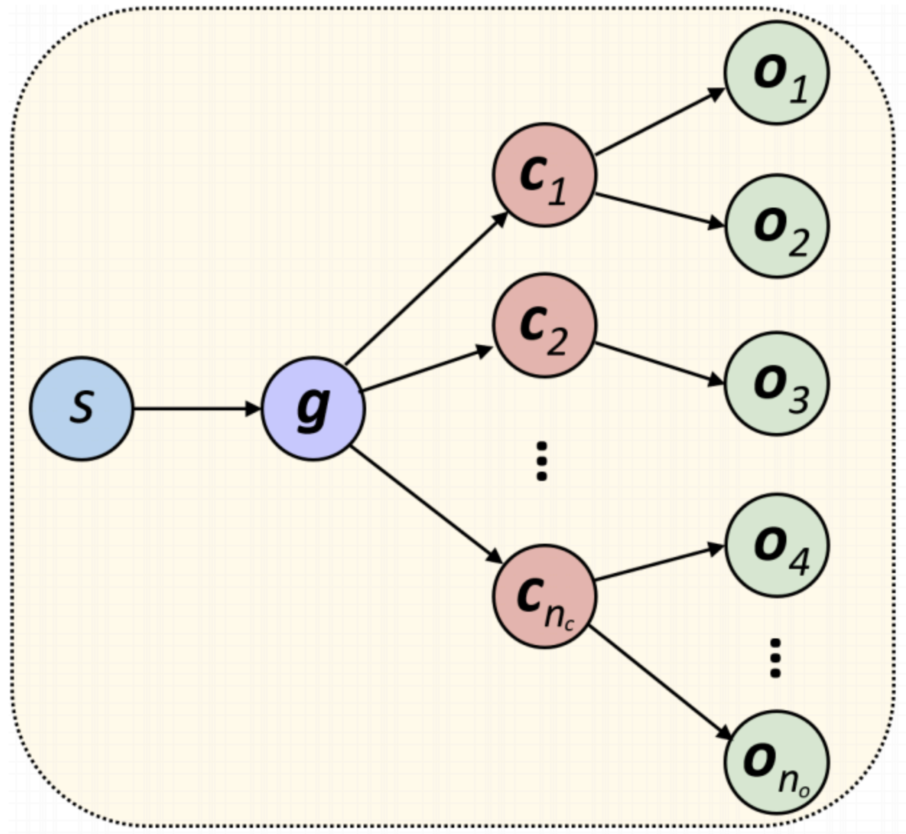
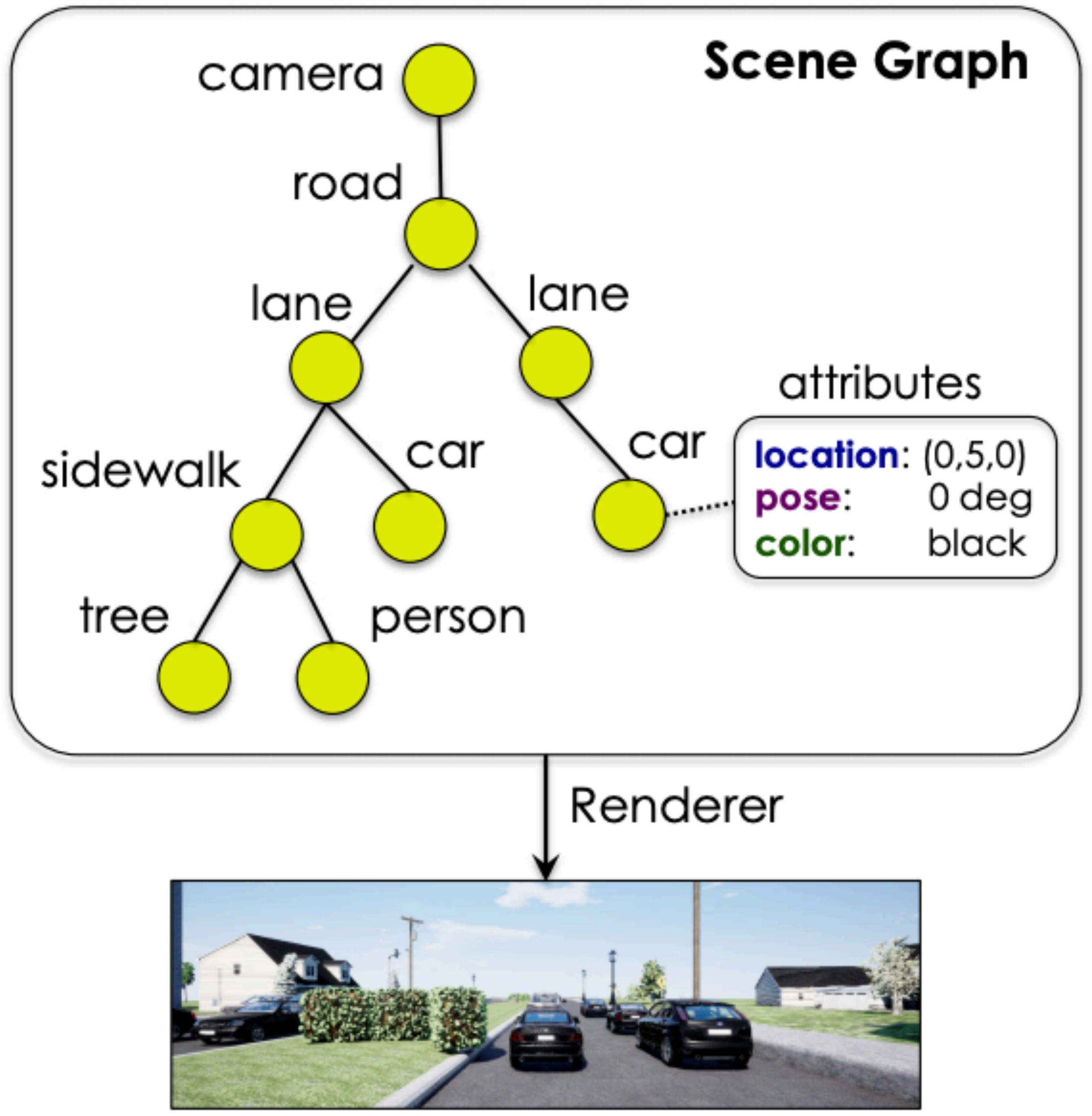
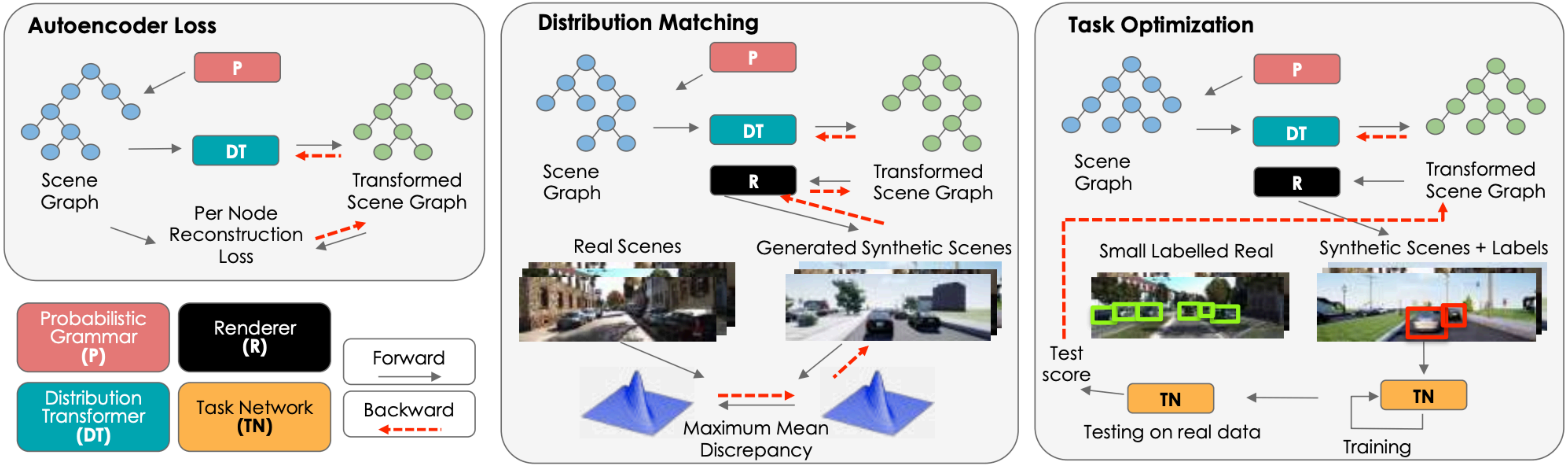


