Model-Based Reinforcement Learning
for Multi-Task Learning

CS 330

Logistics

Midquarter survey is out.

Homework 3 out, due **Wednesday 10/27**

Plan for Today

Recap
goal-conditioned RL & relabeling <- Topic of HW3

Model-based RL
and how it can be used for multi-task learning

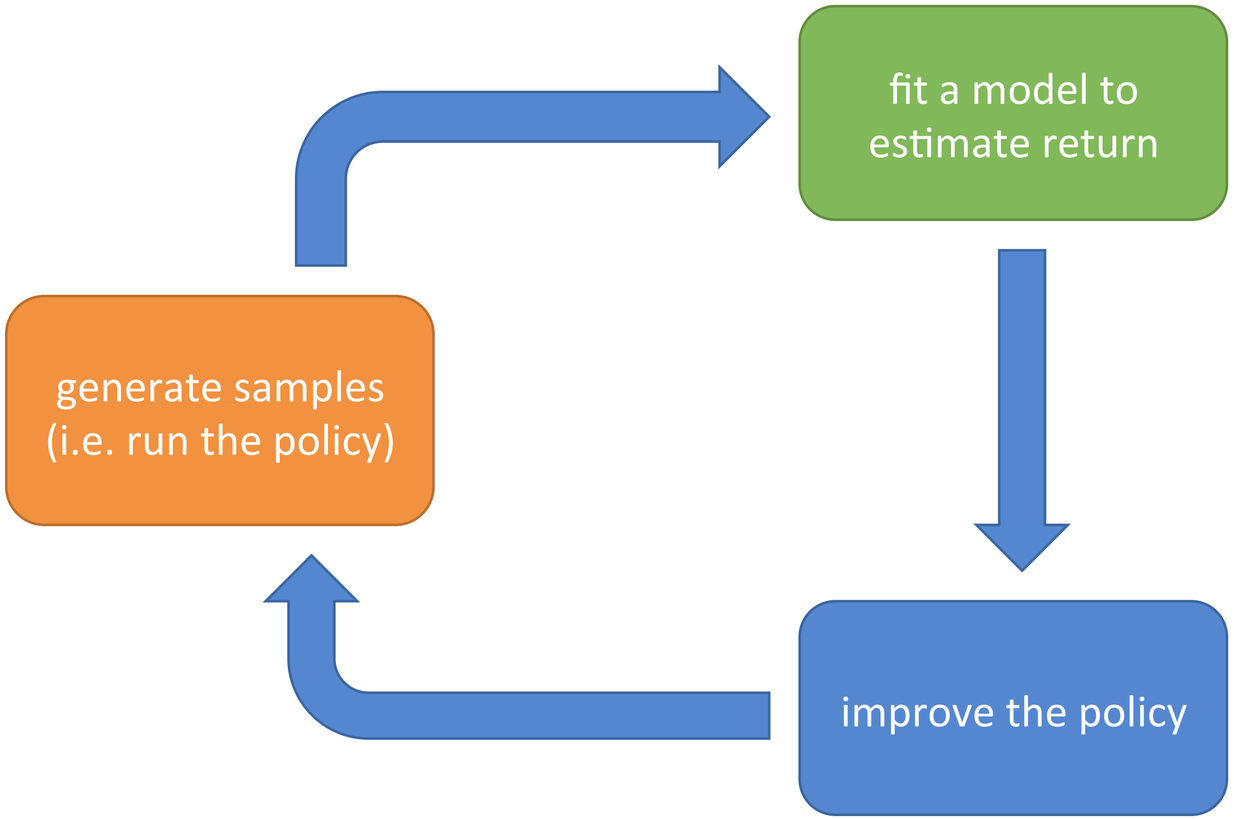
Model-based RL with image observations
or other high-dimensional inputs

Lecture goals:

- Understand how to use & implement **model-based RL**
- Understand how **model-based RL** can be used for **multi-task** RL
- Challenges and strategies for **model-based RL** in **high-dimensional spaces**

Recap: The anatomy of a reinforcement learning algorithm

Last week: introduced model-free RL methods (policy gradient, Q-learning)



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)

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)

$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)

Later in this lecture: focus on **model-based RL** methods

Recap: What is a reinforcement learning task?

action space dynamics

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

state space | initial state distribution | reward

Alternatively: A **task identifier** is part of the state: $\mathbf{s} = (\bar{\mathbf{s}}, \mathbf{z}_i)$ (original state)

Policy: $\pi_\theta(\mathbf{a} \mid \bar{\mathbf{s}}) \rightarrow \pi_\theta(\mathbf{a} \mid \bar{\mathbf{s}}, \mathbf{z}_i)$

Q-function: $Q_\phi(\bar{\mathbf{s}}, \mathbf{a}) \rightarrow Q_\phi(\bar{\mathbf{s}}, \mathbf{a}, \mathbf{z}_i)$

What is $\mathbf{z}_i$?

- one-hot task ID
- language description
- desired goal state, $\mathbf{z}_i = \mathbf{s}_g$ ← "**goal-conditioned** RL"

Recap: Example multi-task RL problem

Task 1: passing

Task 2: shooting goals

What if you accidentally perform a good pass when trying to shoot a goal?

