Model-Based Reinforcement Learning
for Multi-Task and Meta RL

CS 330

Logistics

Homework 3 due **tonight.**

Project milestone due **next Wedesday.**

The Plan

Model-based RL

and how it can be used for multi-task & meta-learning

Model-based RL with image observations

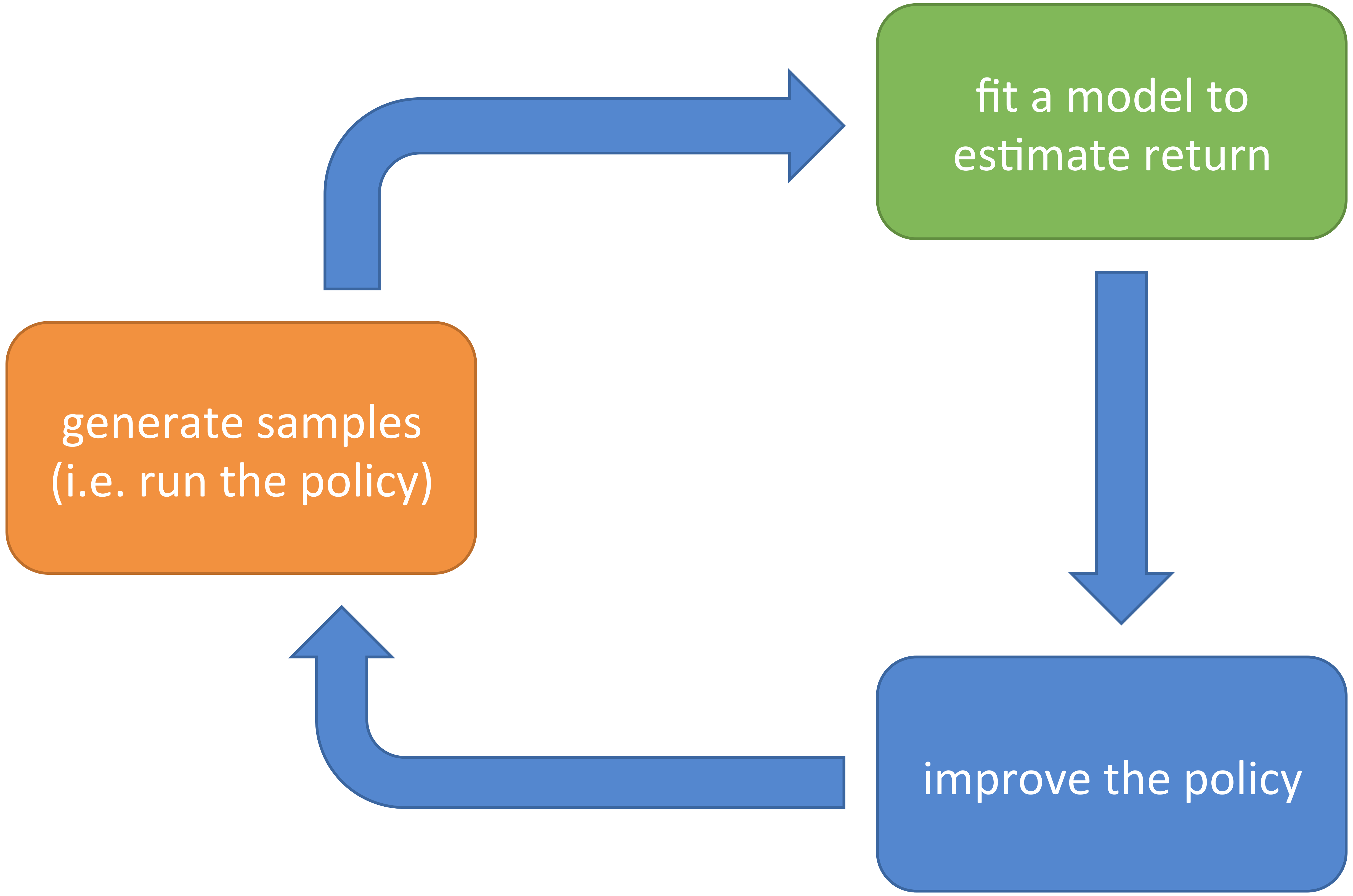
or other high-dimensional inputs

Model-based meta-RL

Lecture objectives:

- understand how to use & implement model-based RL
- challenges and strategies for model-based RL in high-dimensional spaces
- understand how model-based RL relates to multi-task & meta learning

Recall: The anatomy of a reinforcement learning algorithm



compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)
fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)
estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)
$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)
optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

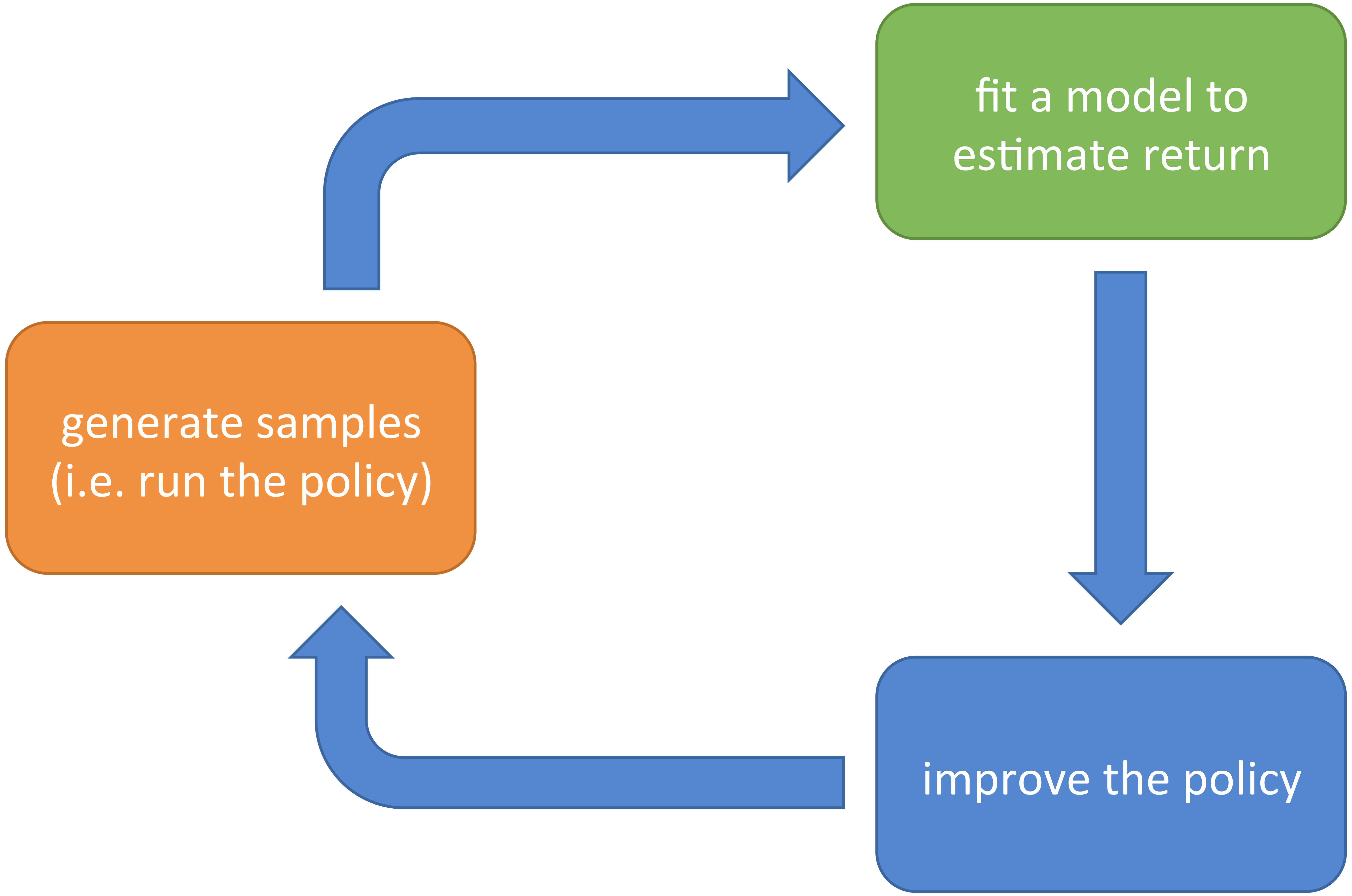
Previous lectures: focus on model-free RL methods (policy gradient, Q-learning)

This lecture: focus on **model-based RL** methods

Model-based RL

Main idea: learn model of environment

Why?
- often leads to better efficiency
- model can be reused



estimate $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$ using $f_\phi(\mathbf{s},\mathbf{a})$

supervised learning

$$\min_\phi \sum_i ||f_\phi(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{s}'_i||^2$$

optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

What does this have to do with multi-task RL and meta RL?

Recall: What is a reinforcement learning task?

Reinforcement learning

action space　　dynamics

A task: $\mathcal{T}_i \triangleq \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p_i(\mathbf{s}' | \mathbf{s}, \mathbf{a}), r_i(\mathbf{s}, \mathbf{a})\}$

state space　　initial state distribution　　reward

a Markov decision process

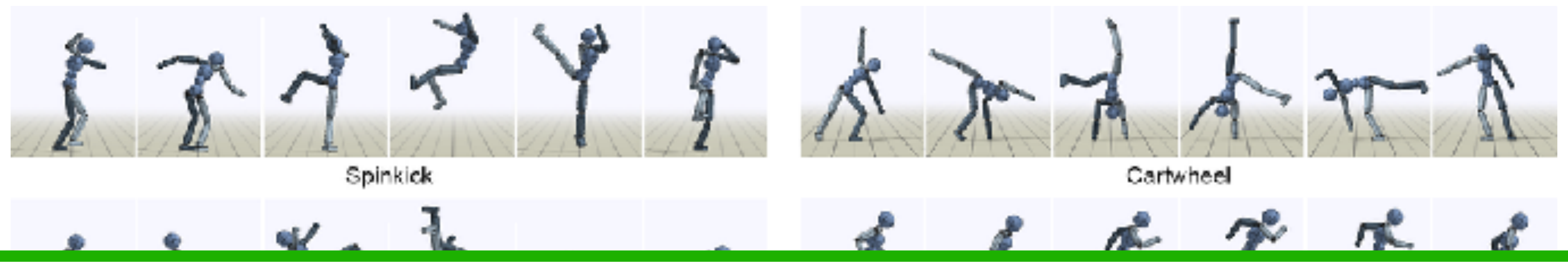
Observation: In many situations: $p_i(\mathbf{s}' | \mathbf{s}, \mathbf{a})$ does not vary across tasks.

The real world:

- object manipulation
- legged locomotion
- navigation

(when env is fully-observed)

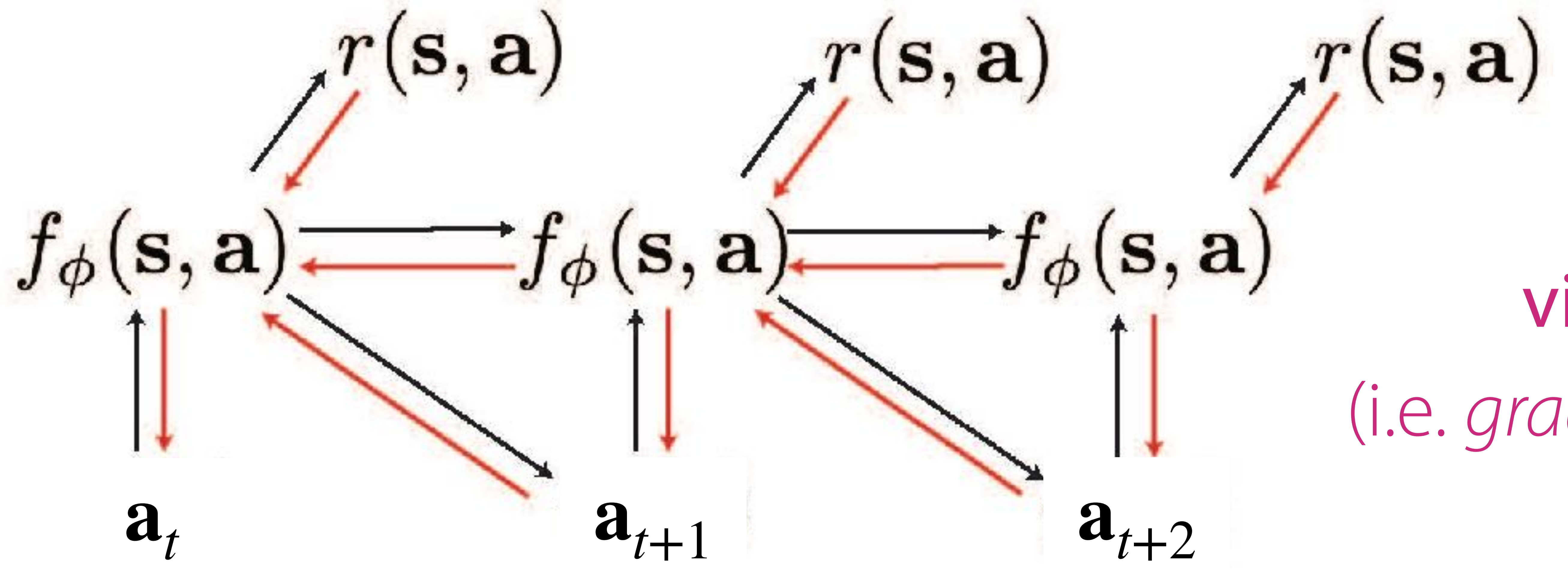
Character animation maneuvers



Task-directed dialog.

In these cases, estimating the model is a single-task problem!

Approach 1: Optimize over actions using model: $\max_{\mathbf{a}_{t:t+H}} \sum_t r(\mathbf{s}_t, \mathbf{a}_t)$

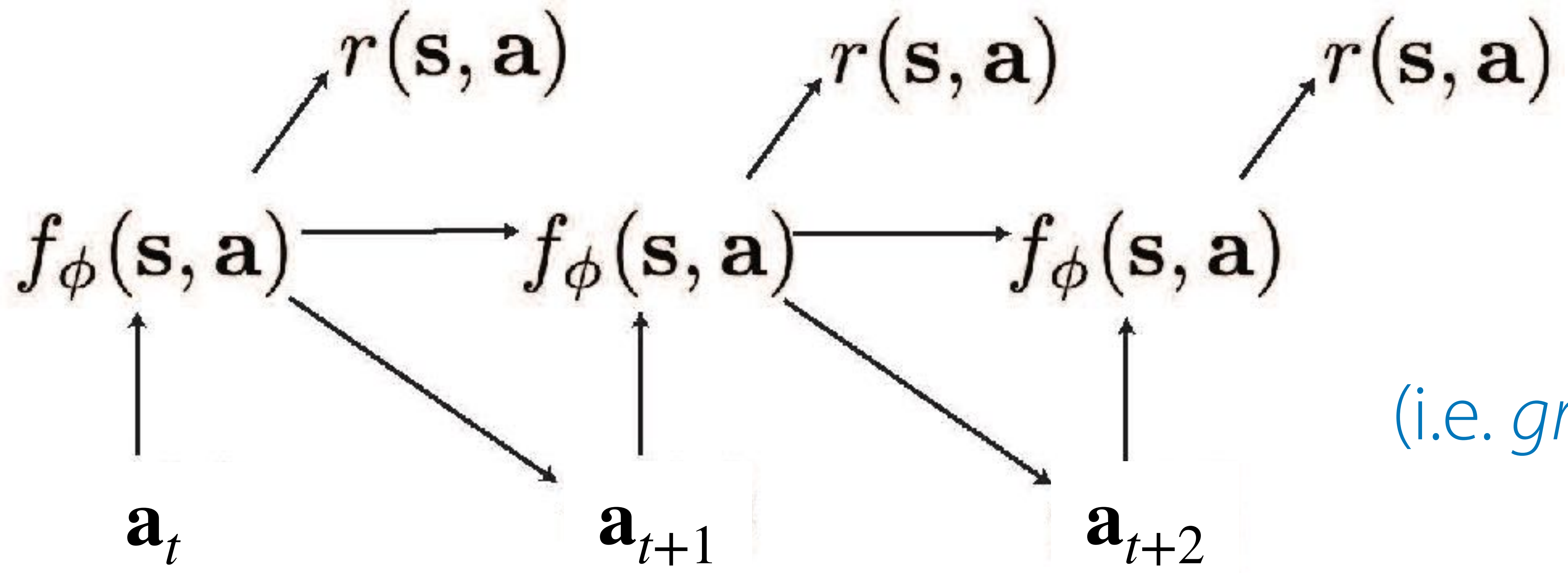


Approach 1a:
via *backpropagation*
(i.e. *gradient-based* optimization)

Algorithm:

1. Run some policy (e.g. random policy) to collect data $\mathscr{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn model $f_\phi(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. backpropagate through $f_\phi(\mathbf{s}, \mathbf{a})$ to choose actions

Approach 1: Optimize over actions using model: $\max_{\mathbf{a}_{t:t+H}} \sum_t r(\mathbf{s}_t, \mathbf{a}_t)$



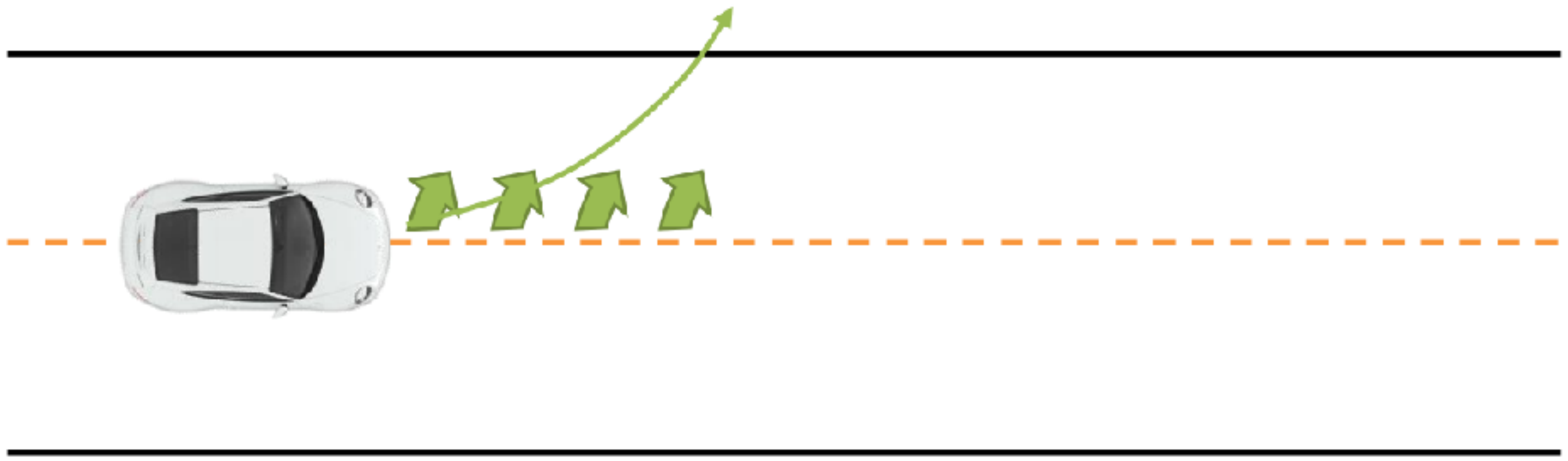
Approach 1b:
via *sampling*
(i.e. *gradient-free* optimization)

Algorithm:

1. Run some policy (e.g. random policy) to collect data $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn model $f_\phi(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. iteratively sample action sequences, run through model $f_\phi(\mathbf{s}, \mathbf{a})$ to choose actions

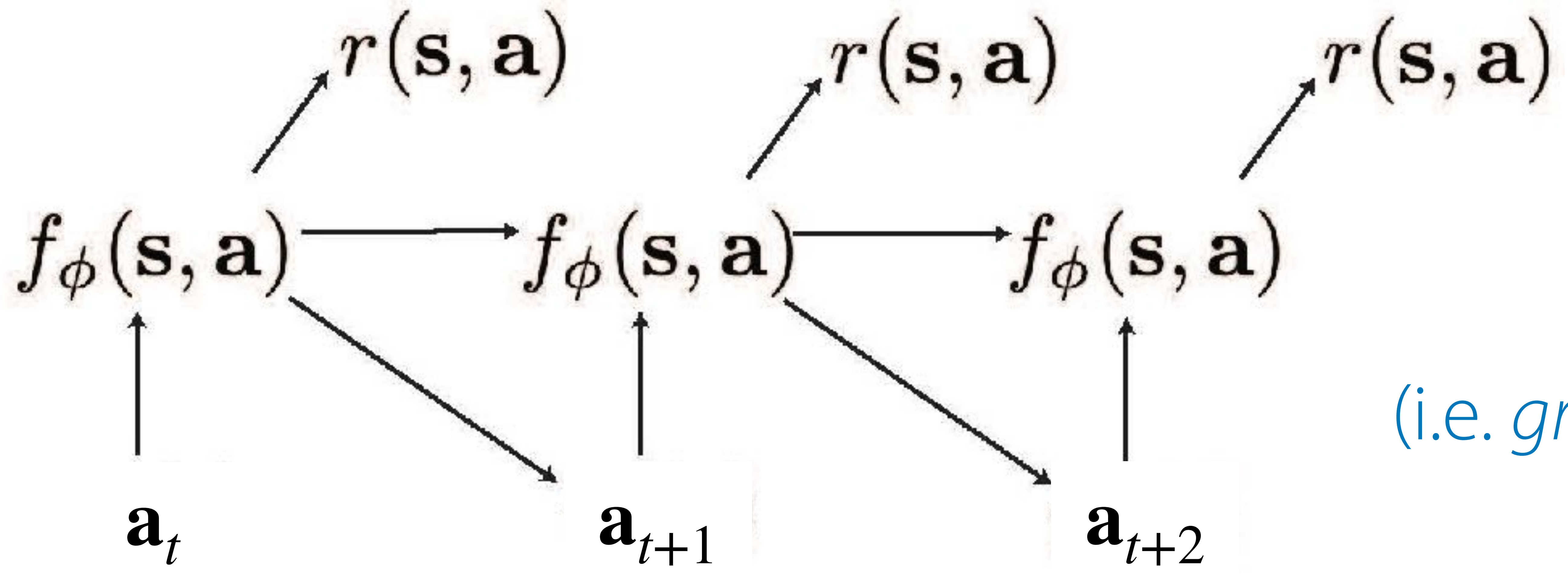
How can this approach fail?

Action optimization will exploit imprecisions in model



Refitting model using new data.

Approach 1: Optimize over actions using model: $\max_{\mathbf{a}_{t:t+H}} \sum_t r(\mathbf{s}_t, \mathbf{a}_t)$



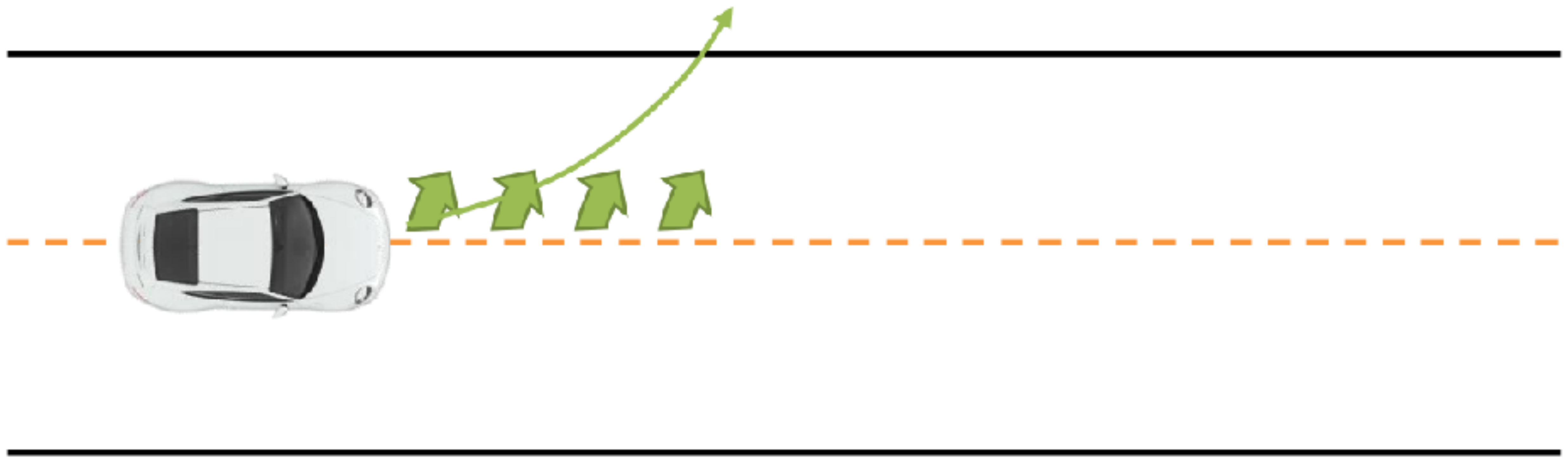
Approach 1b:
via *sampling*
(i.e. *gradient-free* optimization)

Algorithm:

1. Run some policy (e.g. random policy) to collect data $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn model $f_\phi(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}_i'\|^2$
3. iteratively sample action sequences, run through model $f_\phi(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute planned actions, appending visiting tuples $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to $\mathcal{D}$

How can this approach fail?