Fig. 1: Probabilistic relationship among different components of SDR. The scenario (s) determines the global parameters ($\mathbf{g}$), which govern the context splines ($\mathbf{c}_i$), upon which the objects ($\mathbf{o}_j$) are placed. The context splines capture the structure of the scene. The image is rendered from these parameters, splines, and objects.



Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data
[Aayush Prakash et al, 2018]

Meta-Sim



Meta-Sim: Learning to Generate Structured Datasets

[Amlan Kar et al, 2019]

Extensions of domain randomization

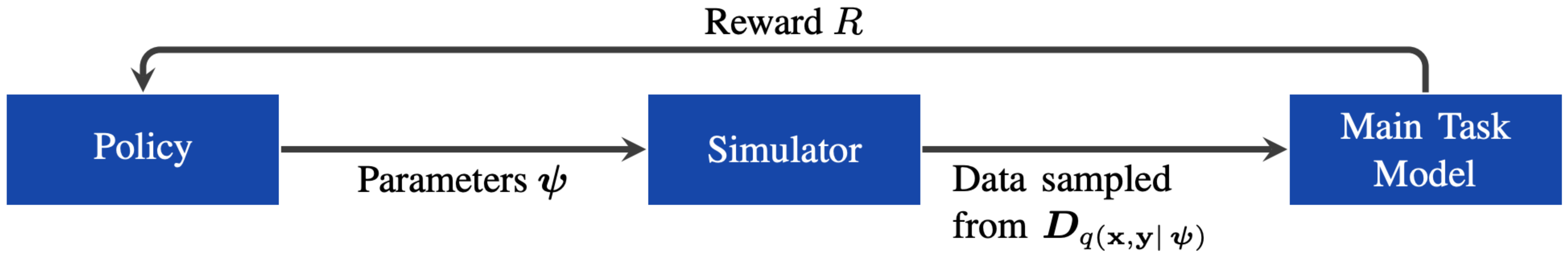
- Network architectures better suited to transfer
- Match the simulator to real data
- **Use real data to improve task performance**
- Automatically surface hard cases in the simulator
- Allow the model to perform well on a wider range of simulations

Learning to Simulate

- Key idea: instead of manually tuning the distribution of sim params, use **meta-learning** to find a **parameter distribution** such that a model trained on that distribution achieves **high accuracy on the validation set**

Learning to Simulate
[Nataniel Ruiz et al, 2019]

Learning to Simulate



- Say we have a distribution of simulator parameters $\psi_k \sim p(\psi)$
- We can compute a **reward R** for ψ_k: sample data from simulators in the distribution, train a model on them, and evaluate its performance on real data
- Key idea: parameterize the space of all possible ψ_k. Train a **policy** π_ω. Actions are ψ_k. Train π_ω using reinforcement learning

Learning to Simulate
[Nataniel Ruiz et al, 2019]

SPOTA

- Key idea: provide a quantitative measure of 'overfitting' to our sim distribution so we can perform early stopping
- Gist of the approach: sample previously-unseen physics parameters η^k. See how much worse our policy performs on them than one trained on them

Simulation-Based Policy Optimization with Transferability Assessment
[Fabio Muratore et al, 2018]

Extensions of domain randomization

- Network architectures better suited to transfer
- Match the simulator to real data
- Use real data to improve task performance
- **Automatically surface hard cases in the simulator**
- Allow the model to perform well on a wider range of simulations

