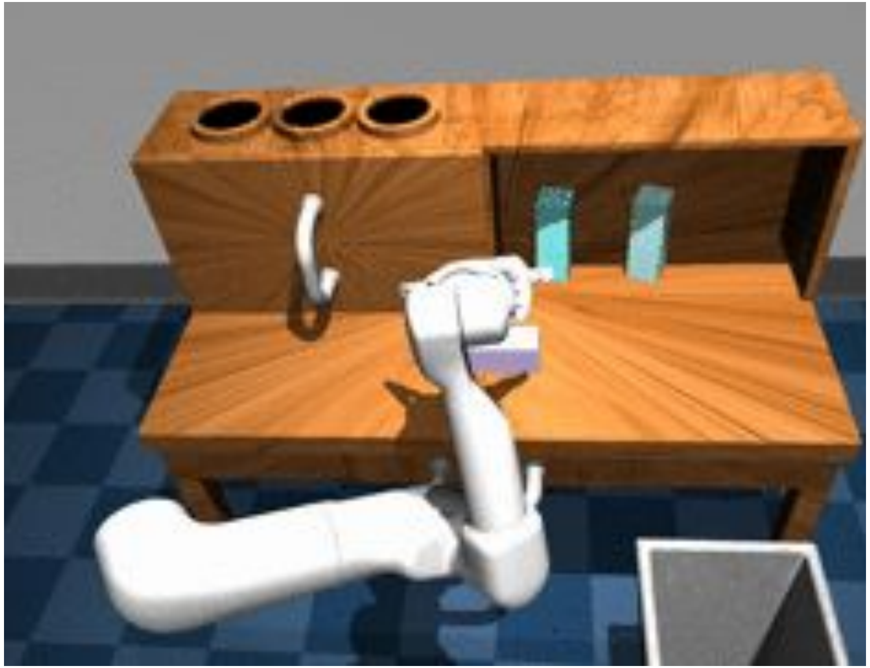
Self-Supervision and Play



Pierre Sermanet

In collaboration with

Corey Lynch, Debidatta Dwibedi, Soeren Pirk, Jonathan Tompson, Mohi Khansari, Yusuf Aytar, Yevgen Chebotar, Yunfei Bai, Jasmine Hsu, Eric Jang, Vikash Kumar, Ted Xiao, Stefan Schaal, Andrew Zisserman, Sergey Levine

Robotics at Google

http://g.co/robotics

Main Message

- Real-world robotics cannot rely on labels and rewards
- Instead, mostly
  - **Self-supervise on unlabeled data**
  - **Use play data**
- We present ways to do this for vision and control

Our mission:

Self-Supervised Robots

i.e.: Autonomously extract learning signals from the world
from play and from others



because: "*Give* a robot *a* label and you feed it for a *second*;
Teach a robot *to* label and you feed it for a *lifetime*."

Supervision Types

Supervised

Signal:

Discrete & human-defined Embedding

1. Hand pose
2. Cups pose
3. Liquid amounts
4. Liquid flowing
5. Liquid color

Unsupervised

Signal:

- Auto-encoding
- Sparsity prior
- Gaussian prior

Continuous embedding

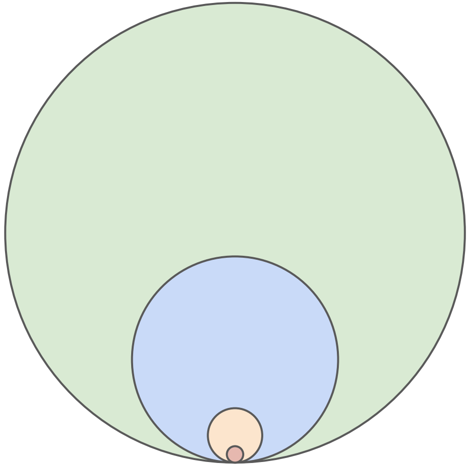
Self-Supervised

Signal:

- Depth
- Time
- Multi-view
- Tactile feedback

Continuous embedding

Supervision Costs

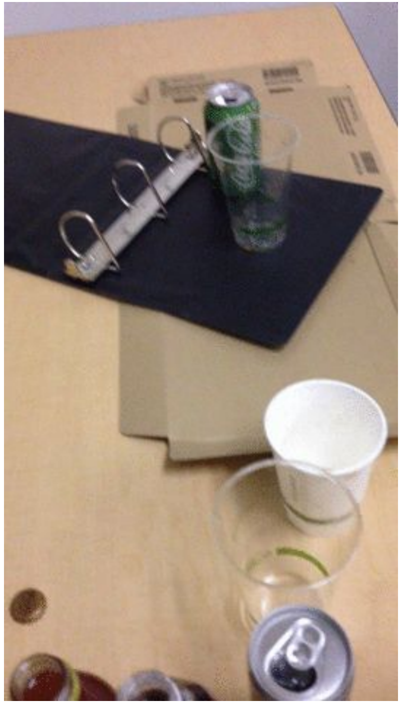


Proportions in which supervision is ok to use (ideally mostly rely on green and blue).

Type of Supervision	Description	Cost
Playing (Intrinsic Motivation)	Alone or with others	**Free**
Play data (Tele-op)		**Very cheap**
Imitation	other agents “playing” for hours, not segmented, not labeled	**Cheap (but not unlimited)**
Demonstrations	Staged, segmented and labeled	**Expensive**
Labeled Frames	e.g. action and object classes / attributes	**Very Expensive**

Why Self-Supervise?

- Be versatile and robust to **different hardware & environments:**
  - **Robot-agnostic and self-calibrating**
  - **Agnostic to sim or real**, train the same way
- **Scaling up** in the real world
- **Can't afford human supervision** given the high dimensionality of the problem
  - Labeling is **not easy to define** even for humans
- **Rich representations** can be discovered through self-supervision and lead to **higher sample efficiency** in RL



Why play?

- Self-Supervision enables using play data
- Cheap
- General
- Rich

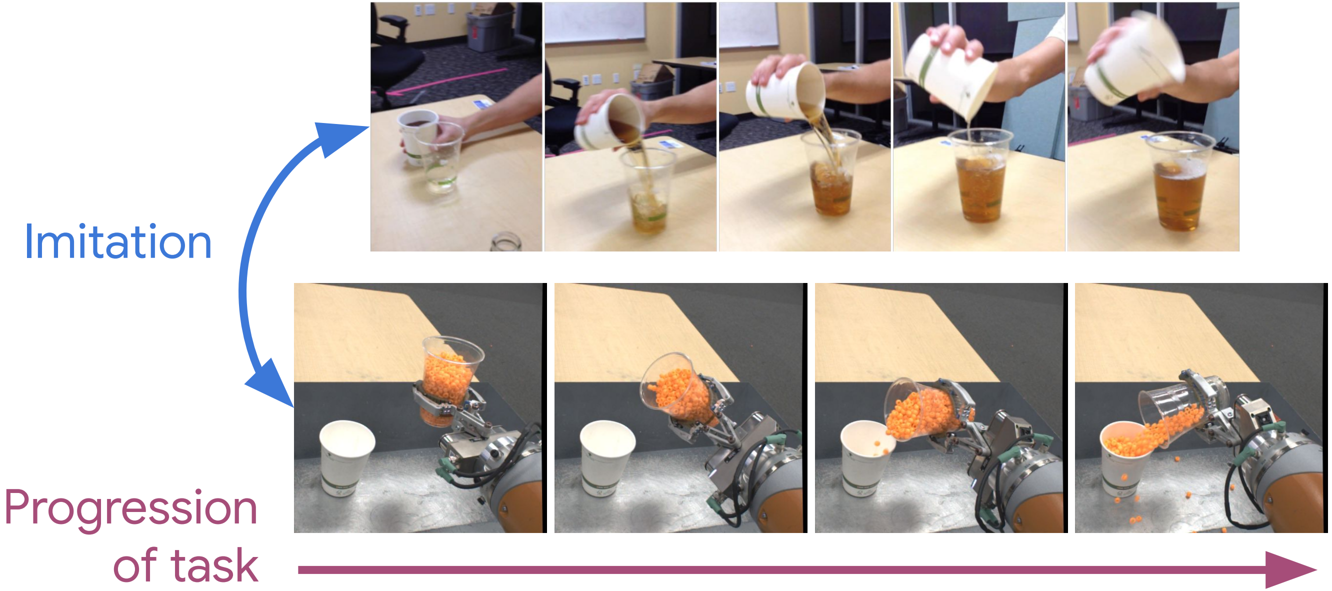

label
free

Self-Supervision and Play
for Vision

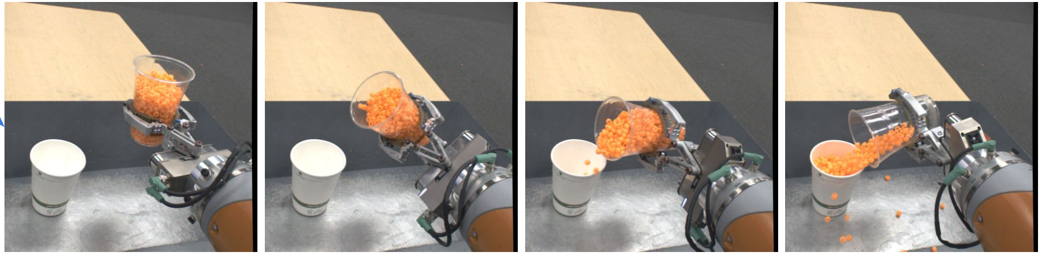
Self-Supervised Visual Representations

- Time-Contrastive Networks (TCN)
- Temporal Cycle-Consistency (TCC)
- Object-Contrastive Networks (OCN)

disentangled / invariant states and attributes



Imitation



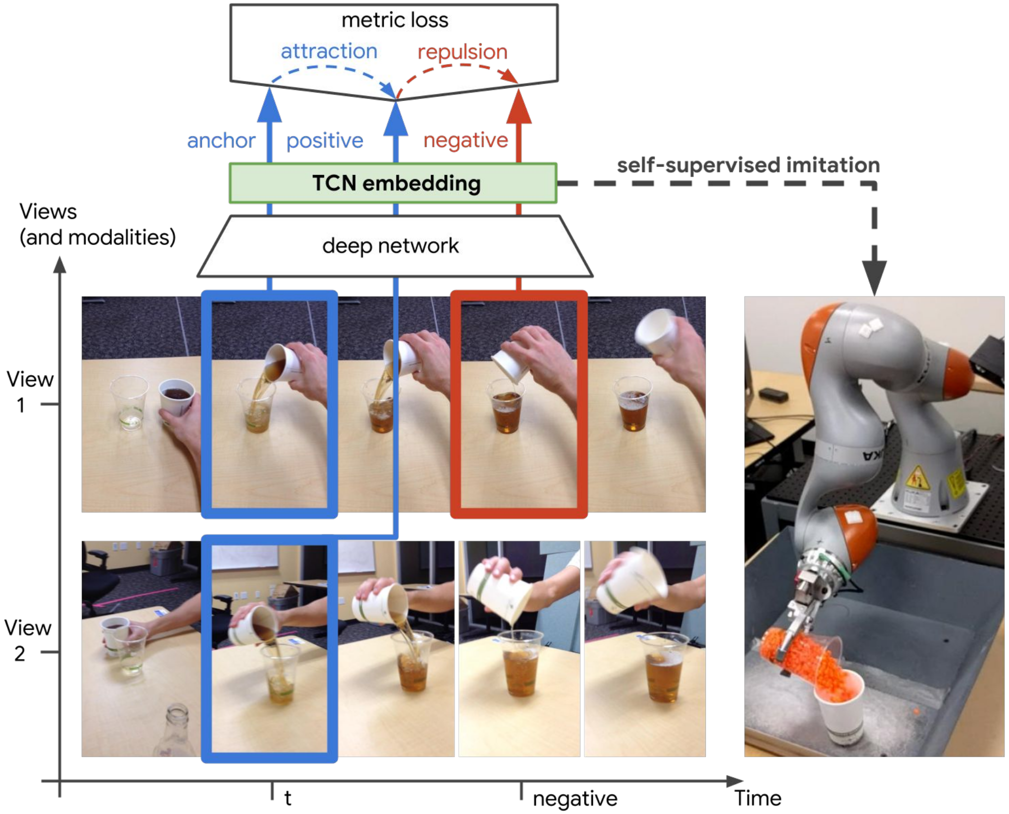
Progression
of task

Play

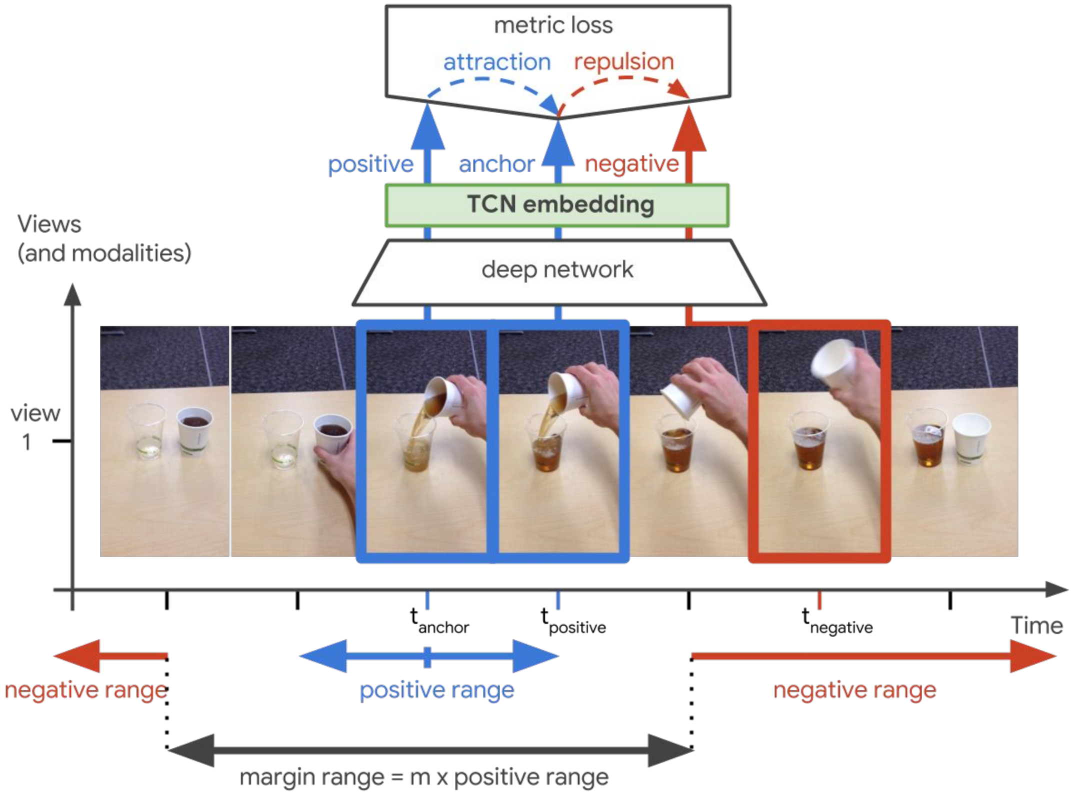
- Detect errors
- Correct
- Retry
- Transition
- General skills
- Data augmentation

Time-Contrastive Networks (TCN)

[Sermanet*, Lynch*, Chebotar*, Hsu, Jang, Schaal, Levine @ ICRA 2018] [sermanet.github.io/imitate]



Single-view TCN

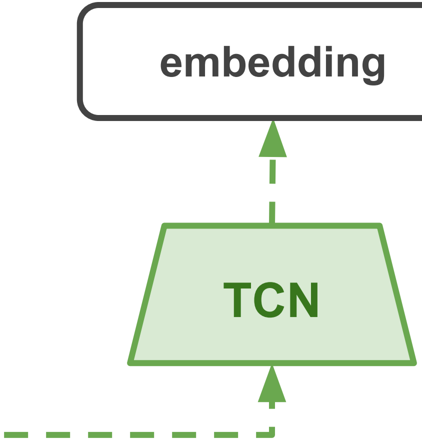


Semantic Alignment with TCN

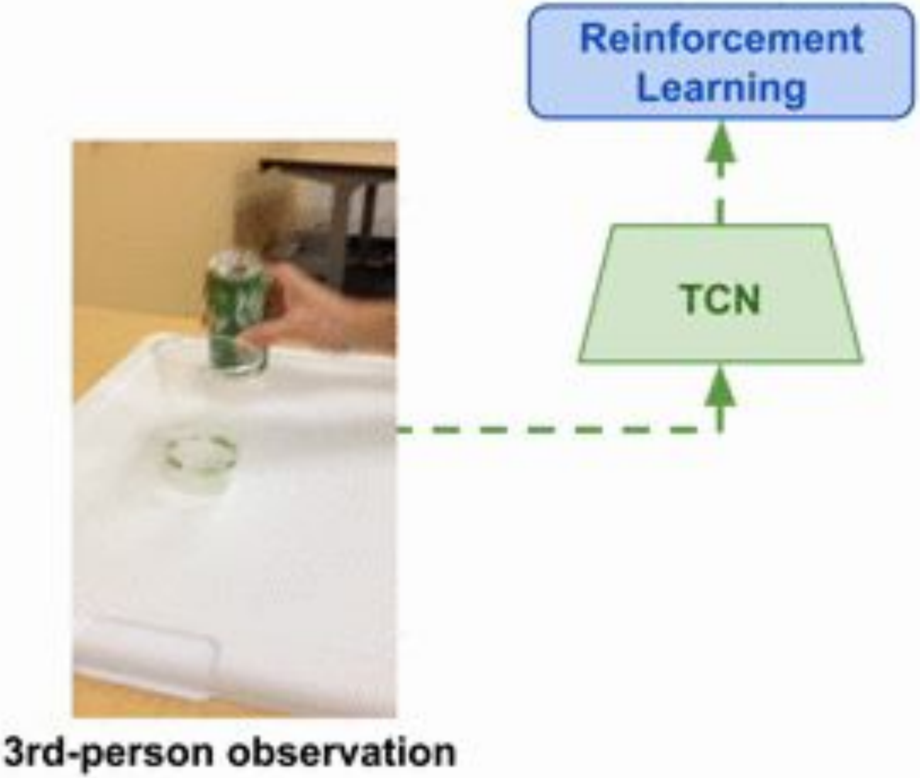
Observation **multi-view TCN**

Robotic Imitation:

Step 1. Self-Supervise on Play data

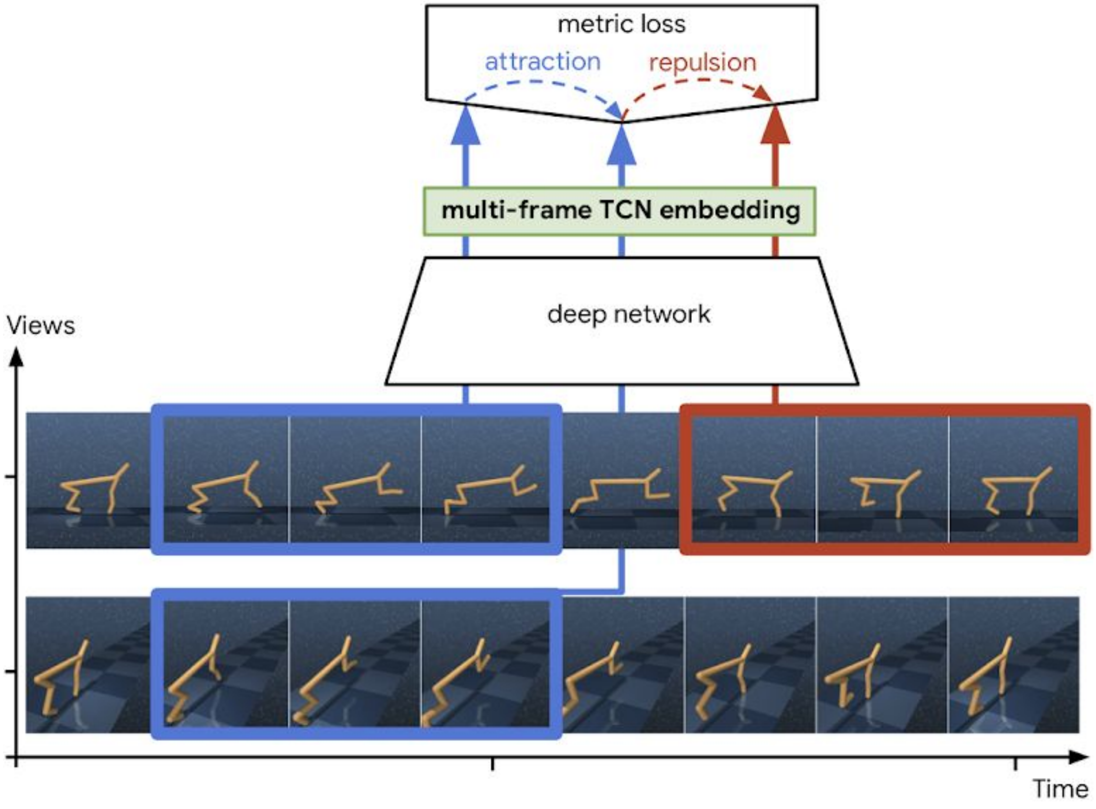


Robotic Imitation:
Step 2. Follow abstract trajectory



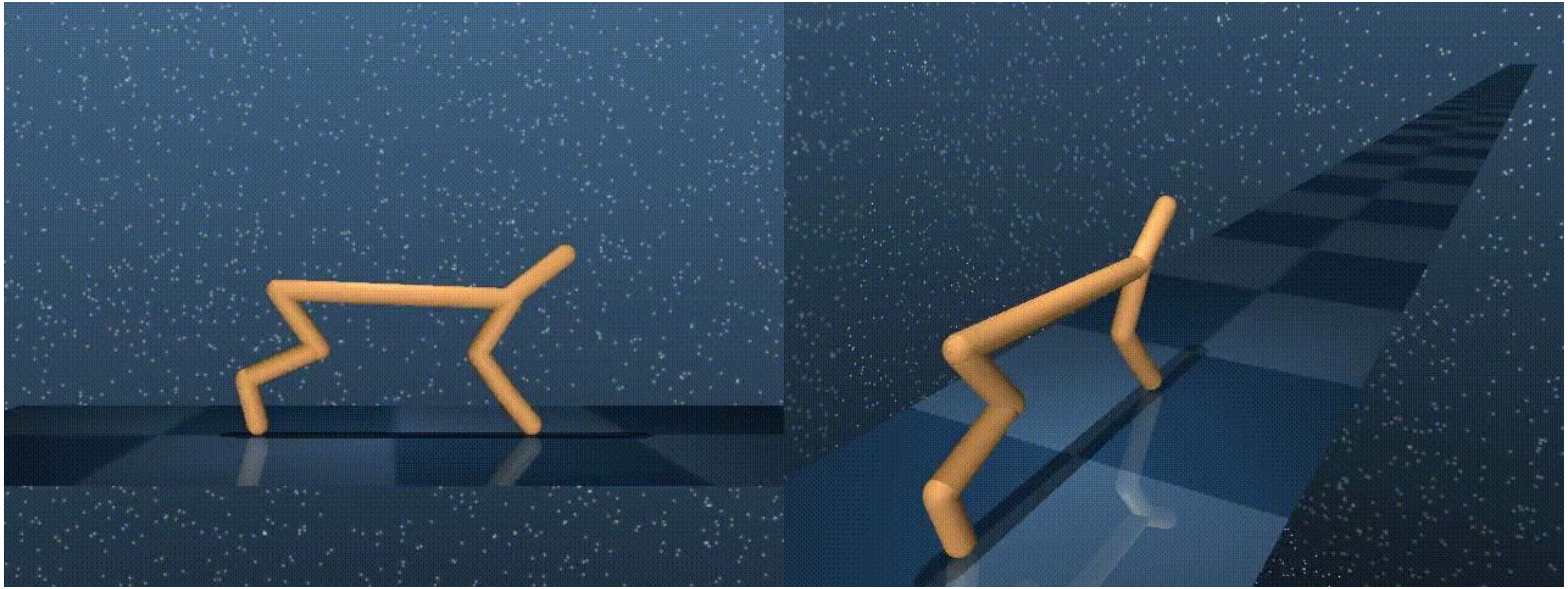
Actionable Representations

[Dwibedi, Tompson, Lynch, Sermanet @ IROS 2018] [sites.google.com/view/actionablerepresentations]



Cheetah Environment

Agent observes another agent demonstrating an action



View 1 View 2

Qualitative Results: Cheetah

PPO on
true state

PPO on
learned visual representations

Input to PPO	Cumulative Reward (Avg of 100 runs)
Random State	28.31
True State	390.16
Raw Pixels	146.14
mfTCN	360.50

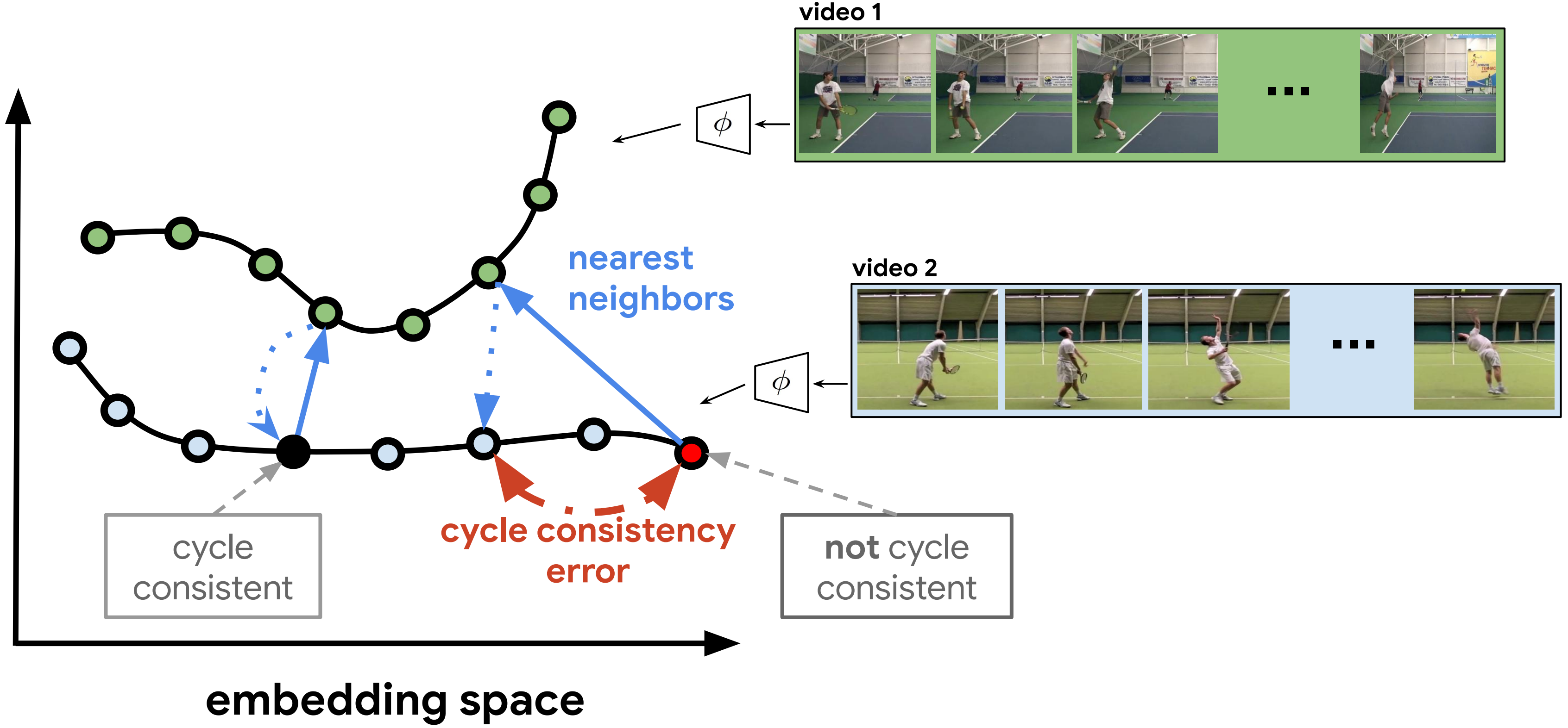
Temporal Cycle-Consistency (TCC)

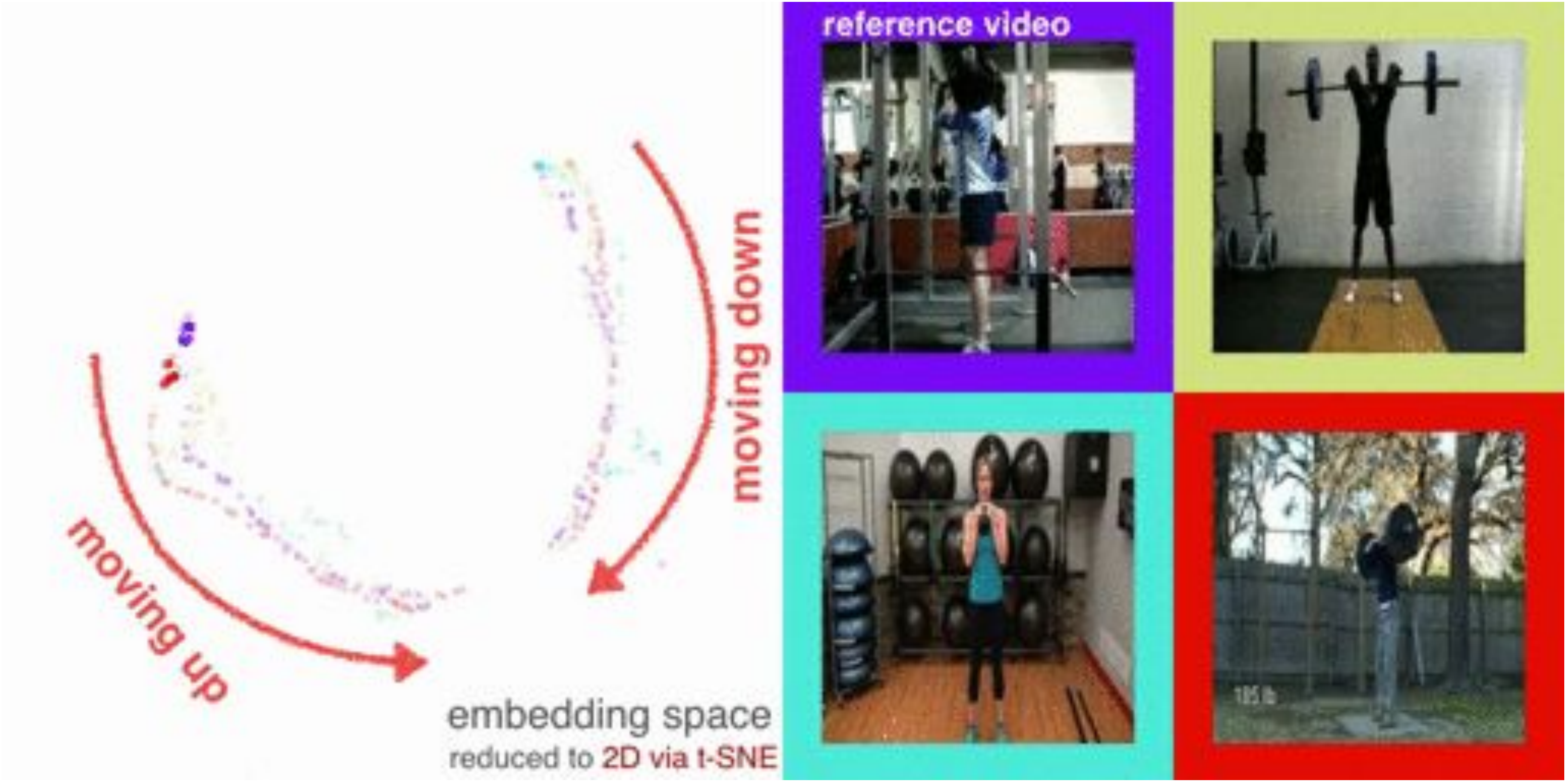
[Dwibedi, Aytar, Tompson, Sermanet, Zisserman @ CVPR 2019] [temporal-cycle-consistency.github.io]

Temporal Cycle-Consistency (TCC)

[Dwibedi, Aytar, Tompson, Sermanet, Zisserman @ CVPR 2019] [temporal-cycle-consistency.github.io]

Temporal Cycle-Consistency

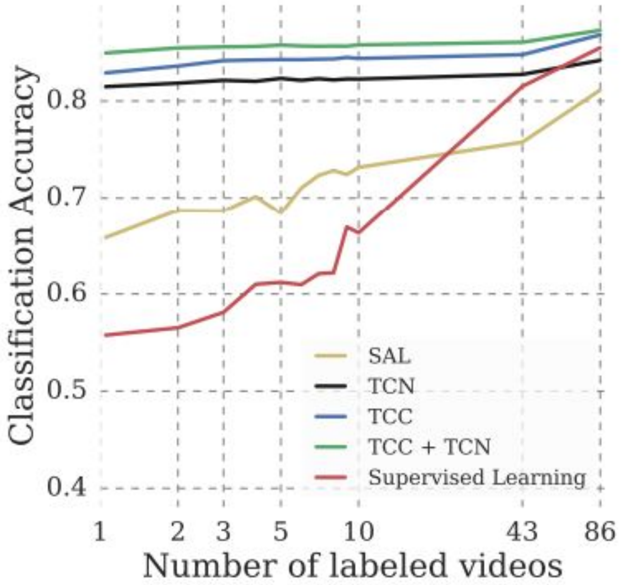



moving up
moving down
embedding space
reduced to 2D via t-SNE
reference video

Action Phase Classification

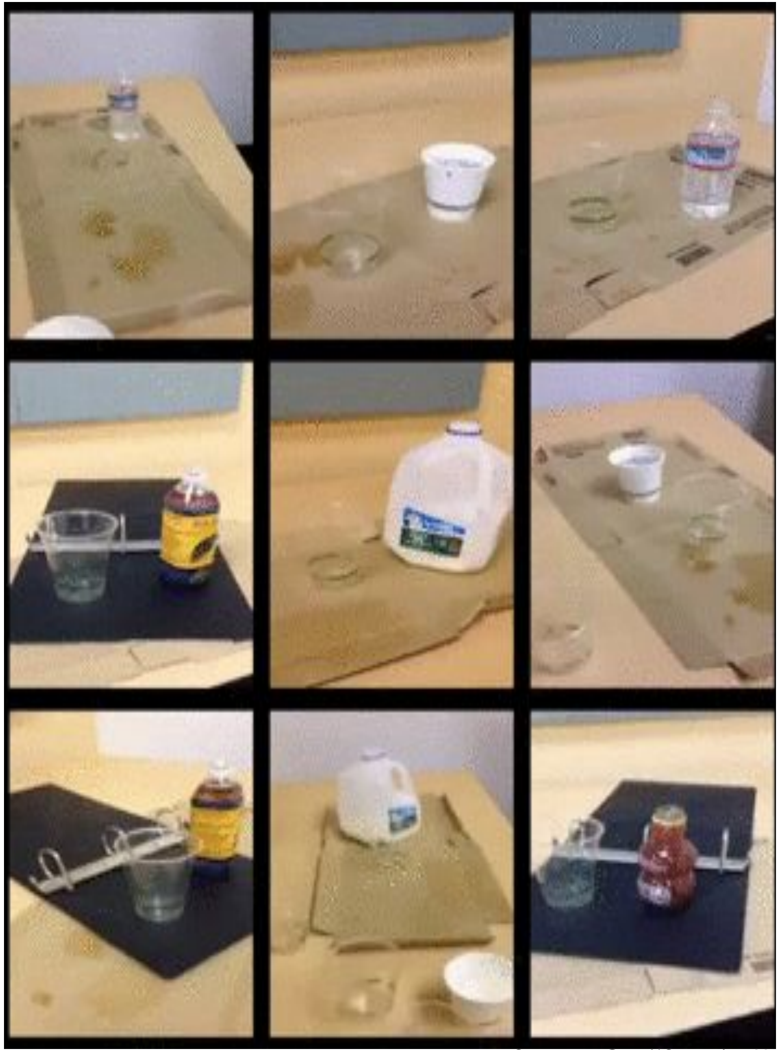
Datasets	% of Labels →	0.1	0.5	1.0
Penn Action	Supervised Learning	57.69	78.55	83.83
	SaL [18]	70.78	74.39	76.35
	TCN [25]	80.21	81.77	82.52
	TCC (ours)	76.41	79.85	81.68
	TCC + SaL (ours)	77.90	81.39	83.11
	TCC + TCN (ours)	**81.59**	**83.50**	**84.11**
Pouring	Supervised Learning	77.31	85.42	90.12
	SaL [18]	79.36	86.62	86.72
	TCN [25]	86.94	88.51	89.14
	TCC (ours)	84.73	88.83	**91.45**
	TCC + SaL (ours)	87.80	89.55	90.61
	TCC + TCN (ours)	**90.97**	**90.17**	90.33

Phase classification results when fine-tuning ImageNet pre-trained ResNet-50.