Action optimization will exploit imprecisions in model



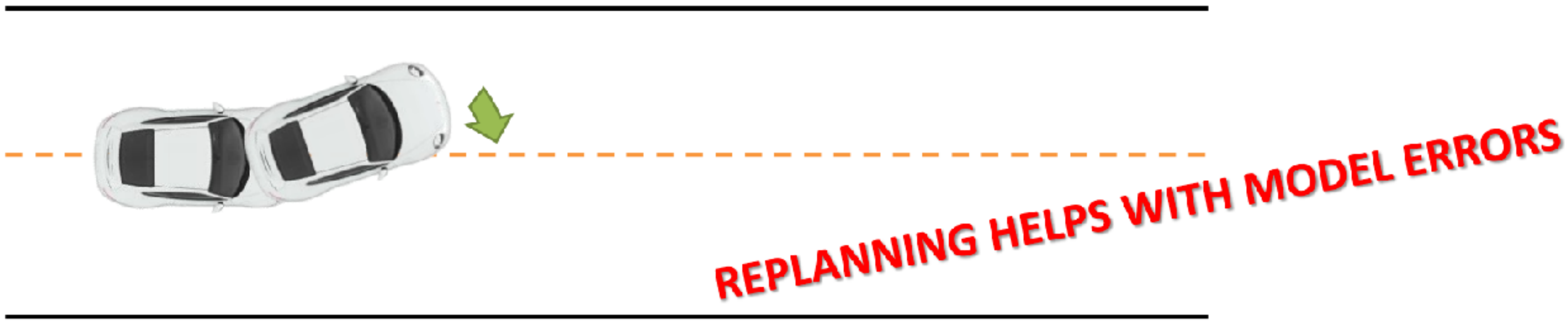
Refitting model using new data will help.

But generally, <u>learning a good global model is hard</u>.

Approach 2: Plan & replan using model

model-predictive control (MPC)

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn model $f_\phi(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i||^2$
3. Use model $f_\phi(\mathbf{s}, \mathbf{a})$ to optimize action sequence
4. execute the first planned action, observe resulting state $\mathbf{s}'$
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$



+ replan to correct for model errors   - compute intensive

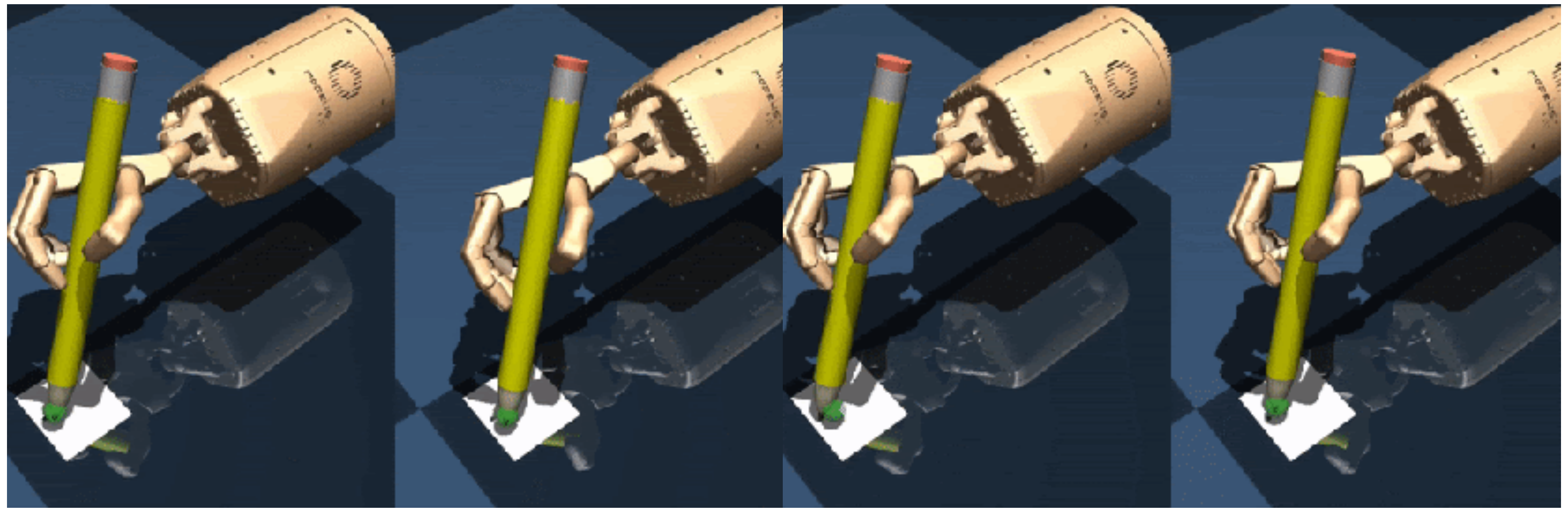
What does this have to do with multi-task RL and meta RL?

1. Do you know the form of the rewards $r_i(\mathbf{s}, \mathbf{a})$ for each task?

If *yes:* learn single model, plan w.r.t. each r_i at test time

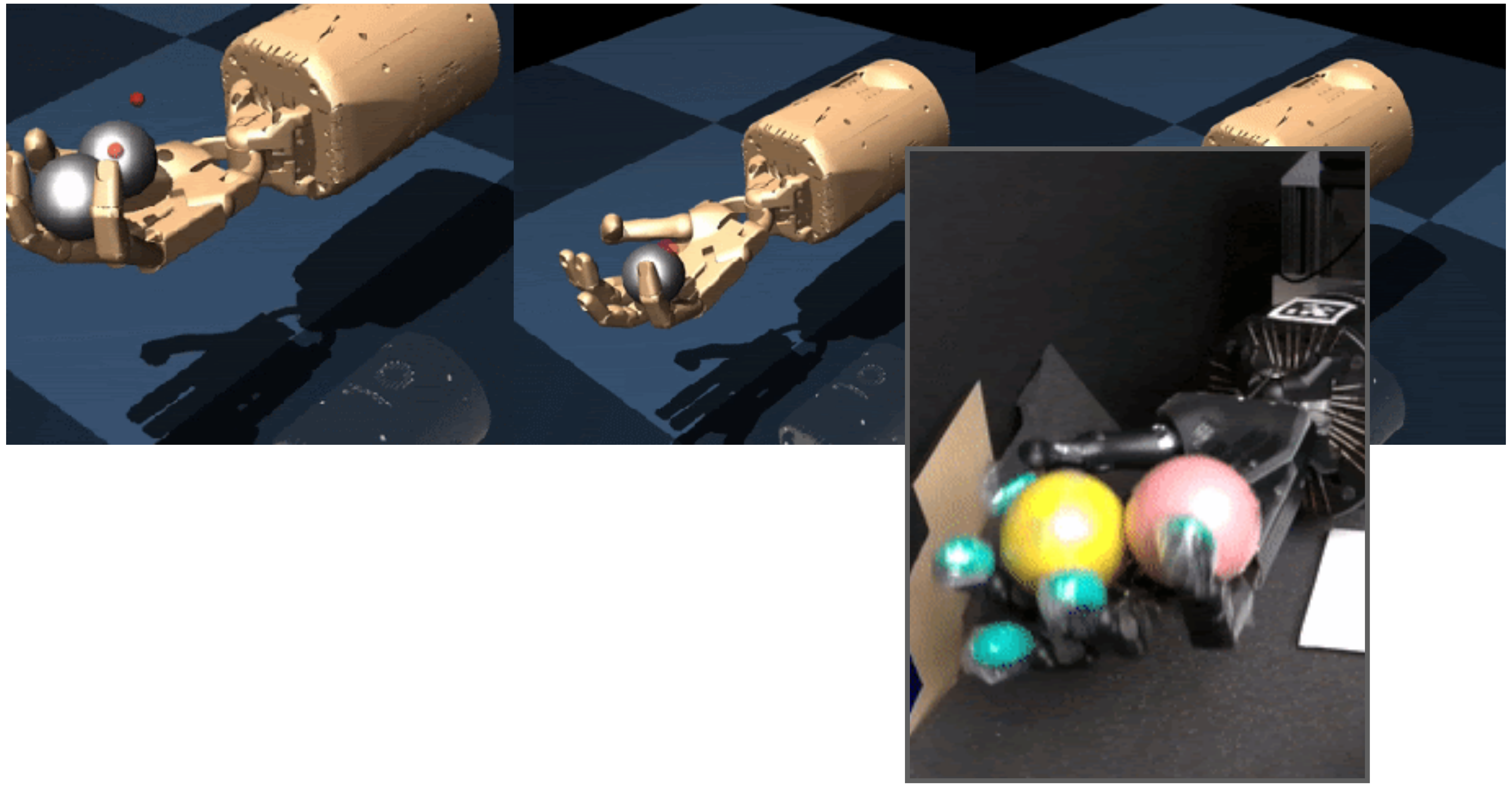
*__Caveat__: reward will change how you collect data.

r_i corresponds to different pencil trajectories



Nagabandi, Konolige, Levine, Kumar. *Deep Dynamics Models for Learning Dexterous Manipulation*. CoRL '19

r_i corresponds to different ball positions/trajectories



Model-based RL is **sample efficient** enough to train on real hardware!

What does this have to do with multi-task RL and meta RL?

1. Do you know the form of the rewards $r_i(\mathbf{s}, \mathbf{a})$ for each task?

If *yes*: learn single model, plan w.r.t. each r_i at test time

If *no*: **multi-task RL**: learn $r_\theta(\mathbf{s}, \mathbf{a}, \mathbf{z}_i)$, use it to plan

meta-RL: meta-learn $r_\theta(\mathbf{s}, \mathbf{a}, \mathscr{D}_i^{\mathrm{tr}})$, use it to plan

Both solve the multi-task RL & meta-RL problem statements.

$\mathscr{D}_i^{\mathrm{tr}}$: a few positive examples

Use it to acquire a binary reward r_θ

Plan using your model to maximize reward

Xie, Singh, Levine, Finn. *Few-Shot Goal Inference for Visuomotor Learning and Planning*. CoRL '18

The Plan

Model-based RL

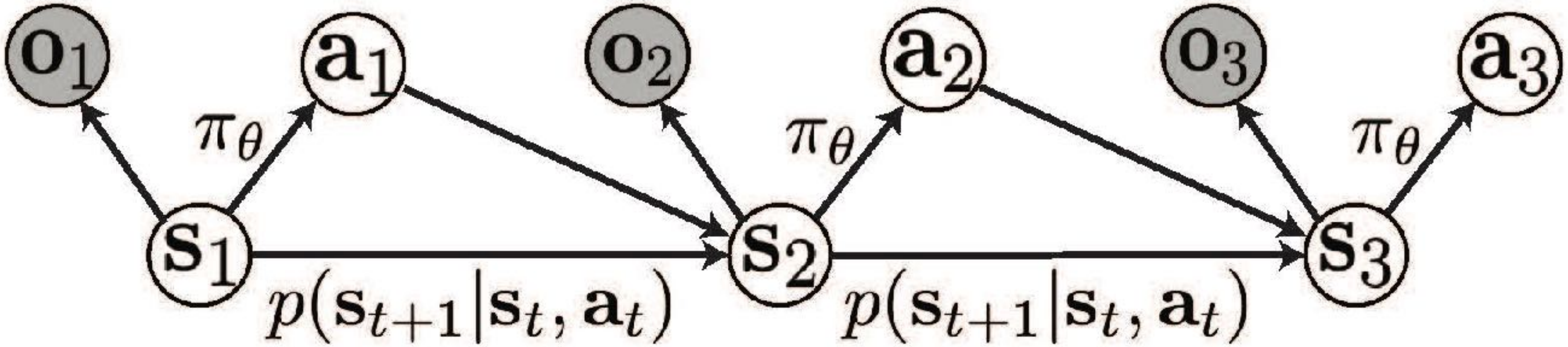
and how it can be used for multi-task & meta-learning

Model-based RL with image observations

or other high-dimensional inputs

Model-based meta-RL

Only access to high-dimensional observations (i.e. images)?

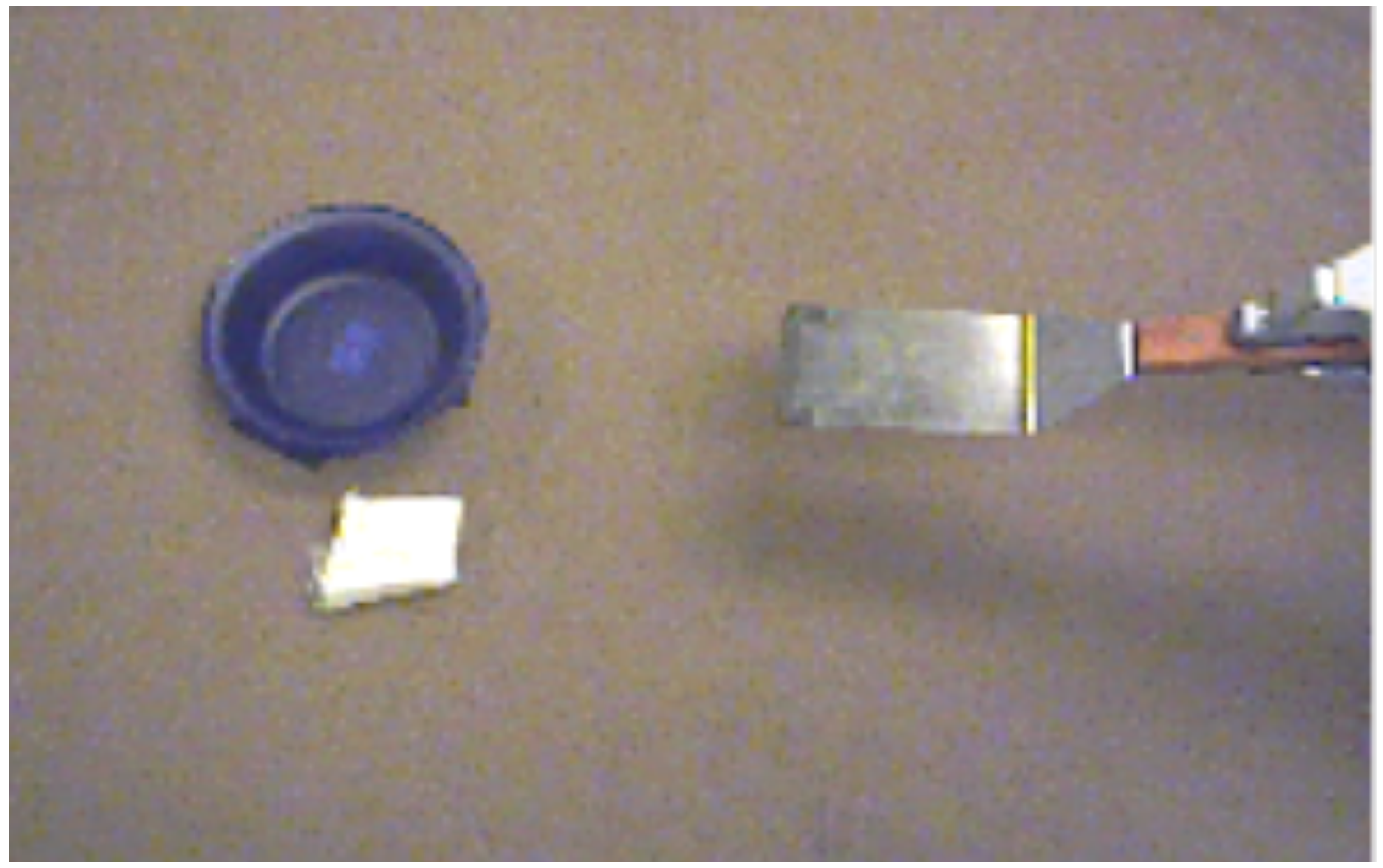


also: no reward signal with only observations

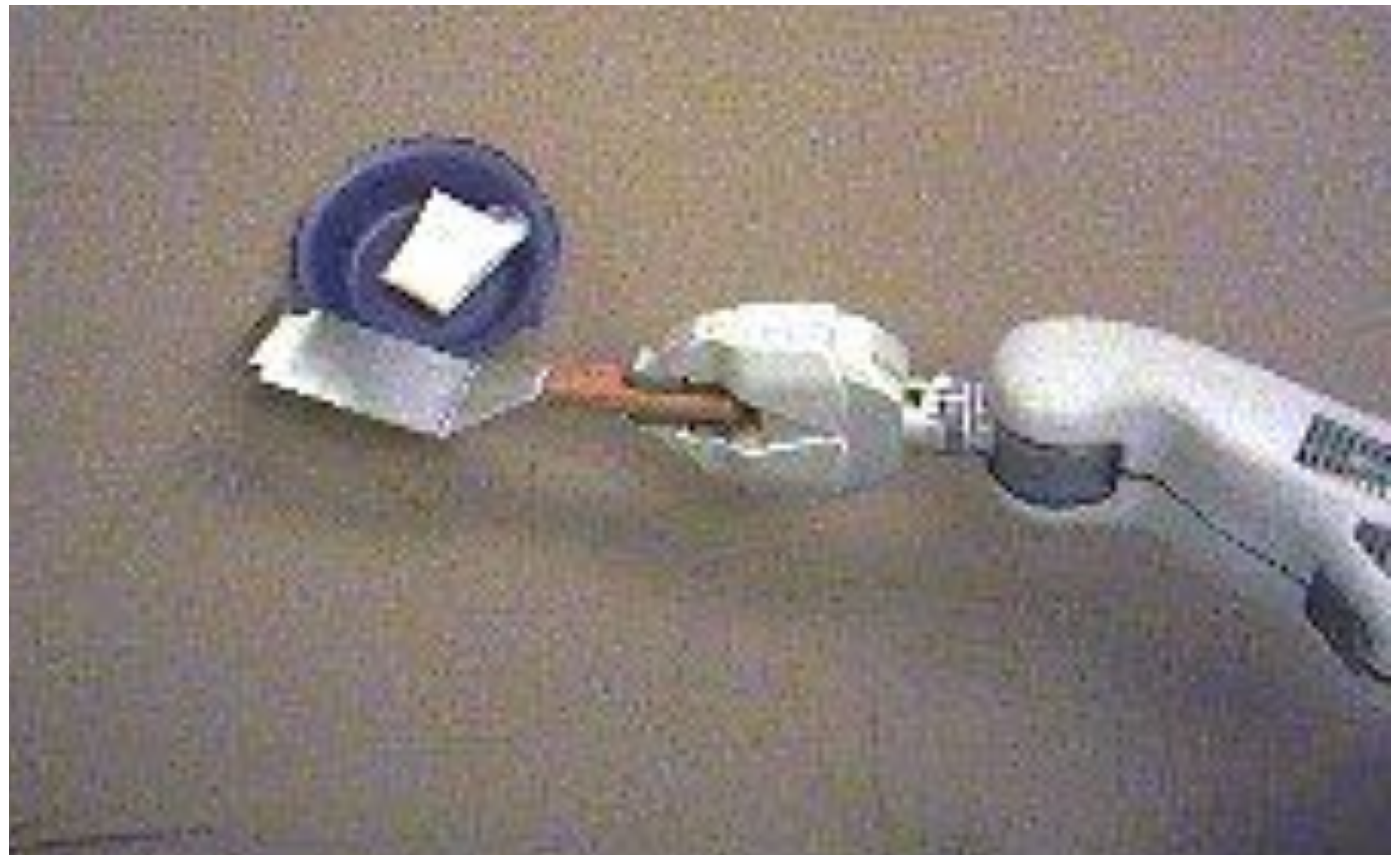
Only access to high-dimensional observations (i.e. images)?

one option: learn an image classifier

another option: provide image of goal
(i.e. goal-conditioned RL)



also: no reward signal with only observations



Approaches

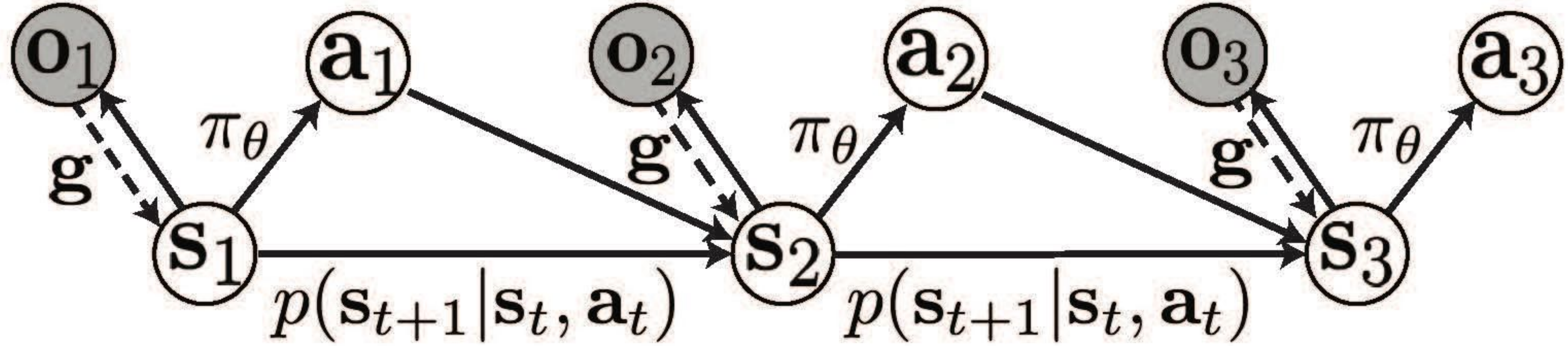
1. Learn model in latent space
2. Learn model of observations (e.g. video)
3. Predict alternative quantities

Learning with Image Observations

1. **Models in latent space**
2. Models directly in image space
3. Model alternative quantities

Learning in Latent Space

Key idea: learn embedding $g(\mathbf{o}_t)$, then learn model in latent space



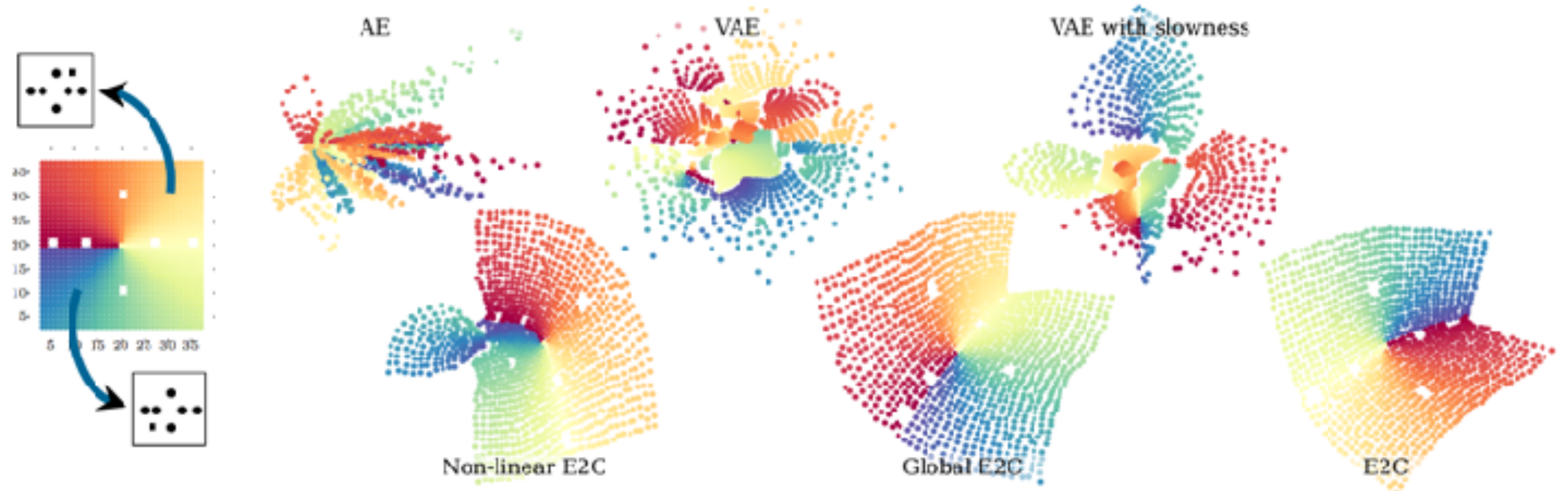
Learning in Latent Space

Key idea: learn embedding $\mathbf{s}_t = g(\mathbf{o}_t)$, then do model-based RL in latent space

Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images

Manuel Watter* **Jost Tobias Springenberg***
Joschka Boedecker
University of Freiburg, Germany
{watterm,springj,jboedeck}@cs.uni-freiburg.de

Martin Riedmiller
Google DeepMind
London, UK
riedmiller@google.com



NIPS 2015

Deep Spatial Autoencoders for Visuomotor Learning

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, Pieter Abbeel

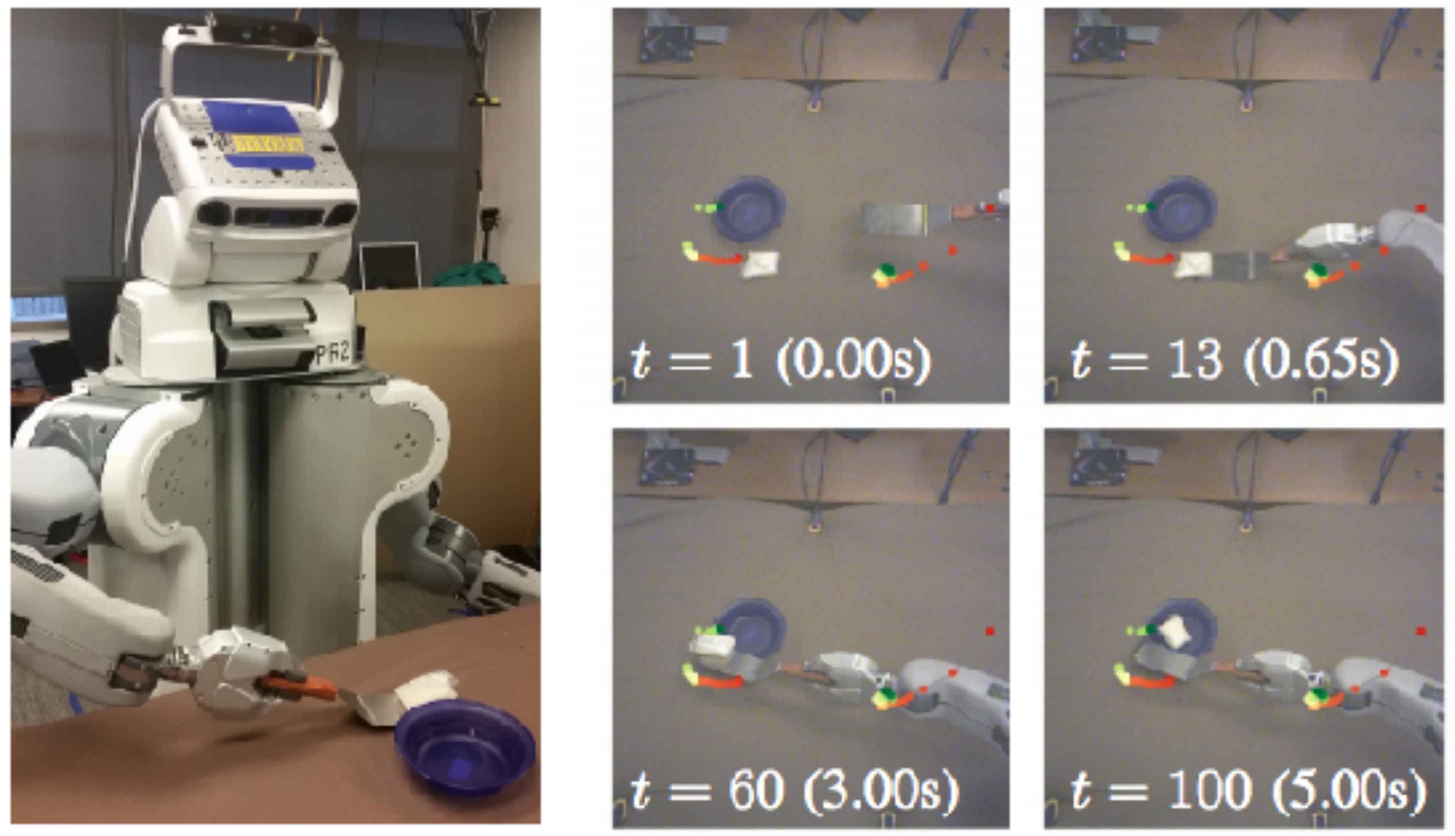


Fig. 1: PR2 learning to scoop a bag of rice into a bowl with a spatula (left) using a learned visual state representation (right).