Active Domain Randomization

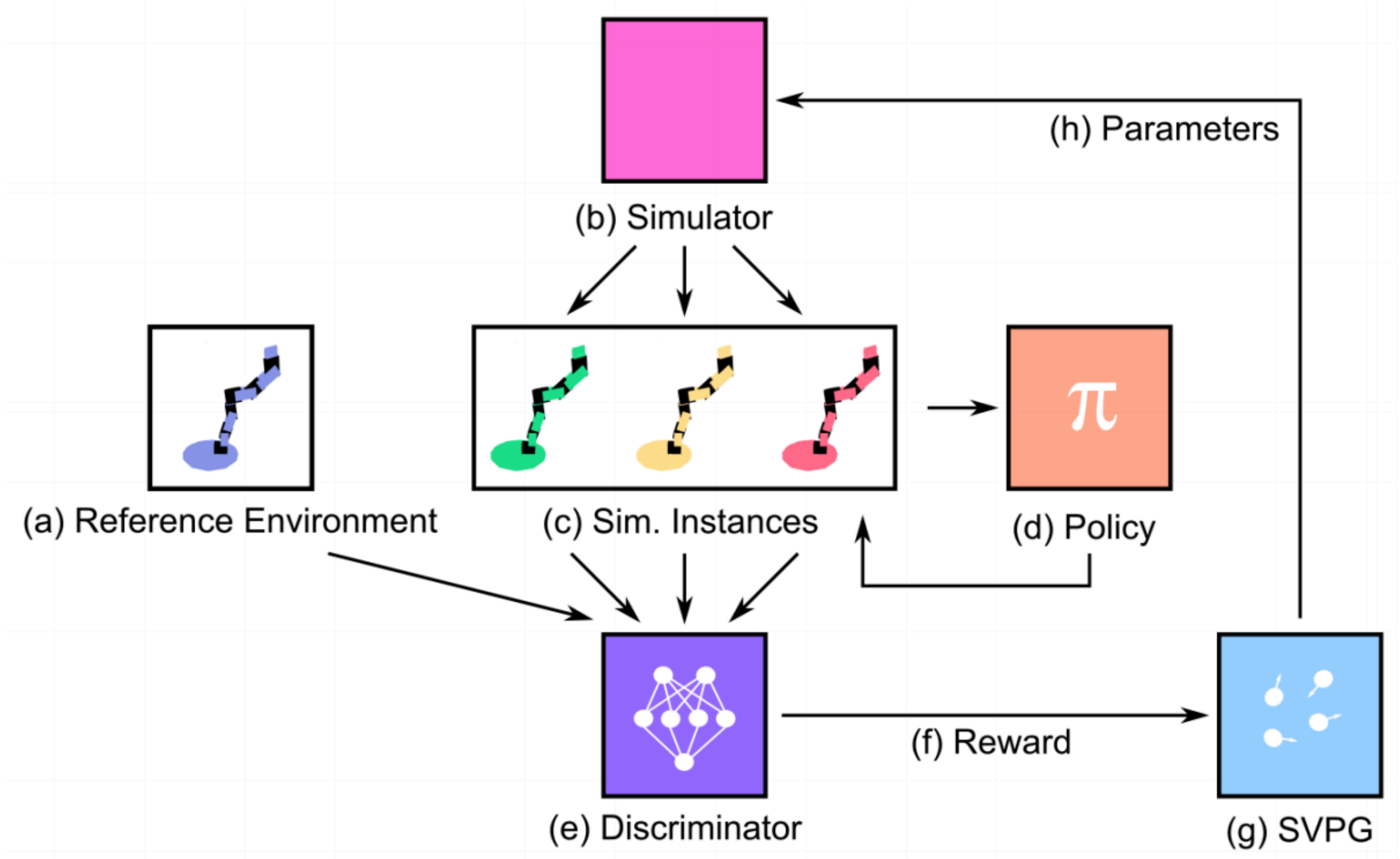
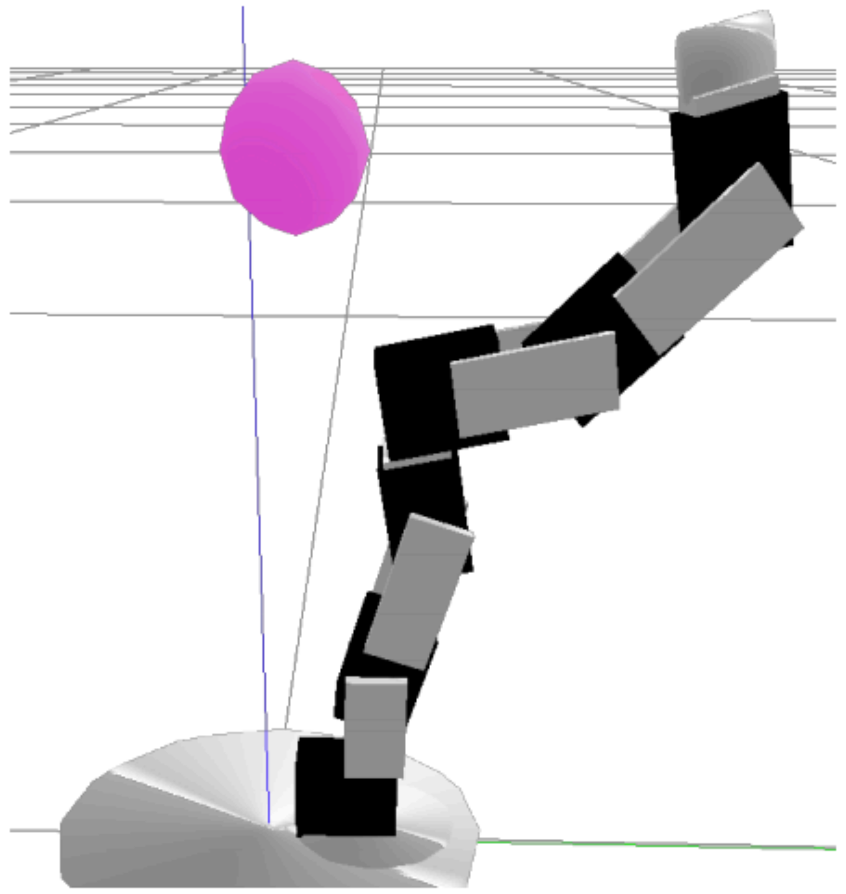
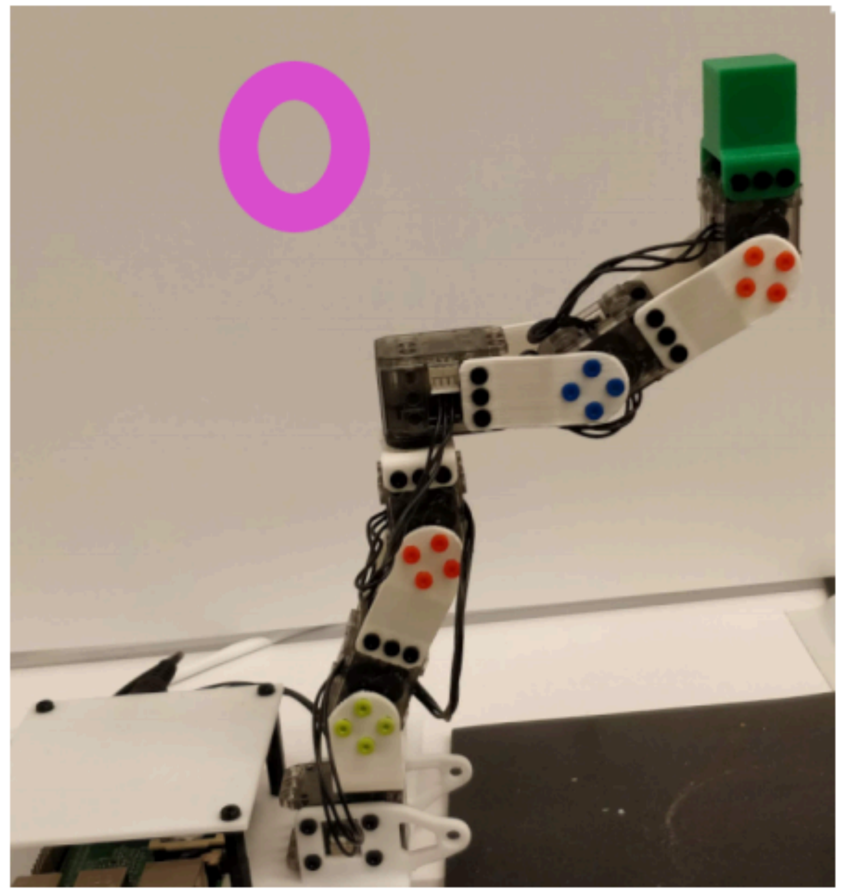