(a) Golf Swing

Self-Supervised Alignment



Object-Contrastive Networks (OCN)

[Pirk, Khansari, Bai, Lynch, Sermanet @ under review]

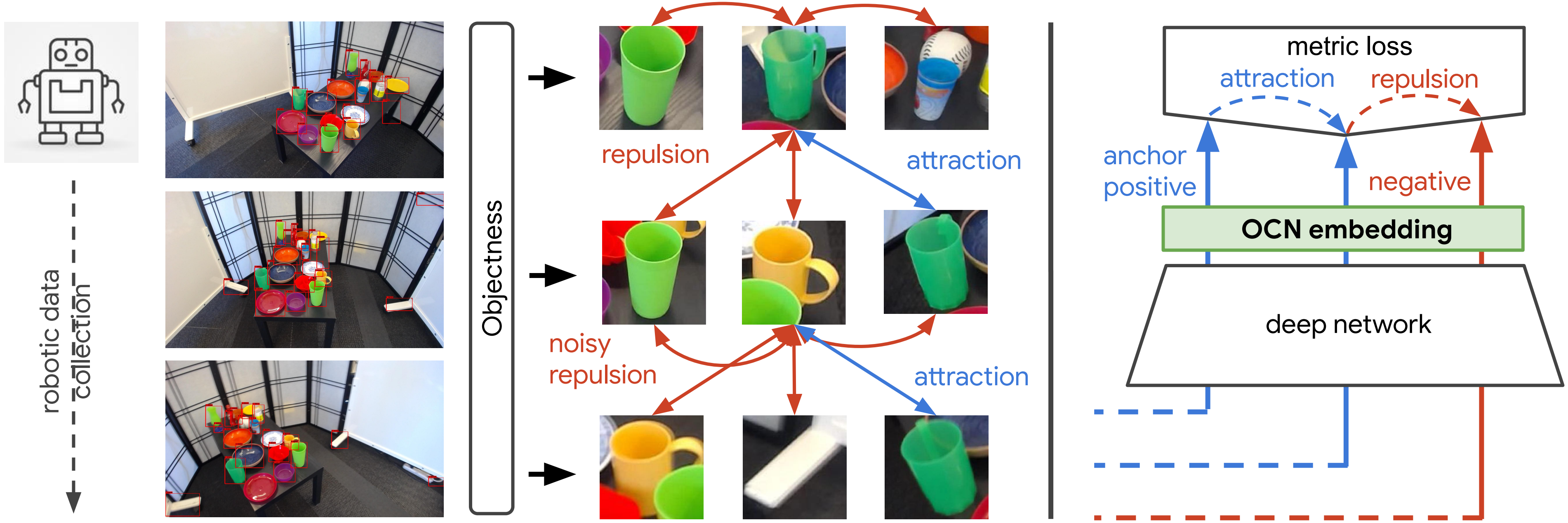
Self-teaching about any object
leads to high robustness, allowing deployment

Robotic Data Collection

Play data for Objects

Object-Contrastive Networks (OCN)

[Pirk, Khansari, Bai, Lynch, Sermanet @ under review]

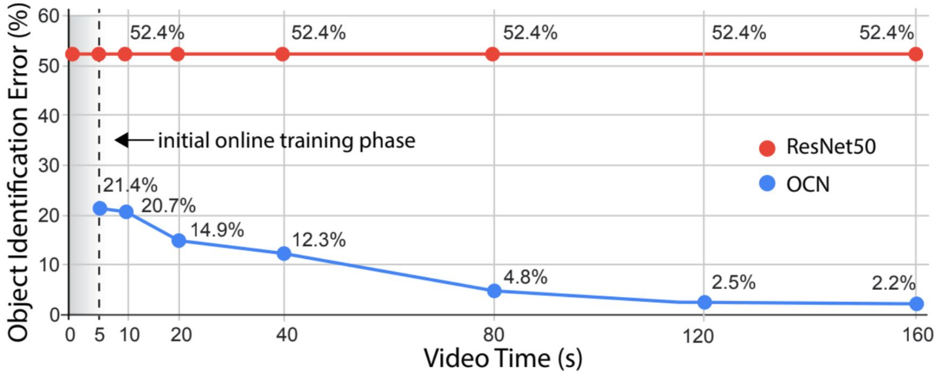


Recovering Continuous Attributes

(Same instance removed)

Online Object Understanding

- Offline average error: 54%
- Online average error: 17% -> 3%
- Do not define states and attributes



Online Adaptation

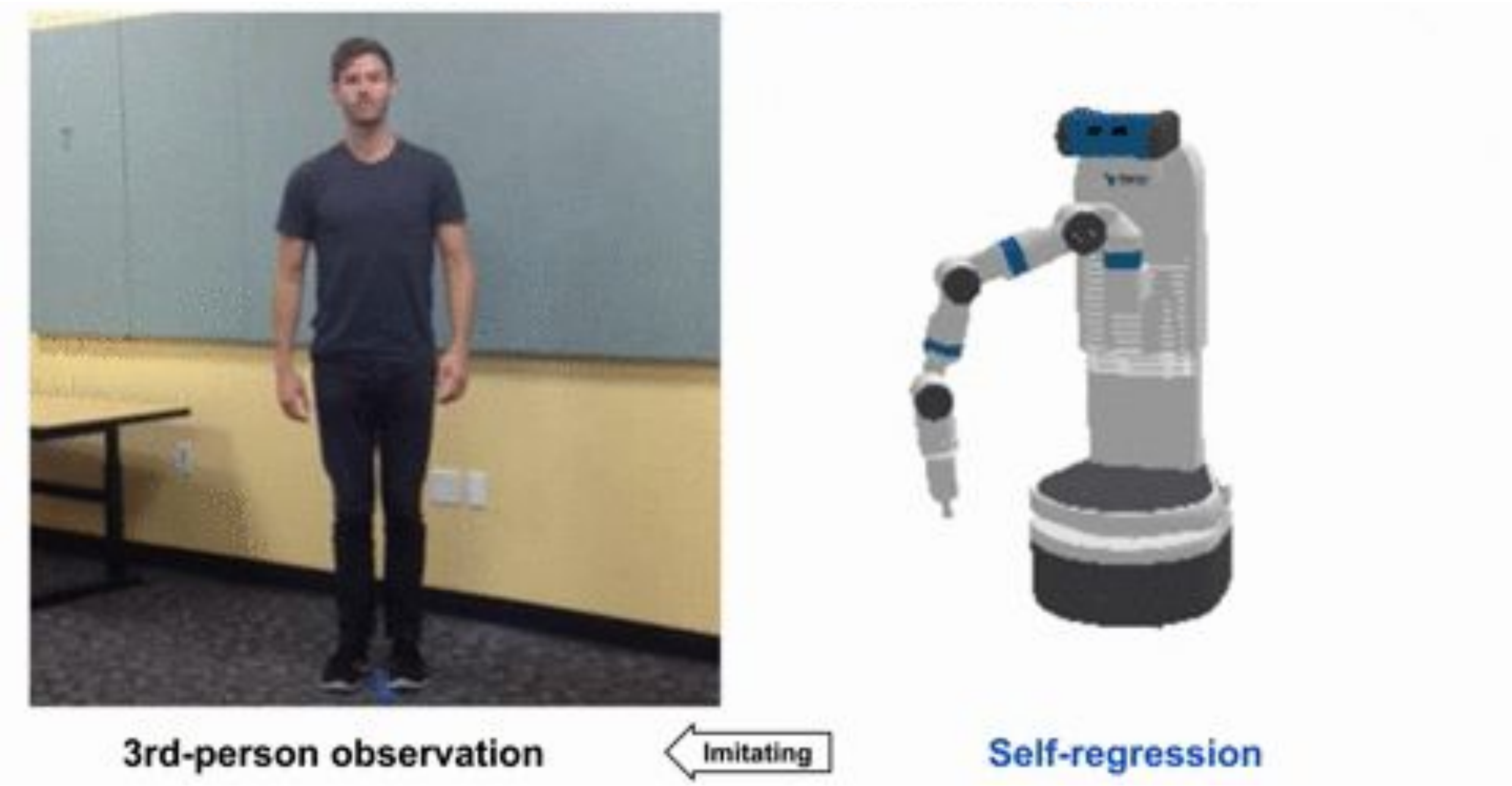
Online Adaptation

ResNet50 - **81.9% error**   Trained on 160s - **40.3% error**

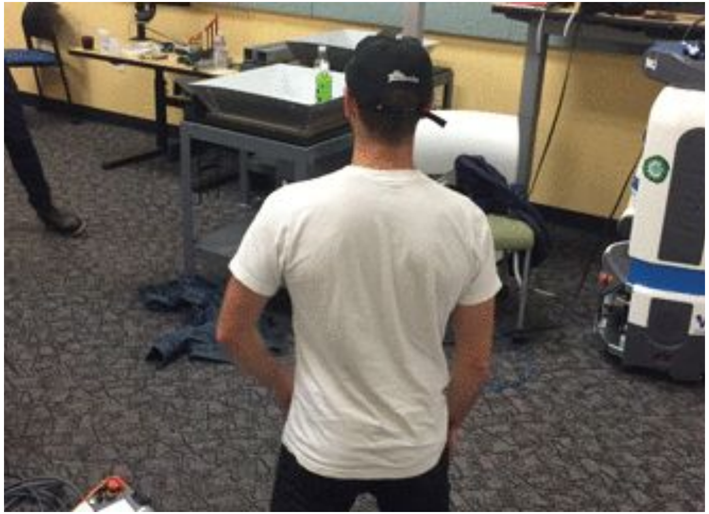
Self-Supervision and Play for Control

Pose Imitation with TCN

[Sermanet*, Lynch*, Chebotar*, Hsu, Jang, Schaal, Levine @ ICRA 2018] [sermanet.github.io/imitate]



Play Data in Pose Space

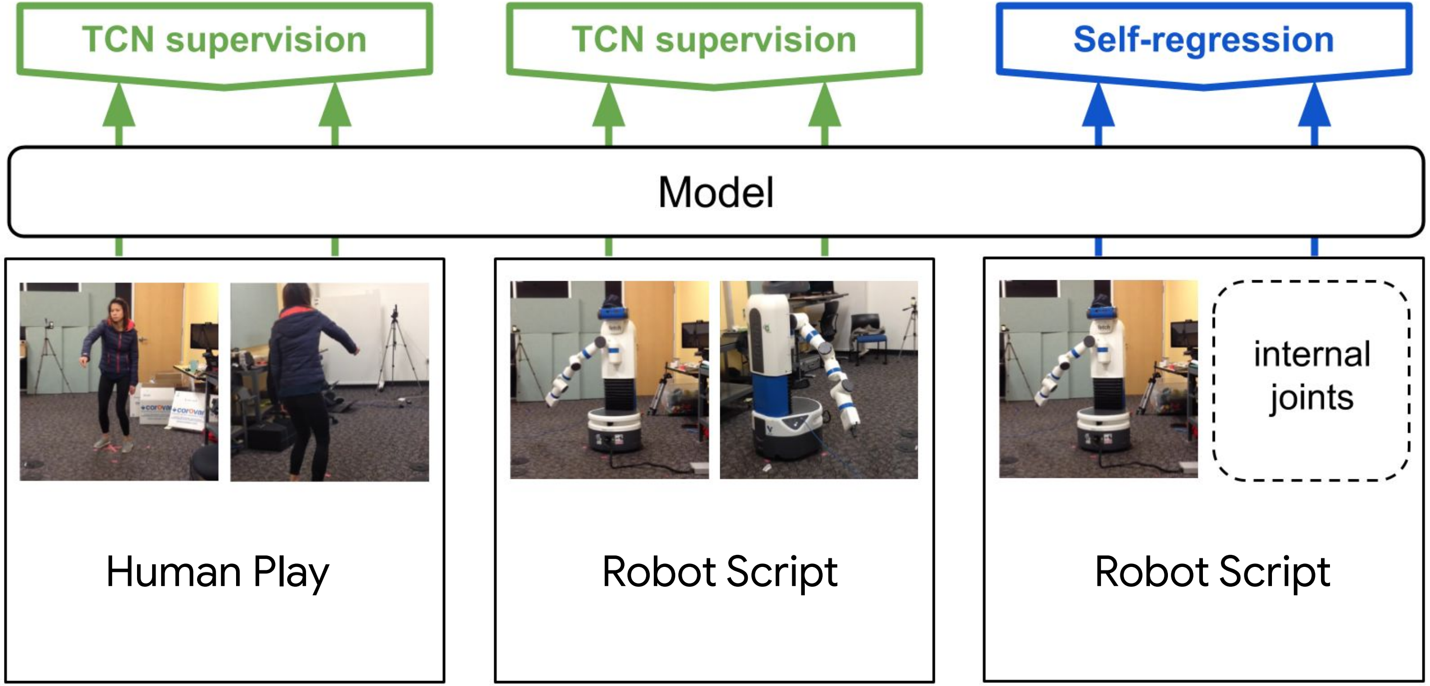


Human Play

Robot Scripting

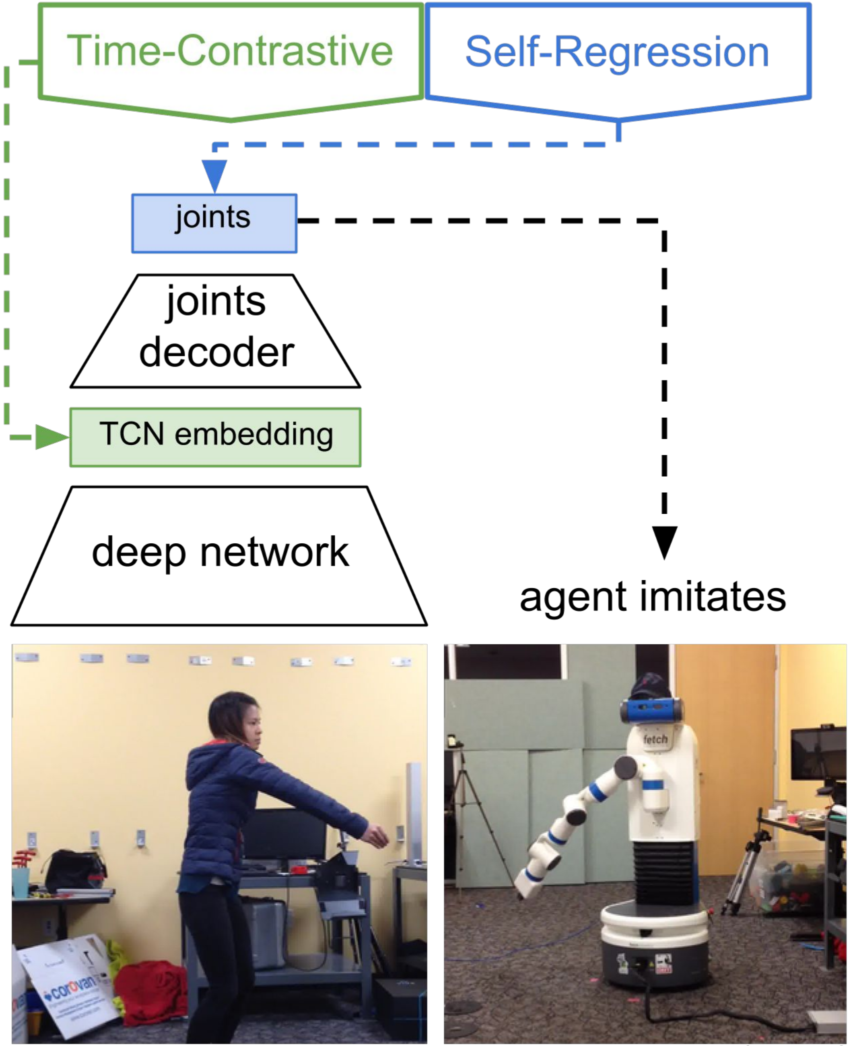
Human imitating Robot

Pose Imitation with Play Data



Pose Imitation

- Self-Supervision + Play recipe
- No explicit task definition.

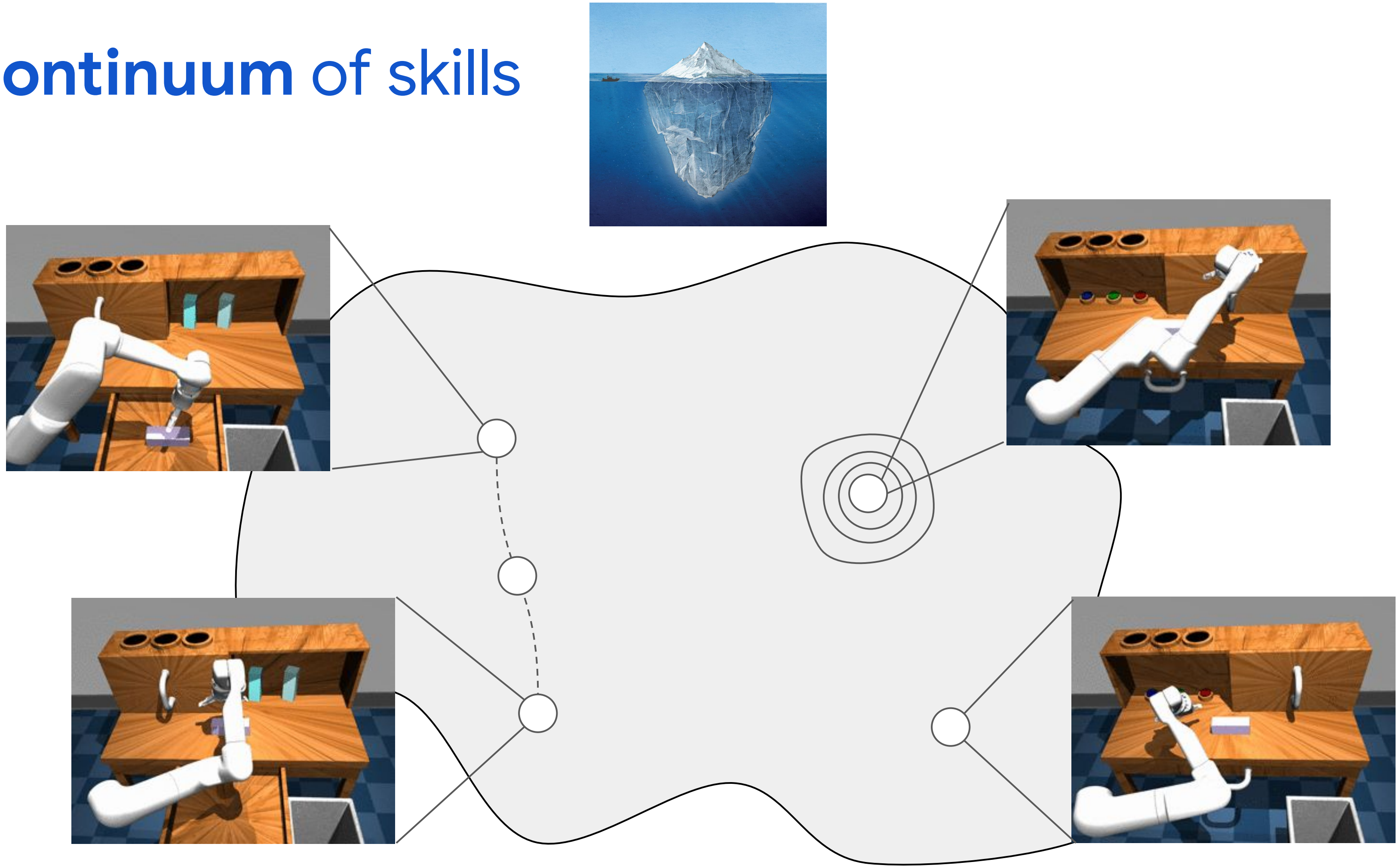


Learning from Play (LfP)

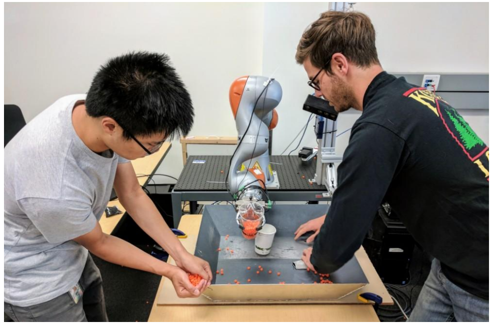
[Lynch, Khansari, Xiao, Kumar, Tompson, Levine, Sermanet @ under review] [learning-from-play.github.io]

- No tasks
- No rewards or RL
- Multiple tasks in zero-shot
- 85% on 18 tasks
- Self-Supervision + Play recipe

Continuum of skills



How can we cover the continuum?