ICRA 2016

Learning in Latent Space

Algorithm

1. Run some policy (e.g. random policy) to collect data $\mathscr{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$
2. learn latent embedding of observation $\mathbf{s}_t = g(\mathbf{o}_t)$ and model $\mathbf{s}' = f_\phi(\mathbf{s}, \mathbf{a})$
3. use model $f_\phi(\mathbf{s}, \mathbf{a})$ to optimize action sequences
4. execute planned actions, appending visiting tuples $(\mathbf{o}, \mathbf{a}, \mathbf{o}')$ to $\mathscr{D}$

What is the reward for optimizing actions?

reward signal: $r(\mathbf{o}, \mathbf{a}) = -\|g(\mathbf{o}) - g(\mathbf{o}_{\text{goal}})\|^2$

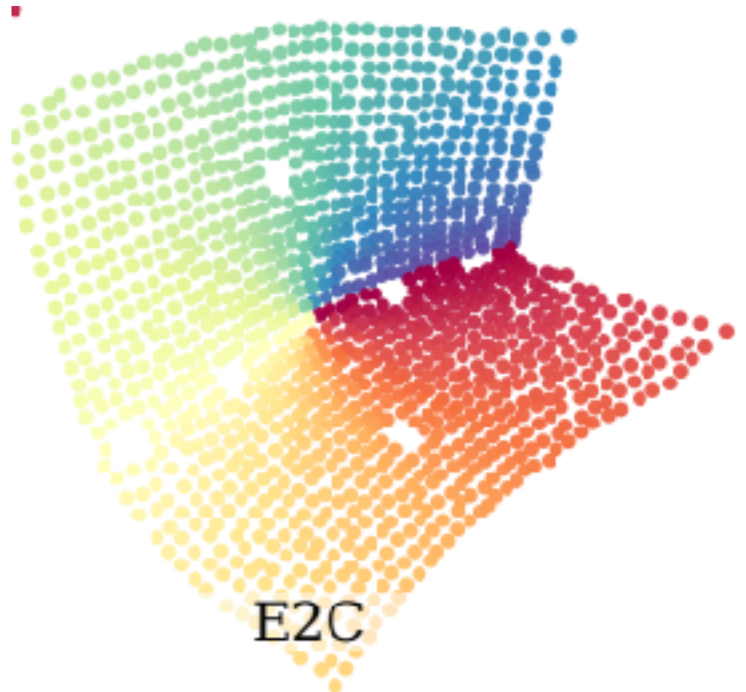
Assumption: distance in latent space is an accurate metric.

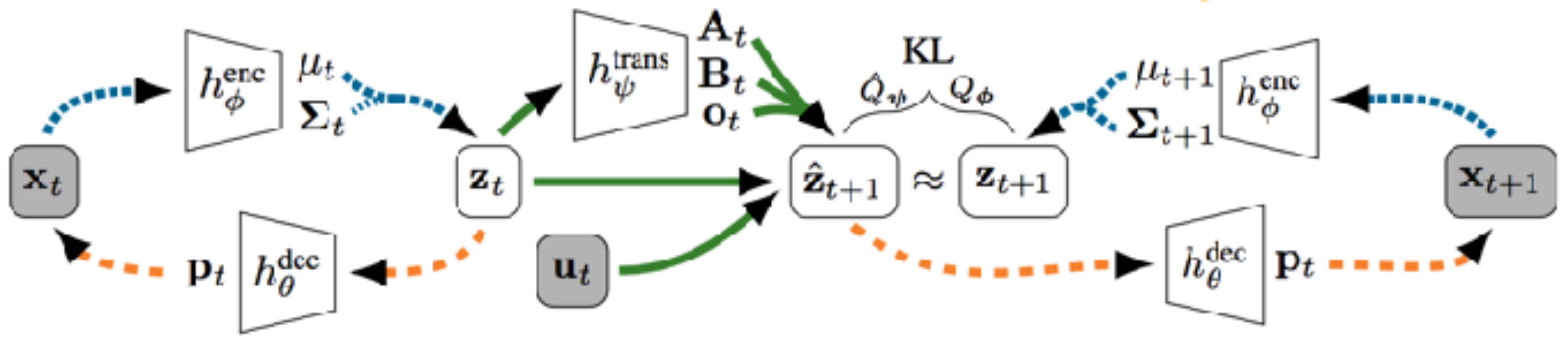
Learning in Latent Space

1. Run some policy (e.g. random policy) to collect data $\mathscr{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn latent embedding of observation $\mathbf{s}_t = g(\mathbf{o}_t)$ and model $\mathbf{s}' = f_\phi(\mathbf{s}, \mathbf{a})$
3. use model $f_\phi(\mathbf{s}, \mathbf{a})$ to optimize action sequences
4. execute planned actions, appending visiting tuples $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to $\mathscr{D}$

How to optimize latent embedding?

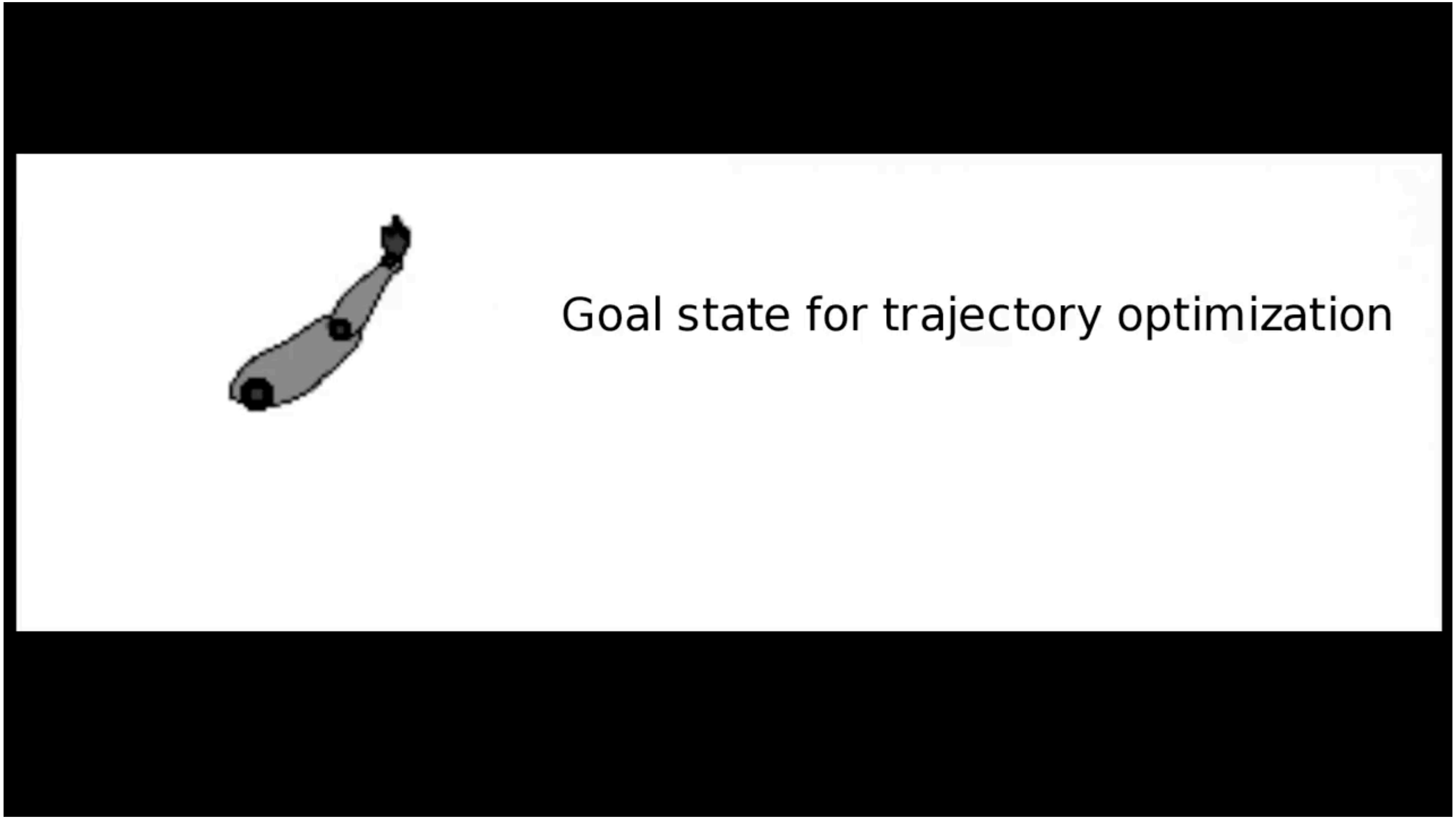
Watter et al. '15





learn embedding & model **jointly**

Learning in Latent Space



~300 trials = ~25 min of robot time (per task)

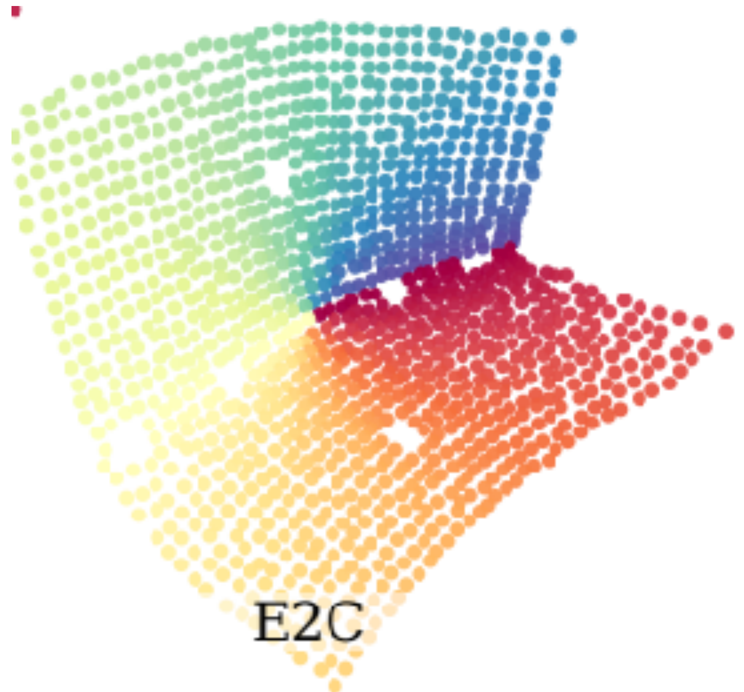
Watter et al. NIPS '15

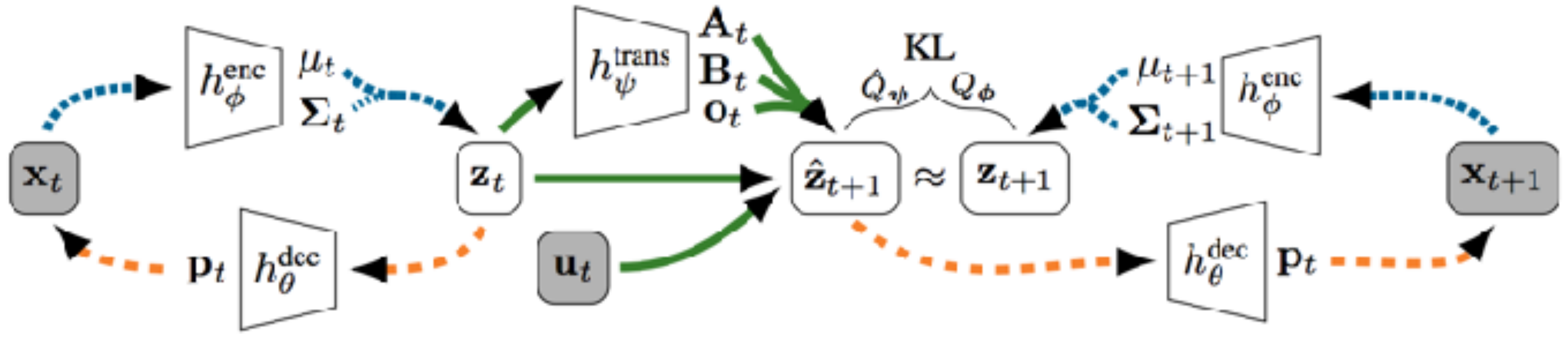
Learning in Latent Space

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., exploratory policy) to collect $\mathcal{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$
2. learn latent embedding of observation $\mathbf{s}_t = g(\mathbf{o}_t)$ and dynamics model $\mathbf{s}' = f_\phi(\mathbf{s}, \mathbf{a})$
3. use model $f_\phi(\mathbf{s}, \mathbf{a})$ to optimize policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
4. run $\pi_\theta(\mathbf{a}_t|g(\mathbf{o}_t))$, appending visited tuples $(\mathbf{o}, \mathbf{a}, \mathbf{o}')$ to $\mathcal{D}$

How to optimize latent embedding?