Store experience as normal. *and* **Relabel** experience with task 2 ID & reward and store.

“hindsight relabeling” "hindsight experience replay” (HER)

Recap: Goal-conditioned RL with hindsight relabeling

1. Collect data $\mathcal{D}_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{s}_g, r_{1:T})\}$ using some policy
2. Store data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$
3. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ using last state as goal: $\mathcal{D}'_k = \{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}, \mathbf{s}_T, r'_{1:T}\}$ where $r'_t = -d(\mathbf{s}_t, \mathbf{s}_T)$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_k$
4. Update policy using replay buffer $\mathcal{D}$

k++

<— Other relabeling strategies?

use any state from the trajectory

Result: exploration challenges alleviated via targeted ***data sharing***

You will implement this algorithm in homework 3!

Kaelbling. Learning to Achieve Goals. IJCAI '93
Andrychowicz et al. Hindsight Experience Replay. NeurIPS '17

The Plan

Recap
goal-conditioned RL & relabeling <- Topic of HW3

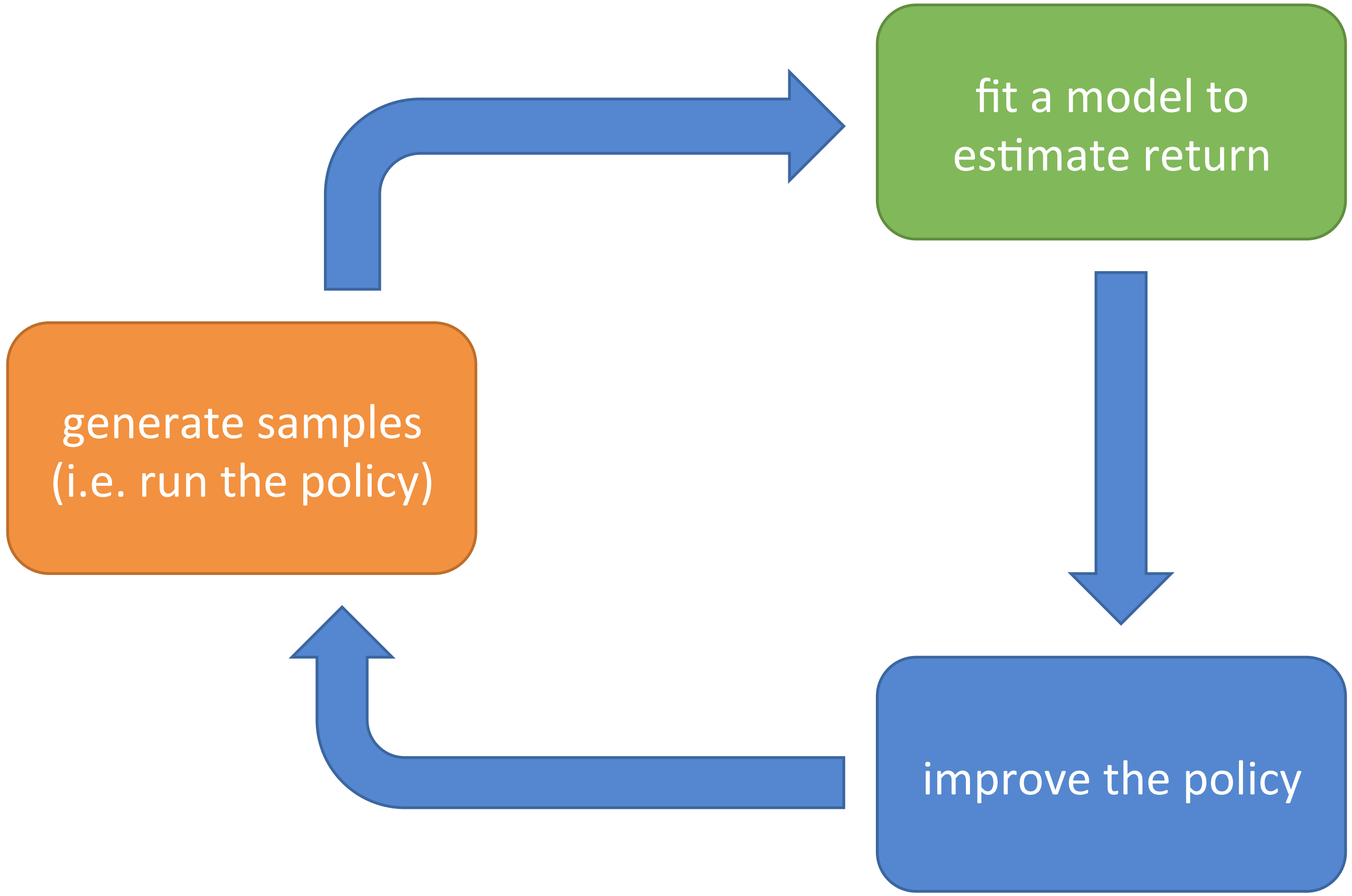
Model-based RL
and how it can be used for multi-task learning

Model-based RL with image observations
or other high-dimensional inputs

Model-based Reinforcement Learning

Main idea: learn model of environment

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



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

supervised learning

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

$$\text{use } f_\phi \text{ to plan actions}$$

Note: can also use f_ϕ to optimize a policy.