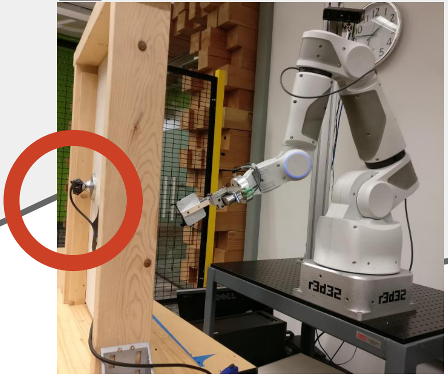


Exploration

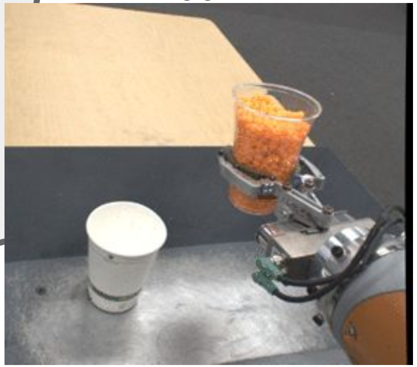
Scripted collection + RL

reward sensor
for door opening

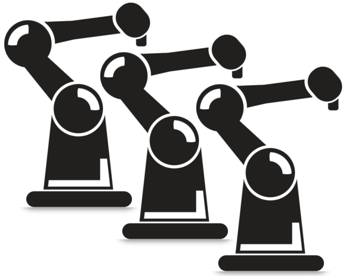
Scripting exploration



Reward engineering

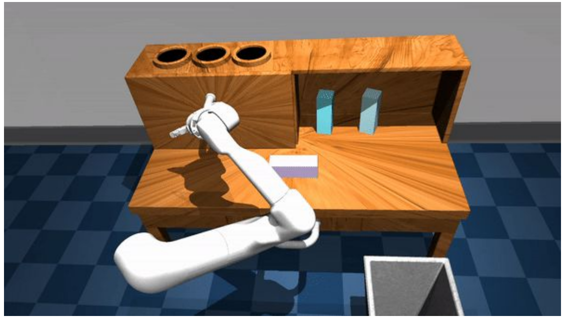


Reset

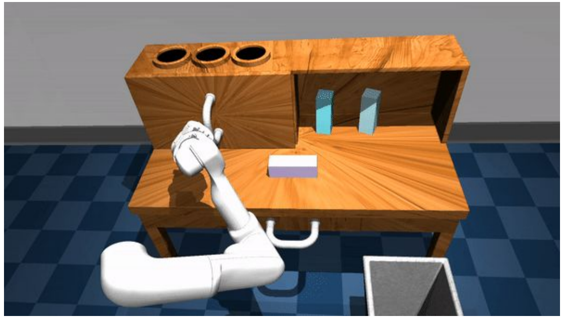


Distributed training

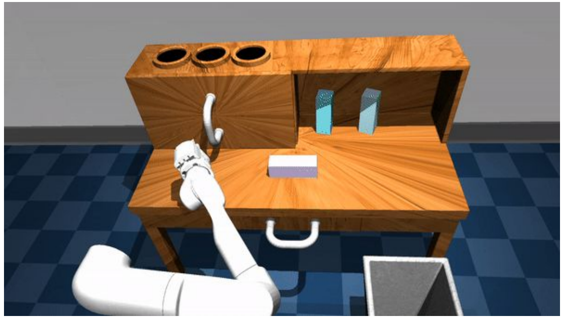
Tasks are not discrete



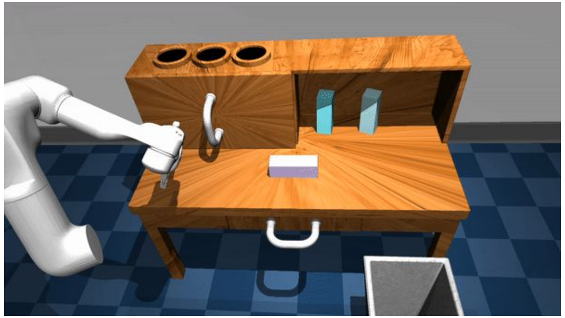
"Grasp fast?"



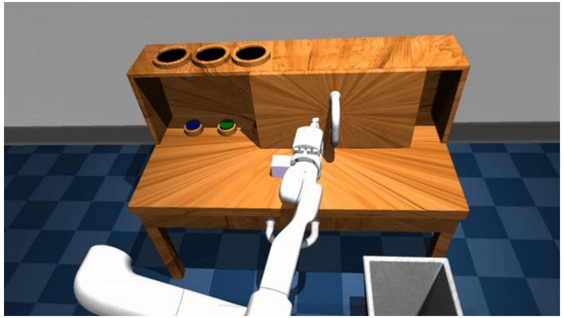
"Nudge slow?"



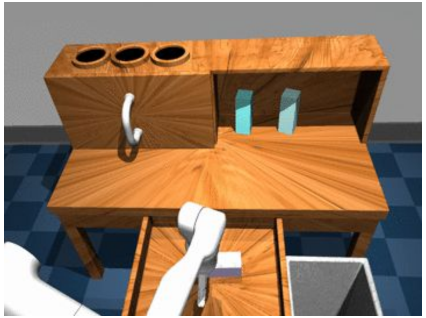
"Nudge + grasp?"



Slide "full"?

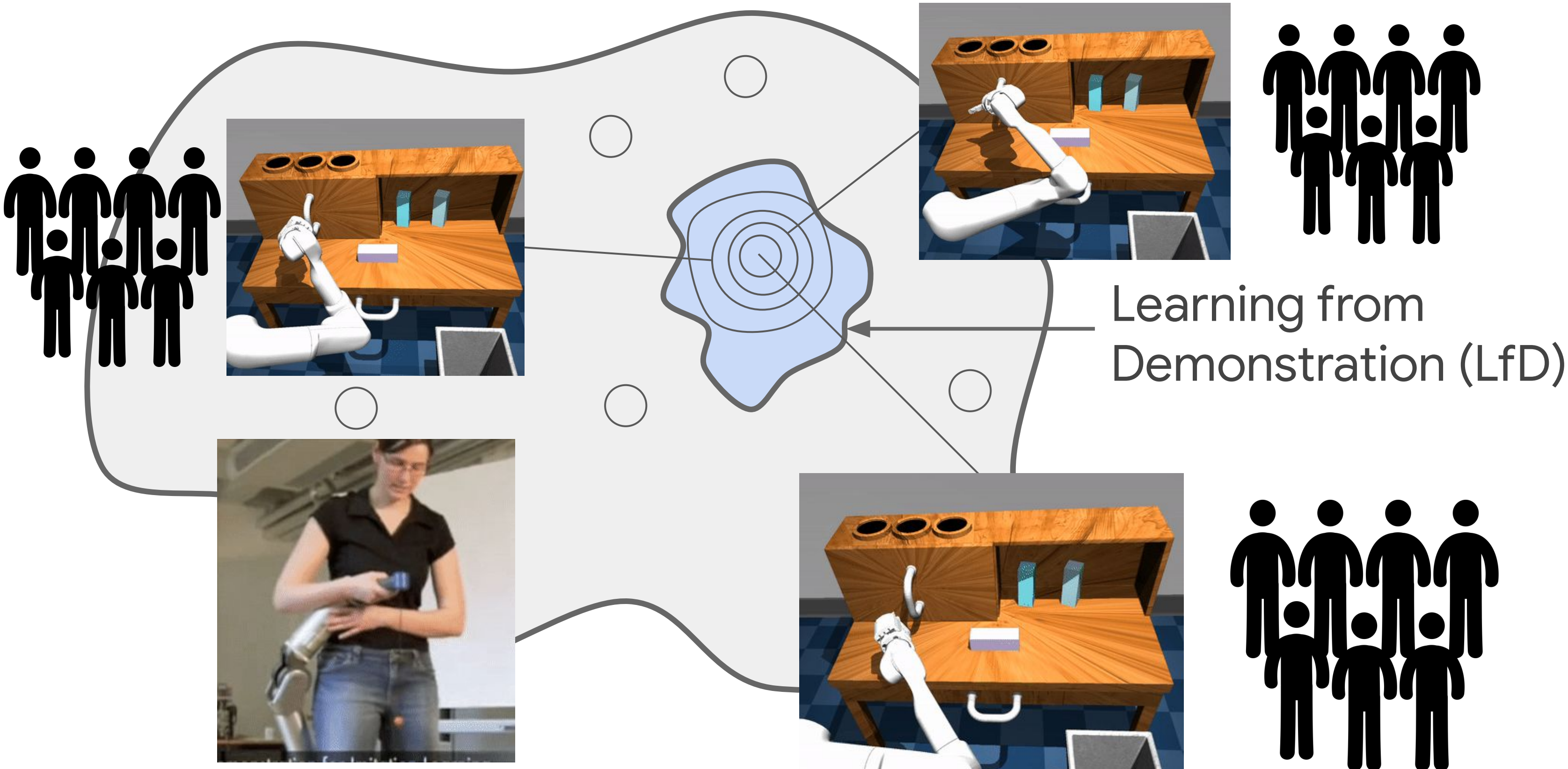


Slide "partial"?



Boundaries between multiple tasks?

How can we cover the continuum?

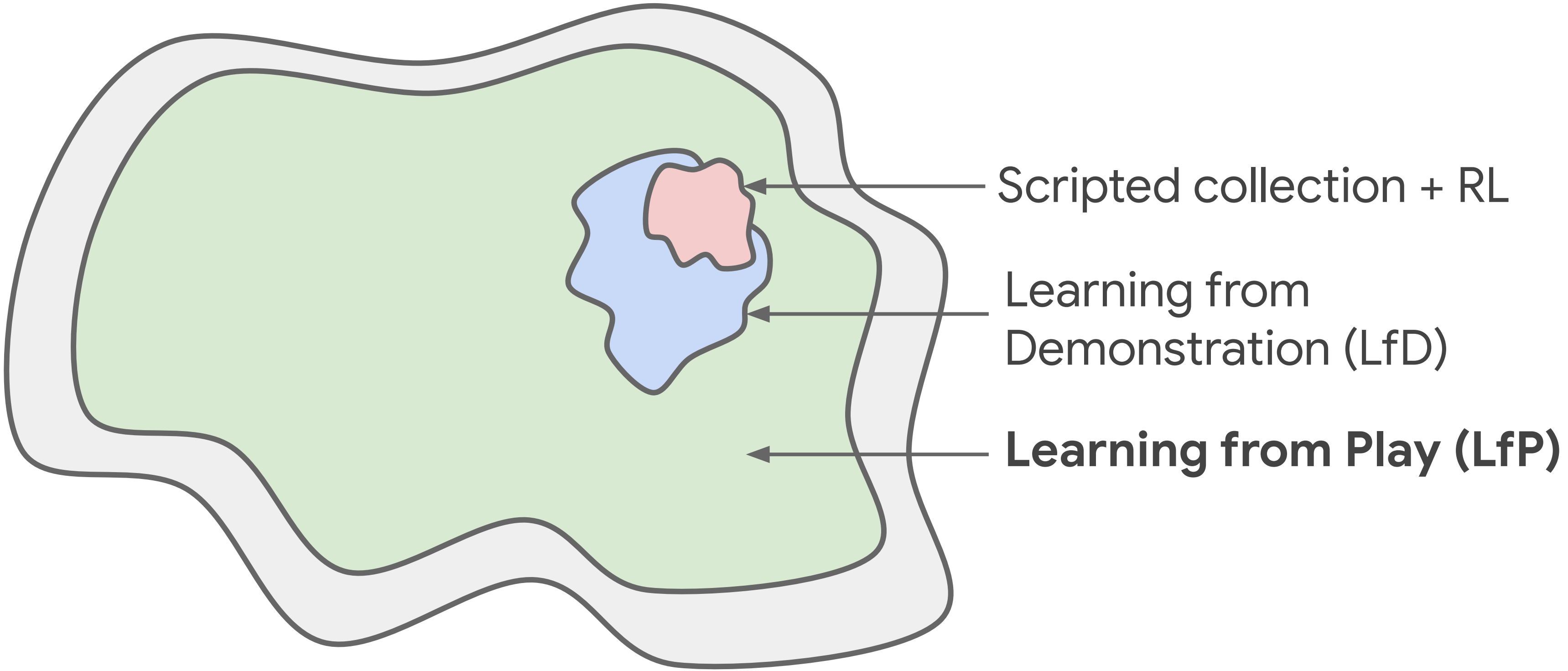


Kinesthetic

[Kober and Peters, 2011]

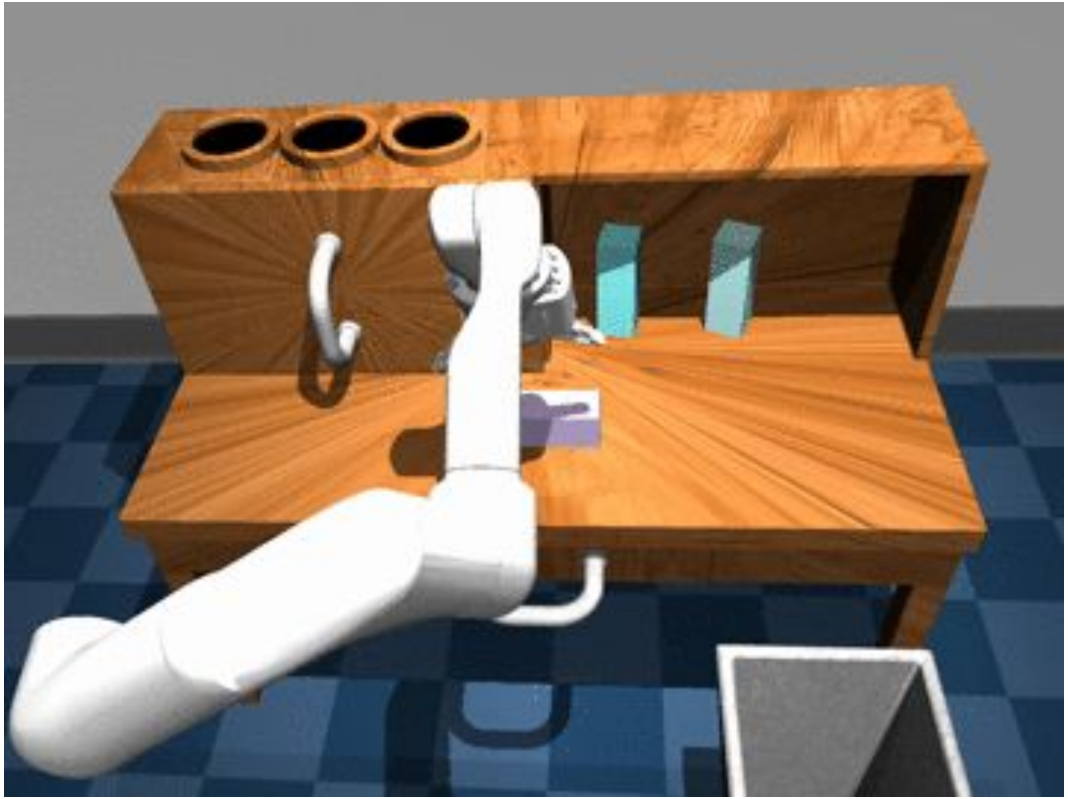
Tele-op, segmented expert demonstrations

How can we cover the continuum?



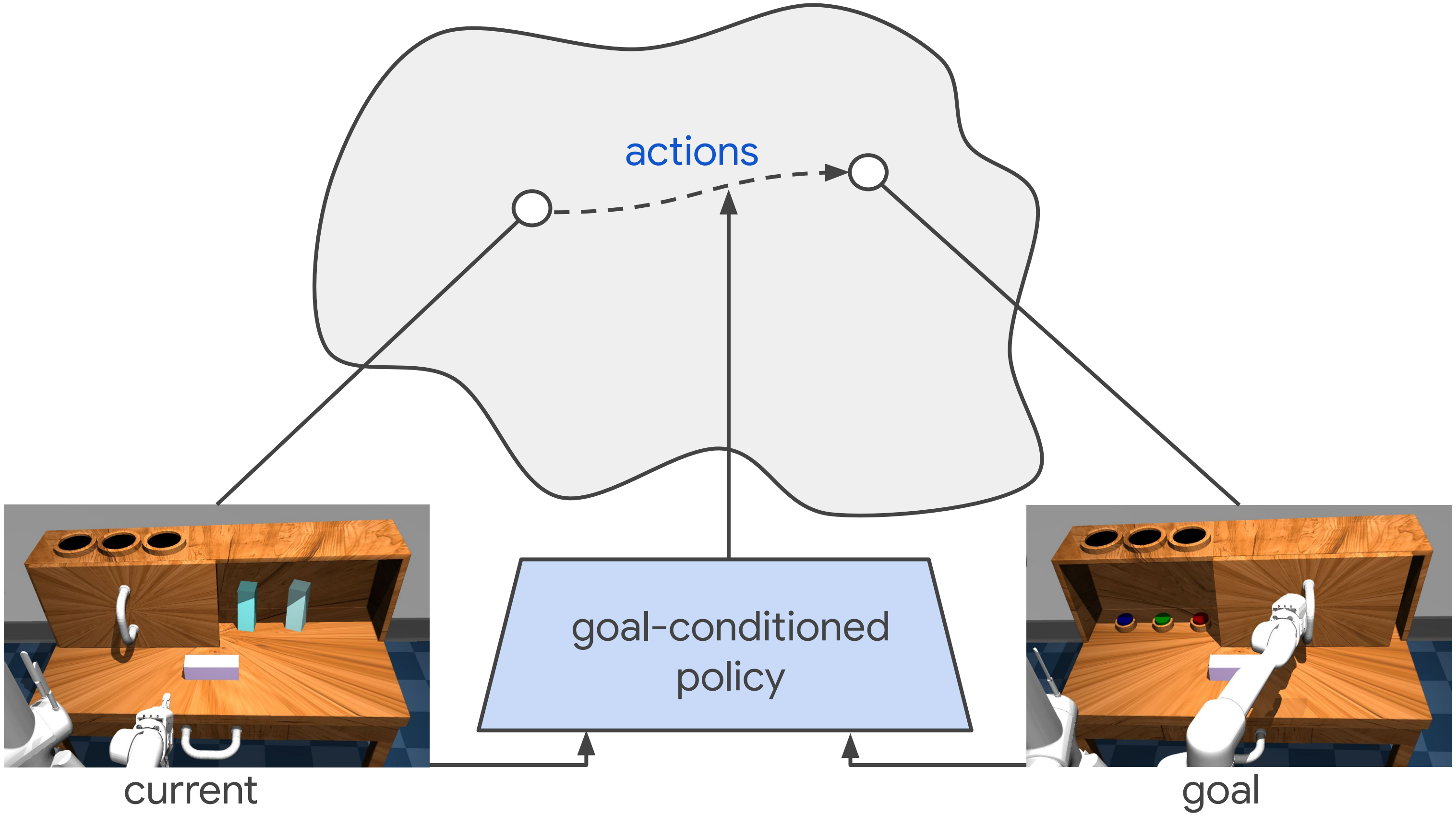
Play data for training

collected from human tele-operation

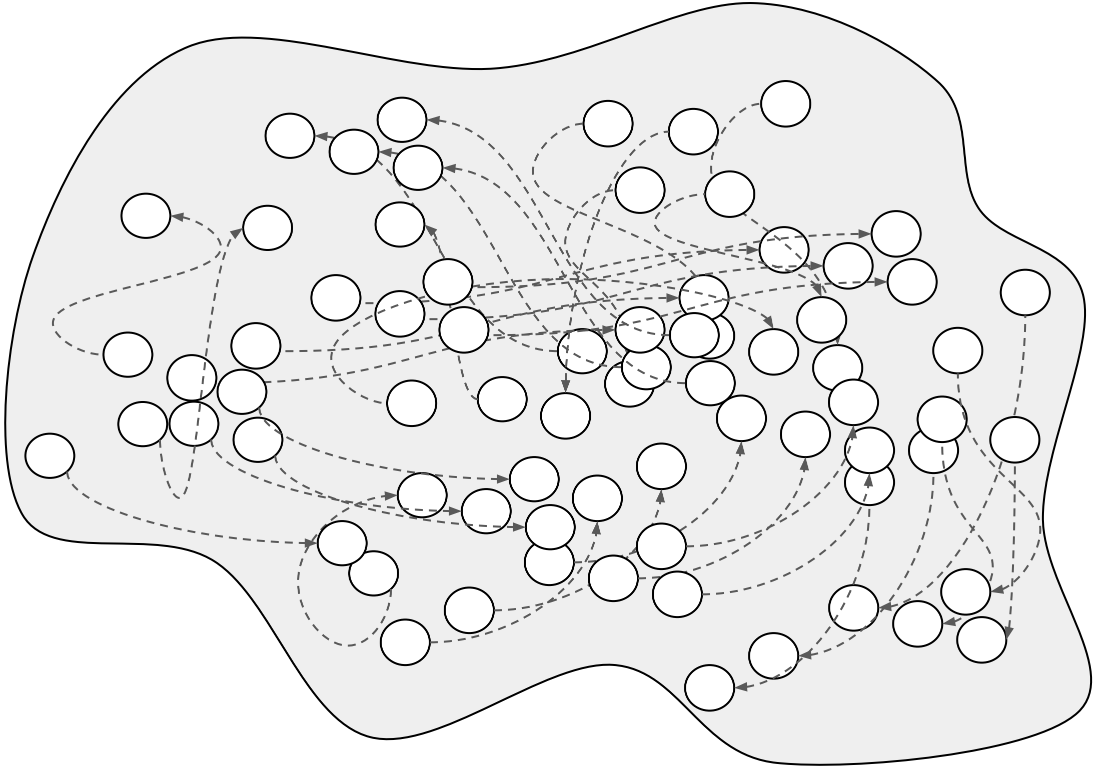


(2.5x speedup)