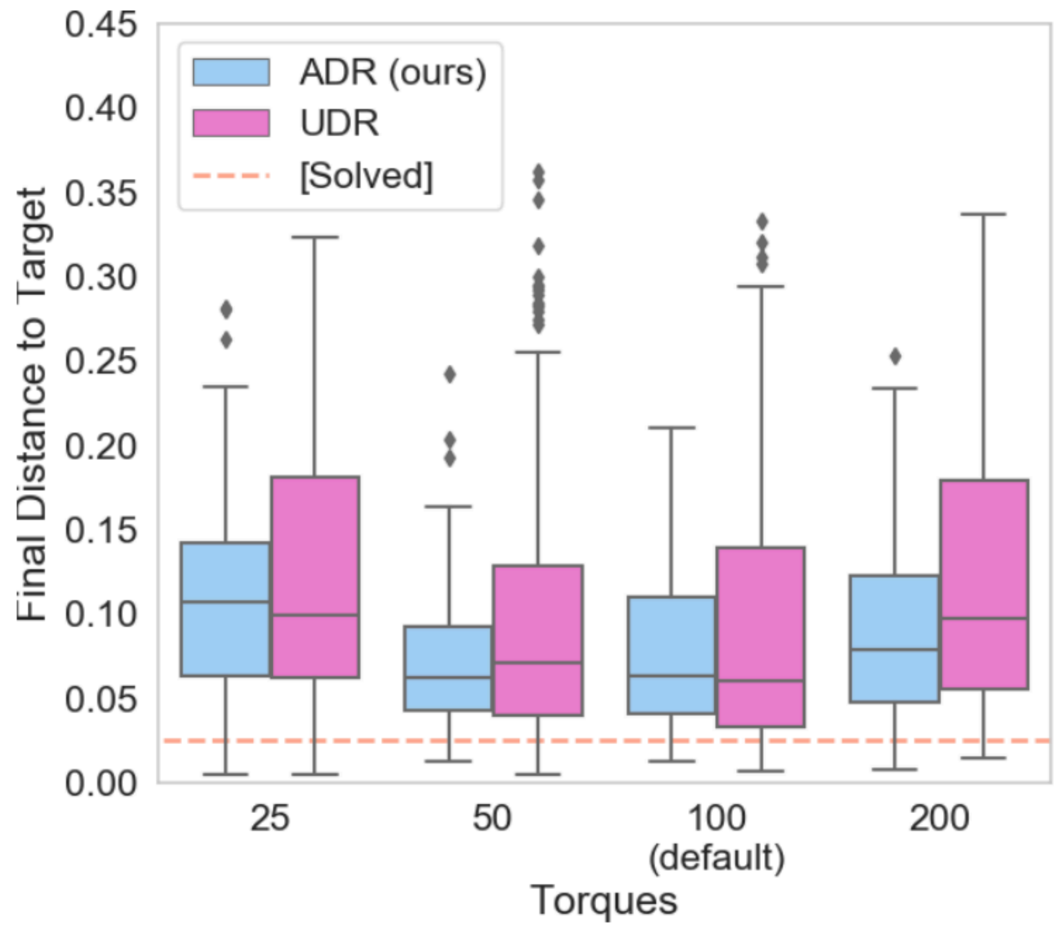
Figure 1: ADR proposes **randomized environments** (c) or simulation instances from a **simulator** (b) and rolls out an **agent policy** (d) in those instances. The **discriminator** (e) learns a **reward** (f) as a proxy for environment difficulty by distinguishing between rollouts in the **reference environment** (a) and randomized instances, which is used to train **SVPG particles** (g). The particles propose a diverse set of environments, trying to find the environment **parameters** (h) that are currently causing the agent the most difficulty.

Active Domain Randomization
[Bhairav Mehta et al, 2019]