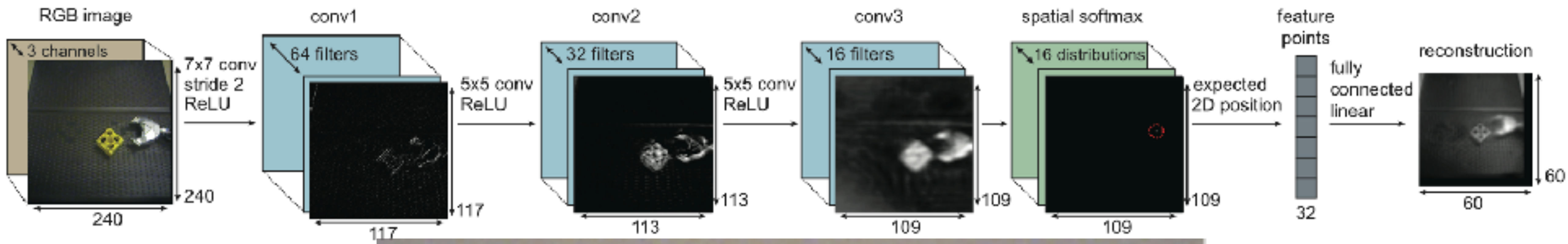
Watter et al. '15

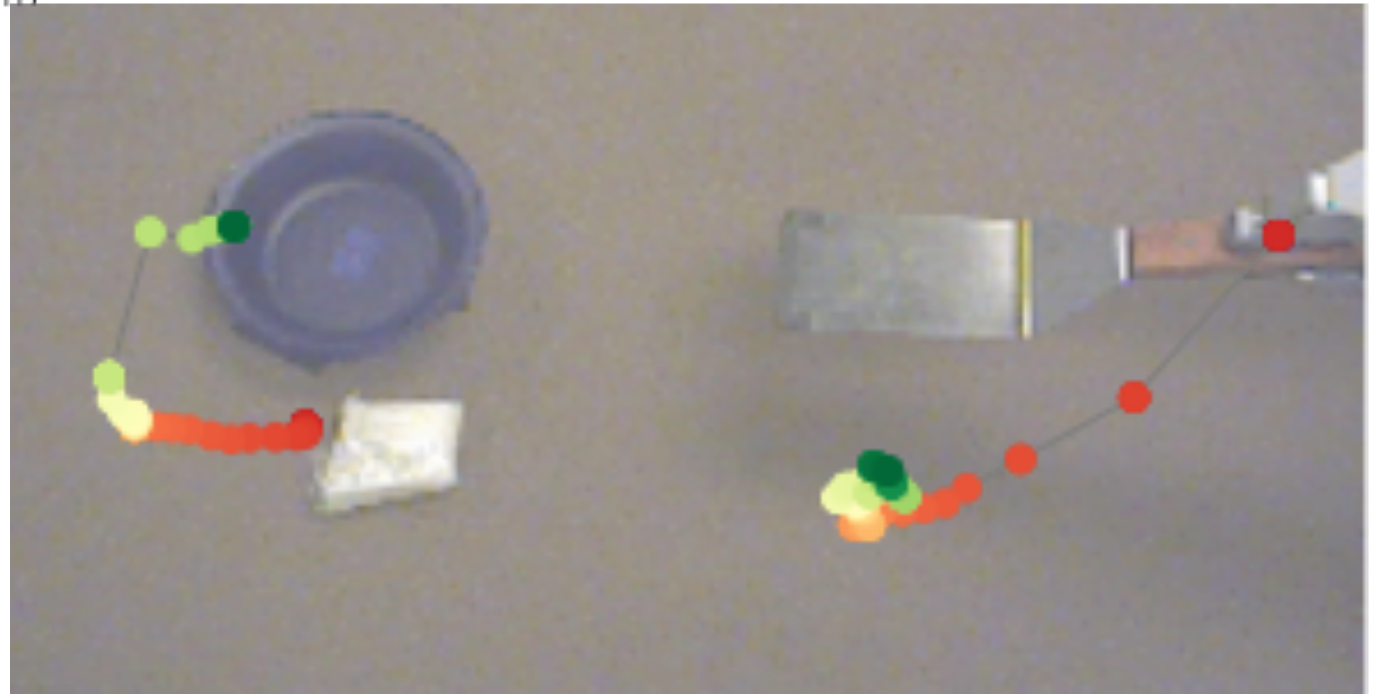




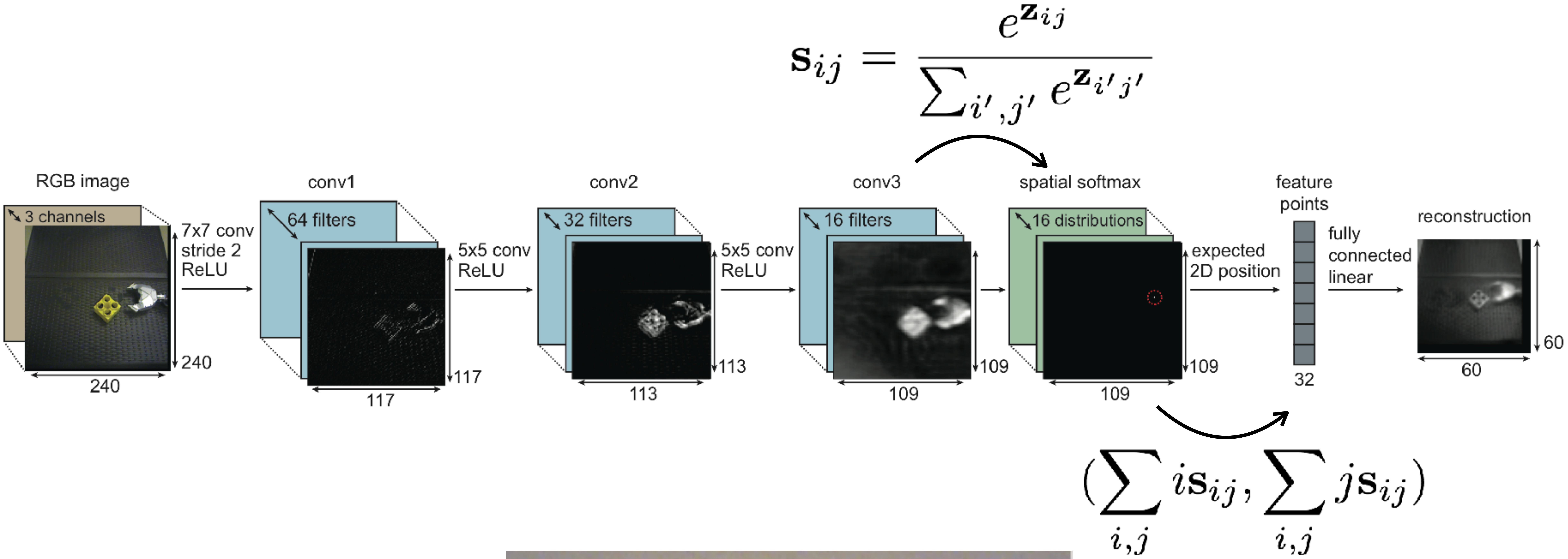
learn embedding & model **jointly**

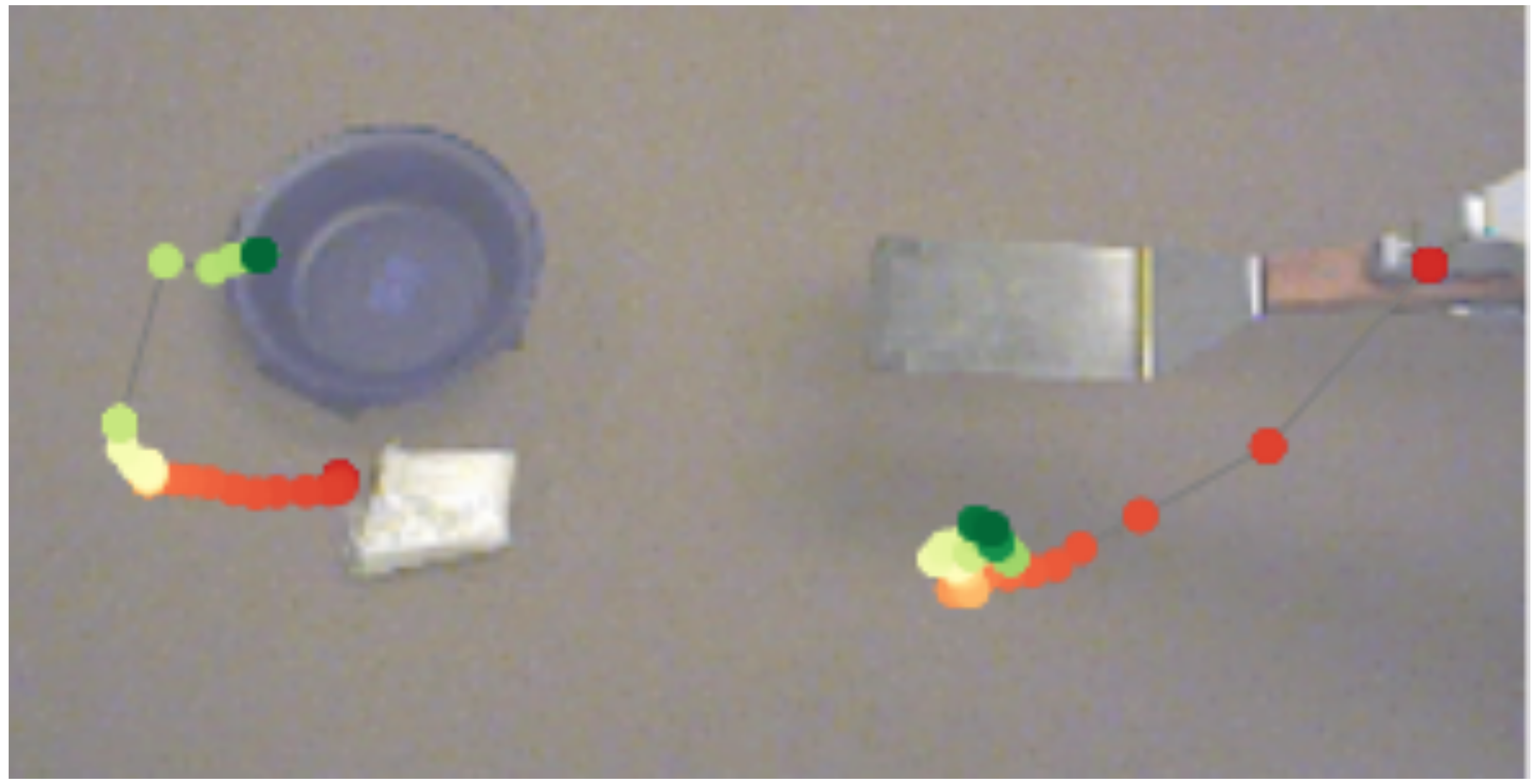
Finn et al. '16





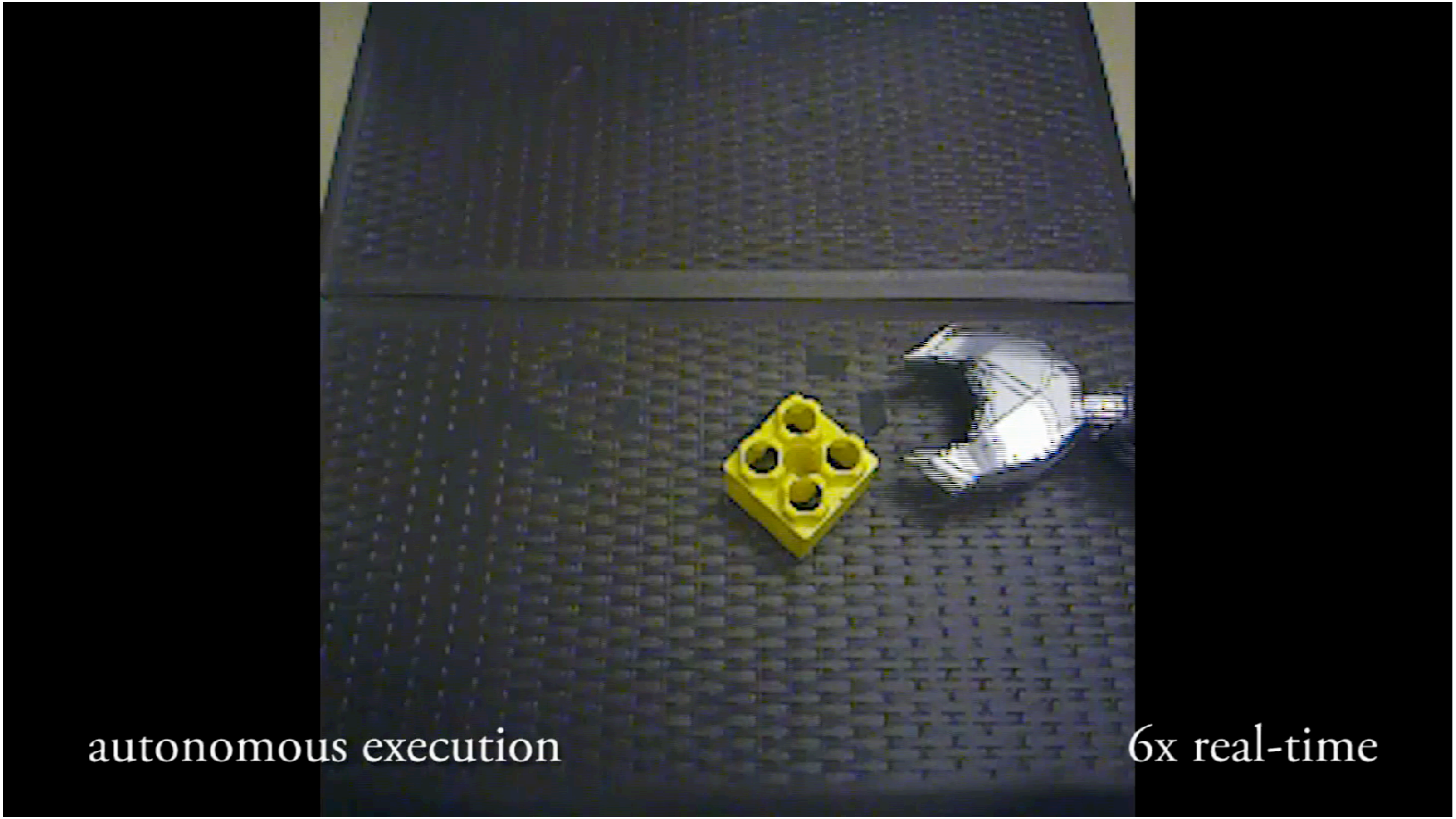
embedding is **smooth and structured**





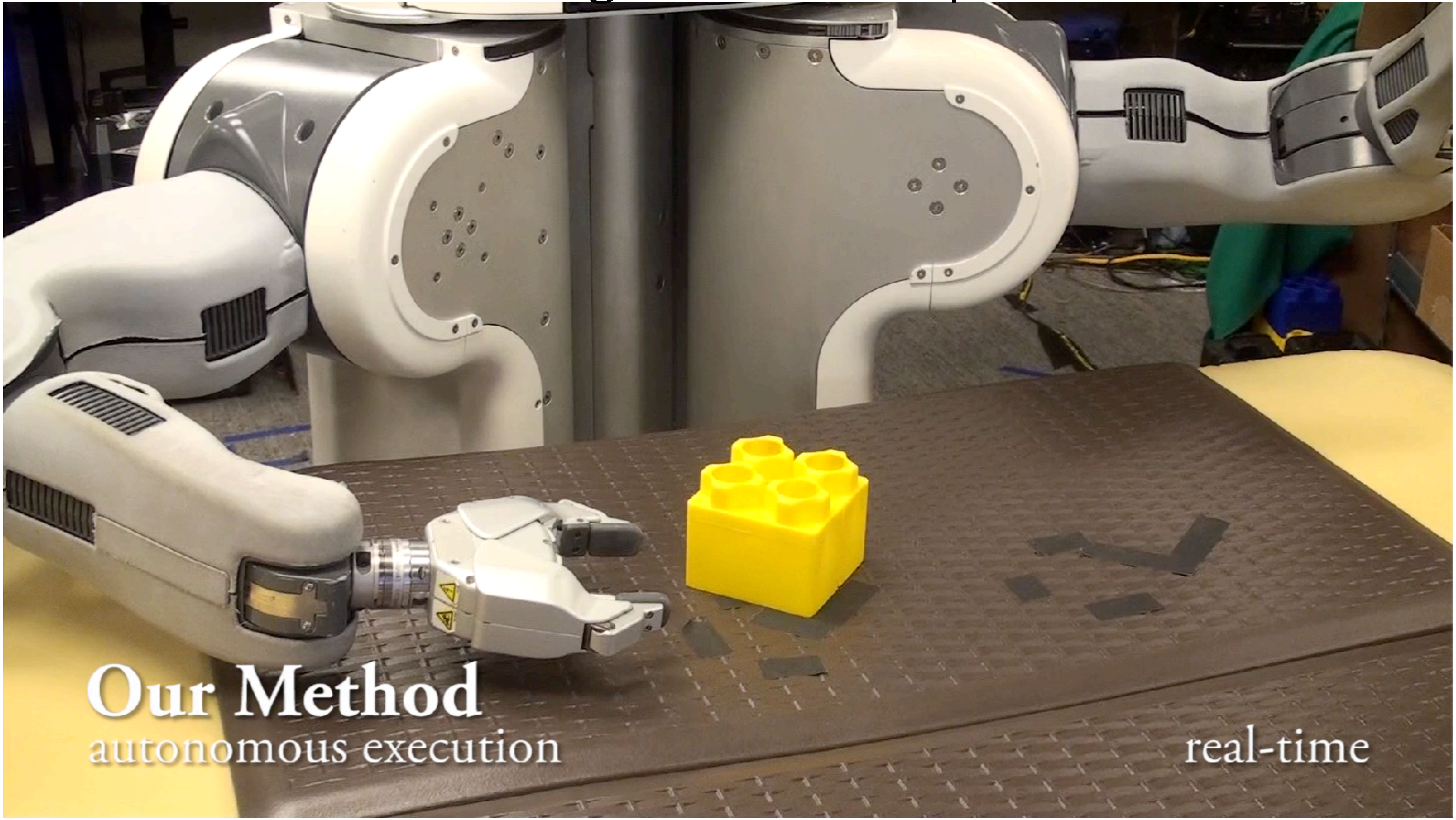
embedding is **structured and smooth**

Learning in Latent Space



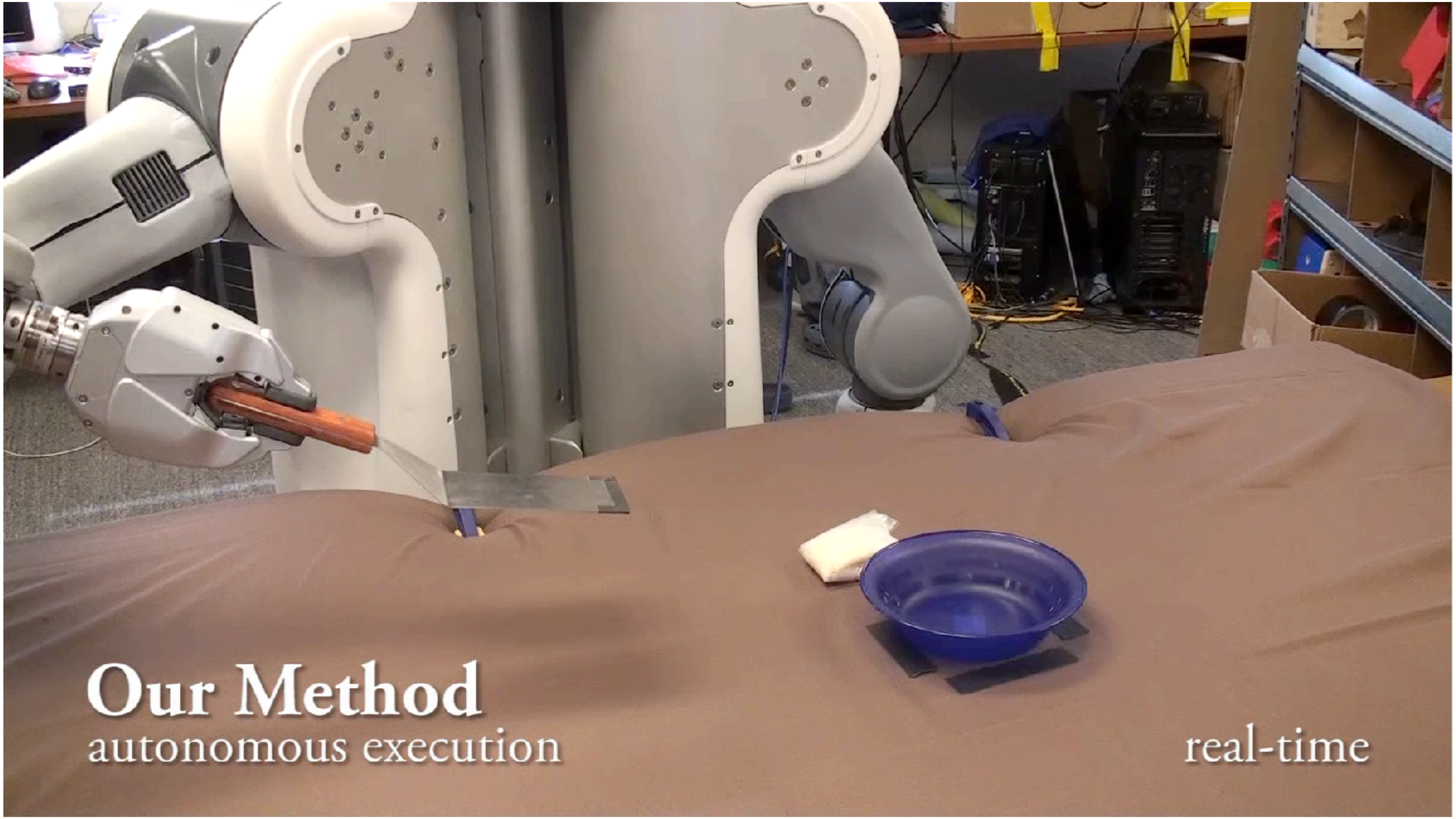
Finn et al. ICRA '16

Learning in Latent Space



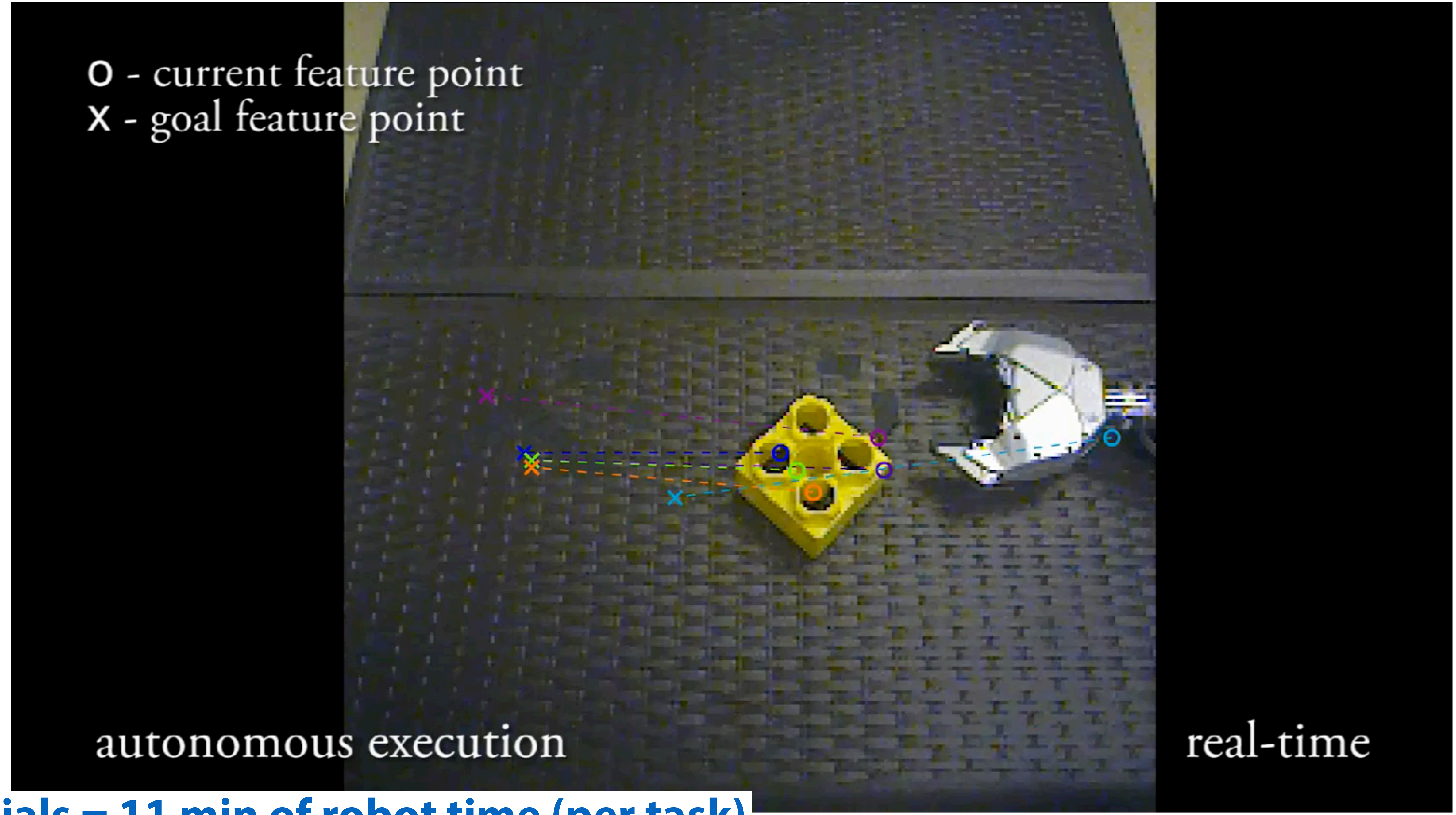
Finn et al. ICRA '16

Learning in Latent Space



Finn et al. ICRA '16

Learning in Latent Space



125 trials = 11 min of robot time (<u>per task</u>)

Caveat: *environment-specific* task representation

Finn et al. ICRA ‘16

Thought exercise:

Why reconstruct the image?

Why not just learn embedding & model w.r.t. model error?

Learning in Latent Space

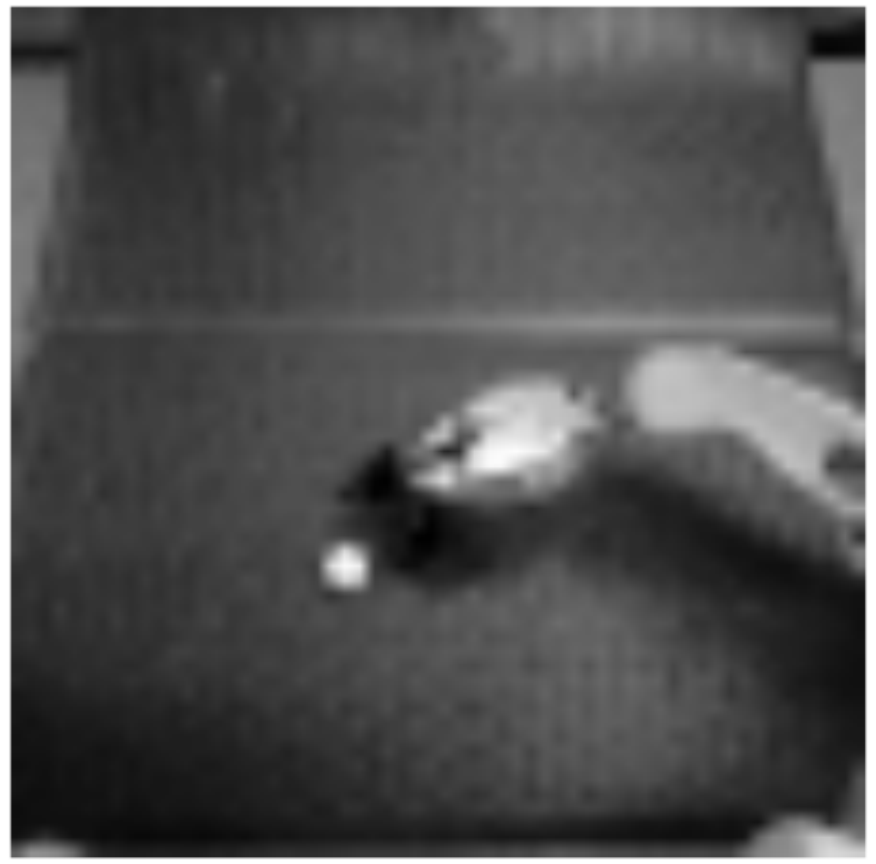
Pros:

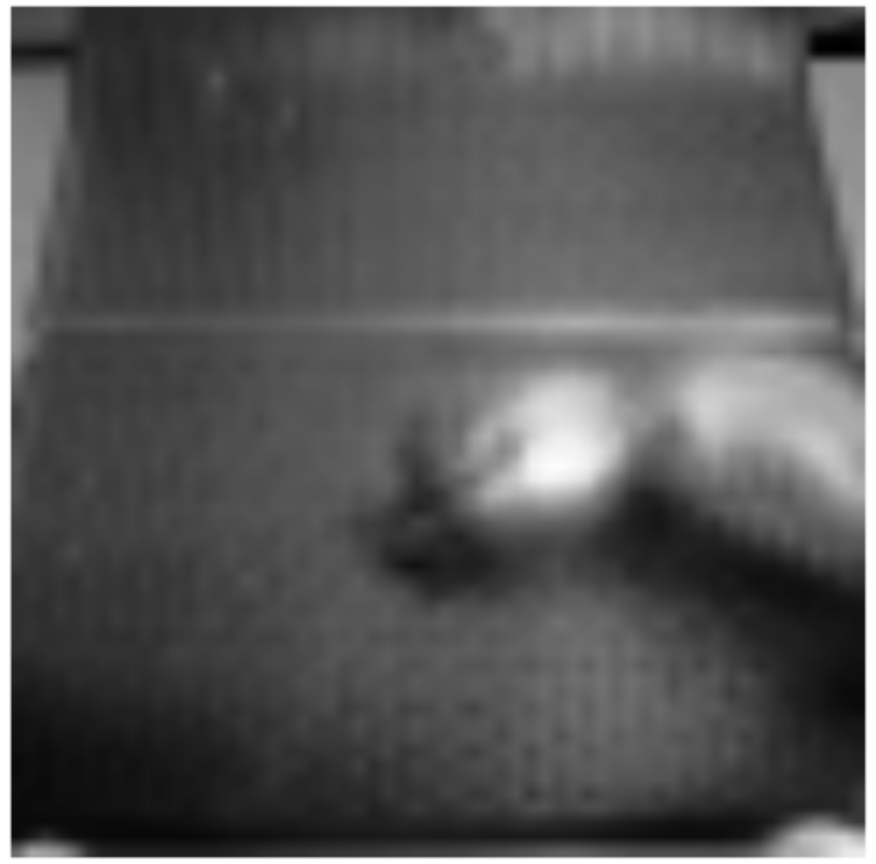
+ Learn complex visual skills very efficiently

+ Structured representation enables effective learning

Cons:

- Reconstruction objectives might not recover the right representation

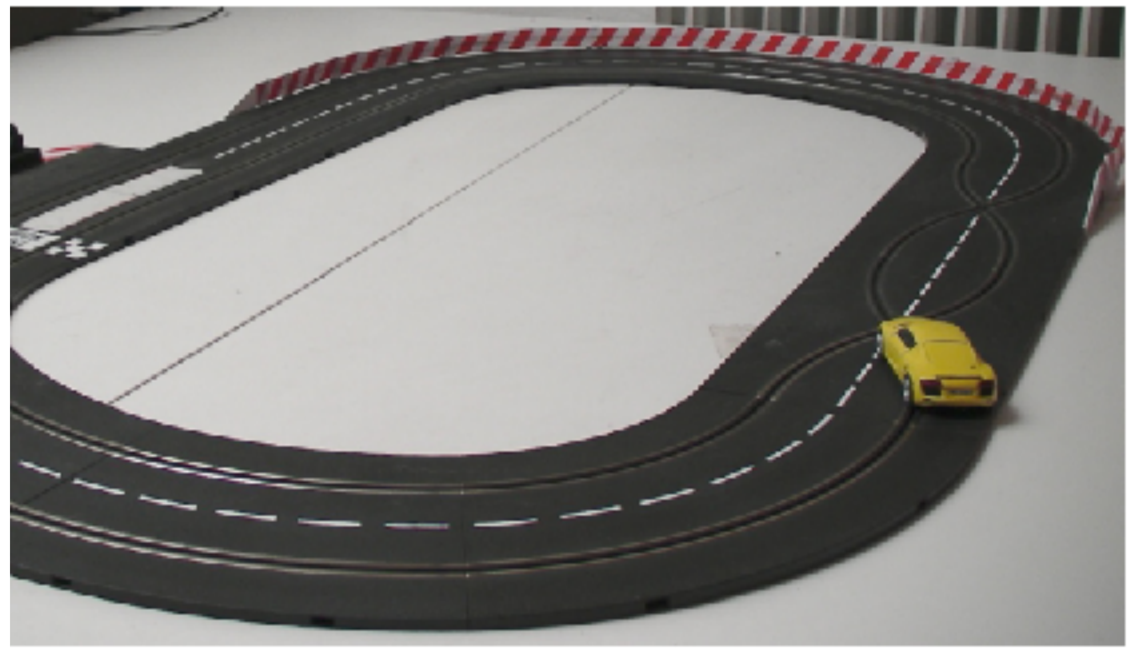




need better unsupervised representation learning methods

Aside: Low-dimensional embedding can also be useful for model-free approaches

model-free RL in latent space

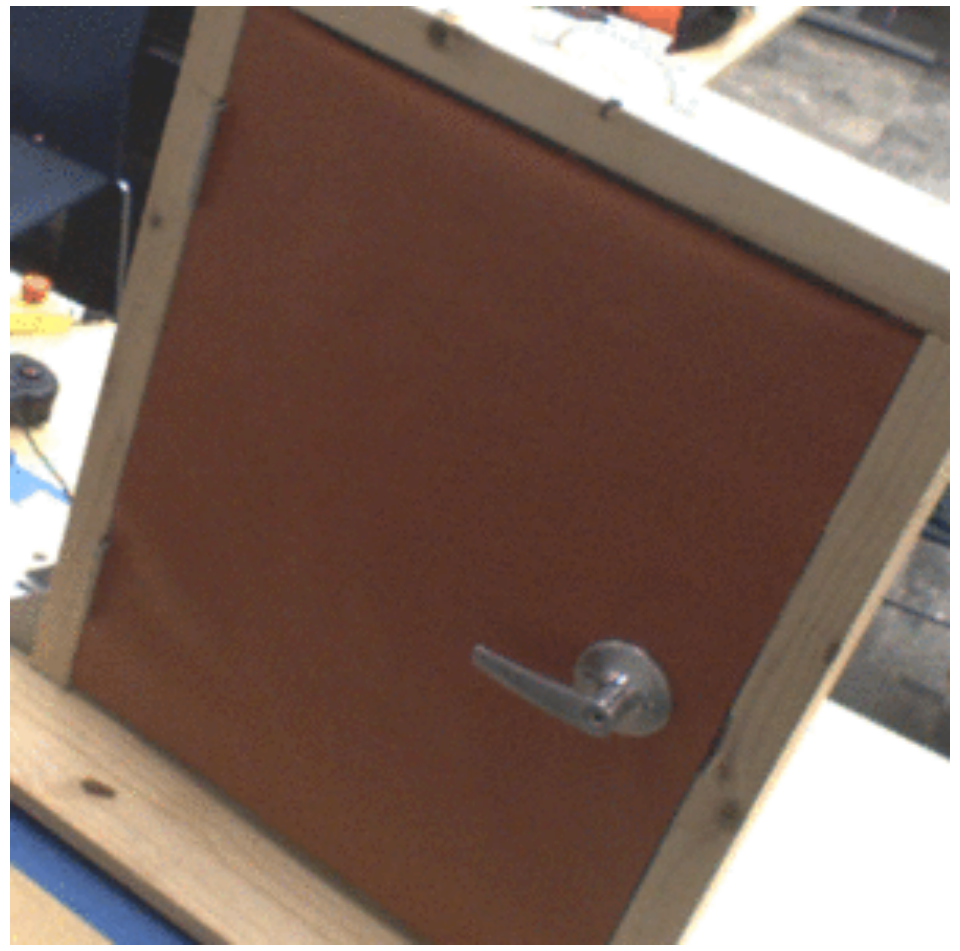


FQI in latent space
Lange et al. '12

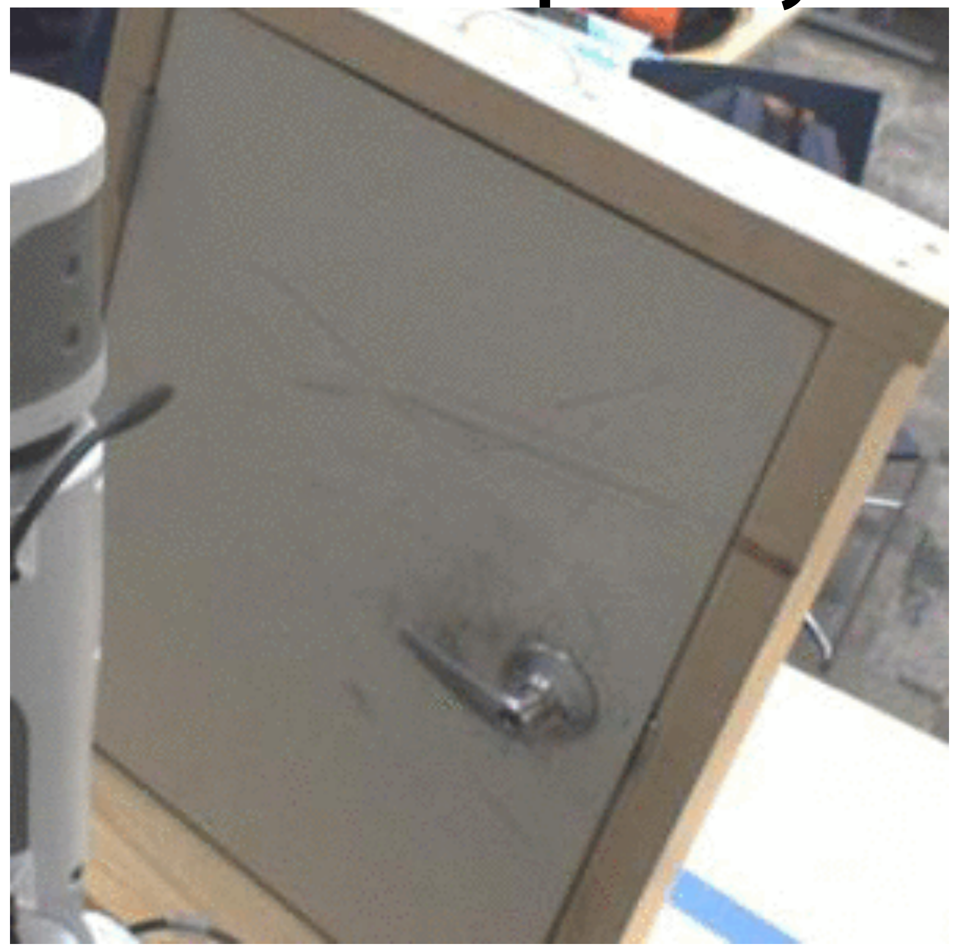
TRPO in latent space
Ghadirzadeh et al. '17

use embedding for reward function

video demonstration



learned policy



acquire reward using ImageNet features + model-free RL
Sermanet et al. RSS '17

If you have a reward, you can predict it to form better latent space.
(Jaderberg et al. '17, Shelhamer et al. '17)

Why not predict reward?