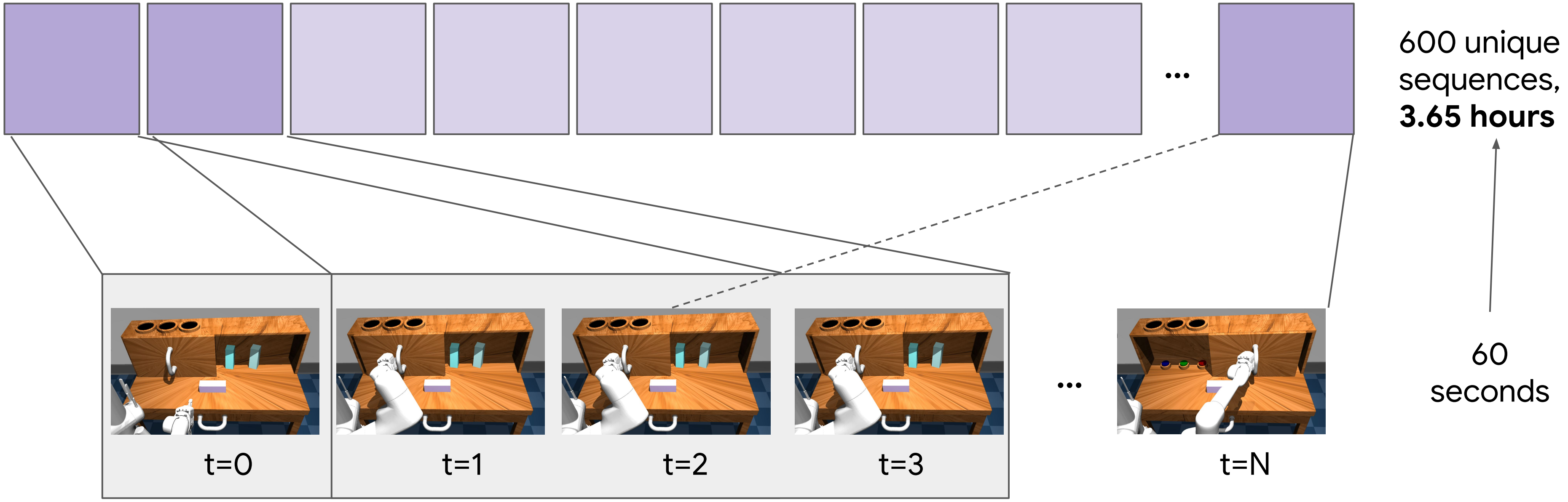
How do we learn control from play?



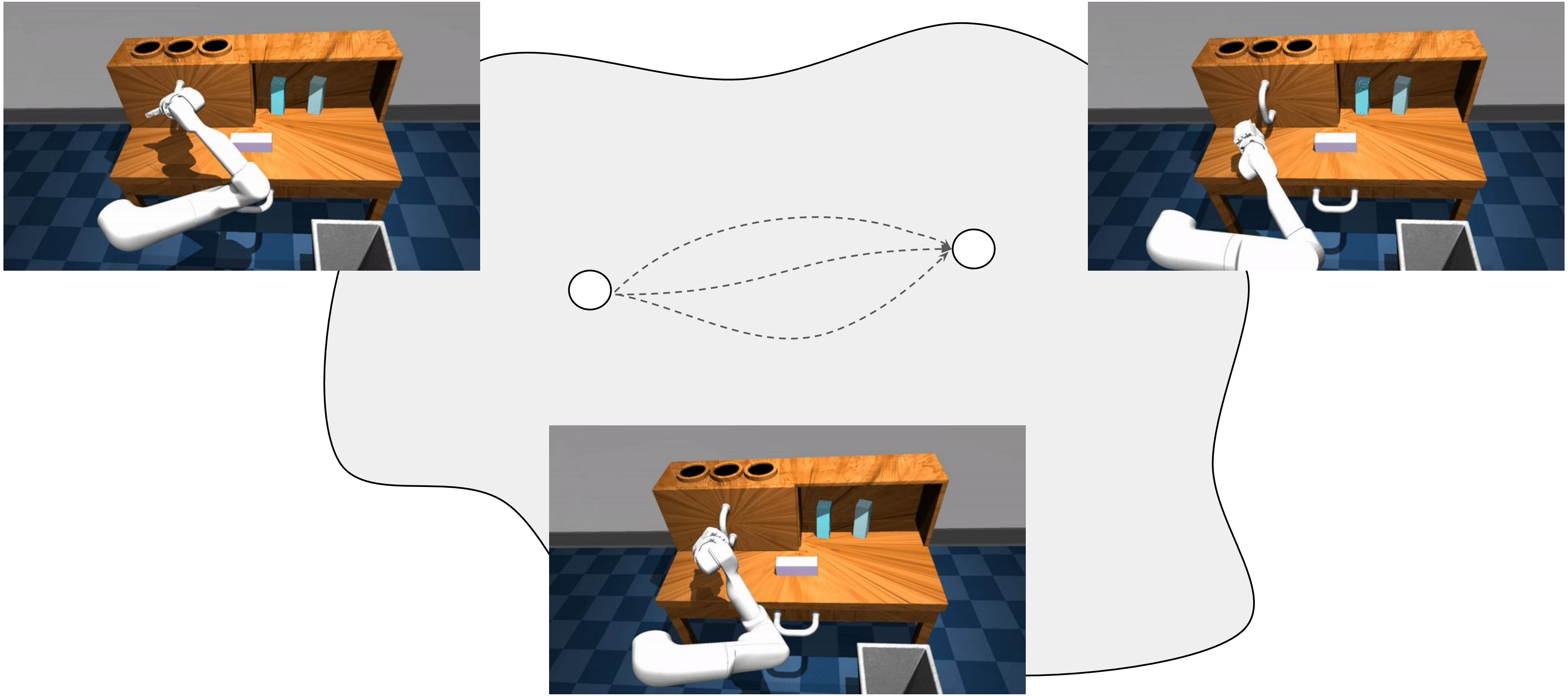
Play covers the continuum



Goal relabeling



Multimodality issue

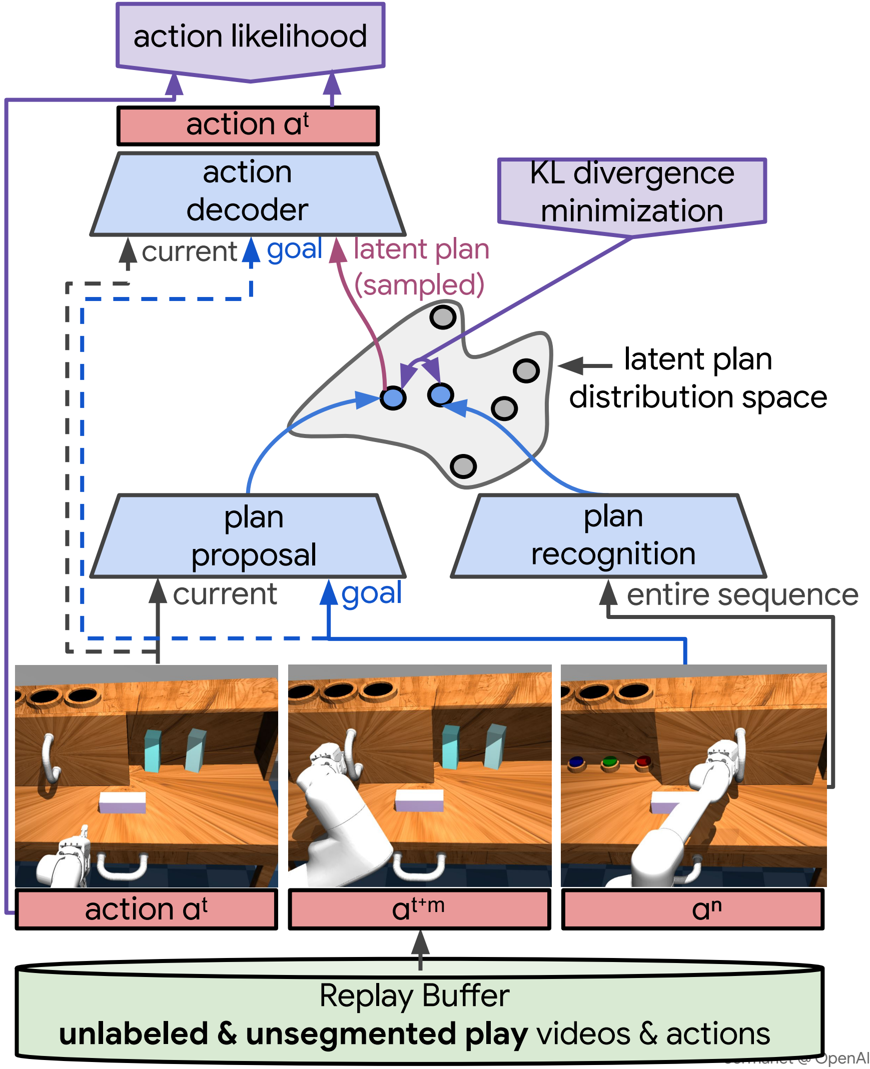


Training Play-LMP

3. **Decode plan**
   to reconstruct actions

2. Learn **latent plans**
   using self-supervision

1. Given **unlabeled play data**

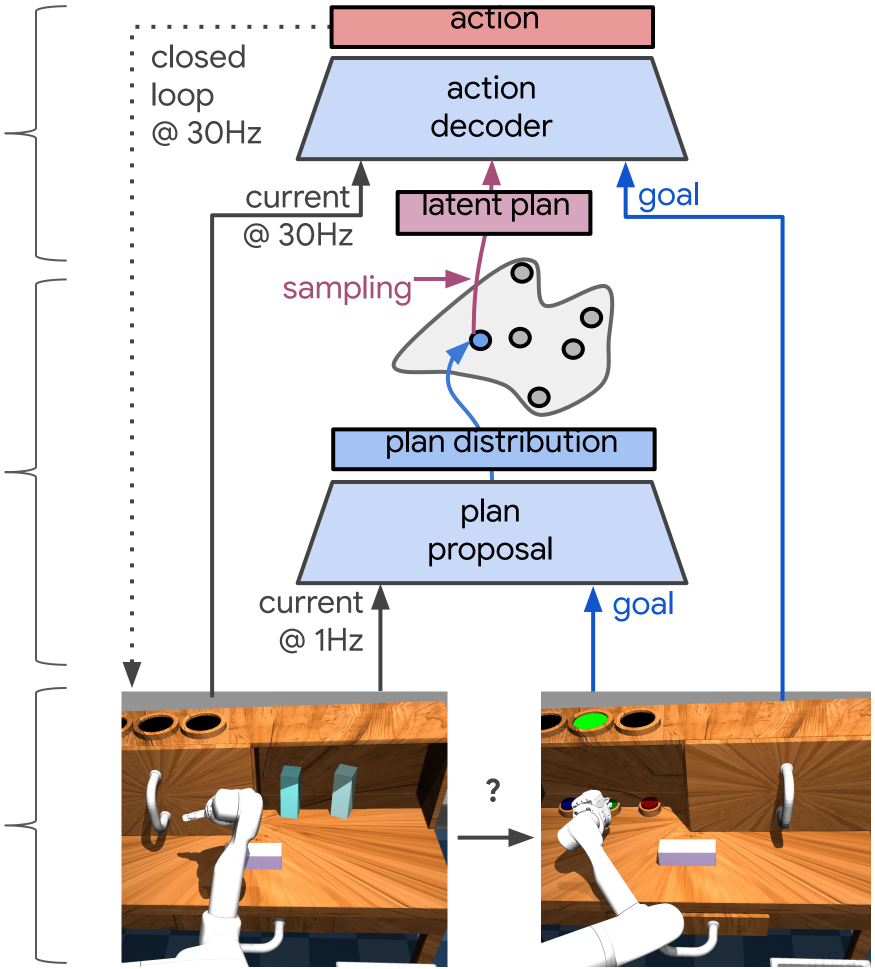


Play-LMP: Test time

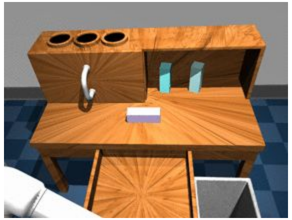
3. **Generate an action** (@30Hz)

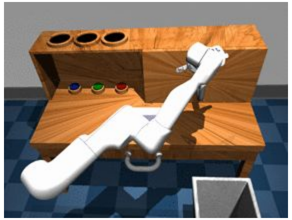
2. **Generate a latent plan** (@1Hz)

1. Given a goal state

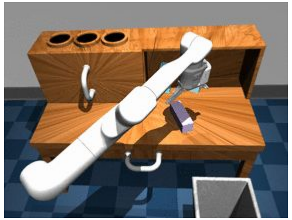


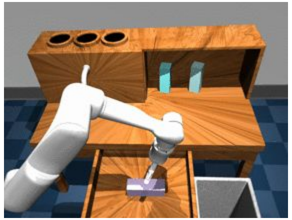
18 tasks (for evaluation only)

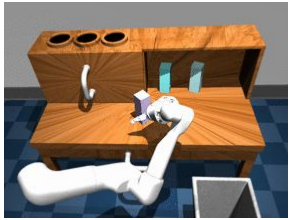

close drawer

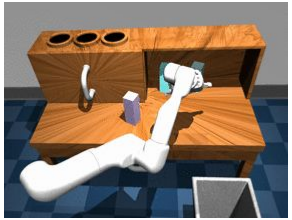

close sliding

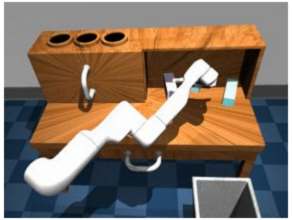

open drawer

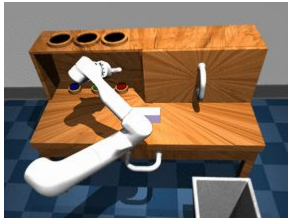

grasp flat

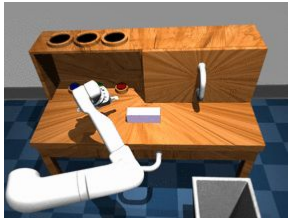

grasp lift


grasp upright

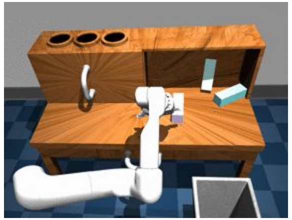

knock

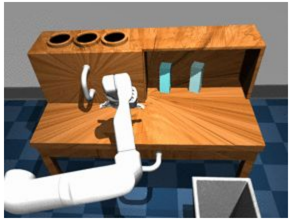

pull out shelf


push blue


push green

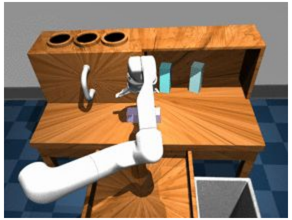

push red

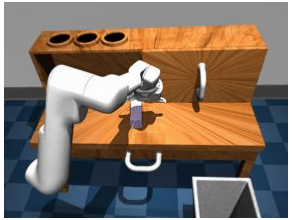

put in shelf

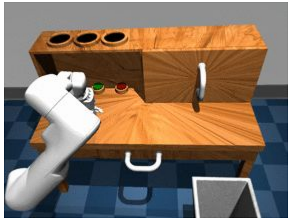

rotate left


rotate right


sliding


sweep

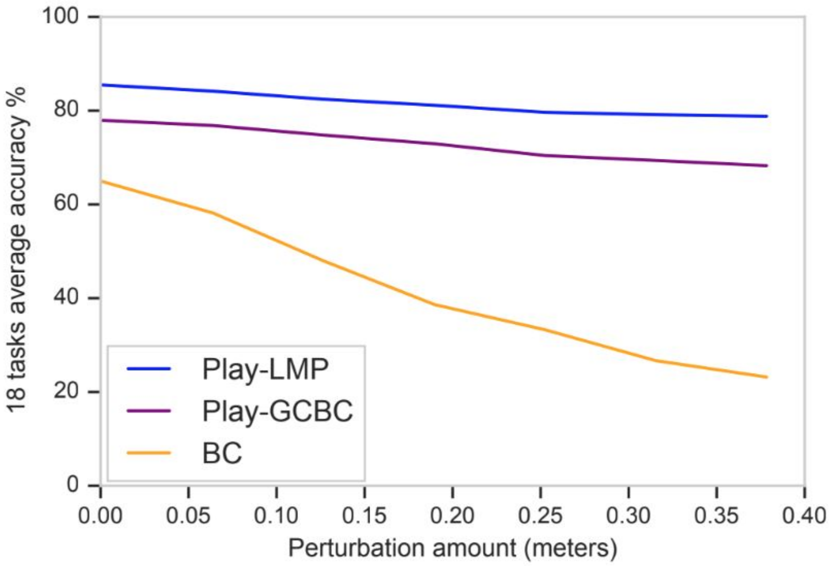

sweep left


sweep right

Quantitative Accuracy

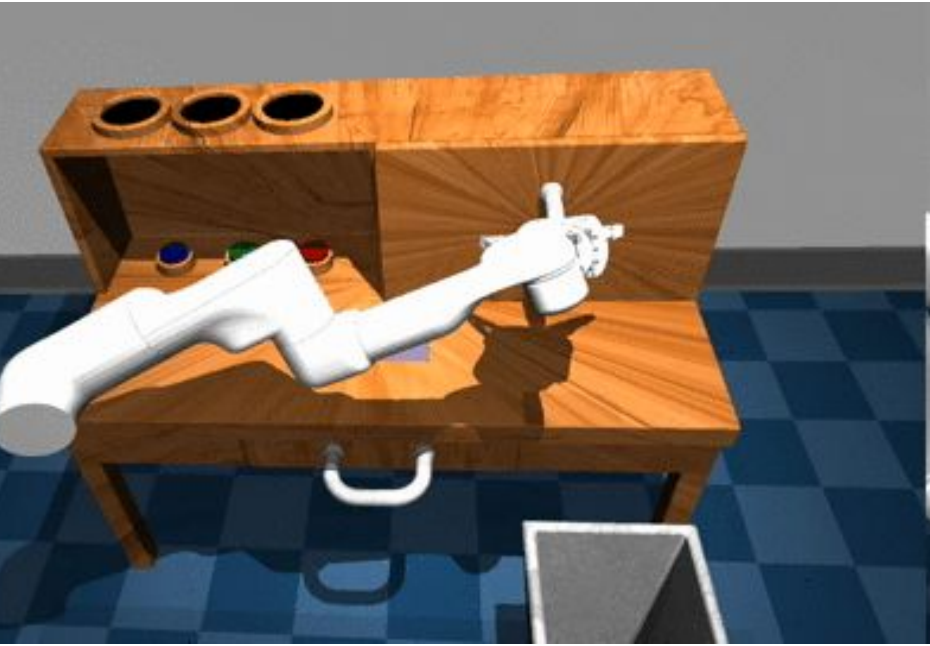
We obtain a **single task-agnostic policy** and evaluate it on 18 zero-shot tasks.