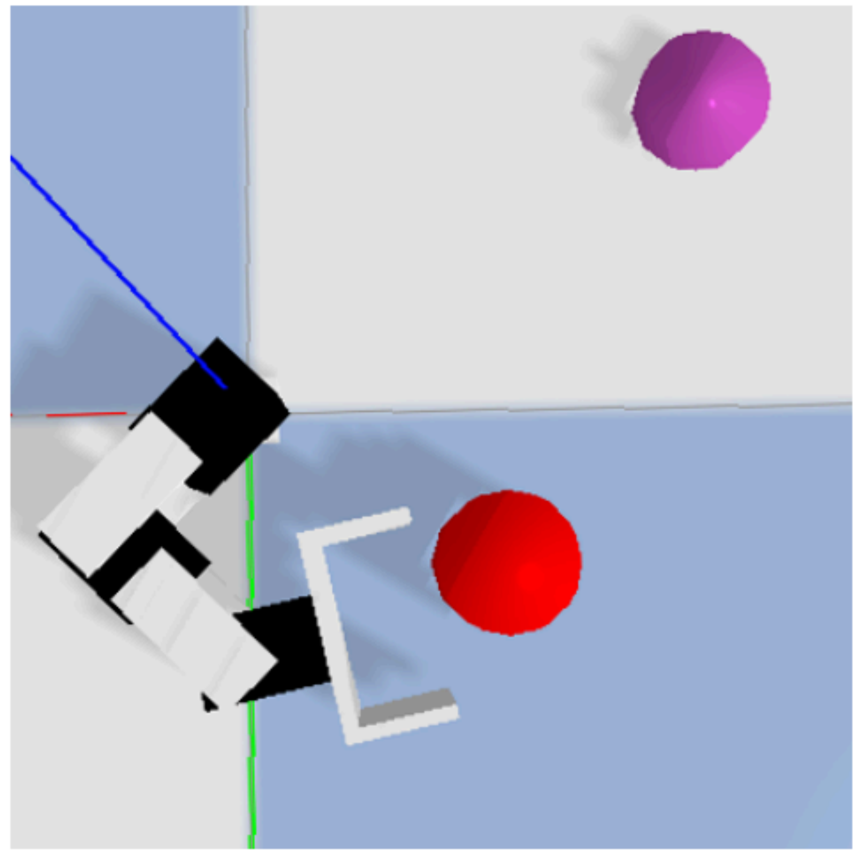
Active Domain Randomization

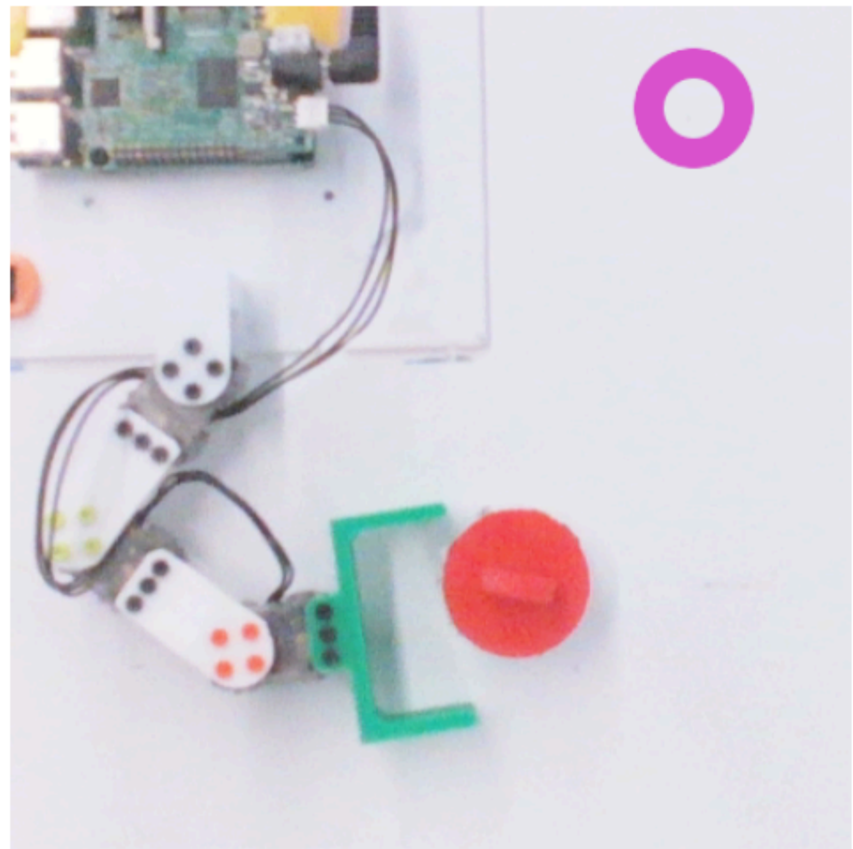


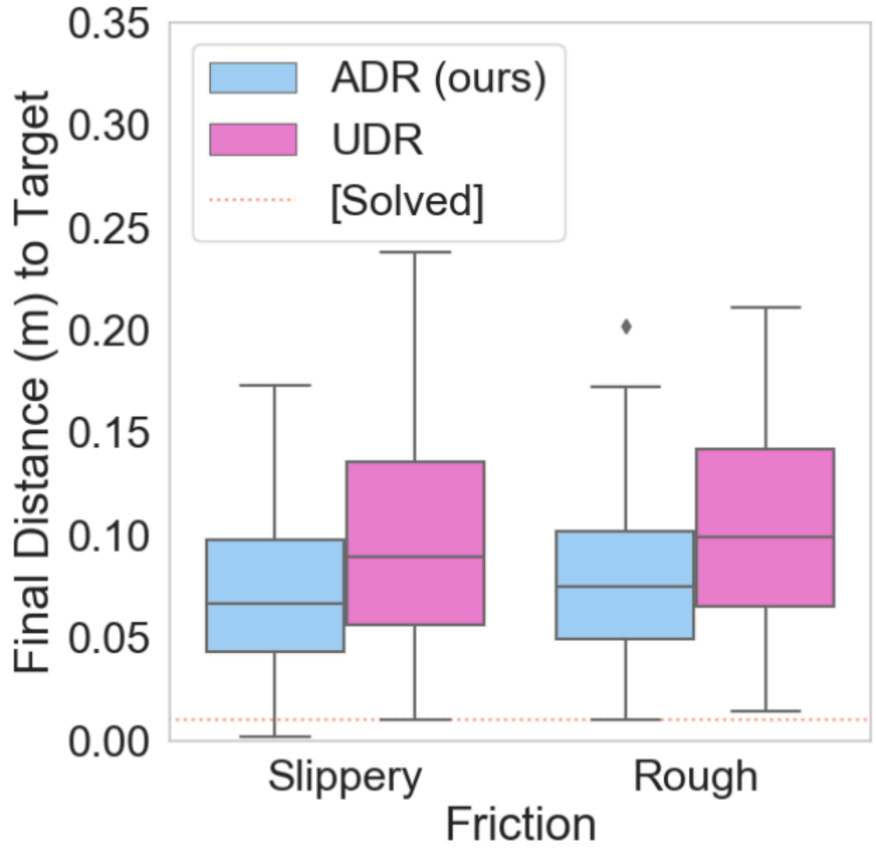




(a)