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

What is a reinforcement learning task?

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

dynamics reward

In many situations:

only r_i varies across tasks.

Today: Focus on special case where: $\mathcal{T}_i \triangleq \{\mathcal{S}, \mathcal{A}, p(\mathbf{s}_1), p(\mathbf{s}'|\mathbf{s},\mathbf{a}), r_i(\mathbf{s},\mathbf{a})\}$

For example:

robotics: physics constant, different task objectives

(do laundry, cook, …)

task-driven dialog: user & language constant, different objectives

(order food, emotional support, …)

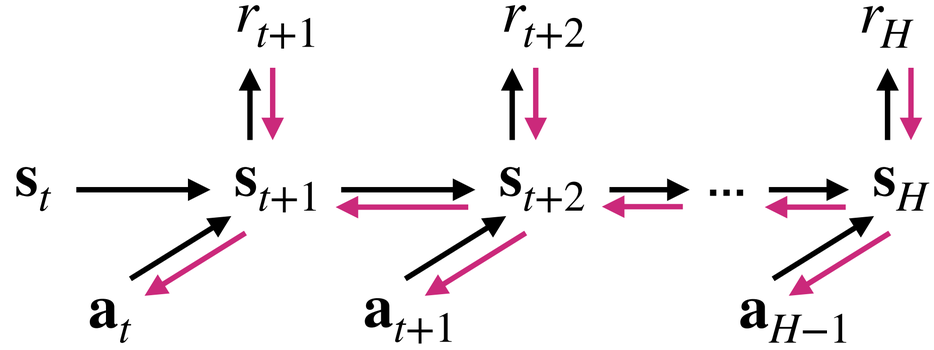
autonomous driving: rules of road constant, user preferences vary

(drive fast, drive smoothly, …)

In these cases, estimating the model $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$ is a single-task problem!

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

"planning"



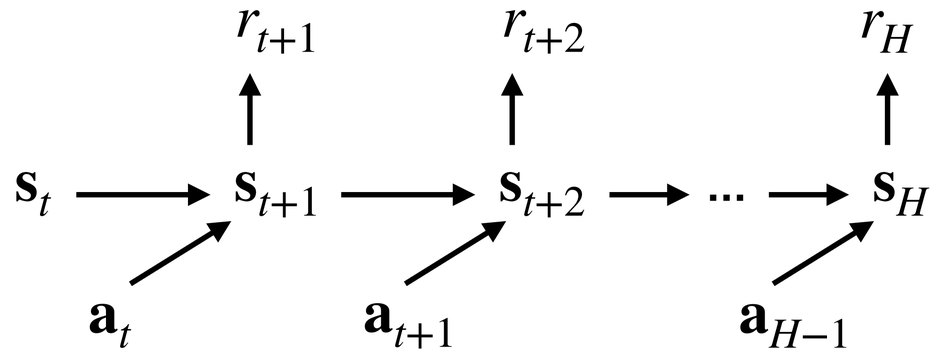
Approach 1a:
via *backpropagation*
(i.e. *gradient-based* optimization)

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. Backpropagate through $f_\phi(\mathbf{s}, \mathbf{a})$ to choose actions

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

"planning"



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

Sampling-Based Optimization

Denote $\mathbf{A} := \mathbf{a}_t, \ldots, \mathbf{a}_{t+H}$

Version 1: guess & check

"random shooting"

a. Sample many $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g. uniform)

b. Choose $\mathbf{A}_i$ based on $\arg\max_i \sum_{t'=t}^{t+H} r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$

Can we improve this distribution?

Version 2: cross-entropy method

a. Sample many $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from $p(\mathbf{A})$

b. Evaluate $J(\mathbf{A}_i) = \sum_{t'=t}^{t+H} r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$

c. Pick the elites $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$ with the largest $J(\mathbf{A}_i)$, where $M < N$

d. Refit $p(\mathbf{A})$ to the elites $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$

Sampling-Based Optimization

Version 1: guess & check

Version 2: cross-entropy method

"random shooting"

Pros:

+ fast, if parallelized

+ simple

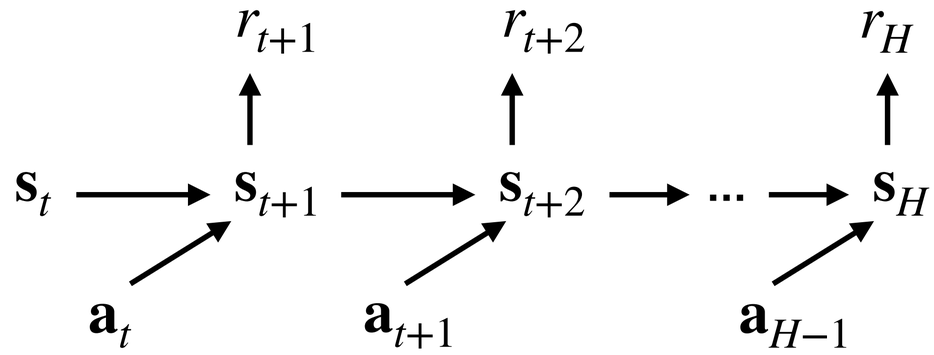
Cons:

- doesn't scale to high-dimensions

(Including both H and $|\mathbf{a}|$)

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

"planning"



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

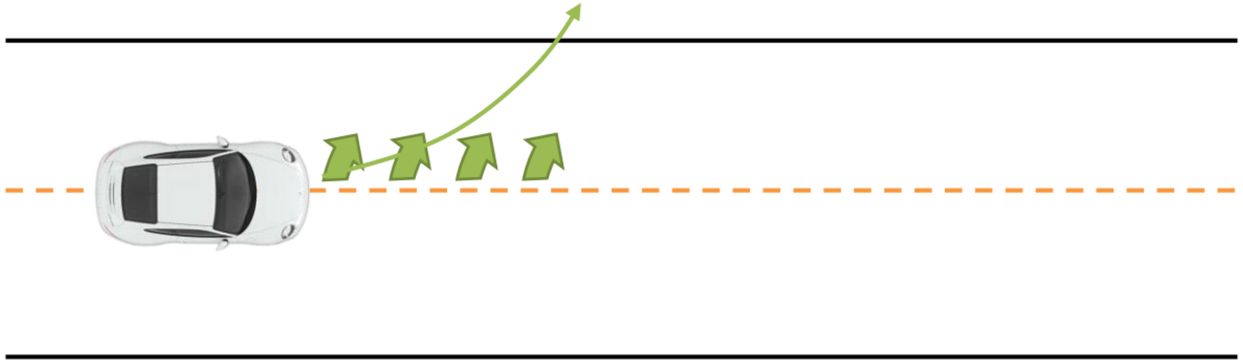
Algorithm:

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

(e.g. with random shooting or cross-entropy method)

How can this approach fail?

Action optimization will exploit imprecisions in model



Thought Exercise: How might you alleviate this issue?

One option: Refitting model using new data.

How can this approach fail?

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

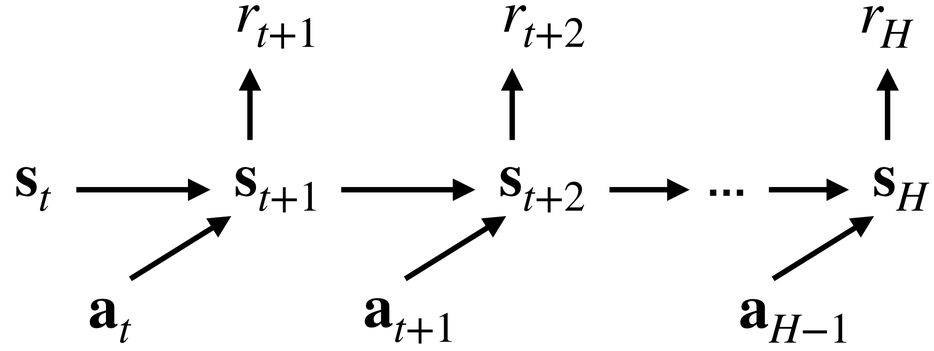
Data distribution mismatch

$$p_{\pi_0}(\mathbf{s}) \neq p_{\pi_f}(\mathbf{s})$$

Going right means that we can go higher!

Thought Exercise: How might you alleviate this issue?

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

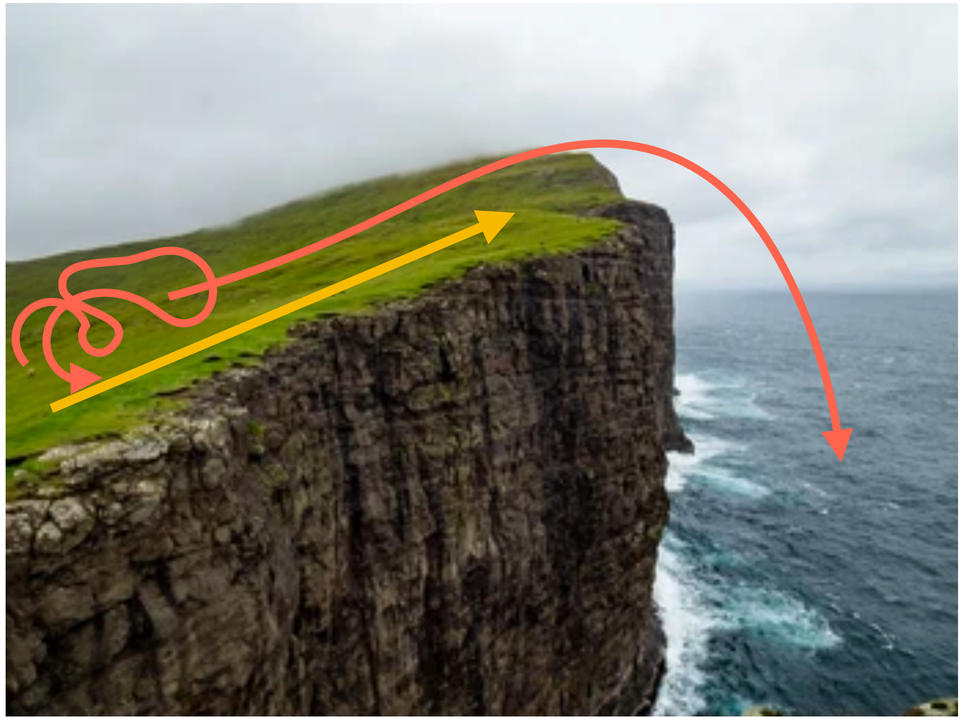


Approach 1b:
via *sampling*
(i.e. *gradient-free* optimization)