Learning with Image Observations

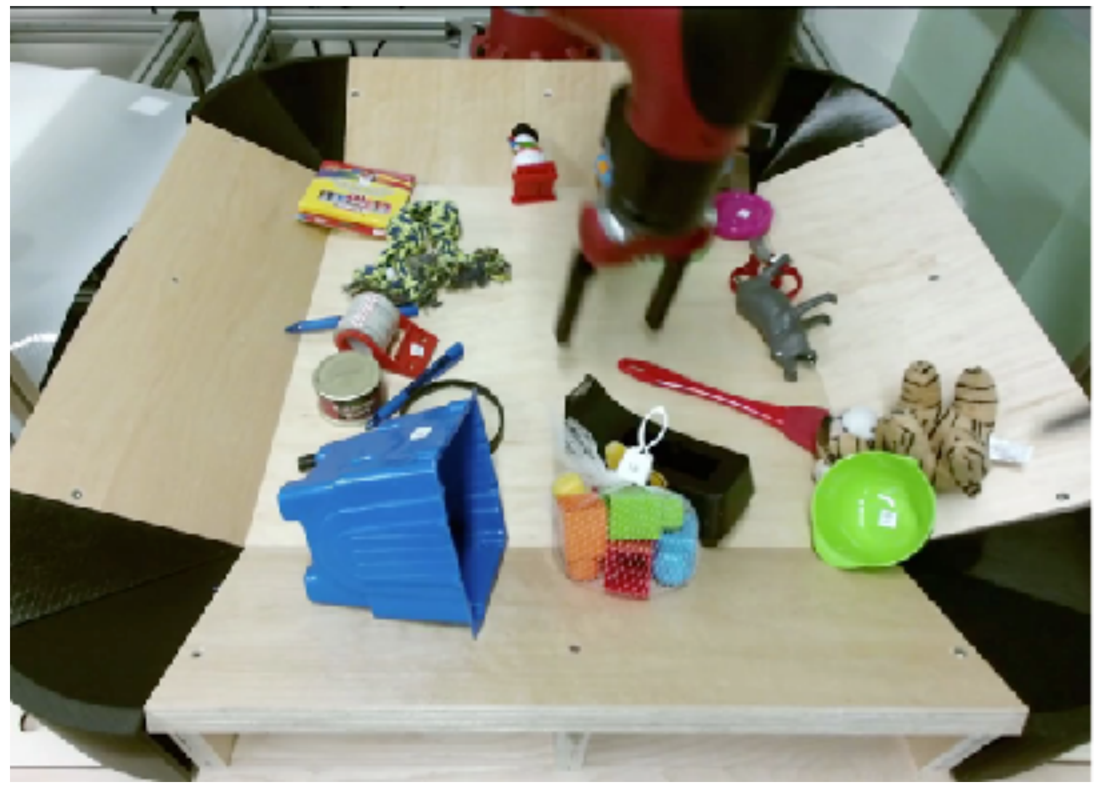
1. Models in latent space
2. **Models directly in image space**
3. Predict alternative quantities

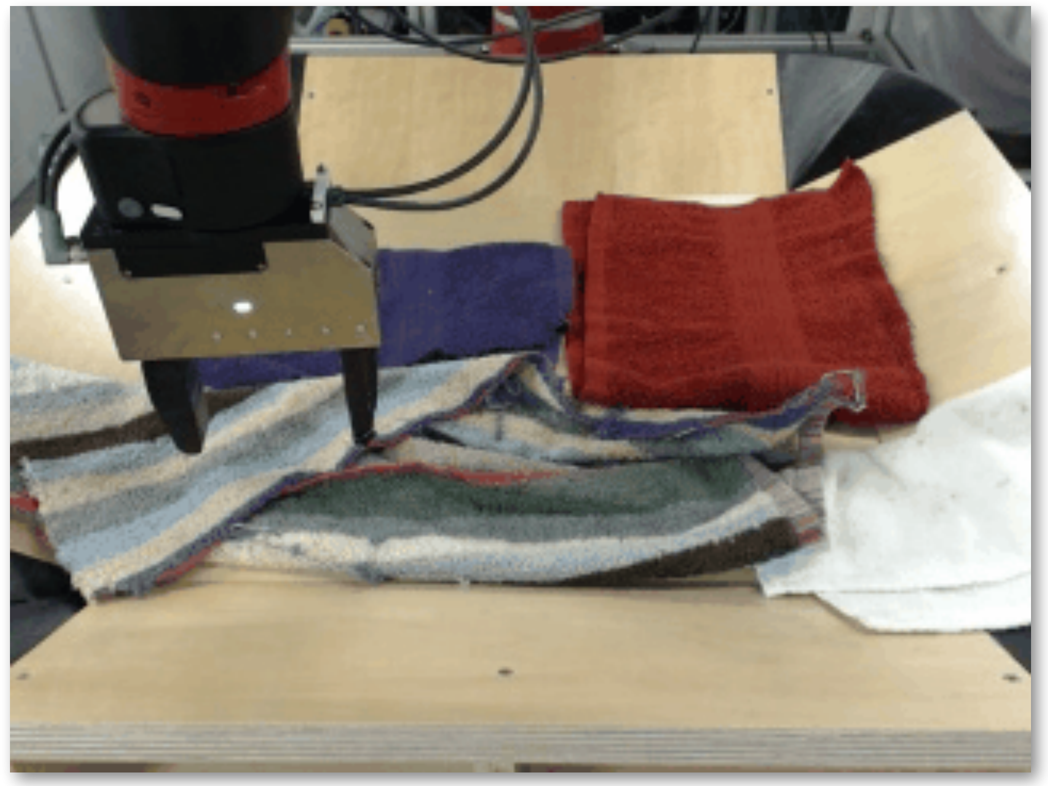
Modeling directly in observation space

Recall MPC

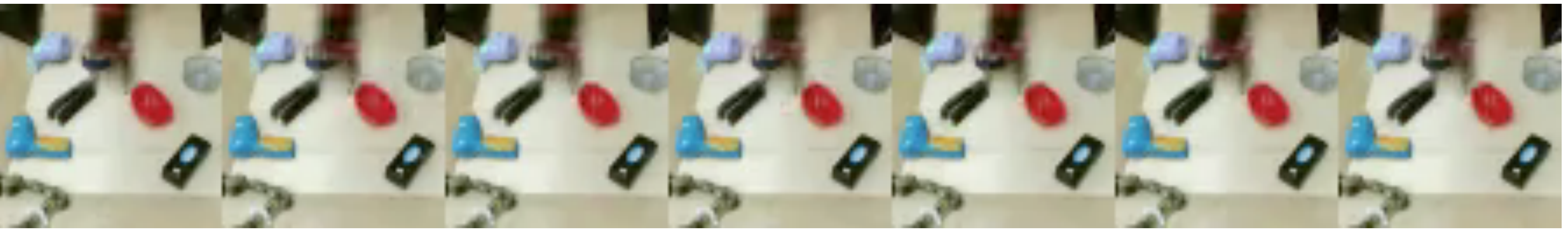
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{o}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$
2. learn model $f_\phi(\mathbf{o}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{o}_i, \mathbf{a}_i) - \mathbf{o}'_i||^2$
3. use model $f_\phi(\mathbf{o}, \mathbf{a})$ to optimize action sequence
4. execute the first planned action, observe resulting state $\mathbf{o}'$
5. append $(\mathbf{o}, \mathbf{a}, \mathbf{o}')$ to dataset $\mathcal{D}$

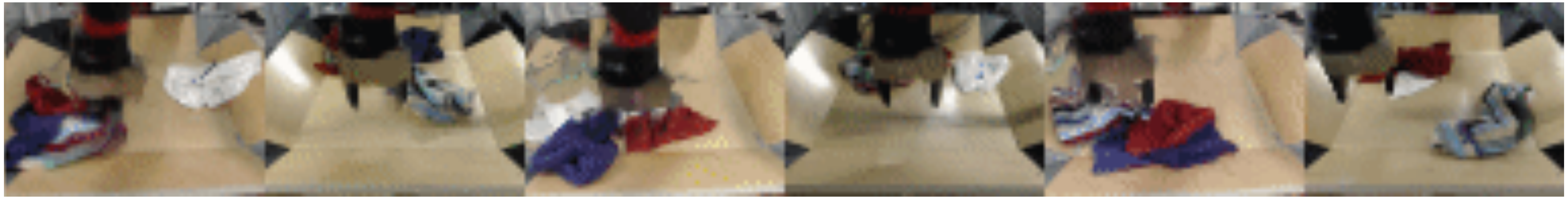
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{o}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$





2. learn model $f_\phi(\mathbf{o}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{o}_i, \mathbf{a}_i) - \mathbf{o}'_i||^2$



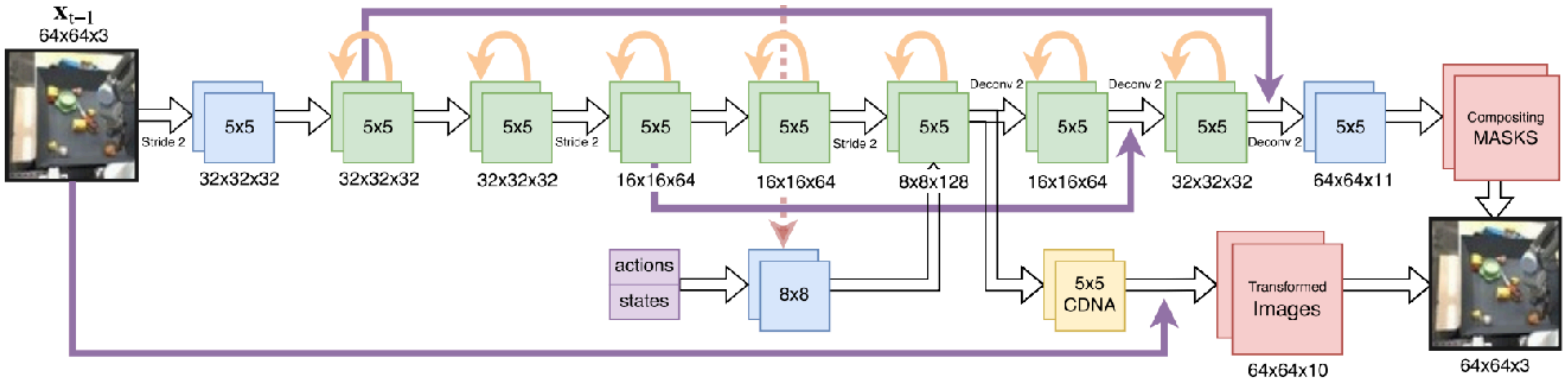


3. use model $f_\phi(\mathbf{o}, \mathbf{a})$ to optimize action sequence

Finn et al. NIPS '16, Finn & Levine ICRA '17, Ebert et al. CoRL '17

How to predict video?

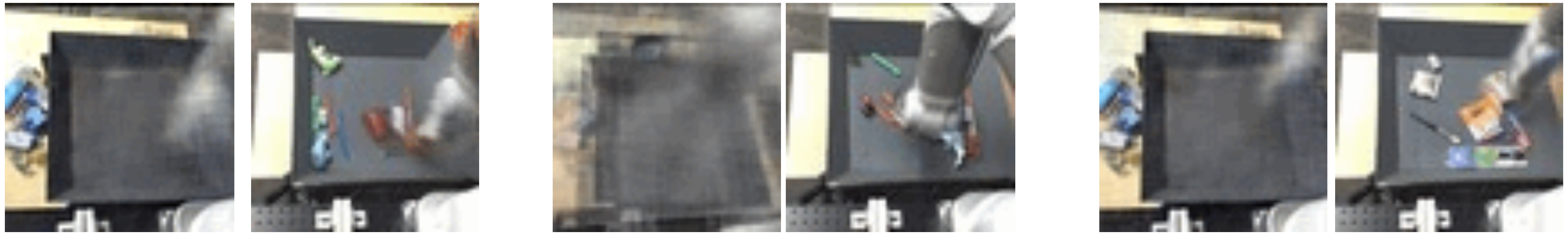
2. learn model $f_\phi(\mathbf{o}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{o}_i, \mathbf{a}_i) - \mathbf{o}'_i||^2$



- deep recurrent network
- multi-frame prediction
- action-conditioned
- explicitly model motion

ground truth video

predicted video



Note: recurrence versus meta-learning?

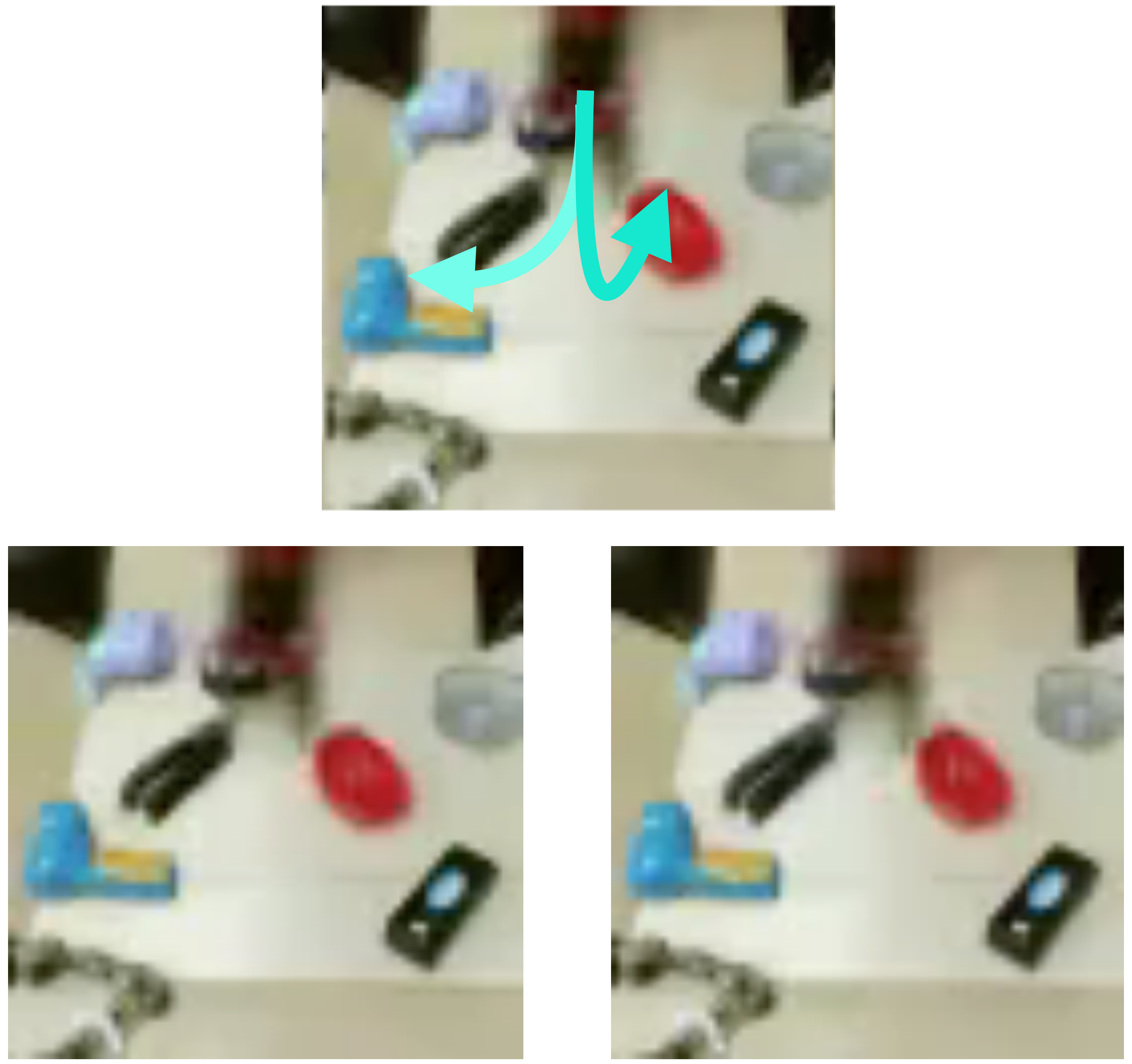
Finn, Goodfellow, Levine NIPS '16

How to plan?

3. use model $f_\phi(\mathbf{o}, \mathbf{a})$ to optimize action sequence

1. Consider potential action sequences
2. Predict the future for each action sequence
3. Pick best future & execute corresponding action
4. Repeat 1-3 to replan in real time

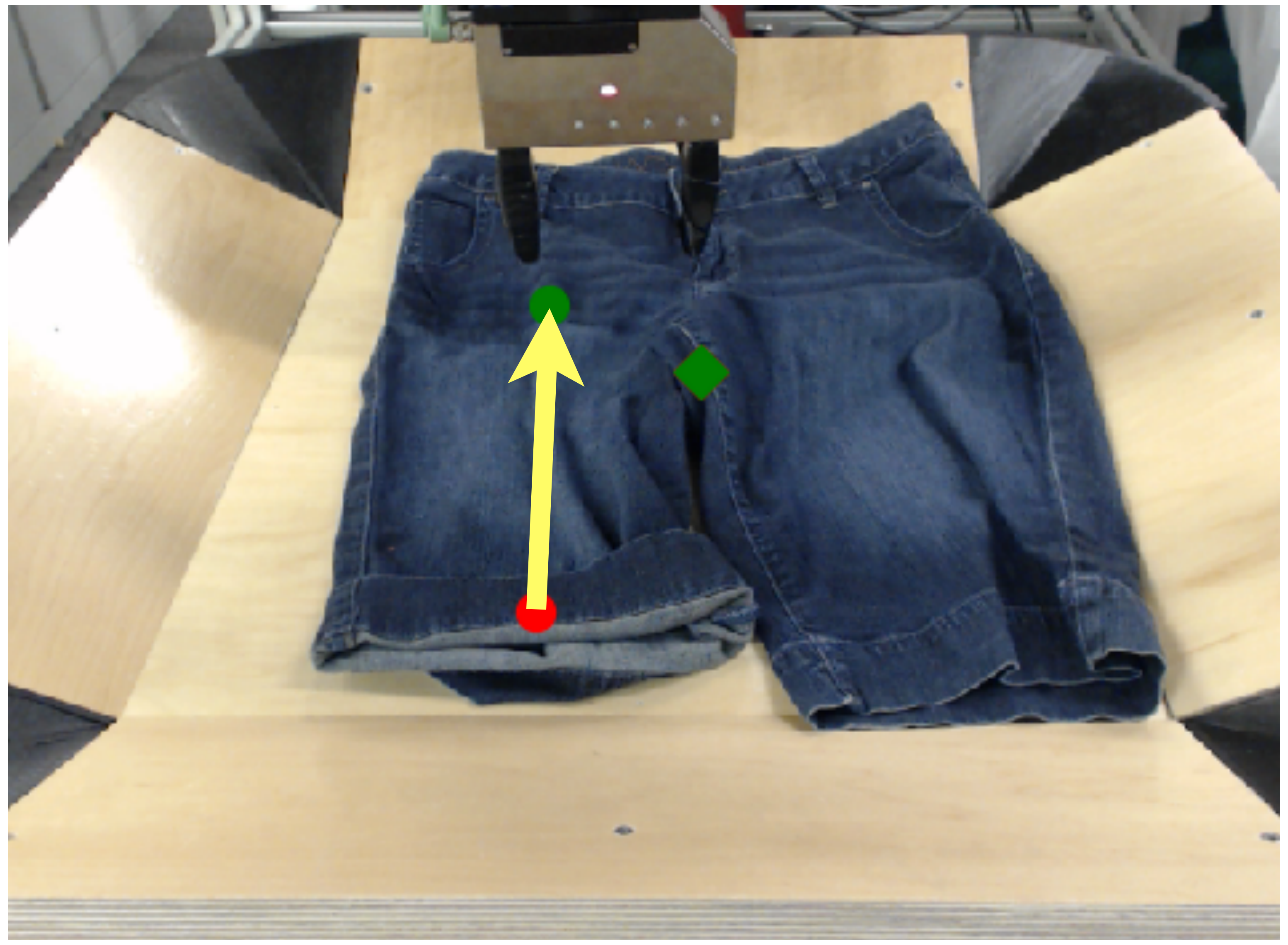
visual "model-predictive control" (MPC)



Finn & Levine. Deep Visual Foresight for Planning Robot Motion. ICRA '17

How it works

Specify goal



Visual MPC execution

Visual MPC w.r.t. goal

Ebert*, Finn*, Dasari, Xie, Lee, Levine. Visual Foresight, arXiv '18

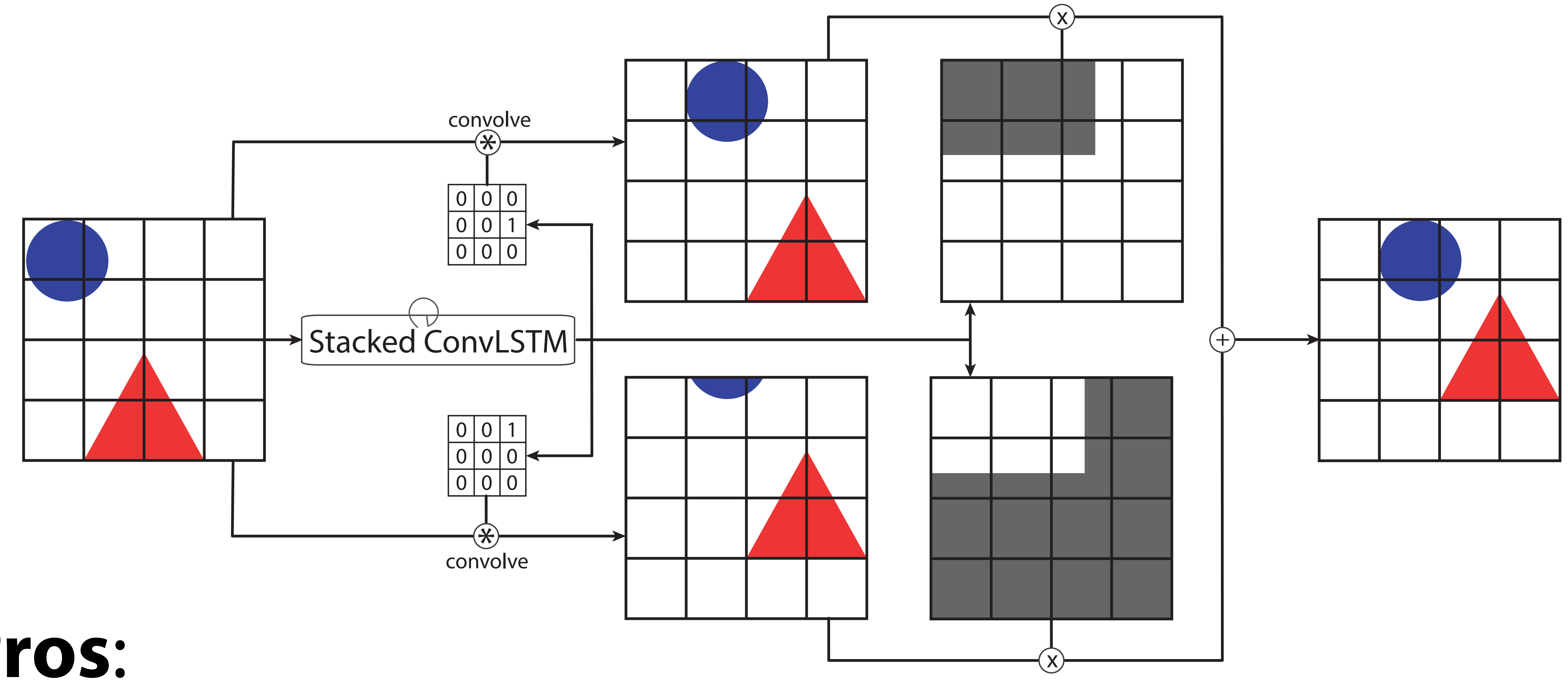
Planning with a **single model** for many tasks