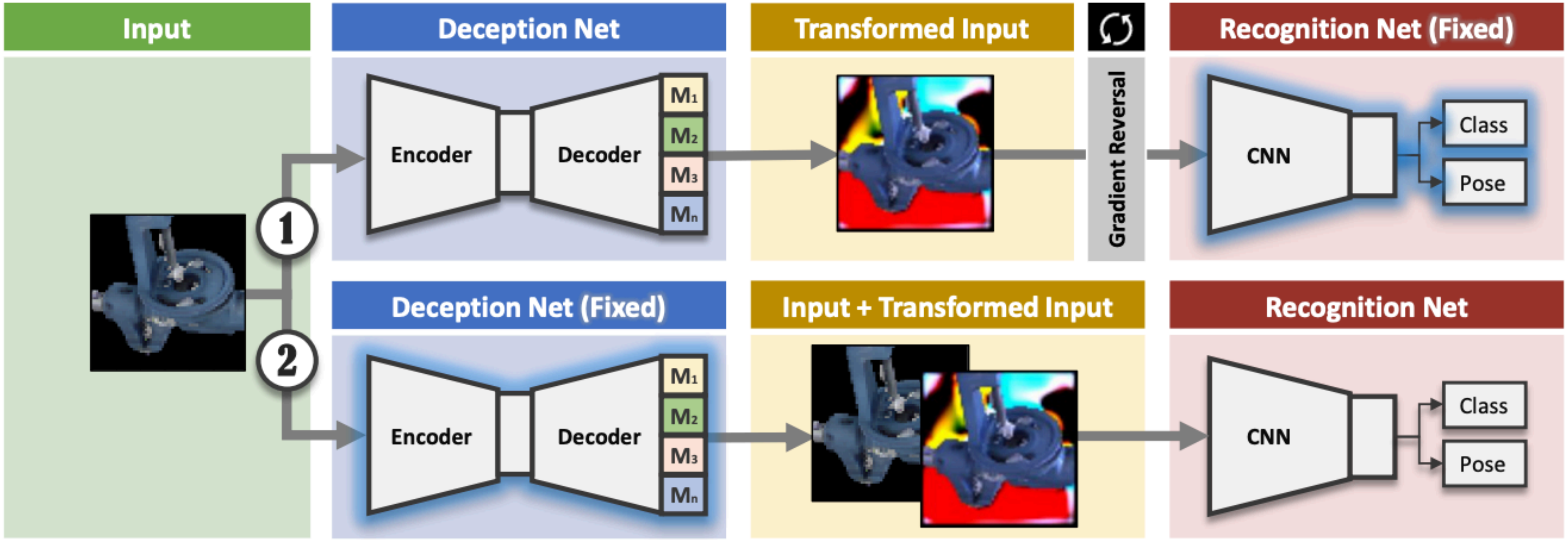






Active Domain Randomization
[Bhairav Mehta et al, 2019]

Network-Driven Domain Randomization



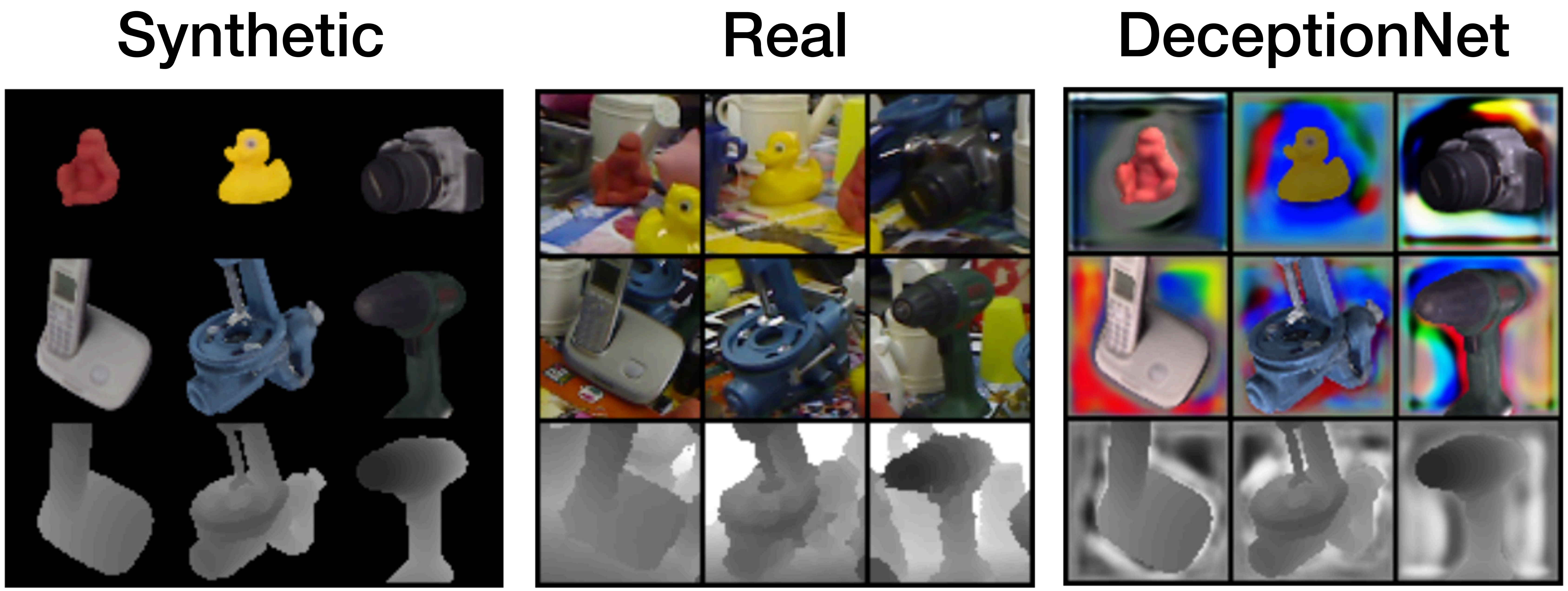
DeceptionNet: Network-Driven Domain Randomization
[Sergey Zakharov et al, 2019]

How to ensure randomizations don't destroy semantic info?

- Restrict randomizations to come from specifically designed differentiable **deception modules**
- **Distortion module**: elastic distortions
- **BG/FG module**: background and foreground colors
- **Noise module**: adding noise to the image

DeceptionNet: Network-Driven Domain Randomization
[Sergey Zakharov et al, 2019]

Network-Driven Domain Randomization



DeceptionNet: Network-Driven Domain Randomization
[Sergey Zakharov et al, 2019]

Network-Driven Domain Randomization

		MNIST → MNIST-COCO	Synthetic Cropped LineMOD → Extended Real Cropped LineMOD	
	Model	**Classification Accuracy (%)**	**Classification Accuracy (%)**	**Mean Angle Error (°)**
	Source (S)	57.2	63.1	78.3
S	Unguided	85.8	77.2	48.5
S	Ours	89.4	99.0	46.5
S + T	DSN [5]	73.2	45.7	76.3
S + T	PixelDA [4]	72.5	76.0	84.2
	Target (T)	96.1	100	14.7

DeceptionNet: Network-Driven Domain Randomization
[Sergey Zakharov et al, 2019]

Extensions of domain randomization

- Network architectures better suited to transfer
- Match the simulator to real data
- Use real data to improve task performance
- Automatically surface hard cases in the simulator
- **Allow the model to perform well on a wider range of simulations**

Policy Transfer with Strategy Optimization

- At training time, instead of optimizing a single best policy $\pi : o \mapsto a$ for *all* physics parameters μ, optimize a conditional policy $\pi_\eta : o \mapsto a$ that gets to observe the physics params