- Play-LMP: **single policy** trained on **cheap unlabelled** data: **85% zero shot**
- Baseline: **18 policies** trained on **expensive labelled** data: **65%**
- When perturbing the start position, the success is:
  - baseline: **23%**
  - Play-LMP: **79%**



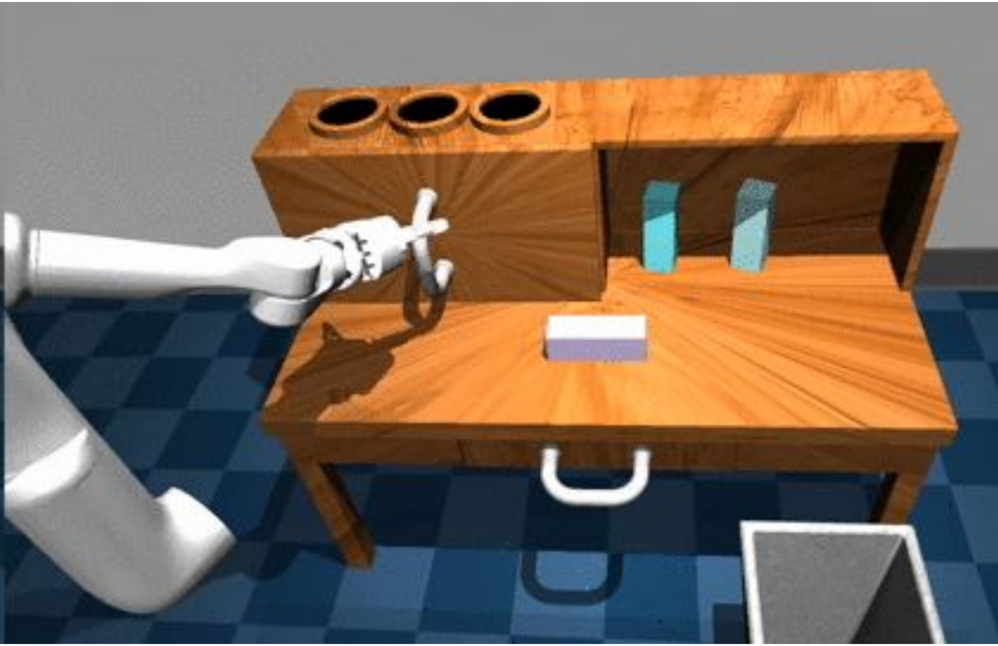
Examples of success runs for Play-LMP

1x



Goal

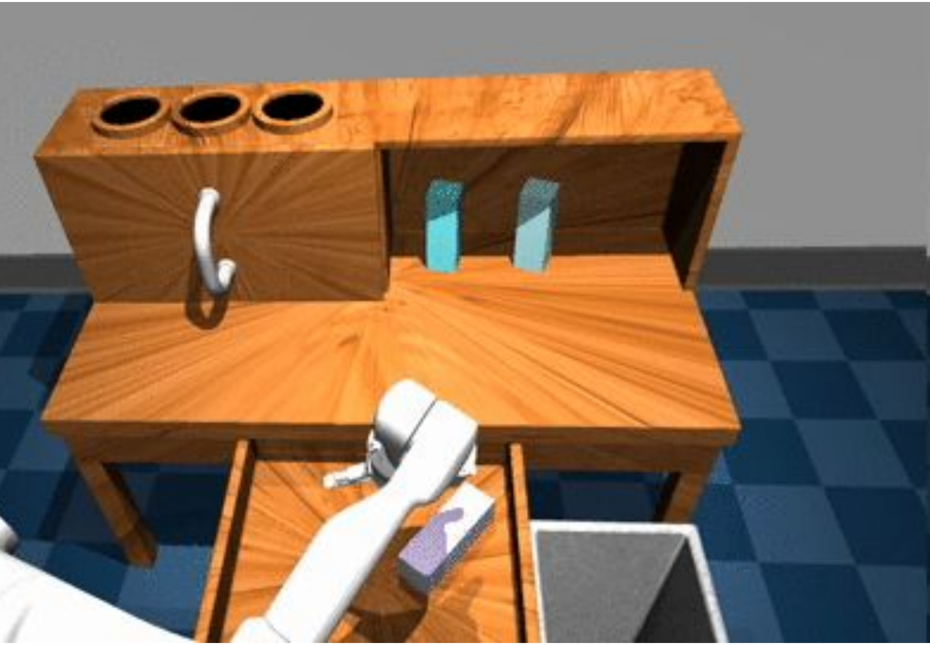
(task: sliding)



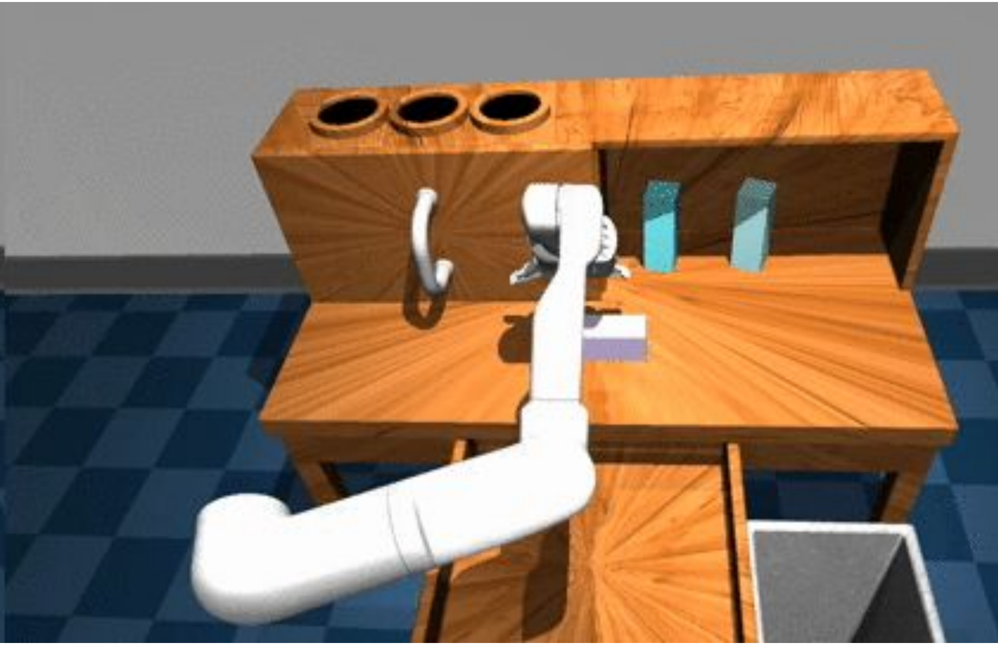
Play-LMP policy

Examples of success runs for Play-LMP

1x



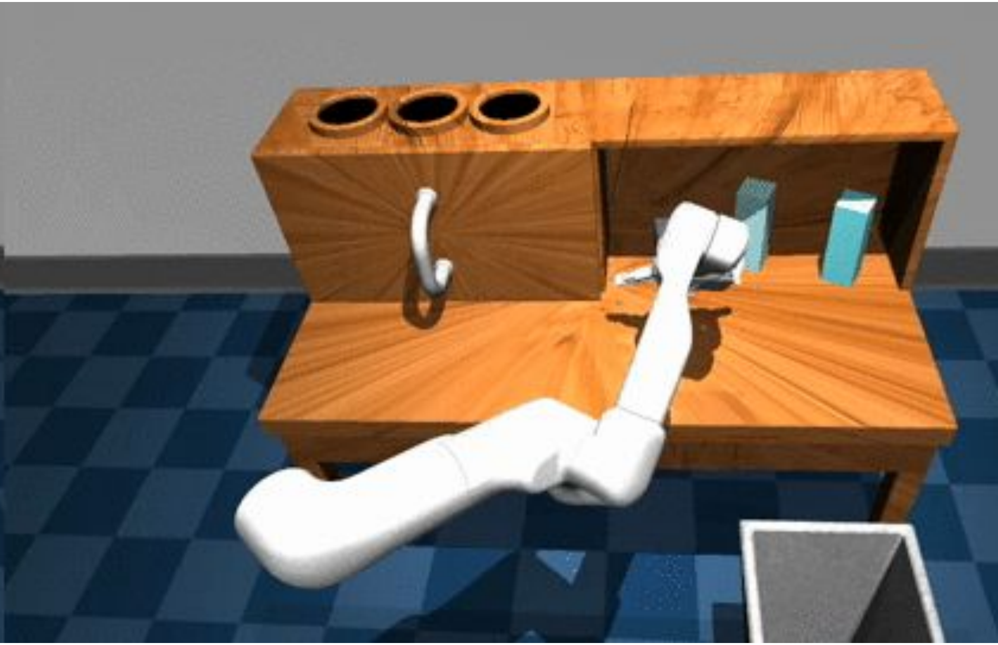
Goal
(task: sweep)



Play-LMP policy

Examples of success runs for Play-LMP

1x

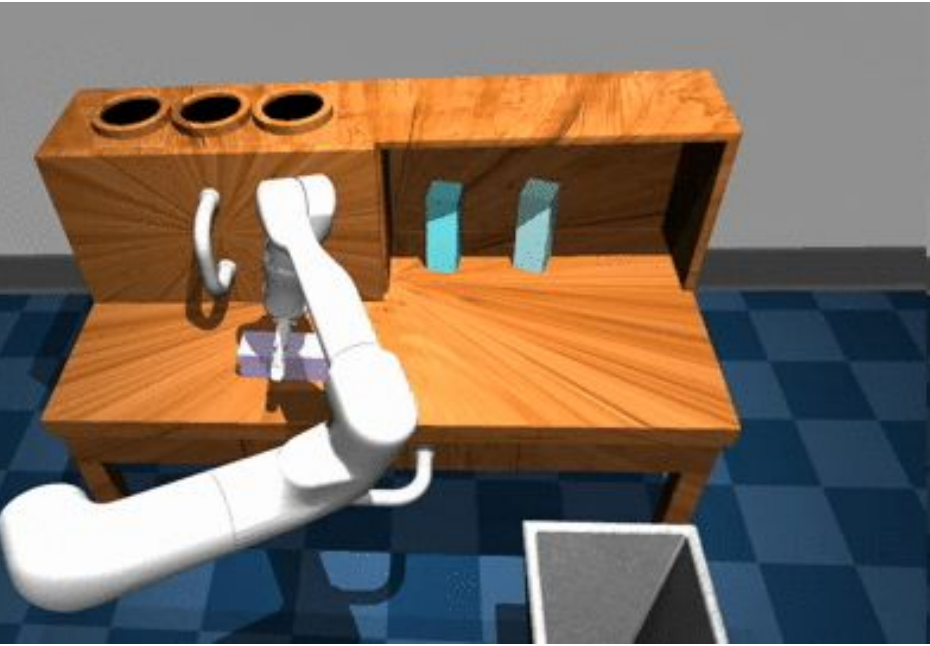


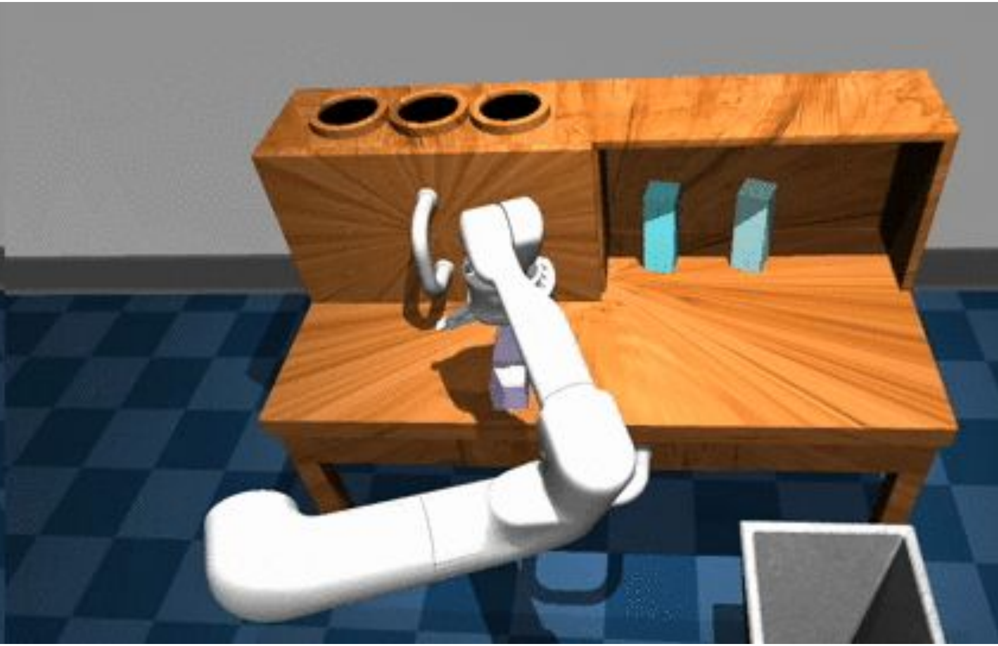
Goal
(task: pull out of shelf)

Play-LMP policy

Examples of success runs for Play-LMP

1x



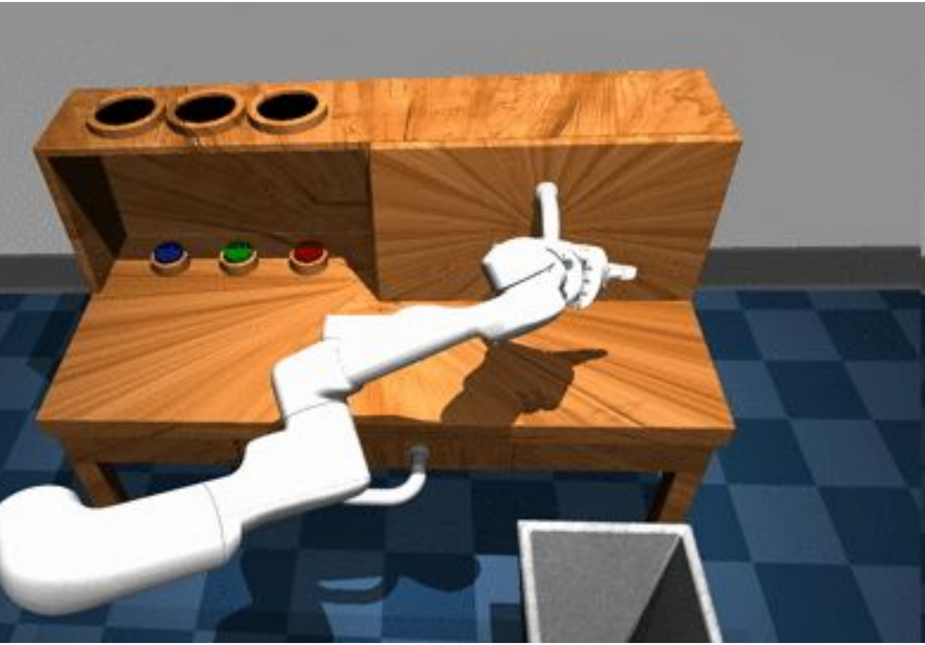


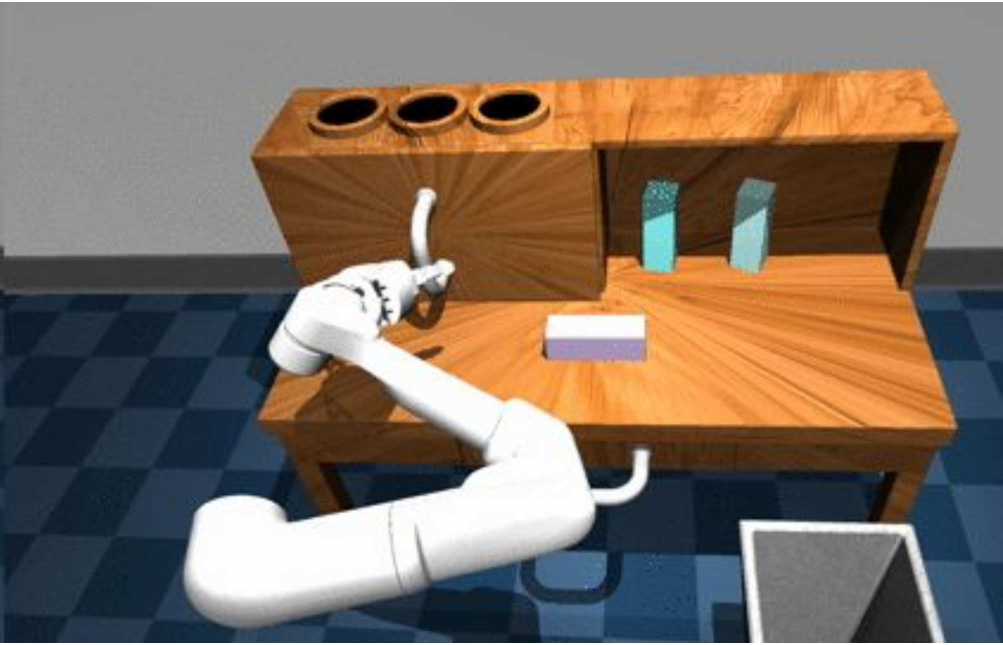
Goal
(task: rotate left)