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 $\mathcal{D}$

Revisiting the cliff



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 $\mathcal{D}$

Going right means that we can go higher!

Final policy: go to the top and stop.

Can we do better?

open-loop vs. closed-loop planning

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

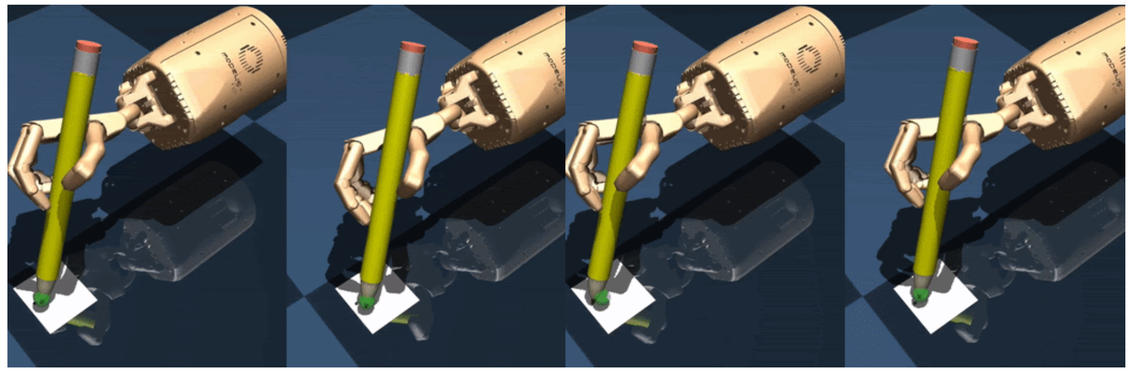
What does this have to do with multi-task RL?

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

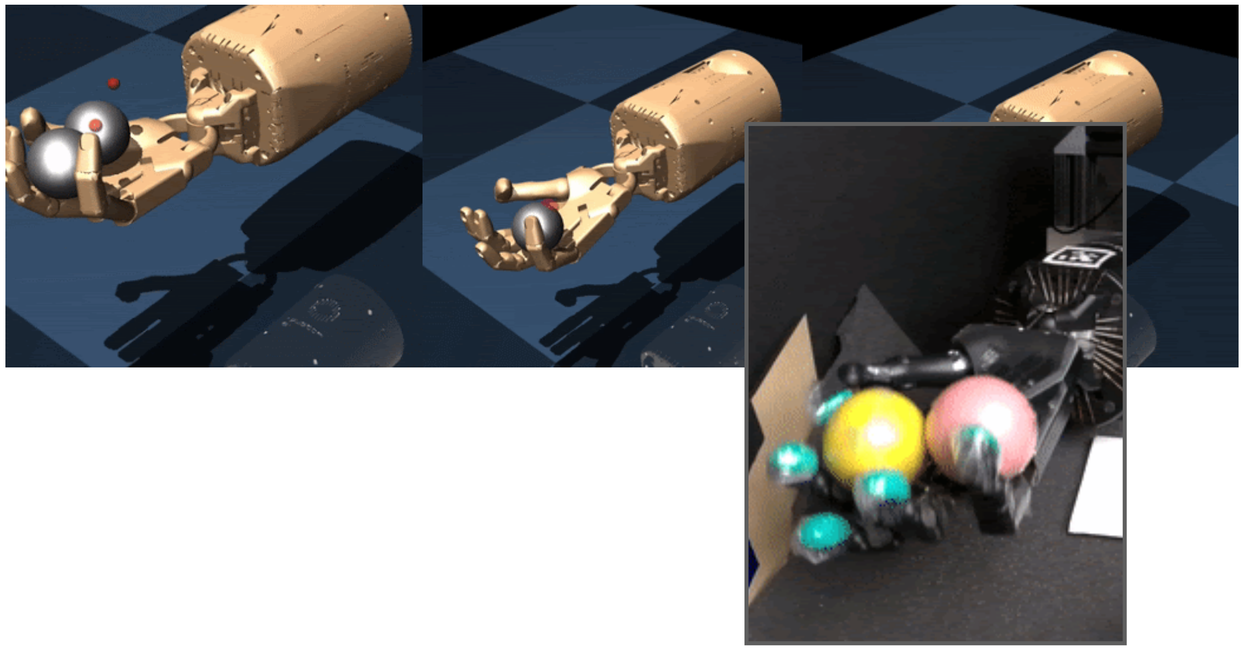
*__Caveat__: reward will change how you collect data.