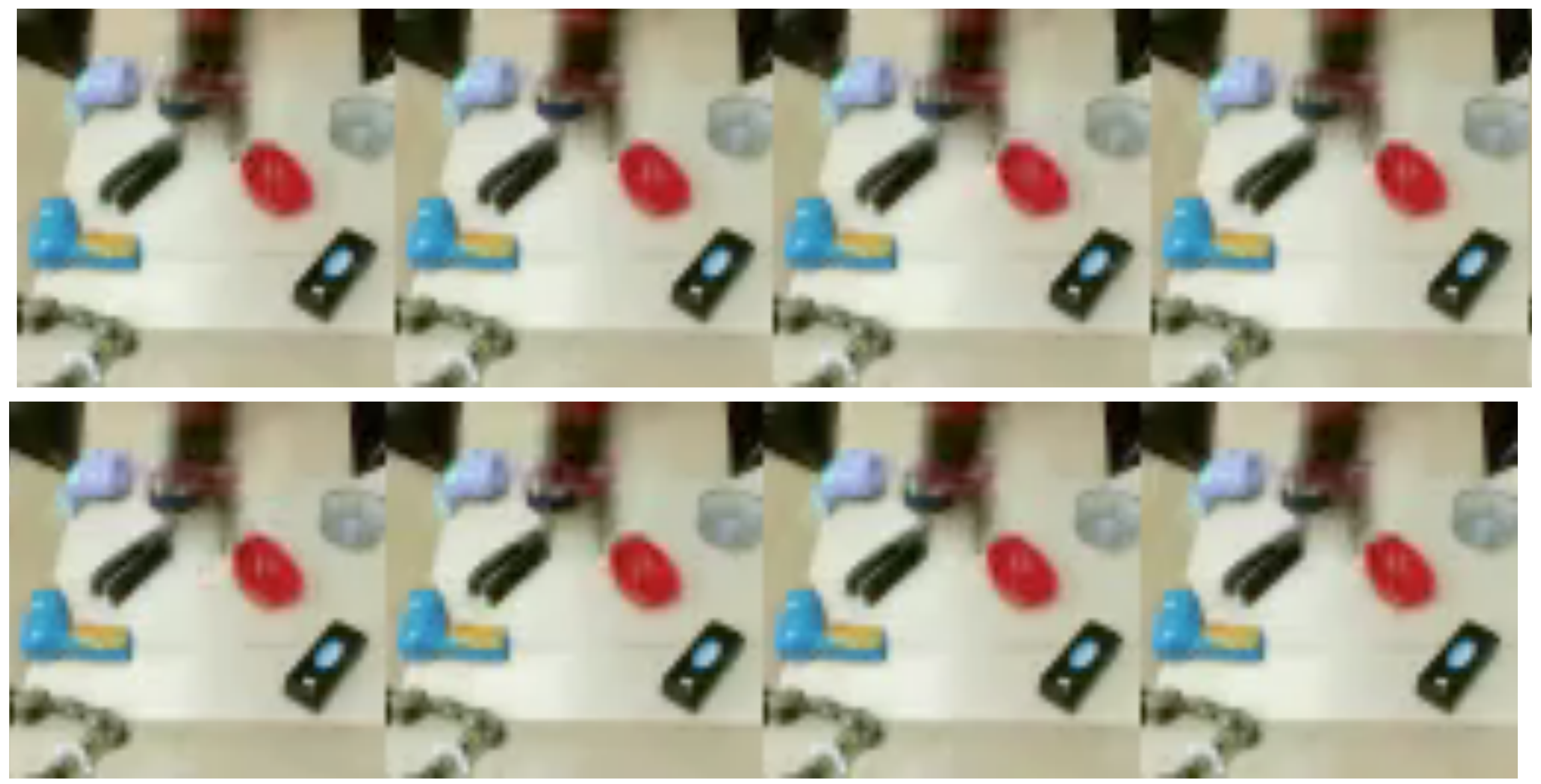
Video speed: 2x

Model training is self-supervised:
Only human involvement during training is:
programming initial primitives and providing objects

Ebert*, Finn*, Dasari, Xie, Lee, Levine. Visual Foresight, arXiv '18

action-conditioned multi-frame video prediction via **flow prediction**





Pros:

+ Real images
+ Very limited human involvement (model training is self-supervised)
+ Can accomplish many tasks with single model

Cons:

- Despite real images, limited background variability
- Can't [yet] handle as complex skills as other methods
- Compute intensive at test-time

Aside (time permitting):
Can we learn a breadth of *complex* skills?

Collect data

Learn to predict

$I_t, a_{t:t+H} \rightarrow I_{t:t+H}$

Plan using model

direct data collection

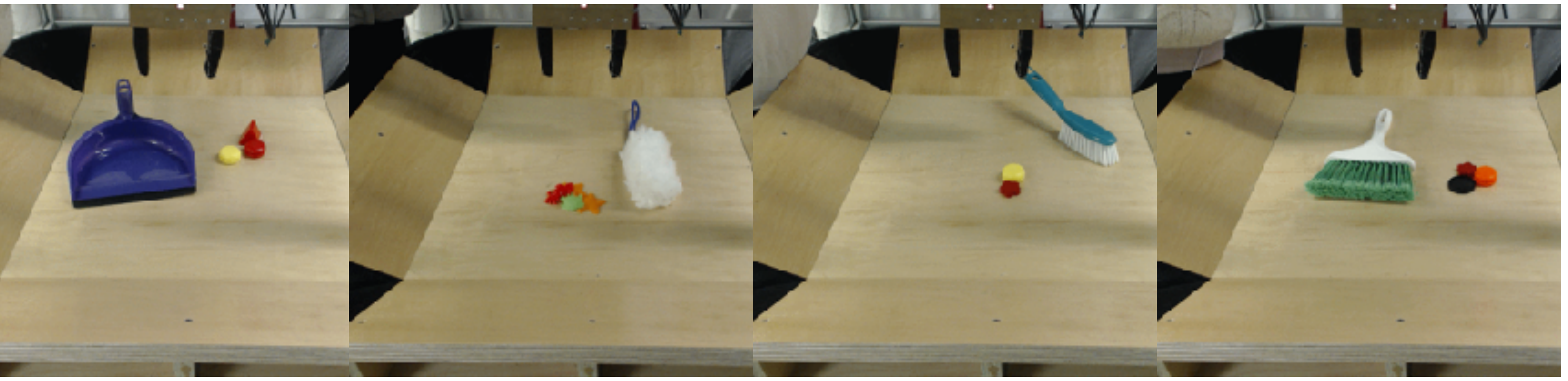
improve model

guide planning

Collect **diverse**, multi-task demonstrations

Fit model of $p(a_{t:t+H} \mid I_t)$ to the demonstration data.

Example multi-task demonstrations:

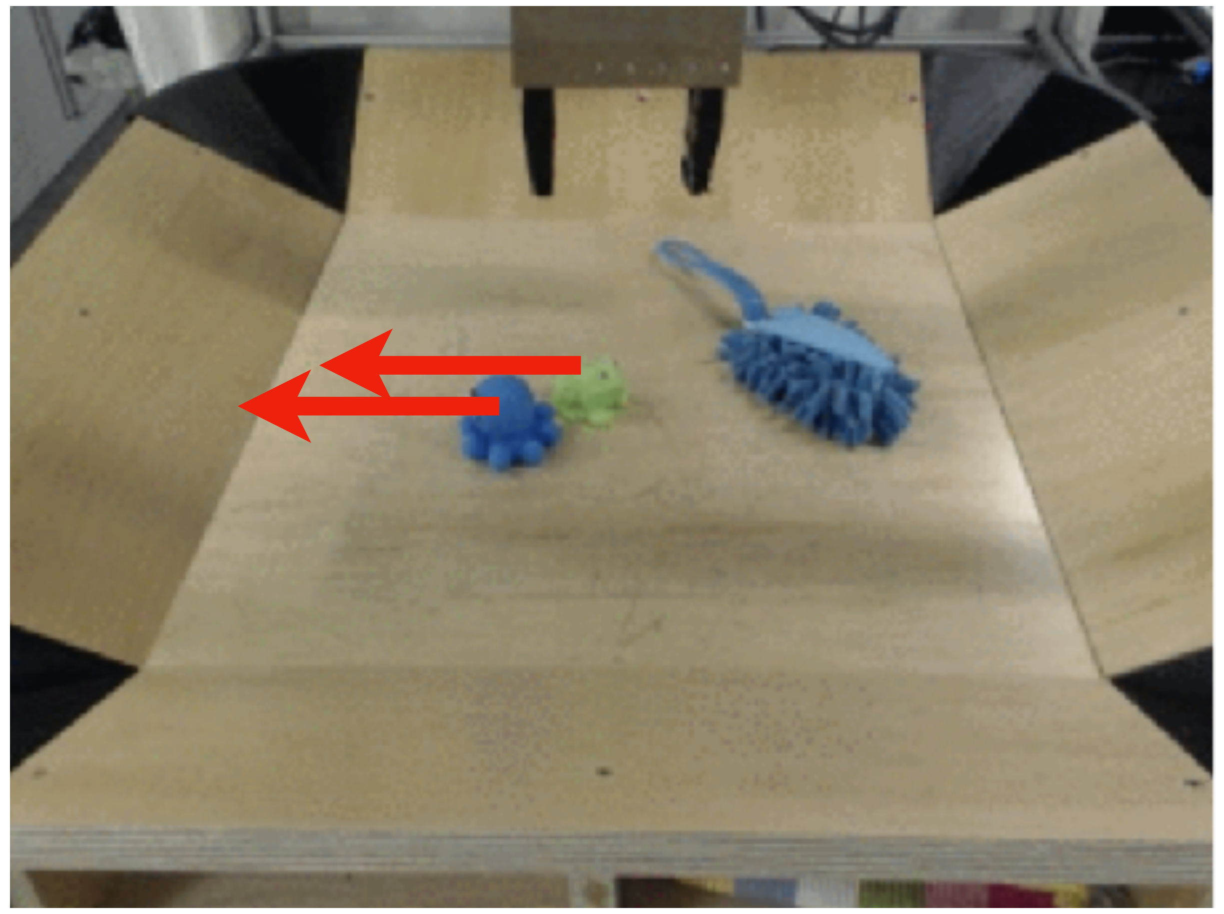


Samples from action proposal model:

Xie, Ebert, Levine, Finn. Improvisation through Physical Understanding, RSS '19

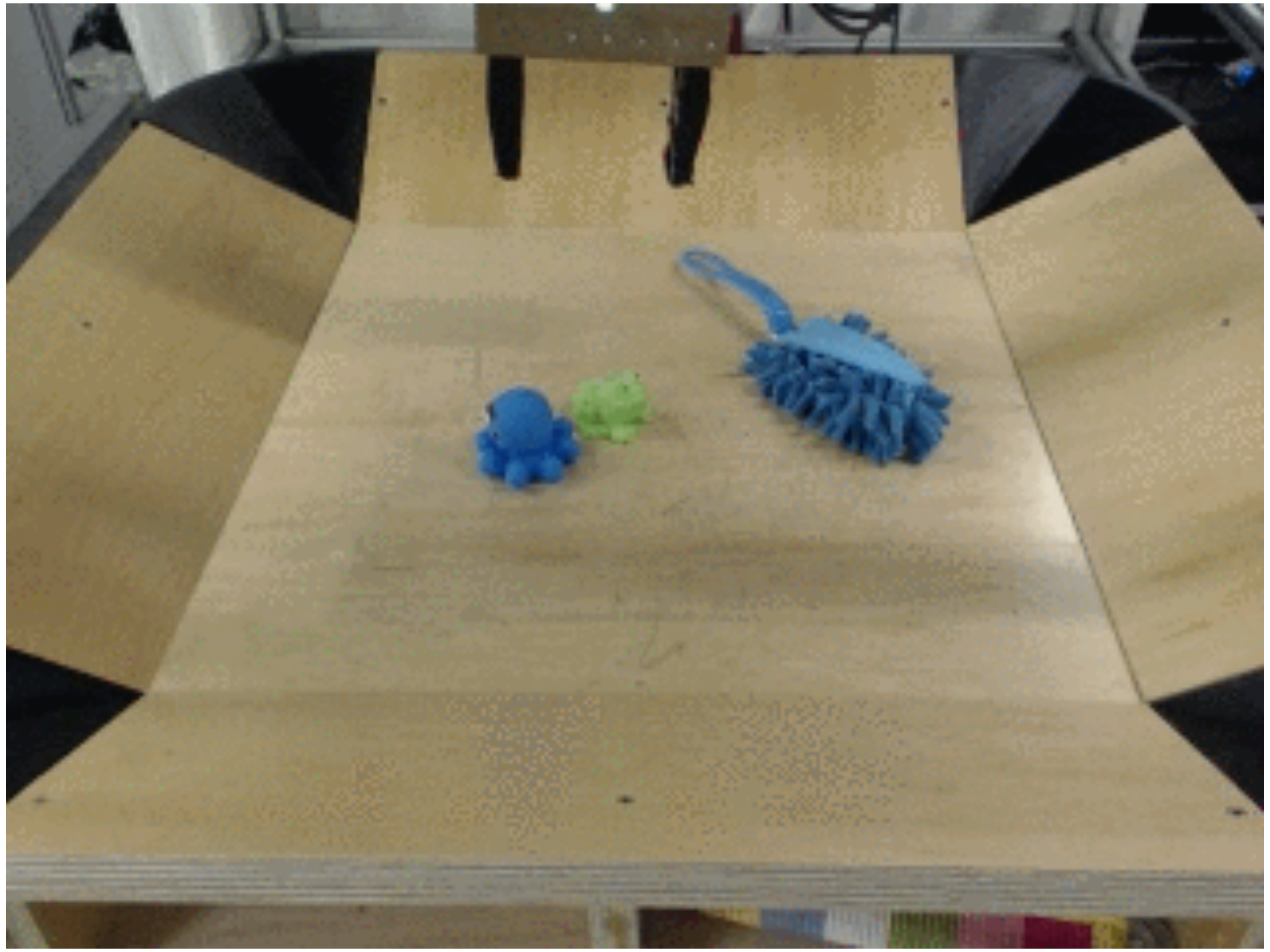
How it works

Specify goal



Guided visual planning w.r.t. goal

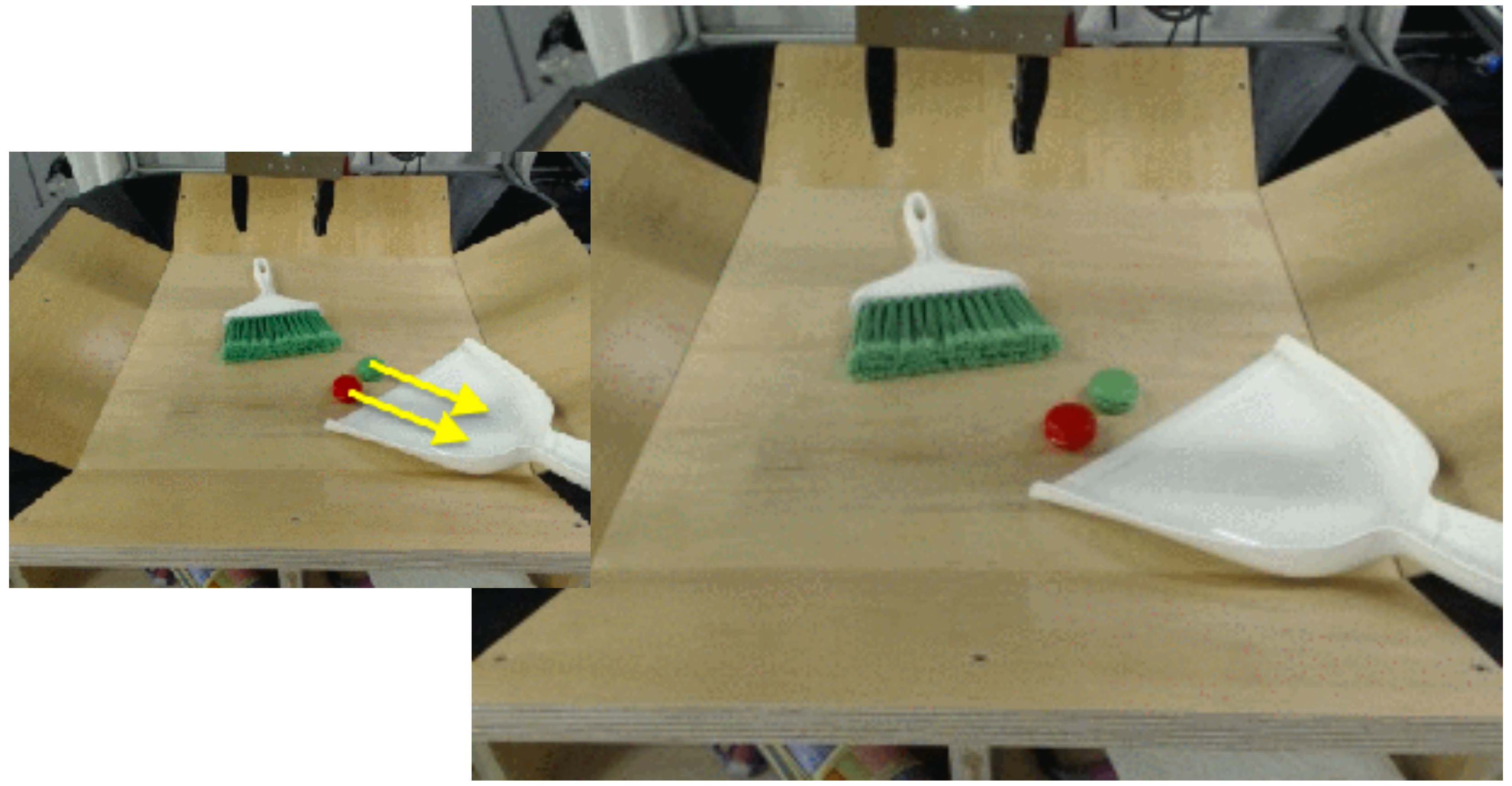
Executing actions



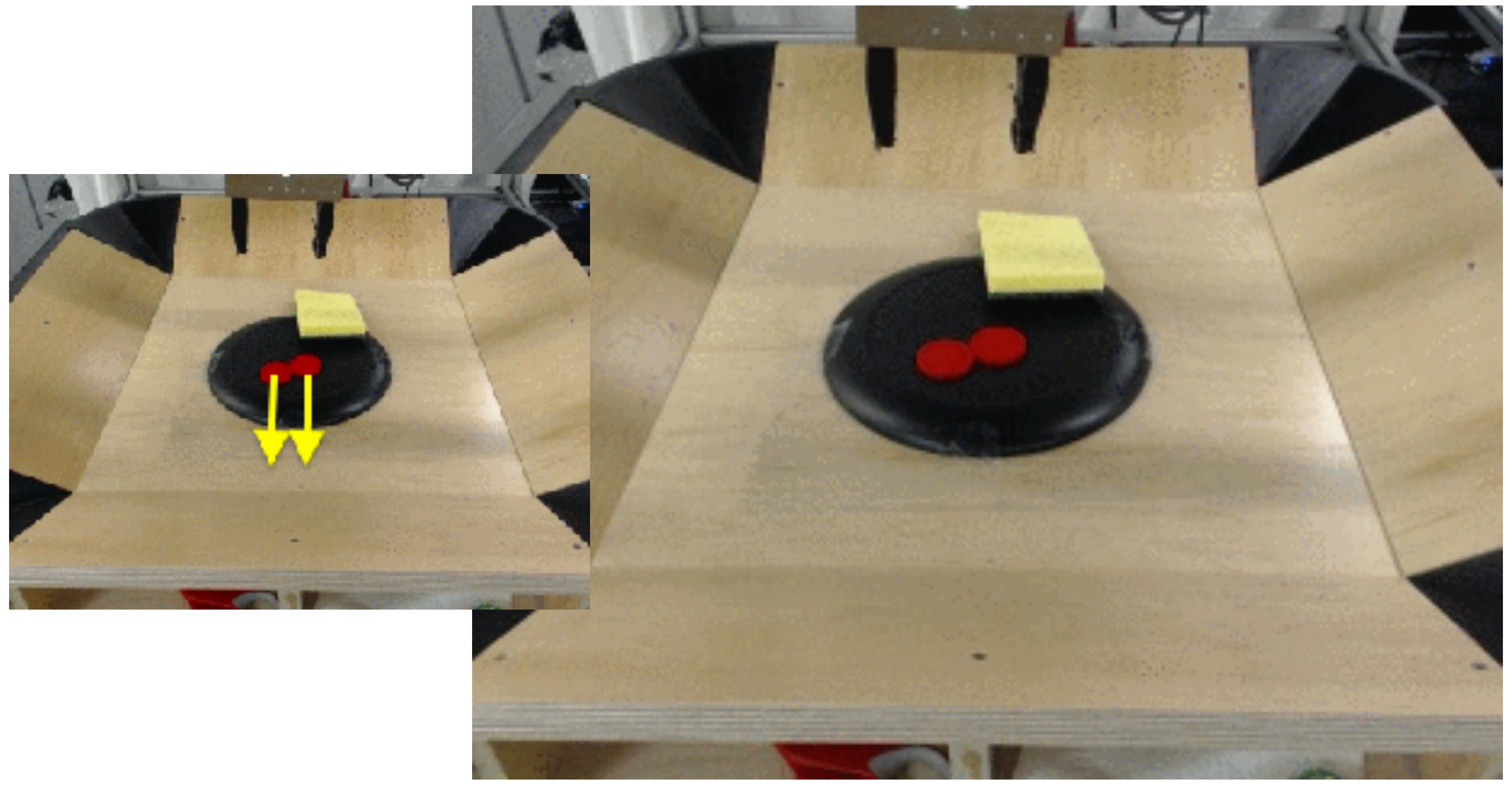
Xie, Ebert, Levine, Finn. Improvisation through Physical Understanding, RSS '19