Play-LMP policy

Some failure cases for Play-LMP

1x



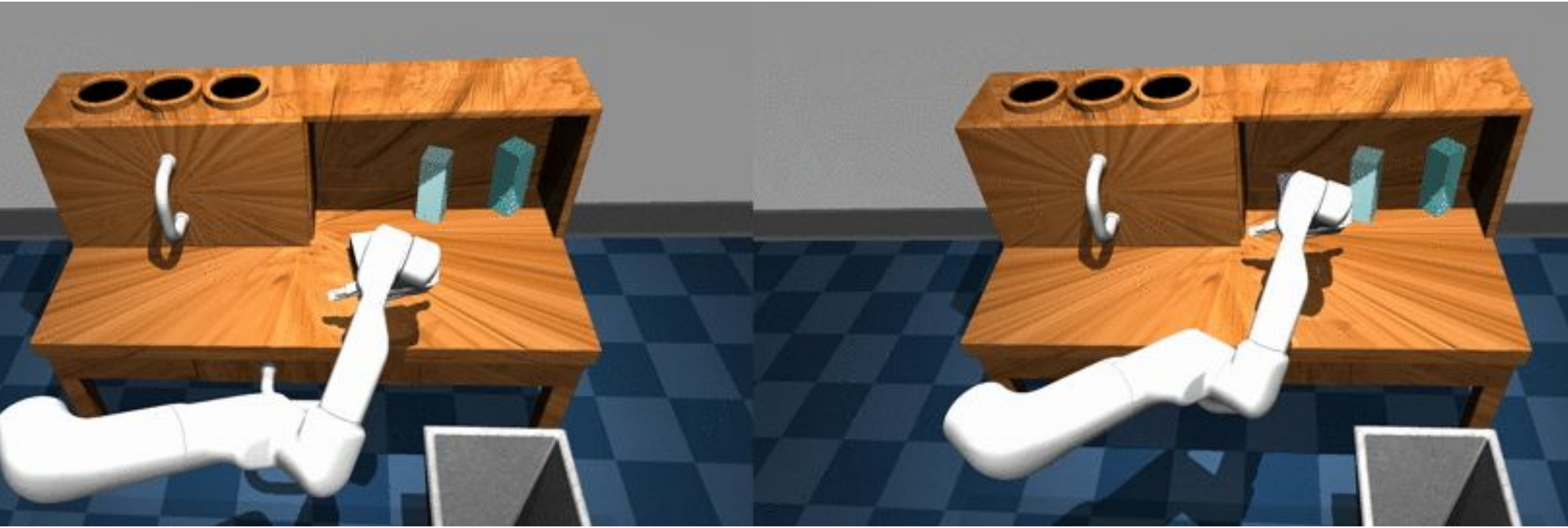


Goal
(task: sliding)

Play-LMP policy

Some failure cases for Play-LMP

1x



Goal
(task: pull out of shelf)

Play-LMP policy

Retrying behavior emerging from Play-LMP

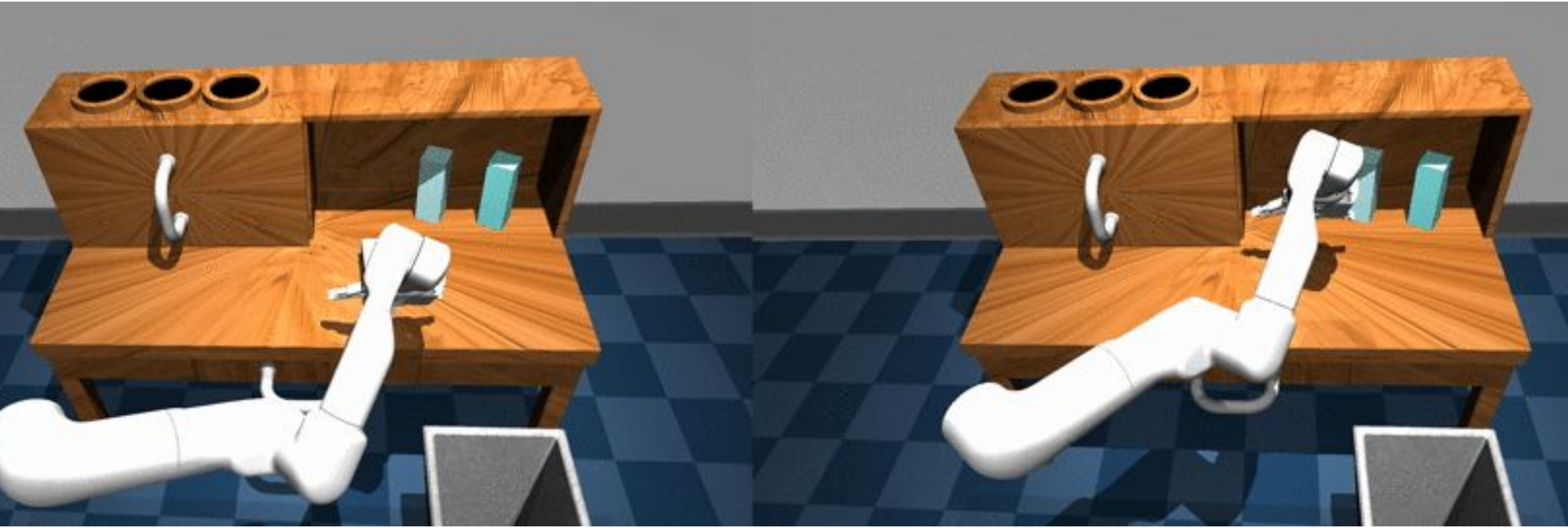
1x

Goal
(task: sliding)

Play-LMP policy

Retrying behavior emerging from Play-LMP

1x



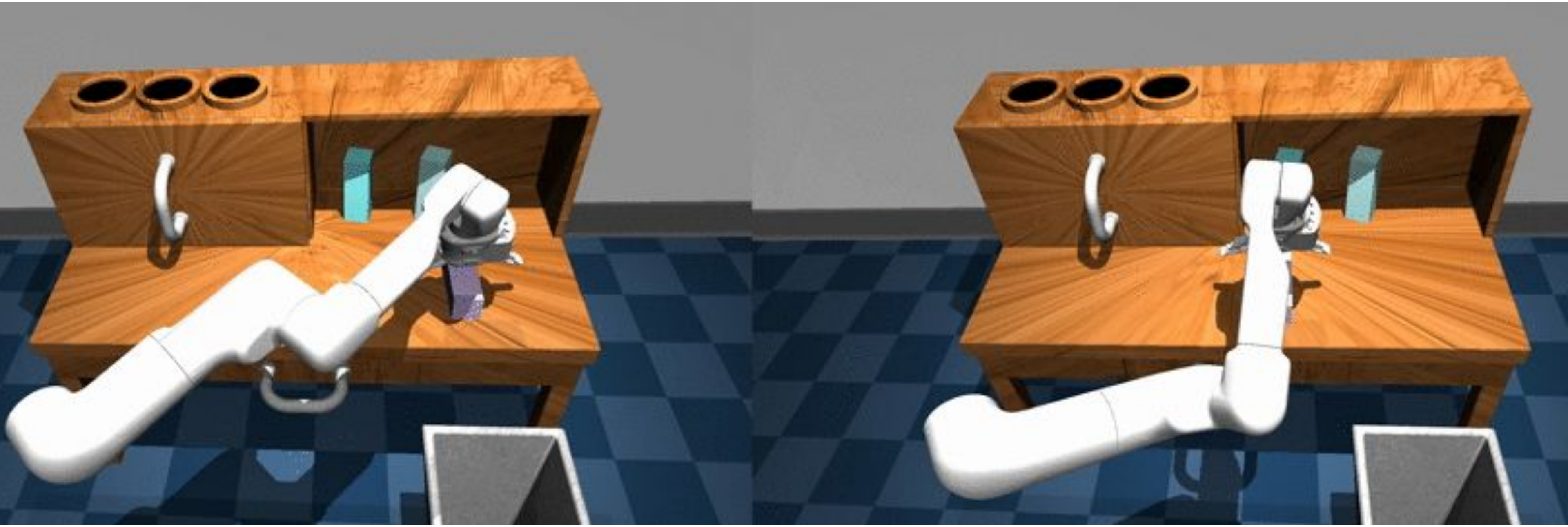
Goal

(task: pull out of shelf)

Play-LMP policy

Retrying behavior emerging from Play-LMP

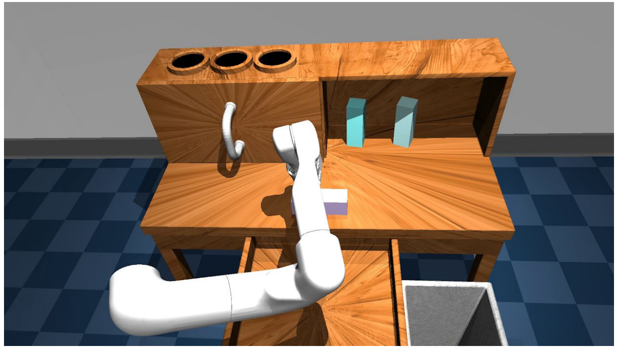
1x



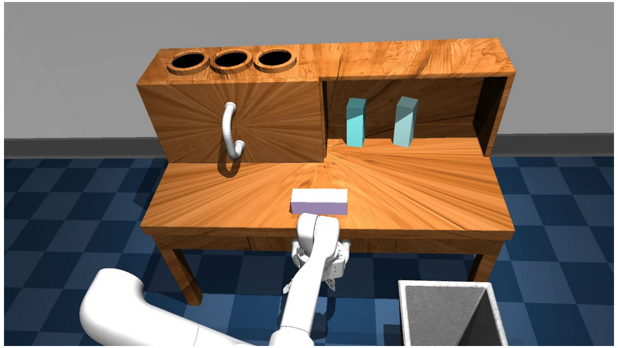
Goal
(task: sweep right)

Play-LMP policy

Composing 2 skills: grasp + close drawer



+



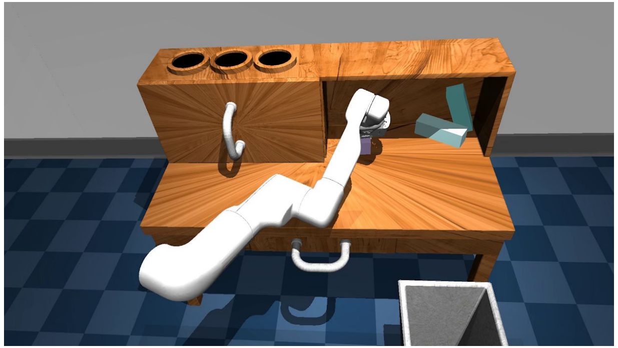
Goals

=

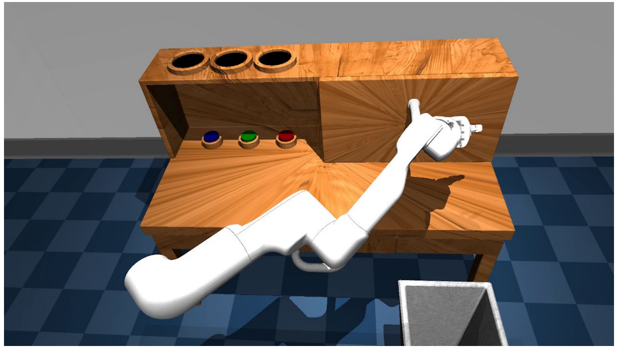
1x

Play-LMP policy

Composing 2 skills: put in shelf + close sliding



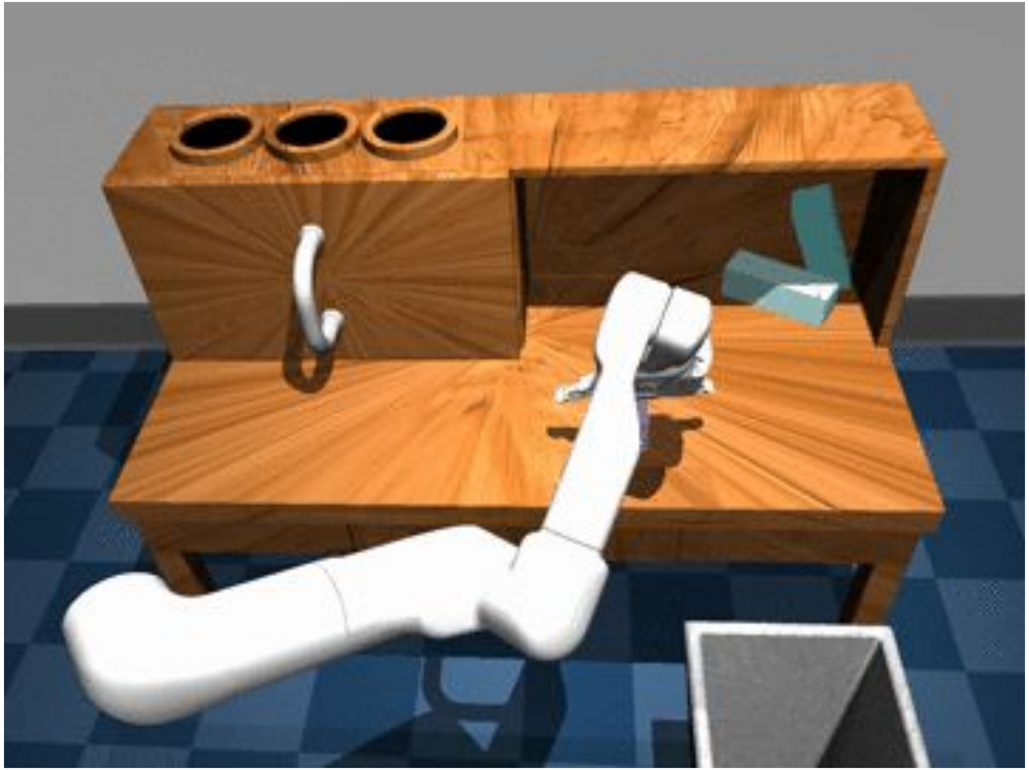
+



Goals

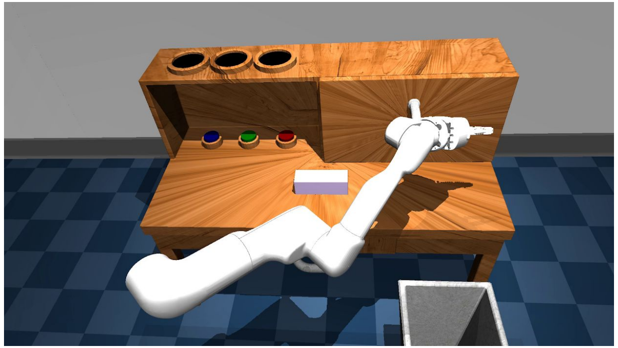
=

1x

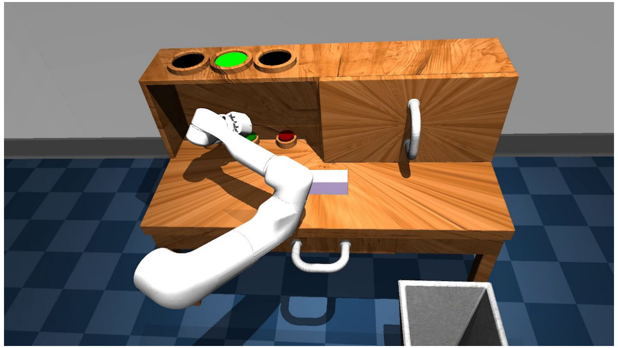


Play-LMP policy

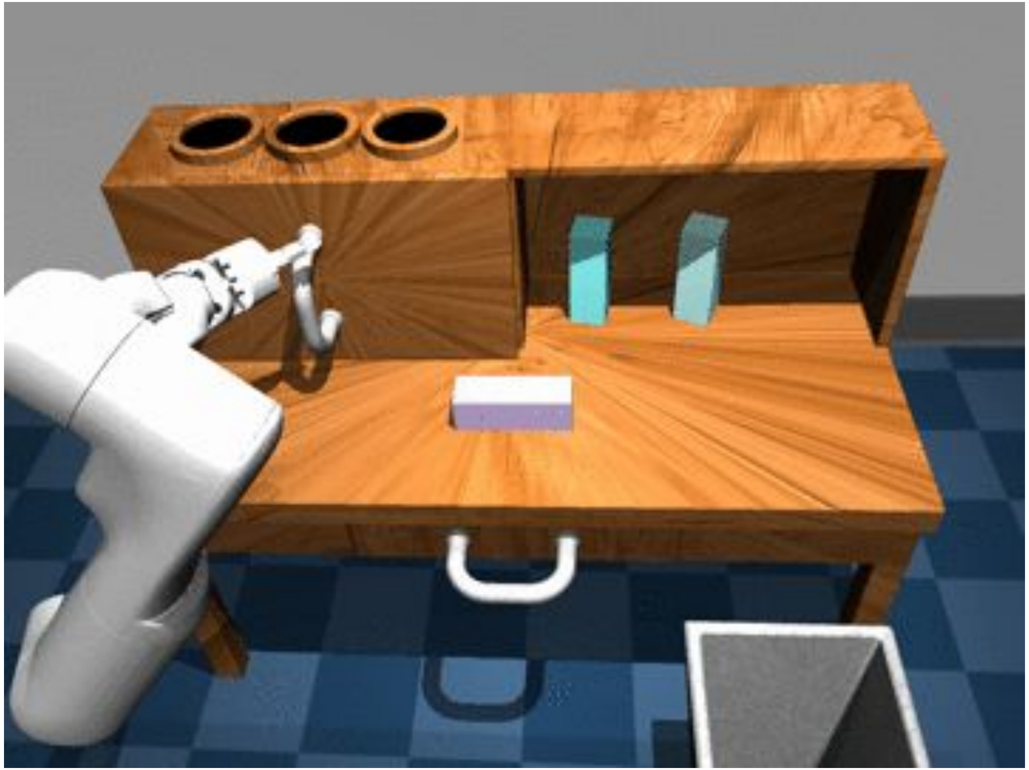
Composing 2 skills: open sliding + push green