- At test time on the real environment, use a black box optimizer to find

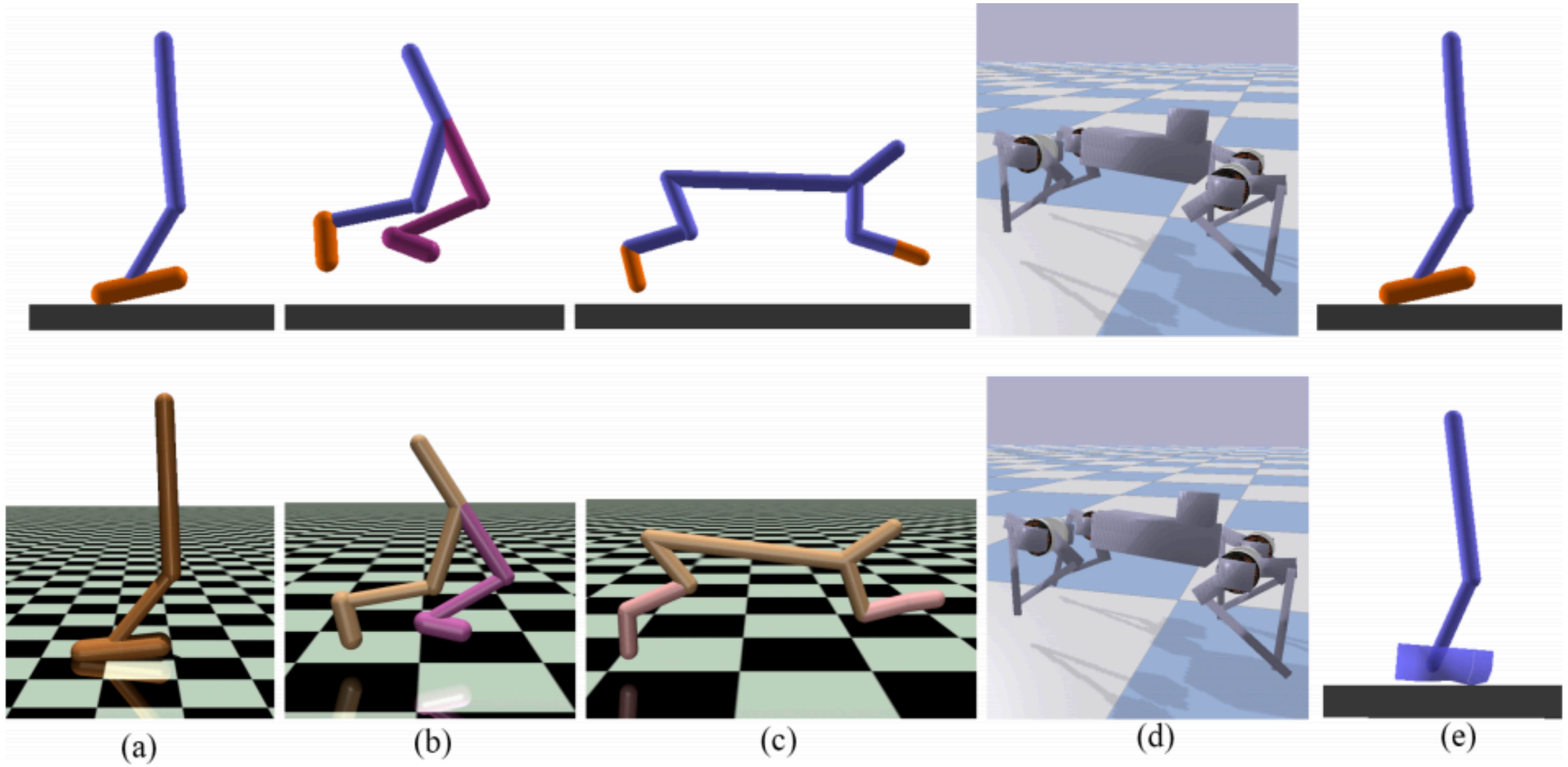
$$\eta * = \arg\max_{\eta} R(\pi_\eta)$$

Policy Transfer with Strategy Optimization
[Wenhao Yu et al, 2018]

Policy Transfer with Strategy Optimization

Dart

MuJoCo



Policy Transfer with Strategy Optimization
[Wenhao Yu et al, 2018]

Automatic Domain Randomization (ADR)

Intuition

- More randomization = better transfer, **given the same performance in sim**
- But wide randomization ranges lead to poor performance
- ADR: automatically create a curriculum of expanding randomization ranges
- How? Widen the distribution if performance is good near the boundary of the range

Solving Rubik's Cube with a Robot Hand
[OpenAI et al, 2019]

Automatic Domain Randomization (ADR)

How parameters are sampled from the randomization distribution

- **Standard domain randomization**: sample sim params $\eta = \left[\eta_1, \cdots, \eta_k\right]$ uniformly from some range $\phi_i^L < \eta_i < \phi_i^H$
- **ADR:** Sample $k - 1$ of the params from the range. Select one parameter to lie on a boundary of the range

Solving Rubik's Cube with a Robot Hand
[OpenAI et al, 2019]

Automatic Domain Randomization (ADR)

How parameter ranges are updated

- **Standard domain randomization**: randomizations are fixed.
- **ADR:** keep track of performance when each parameter is on boundary. Periodically, if average performance is good, widen range. If bad, narrow range.

Solving Rubik's Cube with a Robot Hand
[OpenAI et al, 2019]

Automatic Domain Randomization (ADR)

Algorithm 1 ADR

Require: ϕ^0 ▷ Initial parameter values

Require: $\{D_i^L, D_i^H\}_{i=1}^d$ ▷ Performance data buffers

Require: m, t_L, t_H, where $t_L < t_H$ ▷ Thresholds

Require: Δ ▷ Update step size

$\phi \leftarrow \phi^0$

repeat

  $\lambda \sim P_\phi$

  $i \sim U\{1, \ldots, d\}$, $x \sim U(0, 1)$

  if $x < 0.5$ **then**

    $D_i \leftarrow D_i^L$, $\lambda_i \leftarrow \phi_i^L$ ▷ Select the lower bound in "boundary sampling"

  else

    $D_i \leftarrow D_i^H$, $\lambda_i \leftarrow \phi_i^H$ ▷ Select the higher bound in "boundary sampling"

  end if

  $p \leftarrow \text{EVALUATEPERFORMANCE}(\lambda)$ ▷ Collect model performance on environment parameterized by λ