Planning with a **single model** for many tasks

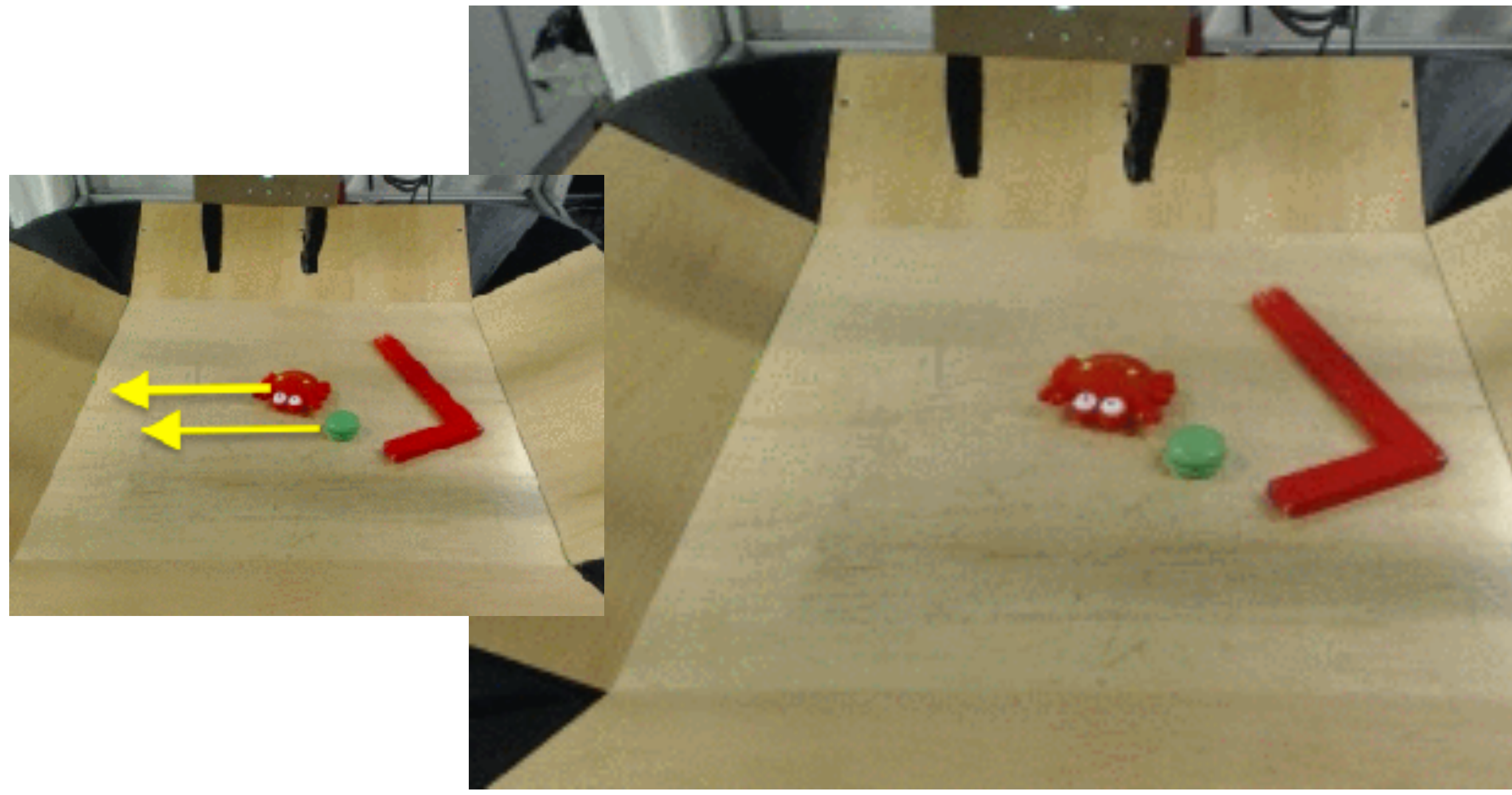
solve new tasks



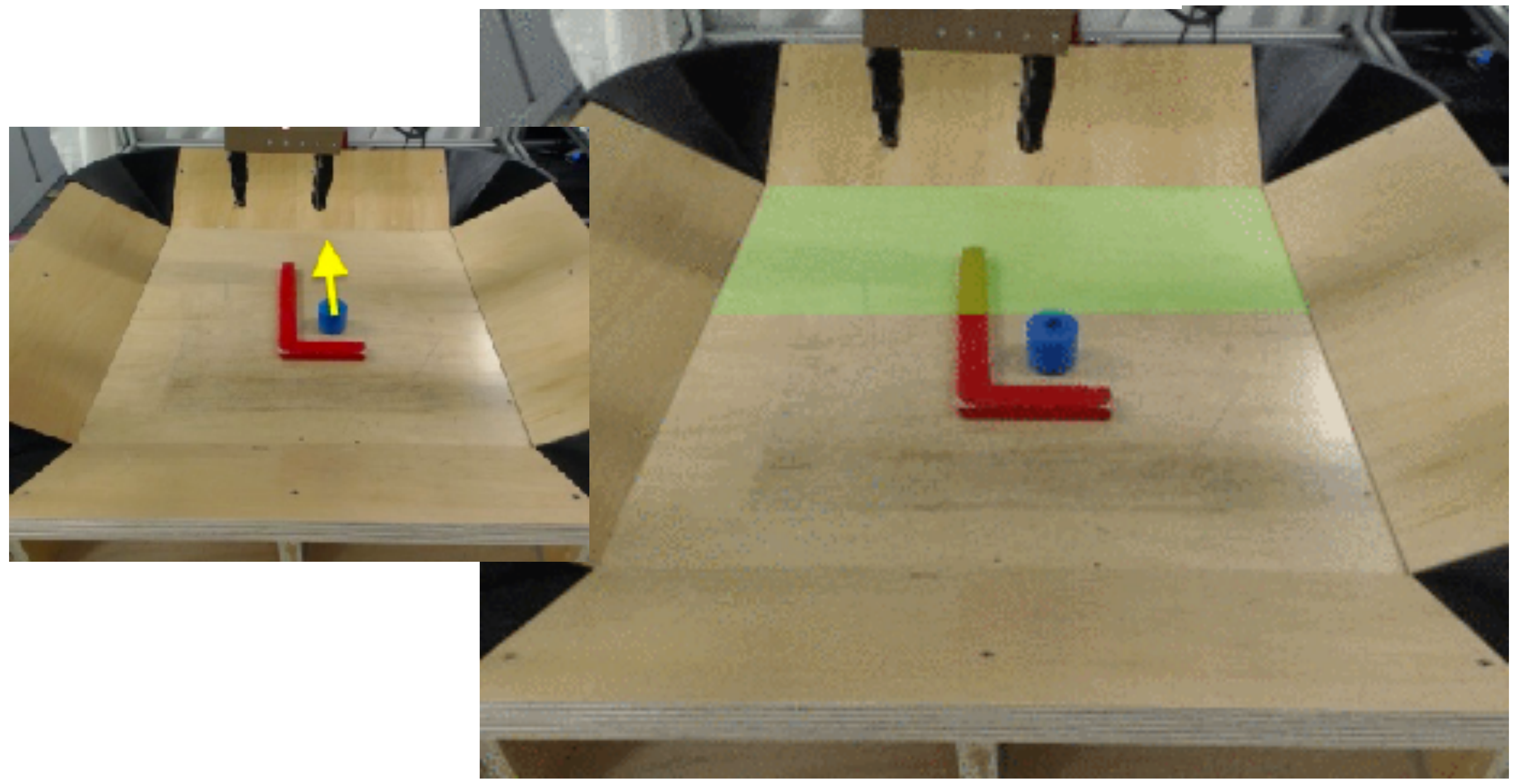
unseen tools



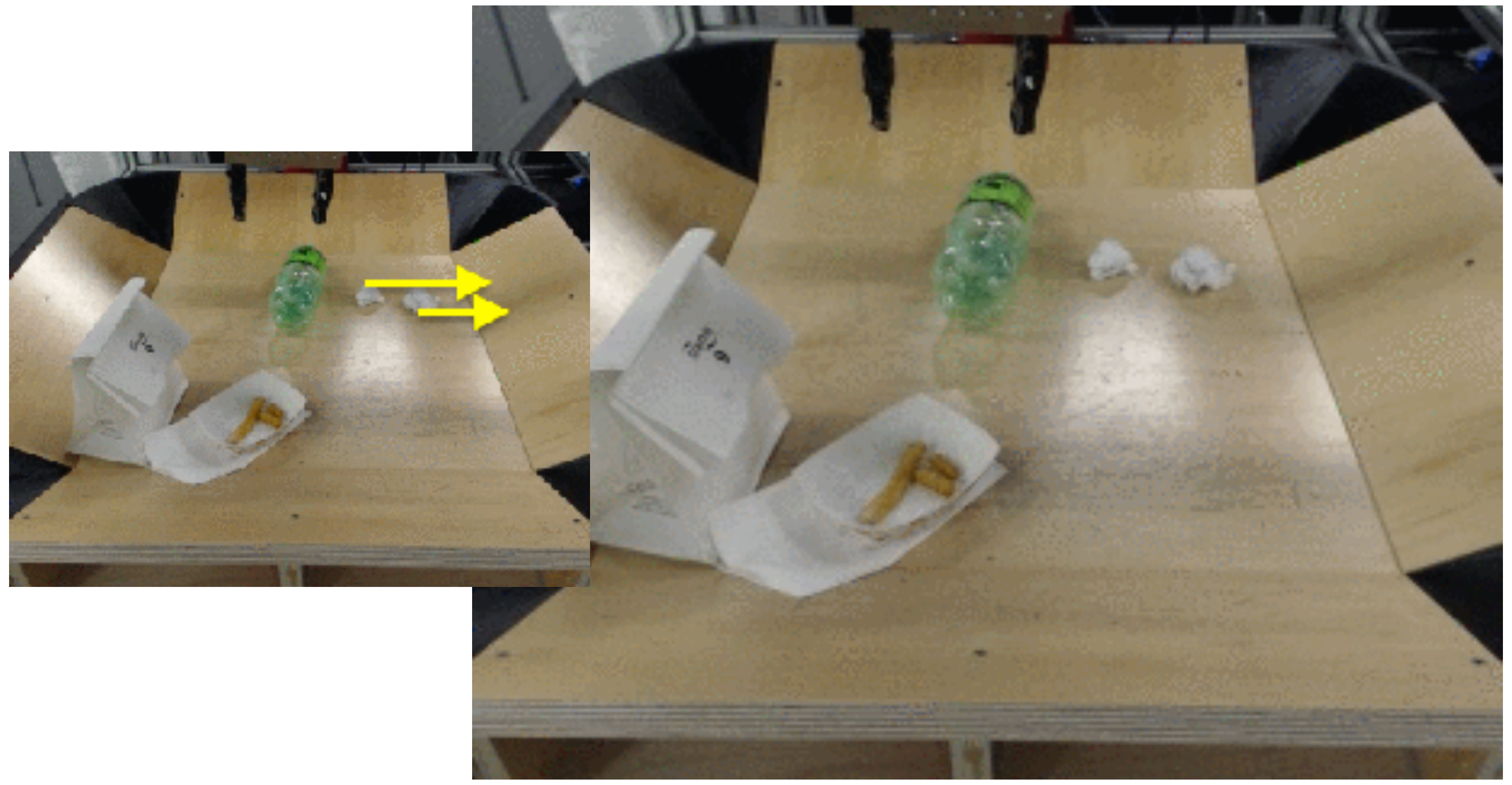
decide when to use a tool…



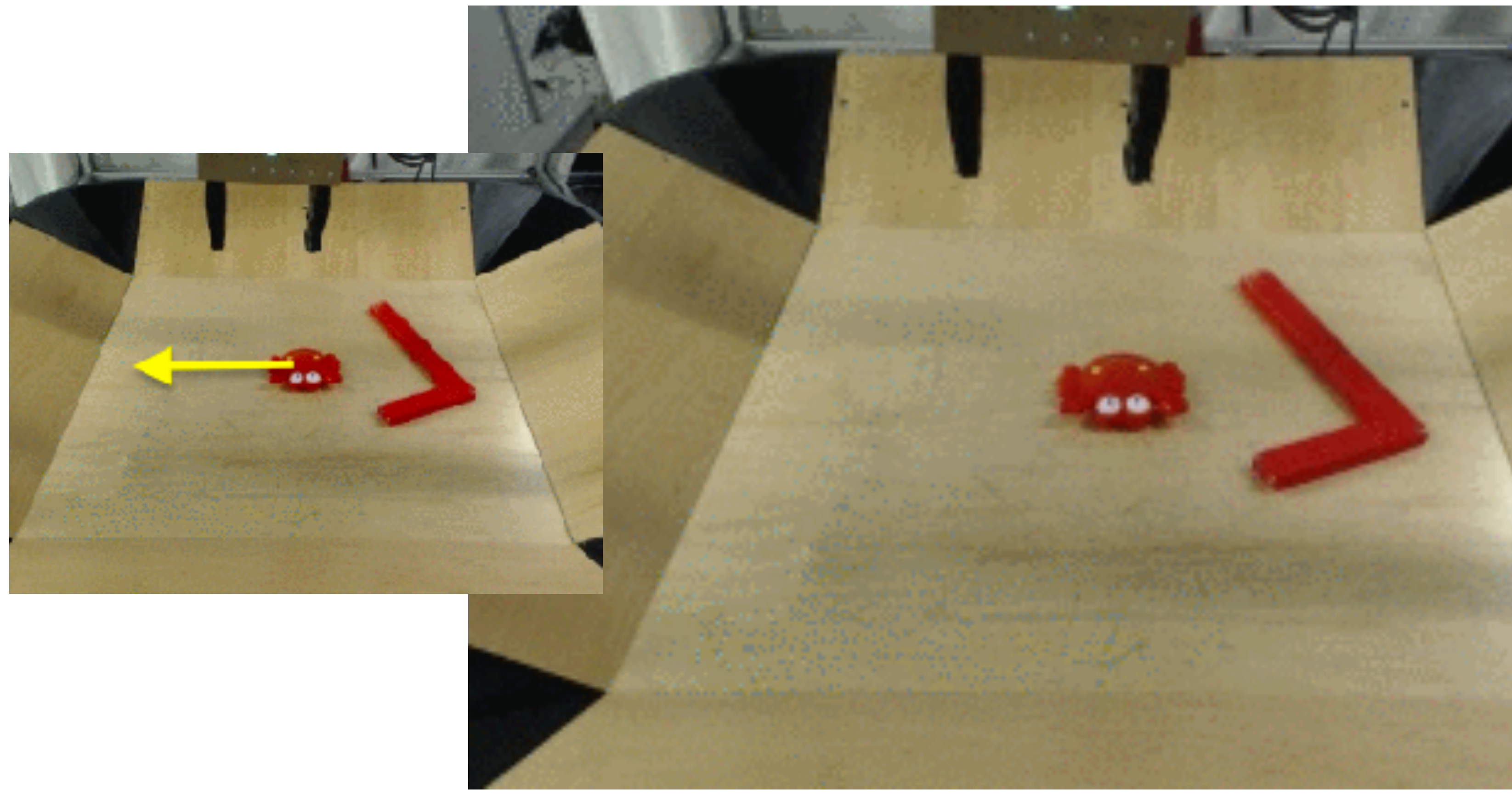
out-of-reach objects



unseen *unconventional* tools



…and when not to



Xie, Ebert, Levine, Finn. Improvisation through Physical Understanding, RSS '19

Learning with Image Observations

1. Models in latent space
2. Models directly in image space
3. **Predict alternative quantities**

Predict alternative quantities

If I take a sequence of actions:

Pinto et al. '16

Will I successfully grasp?

Will I collide?

Kahn et al. '17

Dosovitskiy & Koltun '17

What will health/damage/etc. be?

close connection to Q-learning

(when reward = p(event))

Pros: + Only predict task-relevant quantities!

Cons: - Need to manually pick quantities, must be able to <u>directly observe</u> them

The Plan

Model-based RL

and how it can be used for multi-task & meta-learning

Model-based RL with image observations

or other high-dimensional inputs

Model-based meta-RL

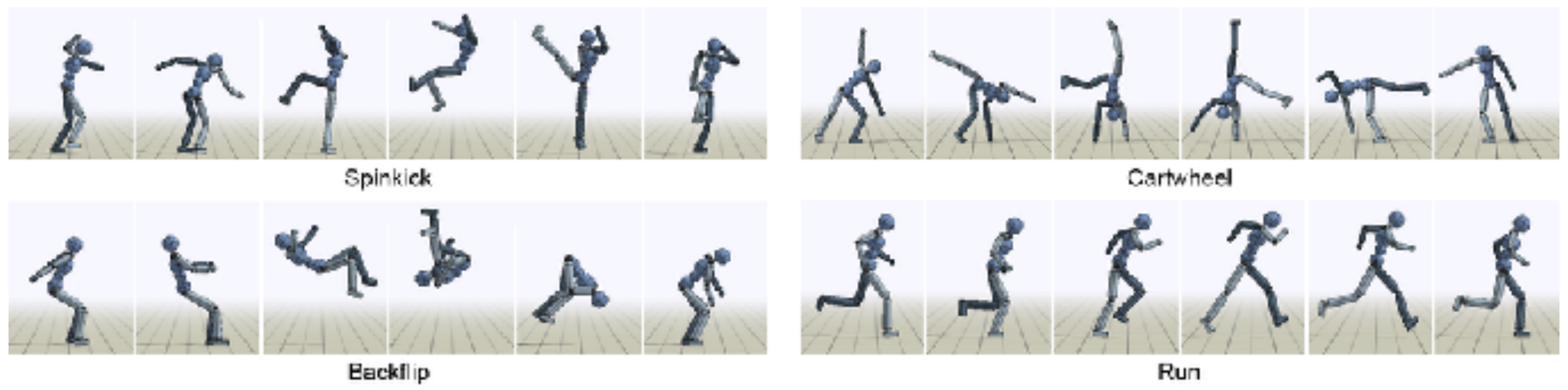
Recall: What is a reinforcement learning task?

Observation: In many situations: $p_i(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ does not vary across tasks.

The real world:

- object manipulation
- legged locomotion
- navigation

Character animation maneuvers



Task-directed dialog.

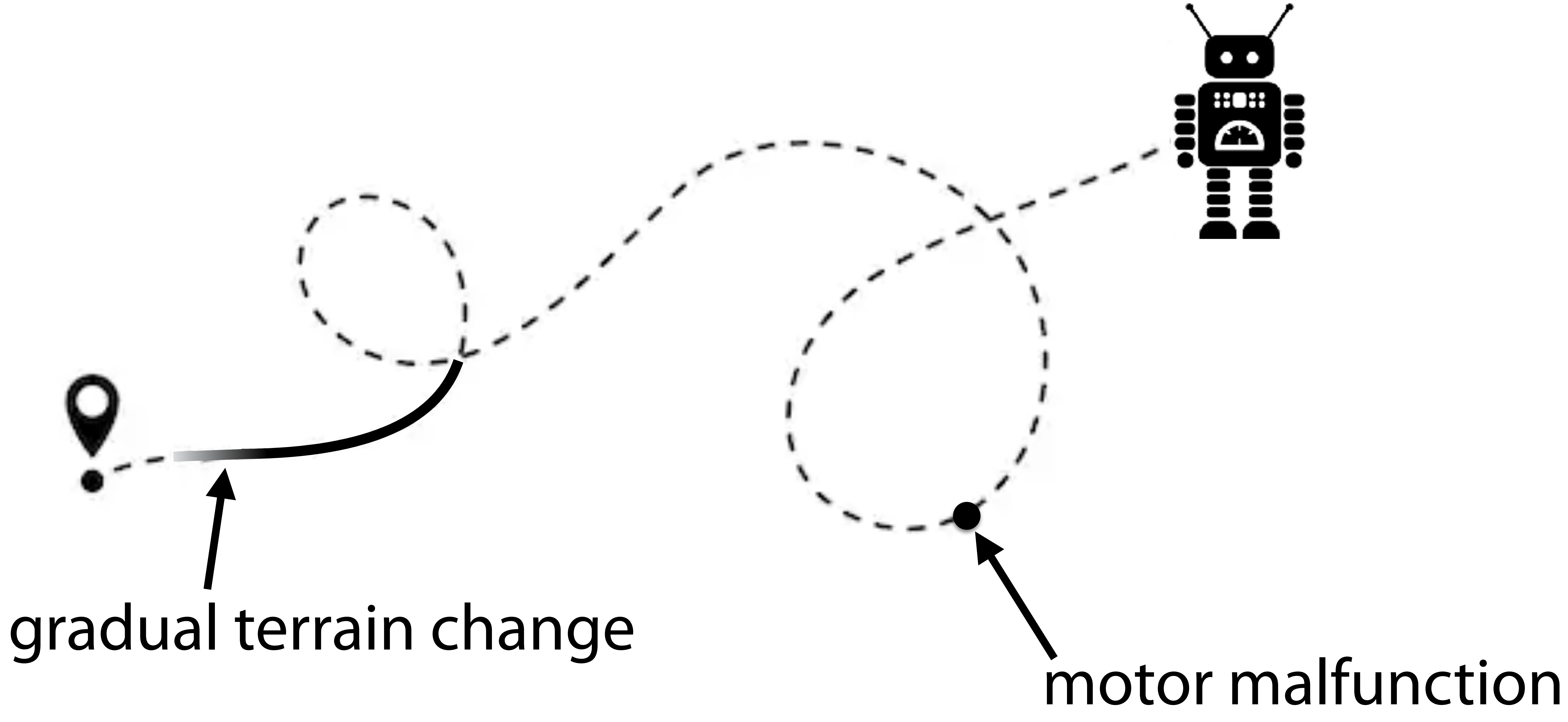
In these cases, estimating the model is a single-task problem!

What about when the dynamics $p_i(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ changes across tasks?

Turn model learning from supervised
learning into meta-learning problem!

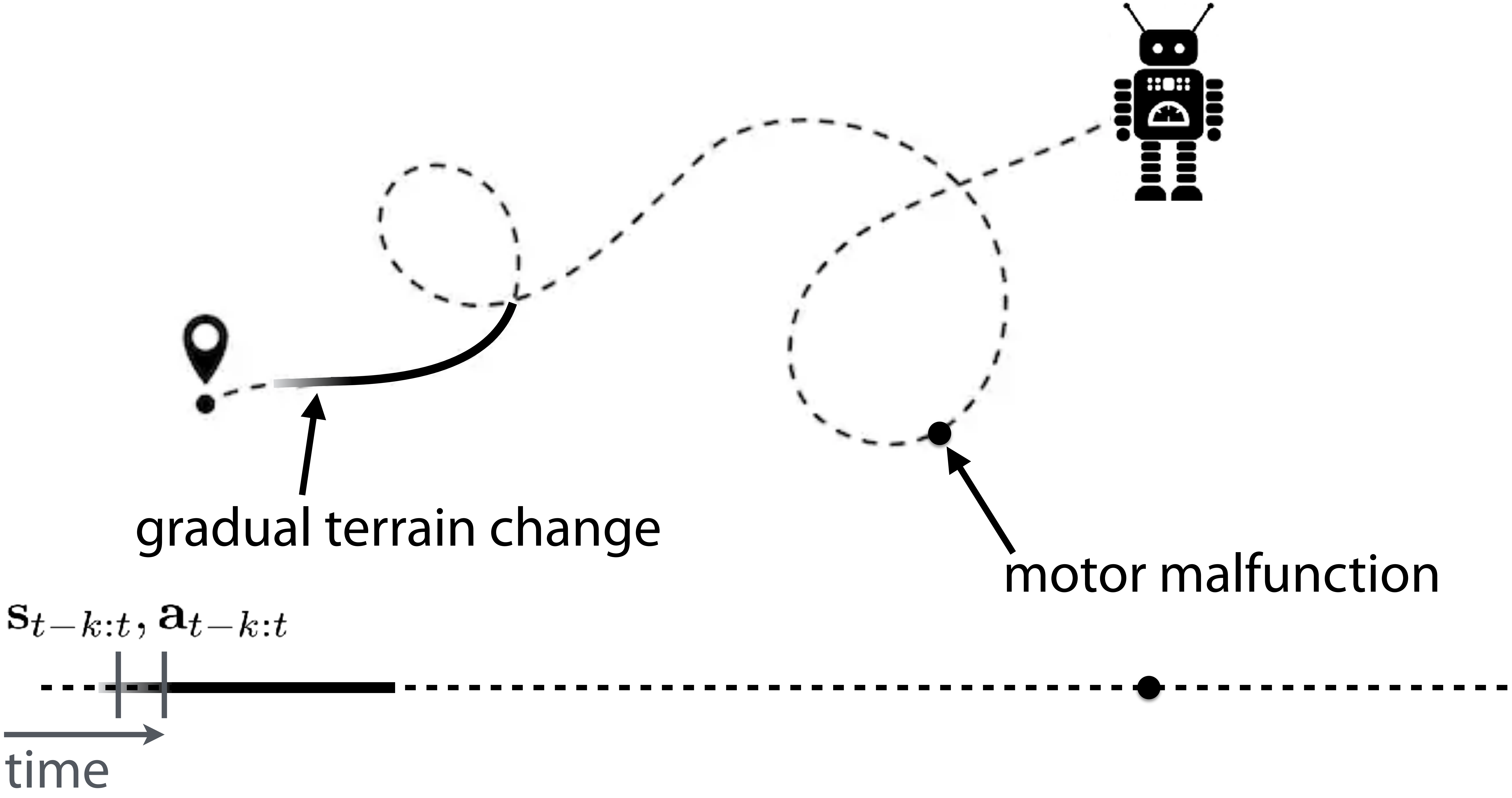
$$p_i(\mathbf{s}'|\mathbf{s}, \mathbf{a}, \mathscr{D}_i^{\mathrm{tr}})$$

Deriving tasks from dynamic environments



Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR'19

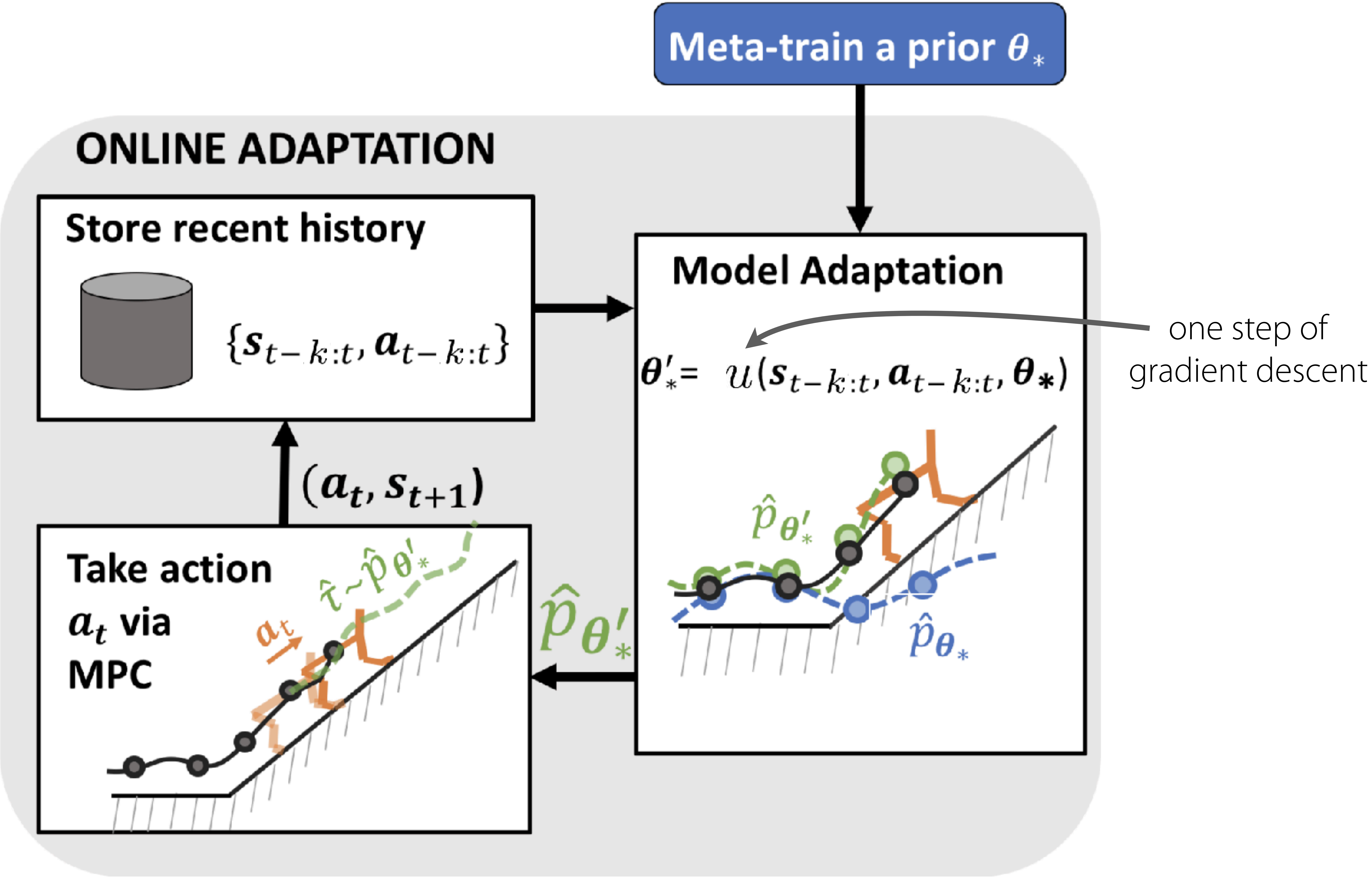
Deriving tasks from dynamic environments