+



=

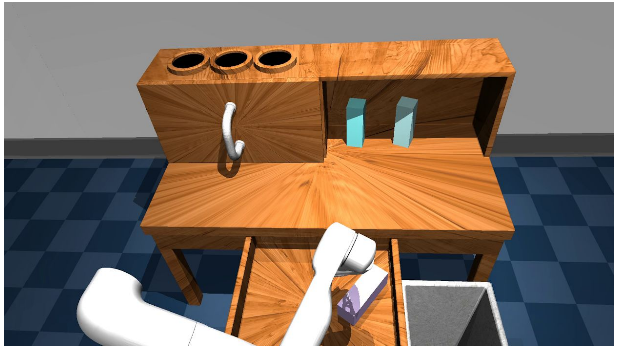


1x

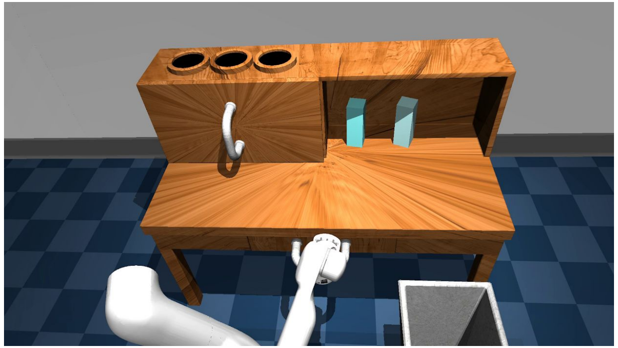
Goals

Play-LMP policy

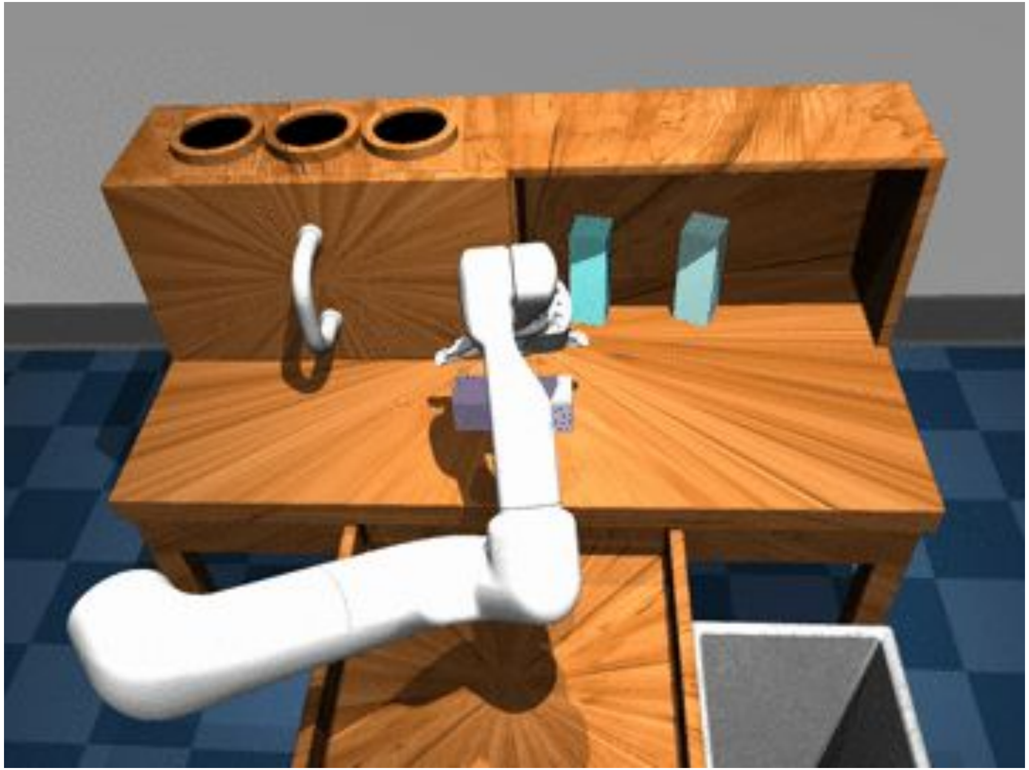
Composing 2 skills: sweep + close drawer



+



=

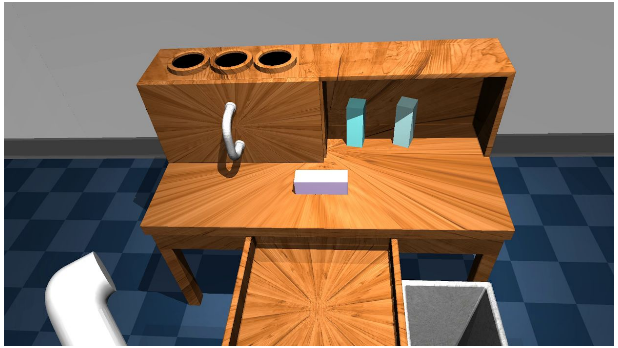


1x

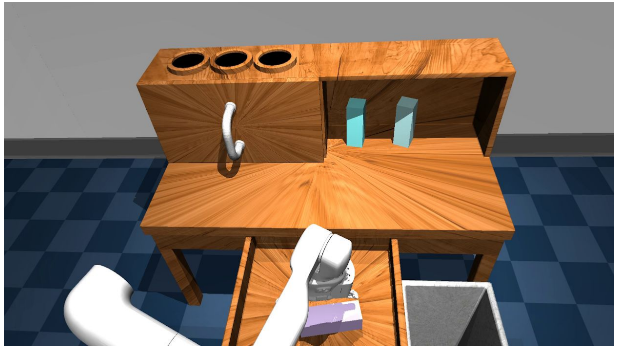
Goals

Play-LMP policy

Composing 2 skills: drawer open + sweep



+



Goals

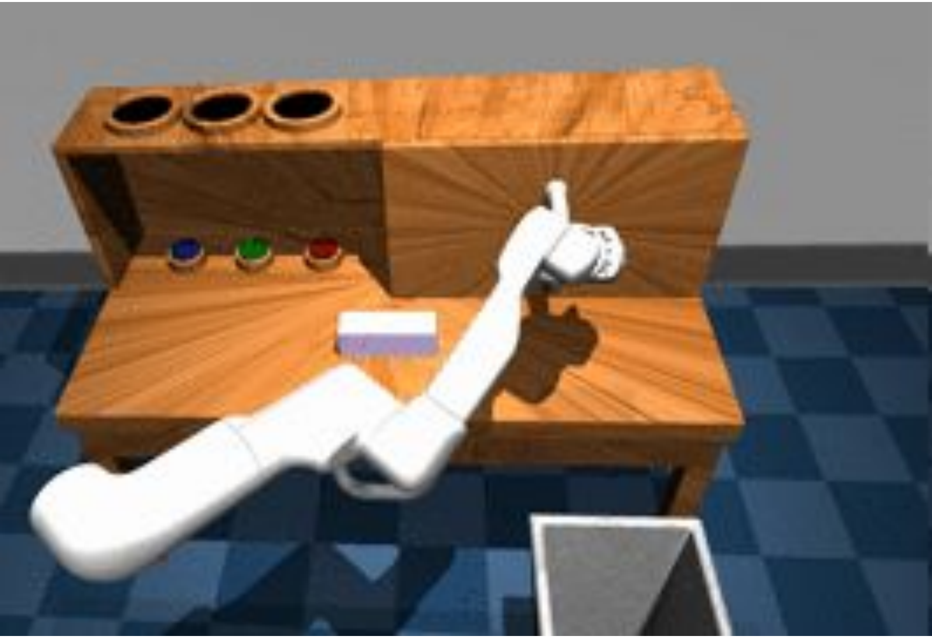
=

1x

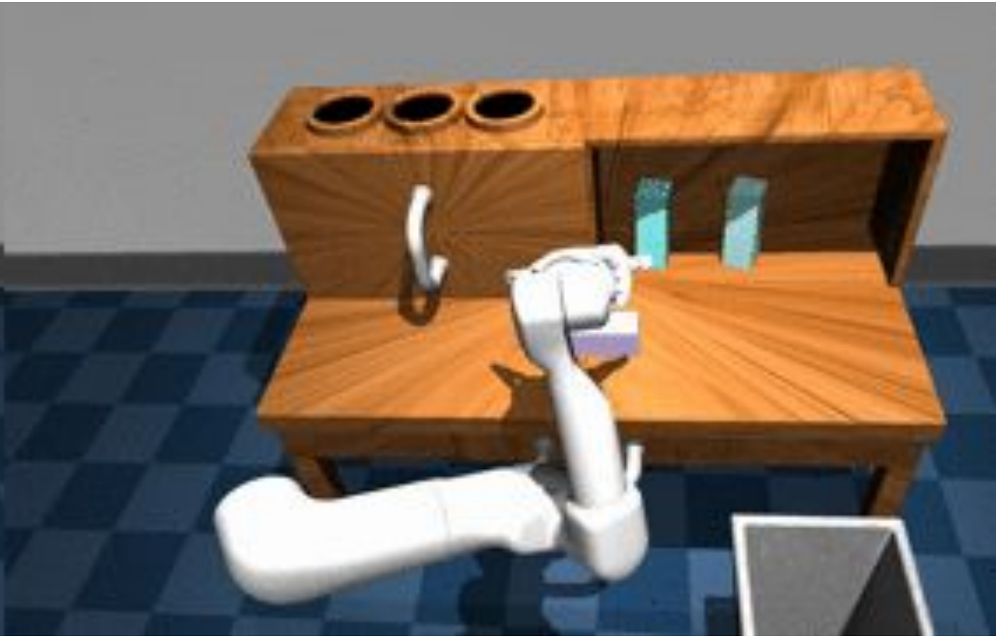
Play-LMP policy

8 skills in a row

1.5x

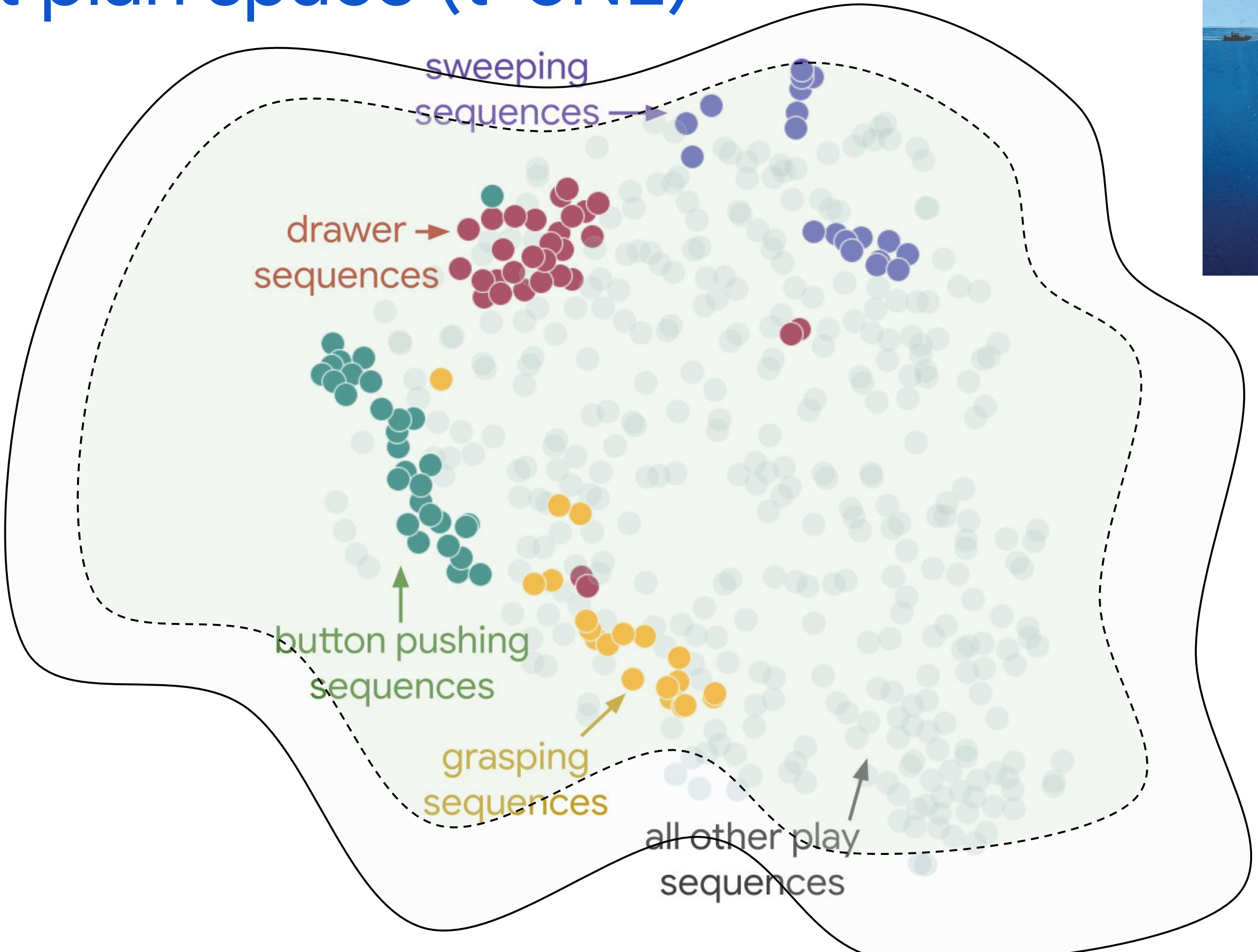


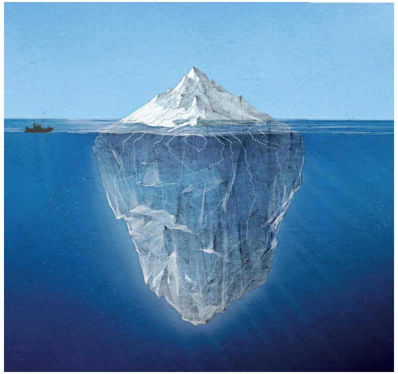
Goal



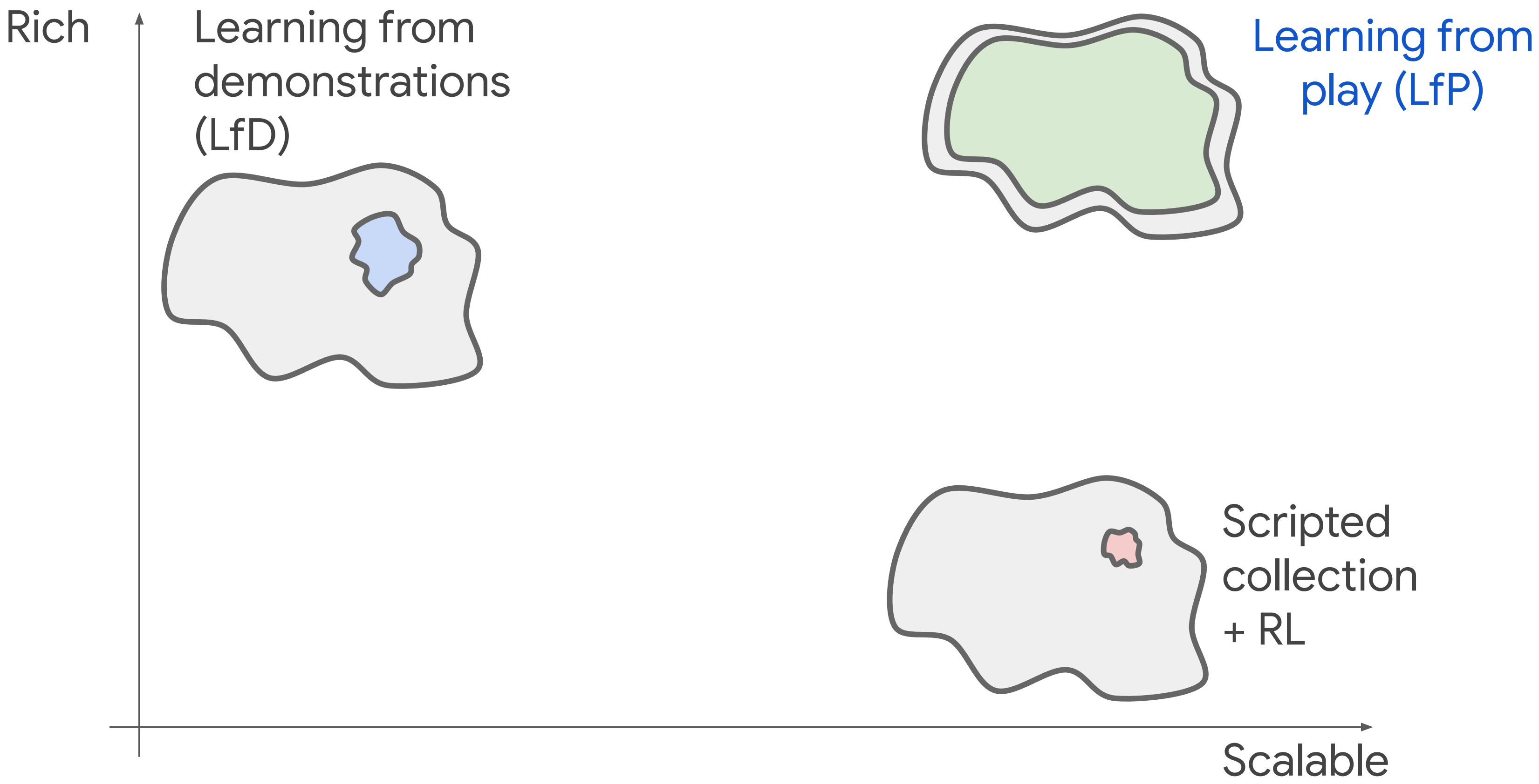
Play-LMP policy

Latent plan space (t-SNE)

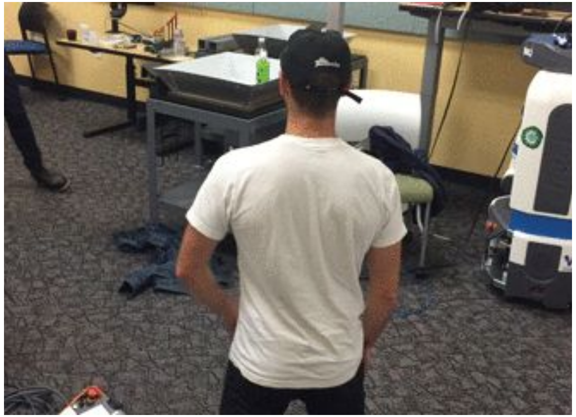


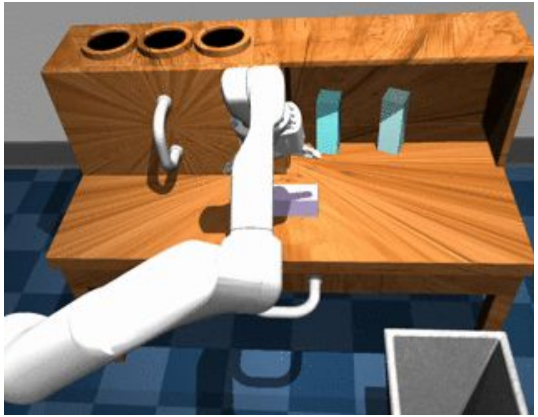


Richness & Scalability of Data



Recipe: Self-Supervision + Play



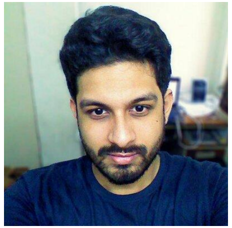


Takeaways

- **Self-Supervision + Play** recipe:
    - Self-supervise on lots of unlabeled data
    - Use play data
- **Delay definitions** of tasks, states or attributes,

  Let self-supervision organize continuous spaces:
    - **Continuum of states and attributes**
    - **Continuum of skills**


Corey Lynch


Debidatta Dwibedi

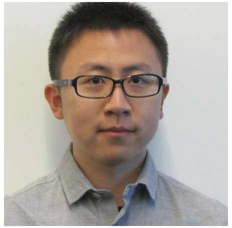

Soeren Pirk


Jonathan Tompson


Mohi Khansari


Yusuf Aytar

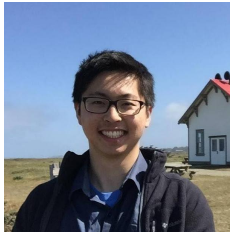

Yevgen Chebotar


Yunfei Bai


Jasmine Hsu


Eric Jang


Vikash Kumar


Ted Xiao


Stefan Schaal


Andrew Zisserman


Sergey Levine


Pierre Sermanet

Questions?

g.co/robotics

sermanet.github.io

sermanet@google.com