  $D_i \leftarrow D_i \cup \{p\}$ ▷ Add performance to buffer for λ_i, which was boundary sampled

  if $\text{LENGTH}(D_i) \geq m$ **then**

    $\bar{p} \leftarrow \text{AVERAGE}(D_i)$

    $\text{CLEAR}(D_i)$

    if $\bar{p} \geq t_H$ **then**

      $\phi_i \leftarrow \phi_i + \Delta$

    else if $\bar{p} \leq t_L$ **then**

      $\phi_i \leftarrow \phi_i - \Delta$

    end if

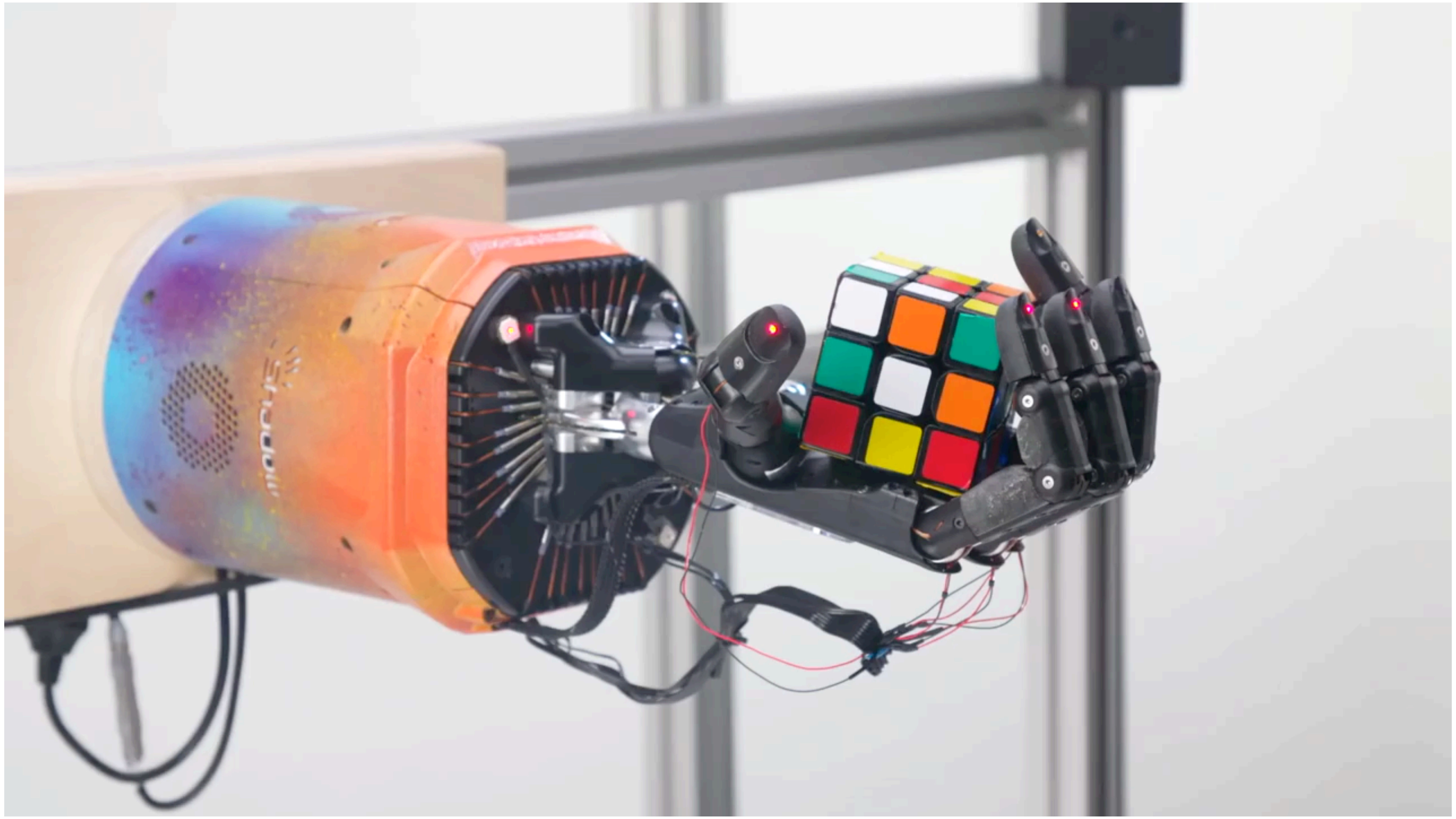
  end if

until training is complete

Solving Rubik's Cube with a Robot Hand

[OpenAI et al, 2019]

ADR - results



Solving Rubik's Cube with a Robot Hand
[OpenAI et al, 2019]

DR vs ADR on block rotation

Policy	Training Time	ADR Entropy	Successes (Sim)		Successes (Real)	
			Mean	Median	Mean	Median
Baseline (data from [77])	—	—	43.4 ± 0.6	**50**	18.8 ± 5.4	13.0
Baseline (re-run of [77])	—	—	33.8 ± 0.9	**50**	4.0 ± 1.7	2.0
Manual DR	13.78 days	-0.348^{*} npd	42.5 ± 0.7	**50**	2.7 ± 1.1	1.0
ADR (Small)	0.64 days	-0.881 npd	21.0 ± 0.8	15	1.4 ± 0.9	0.5
ADR (Medium)	4.37 days	-0.135 npd	34.4 ± 0.9	**50**	3.2 ± 1.2	2.0
ADR (Large)	13.76 days	0.126 npd	40.5 ± 0.7	**50**	13.3 ± 3.6	11.5
ADR (XL)	—	0.305 npd	45.0 ± 0.6	**50**	16.0 ± 4.0	12.5
ADR (XXL)	—	**0.393 npd**	$\mathbf{46.7 \pm 0.5}$	**50**	$\mathbf{32.0 \pm 6.4}$	**42.0**

Solving Rubik's Cube with a Robot Hand
[OpenAI et al, 2019]

Outline of the rest of the talk

- Why is it so hard to use simulated training data?
- How can you use simulation without solving sim2real?
- Building a good simulation
- Domain adaptation
- Domain randomization
- **What's next?**

What's next in Sim2Real

- More / better domain randomization tools
- More accurate / scalable simulations?
- The next generation of Sim2Real techniques:
  - More / better automatic domain randomization
  - Domain randomization, domain adaptation, and model-based RL converge
- The next generation of Sim2Real use cases:
  - Synthetic data for edge cases
  - Synthetic data for reducing bias
  - Synthetic data on a hard, real-world dataset (drive an RC car around UC Berkeley?)

The dream: real-sim-real

Collect data about the world

Automatically design and randomize a sim

Optimize a policy in that simulation

Figures from: **Learning Dextrous In-Hand Manipulation**
[OpenAI et al, 2018]

Where to learn more?

- Sim2Real AI (https://sim2realai.github.io/, https://twitter.com/sim2realAIorg)
- Domain Randomization for Sim2Real Transfer (https://lilianweng.github.io/lil-log/2019/05/05/domain-randomization.html)
- RSS Sim2Real Workshop, 2019 (https://sim2real.github.io/)
- Unreal CV (https://github.com/unrealcv/synthetic-computer-vision)

Thank you!