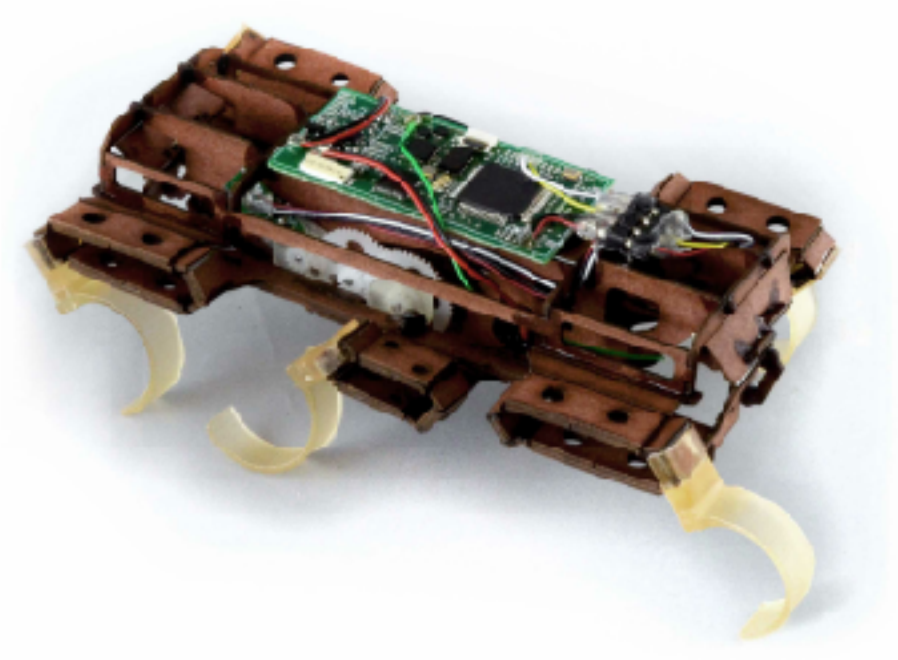
online adaptation = few-shot learning

tasks are **temporal slices** of experience

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR'19



Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR'19



VelociRoACH Robot

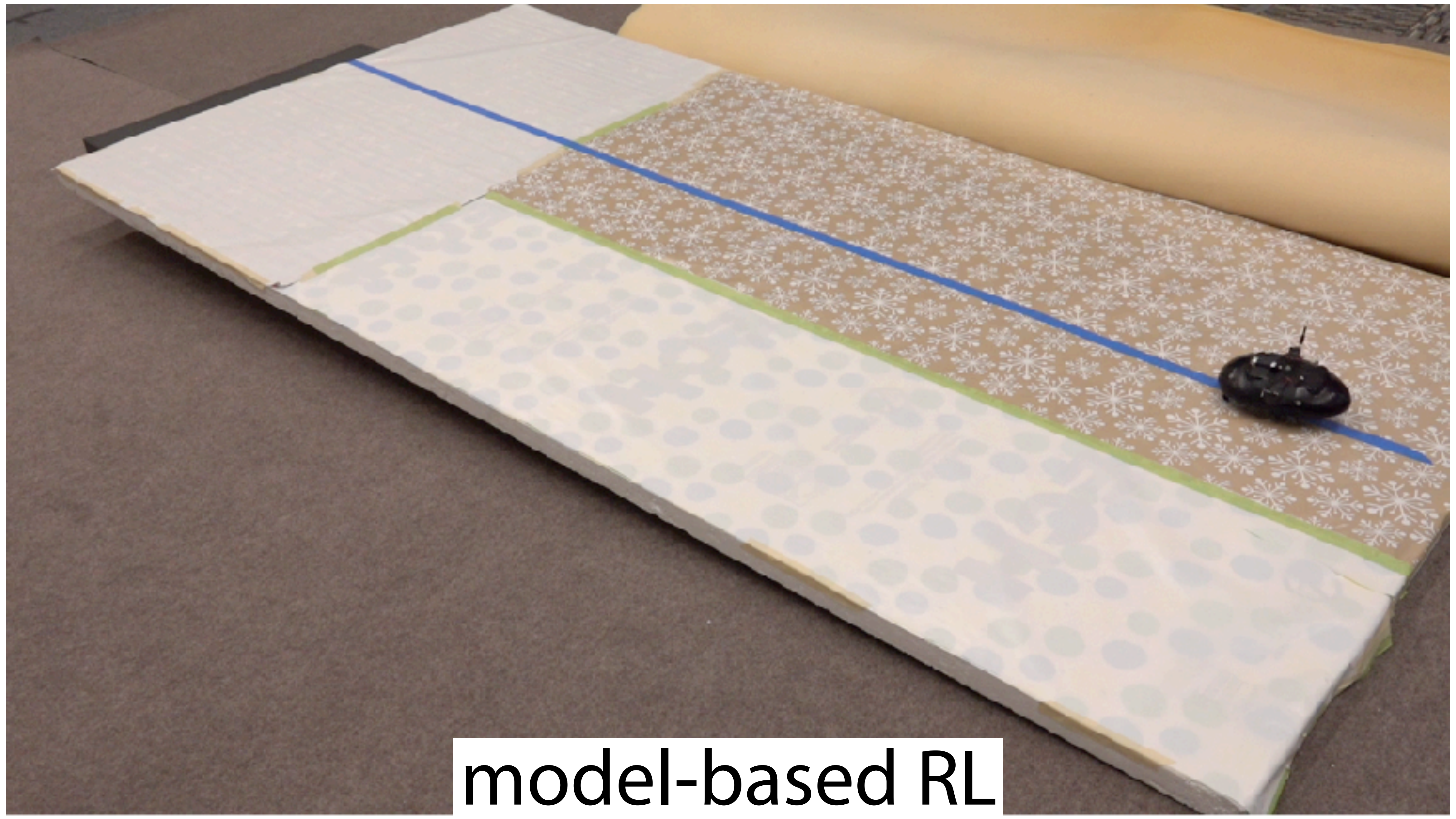
Meta-train on **variable terrains**

Meta-test with **slope, missing leg, payload, calibration errors**

Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR'19

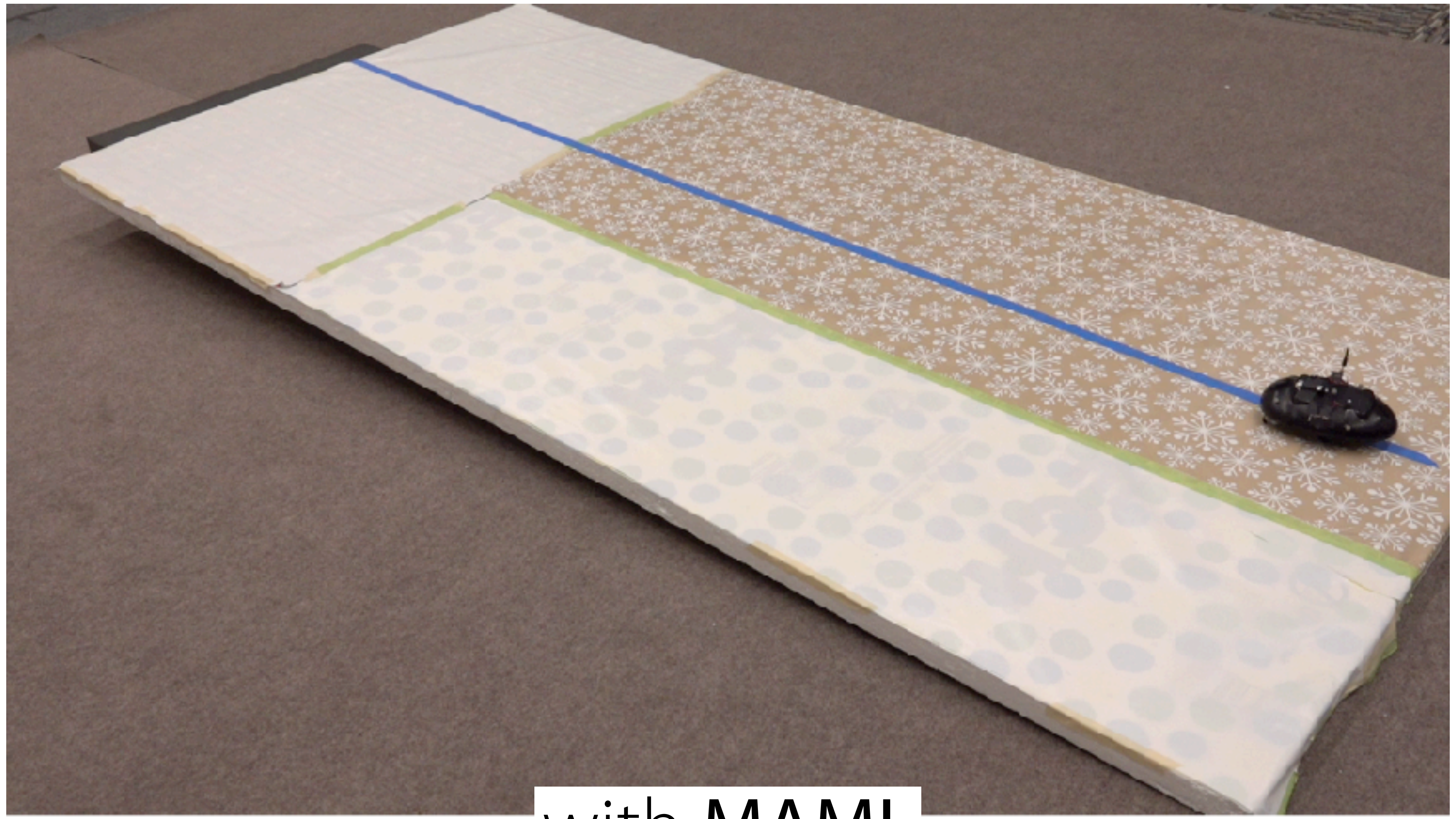
VelociRoACH Robot

Meta-train on **variable terrains** **Meta-test** with **<u>slope</u>**, **missing leg, payload, calibration errors**

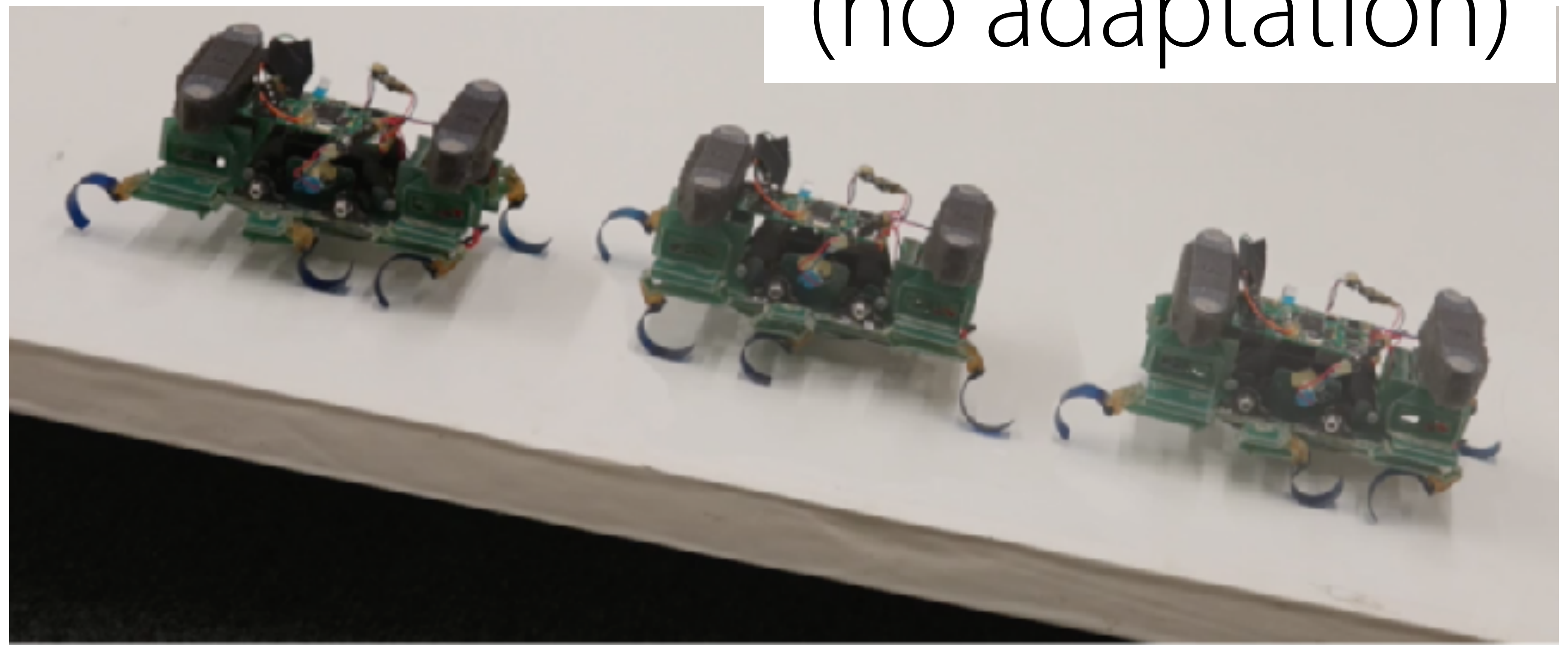


model-based RL

(no adaptation)



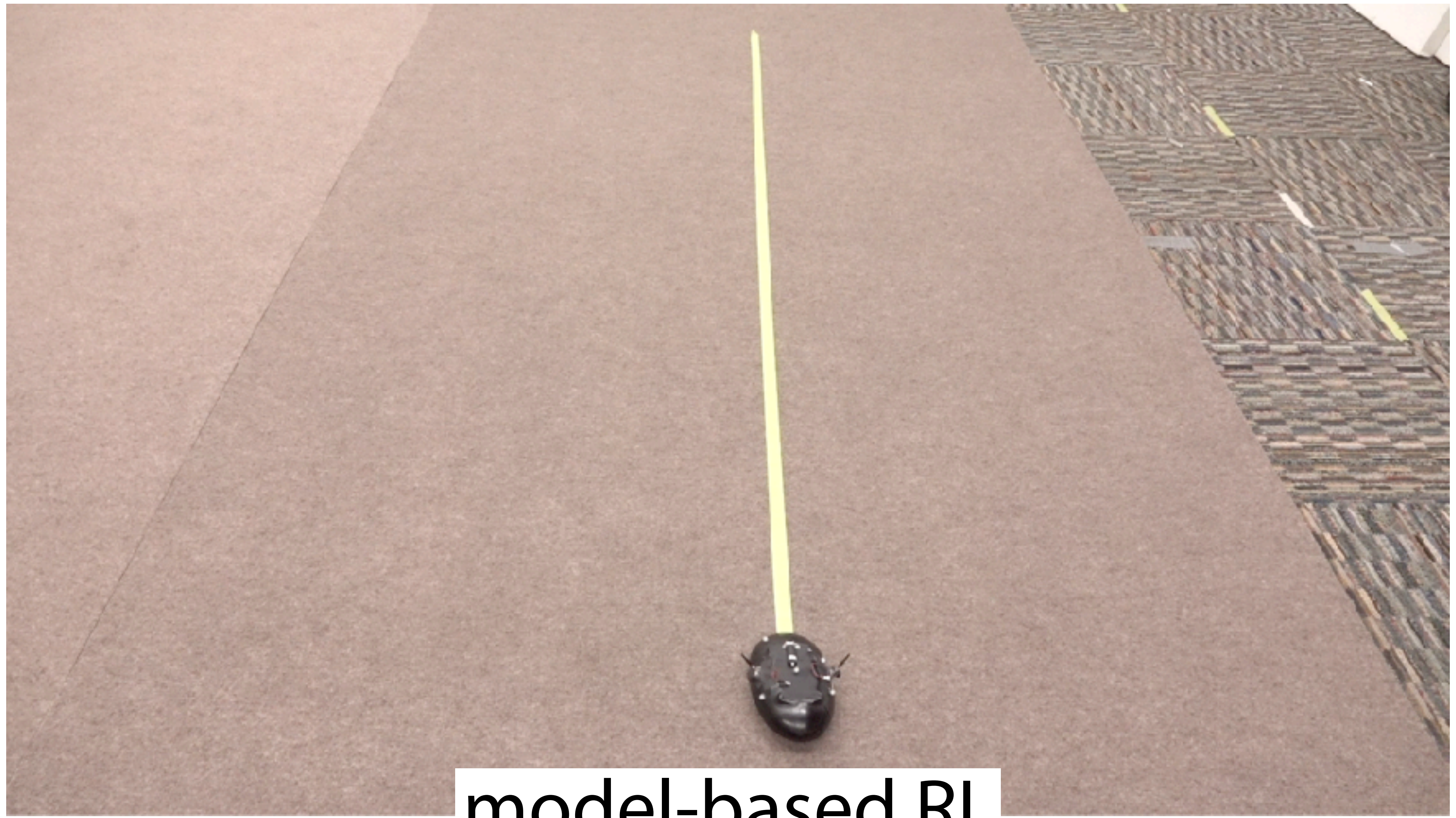
with **MAML**



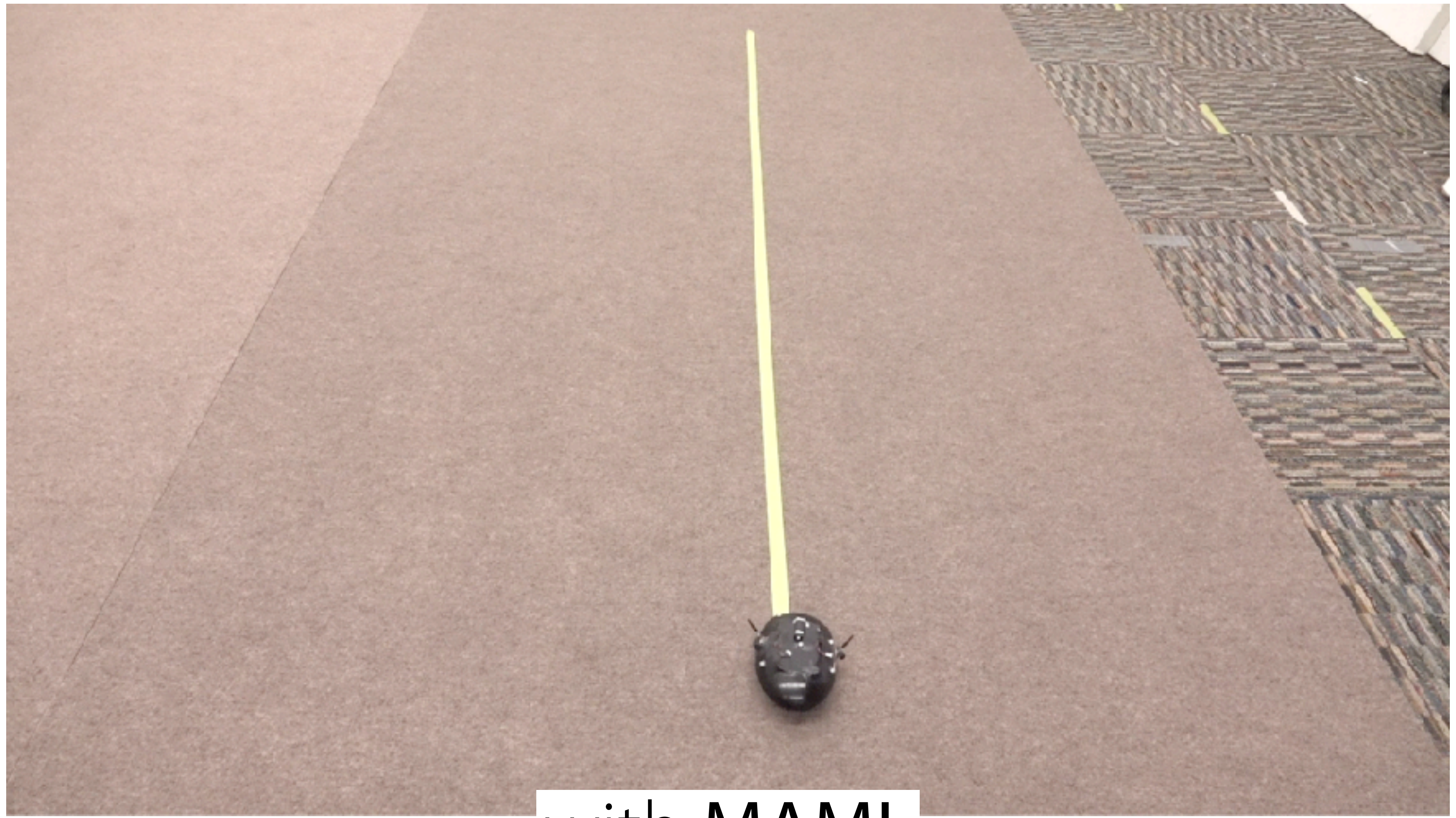
Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR '19

VelociRoACH Robot

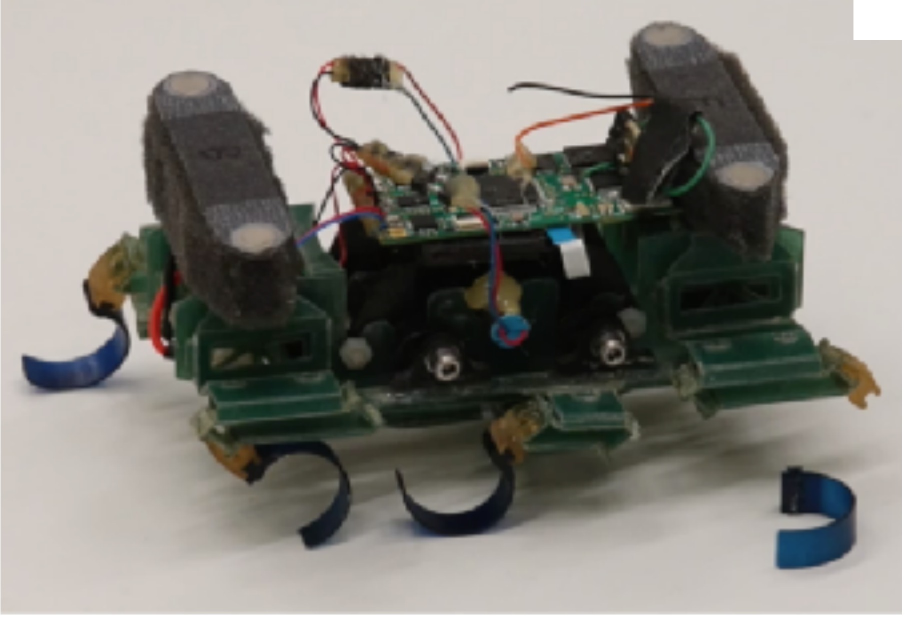
Meta-train on **variable terrains** **Meta-test** with **slope**, **<u>missing leg</u>**, **payload**, **calibration errors**



model-based RL
(no adaptation)



with **MAML**



Nagabandi*, Clavera*, Liu, Fearing, Abbeel, Levine, Finn. Learning to Adapt in Dynamic Environments through Meta-RL. ICLR'19

Takeaways: Model-Based vs. Model-Free Learning

Models:

+ Easy to collect data in a scalable way (self-supervised)
+ Easy to transfer across rewards
+ Typically require a smaller quantity of reward-supervised data
- Models don't optimize for task performance
- Sometimes harder to learn than a policy
- Often need assumptions to learn complex skills (continuity, resets)

Model-Free:

+ Makes little assumptions beyond a reward function
+ Effective for learning complex policies
- Require a lot of experience (slower)
- Harder optimization problem in multi-task setting

Ultimately we will want elements of both!

Remaining Questions: The Next Few Weeks

What about seeing tasks **in sequence**?

Other topics not covered?

What are the current research frontiers?

Monday lecture
Lifelong learning

Wednesday paper presentations
Misc topics: task interference, differentiability, sim2real, hybrid RL

Monday 11/18:
Jeff Clune guest lecture
(evolutionary methods, lifelong learning, meta-learning)

Wednesday 11/20
Sergey Levine
information-theoretic exploration

Monday 12/2
My perspective on outstanding challenges & frontiers

Reminders

Homework 3 due **tonight.**

Project milestone due **next Wedesday.**