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?

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

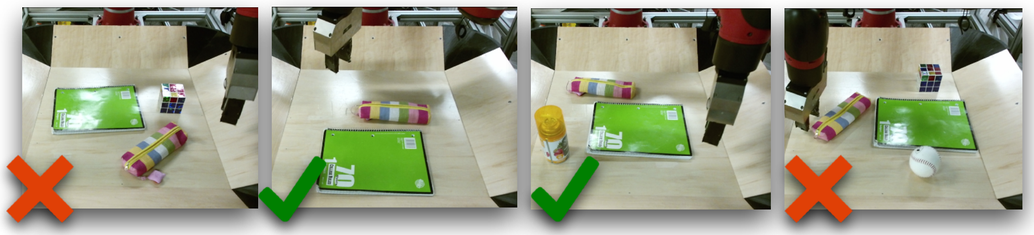
If *no*: **multi-task RL**: learn $r_\theta(\mathbf{s}, \mathbf{a}, \mathbf{z}_i)$, use it to plan

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

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

$\mathcal{D}_i^{\text{tr}}$: 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

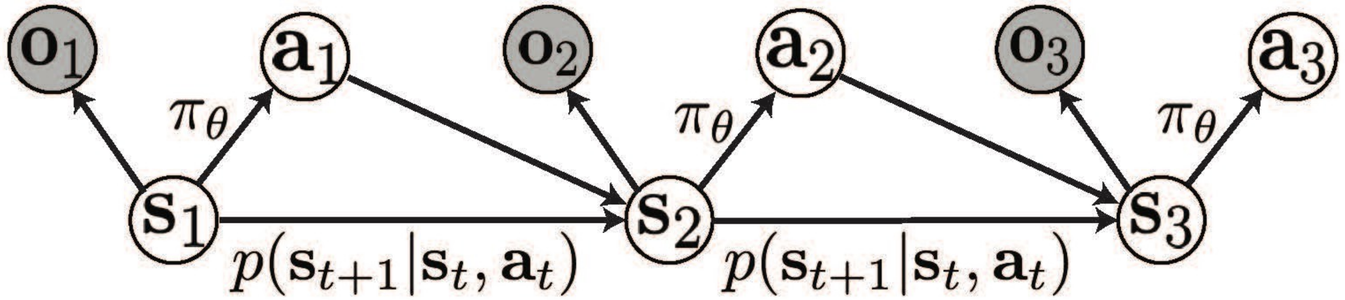
The Plan

Recap
goal-conditioned RL & relabeling

Model-based RL
and how it can be used for multi-task learning

Model-based RL with image observations
or other high-dimensional inputs

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

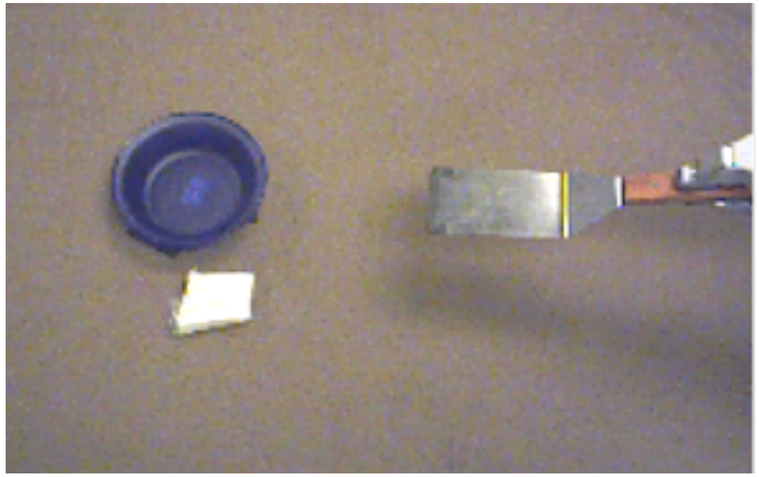


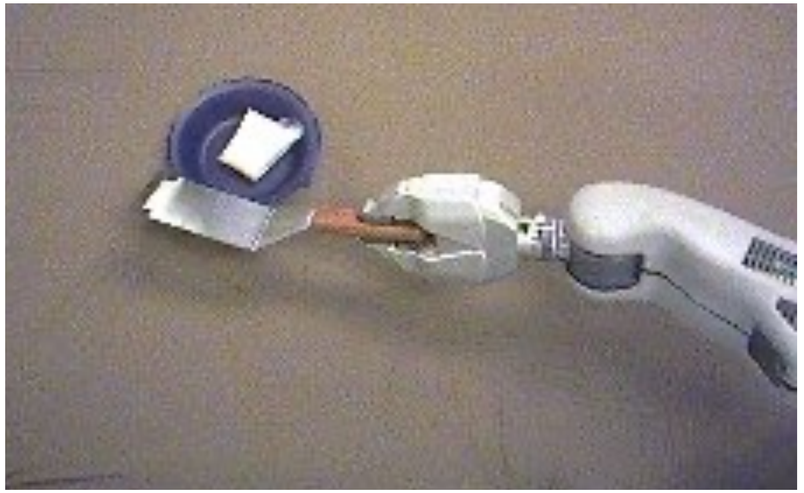
also: no reward signal with only observations

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

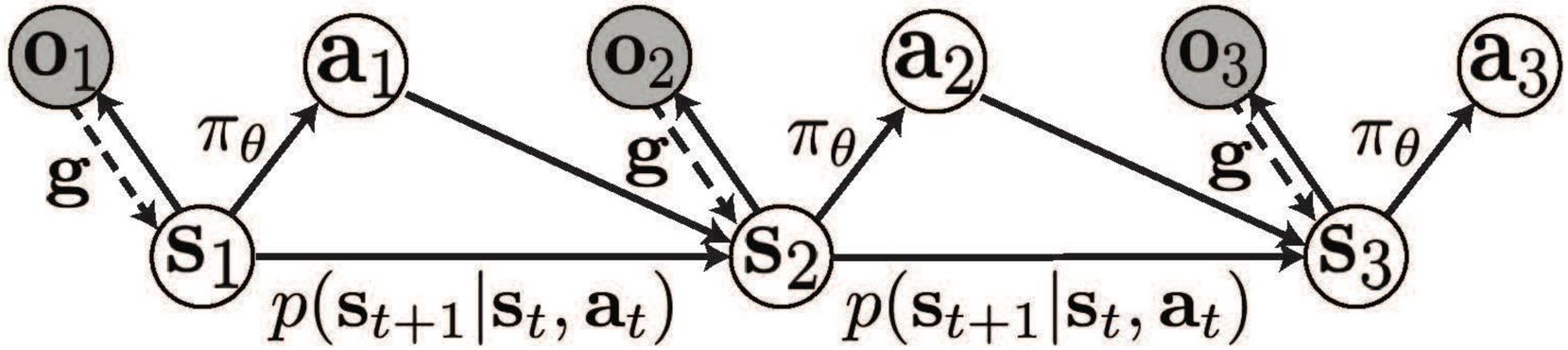
1. Learn model in latent space
2. Learn model of observations (e.g. video)
3. Predict alternative quantities

Model-Based RL 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



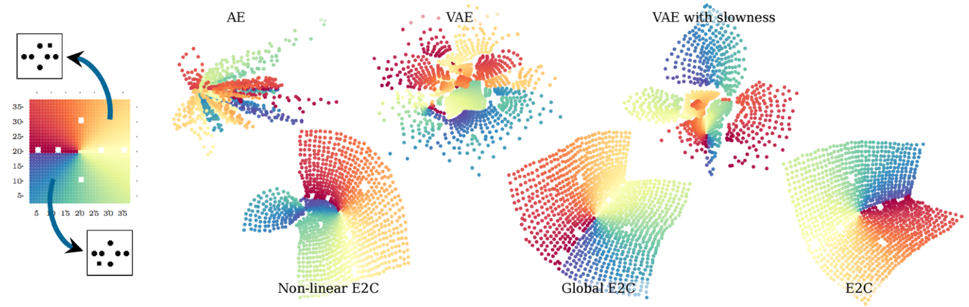
Learning in Latent Space

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



NeurIPS 2015

Deep Spatial Autoencoders for Visuomotor Learning

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

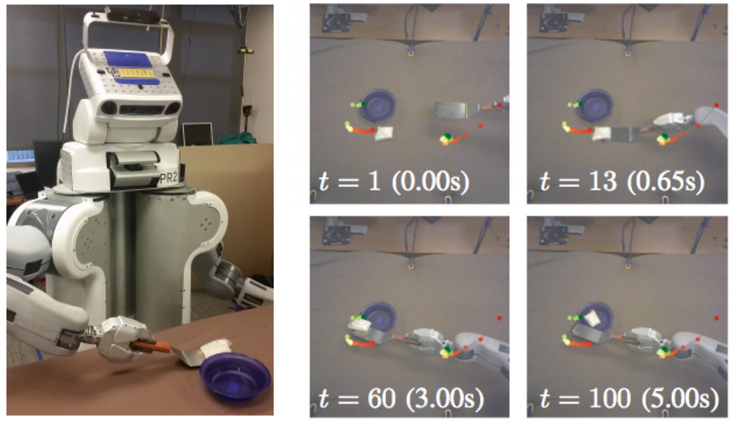


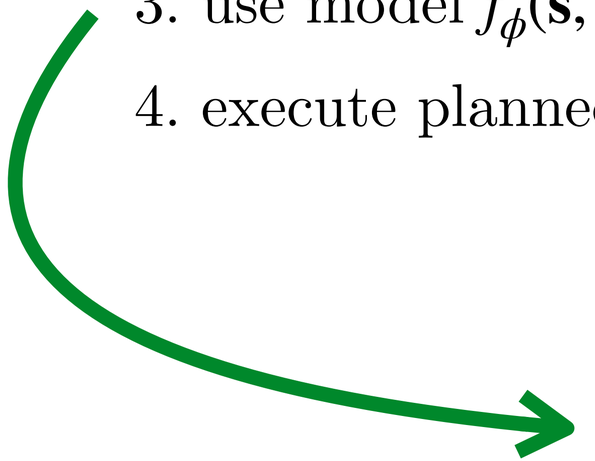
Fig. 1: PR2 learning to scoop a bag of rice into a bowl with a spatula (left) using a learned visual state representation (right).

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

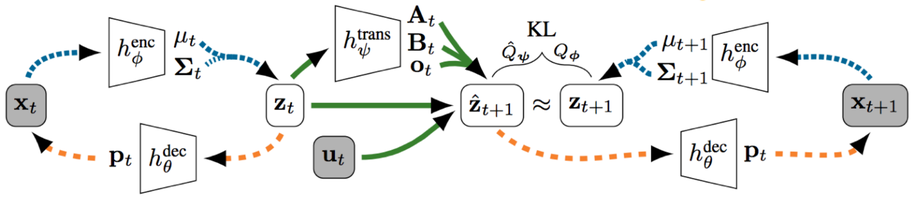
Assumption: distance in latent space is an accurate metric.

Learning in Latent Space

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

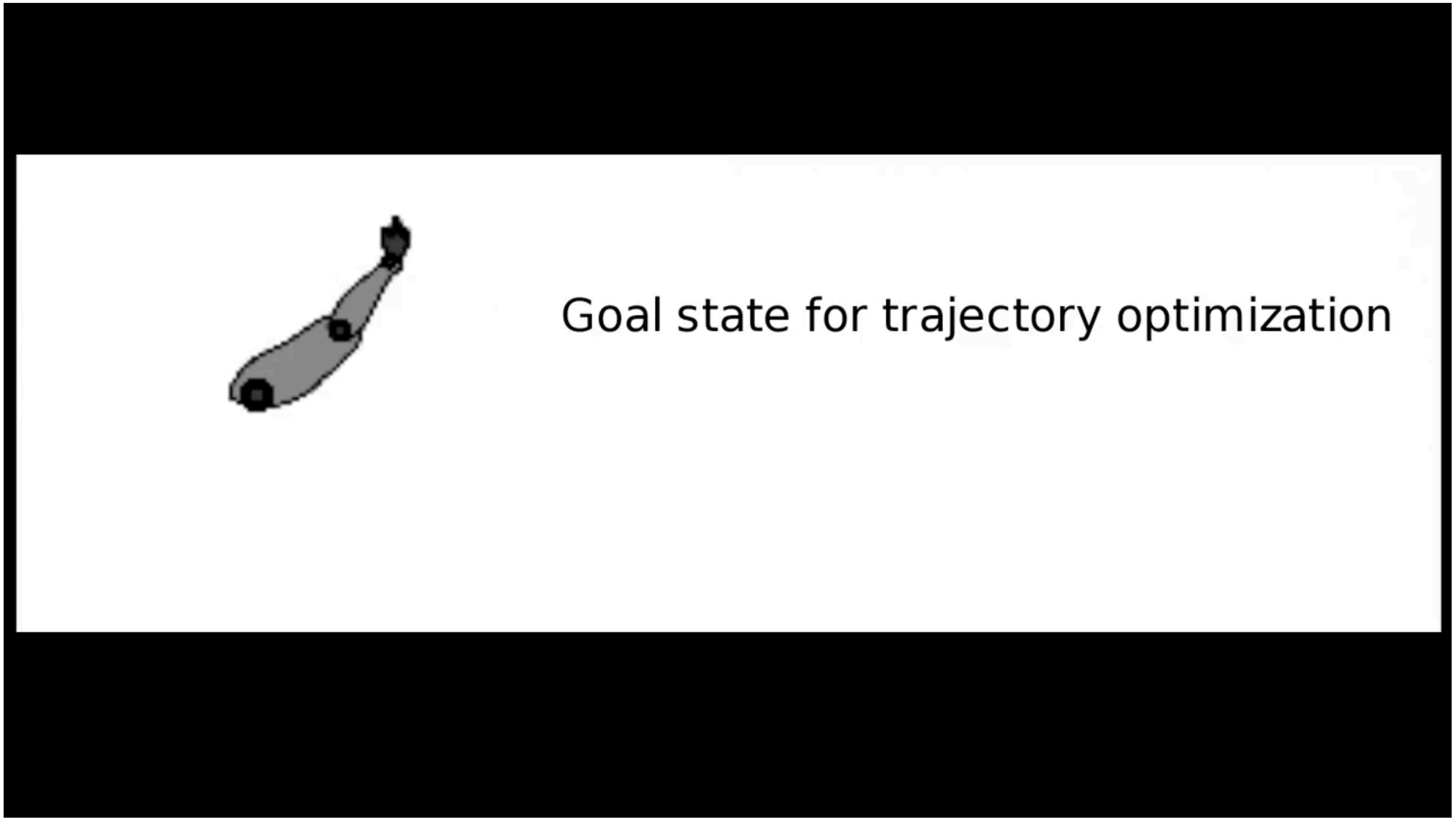
How to optimize latent embedding?

Watter et al.'15



learn embedding & model **jointly**

Learning in Latent Space



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

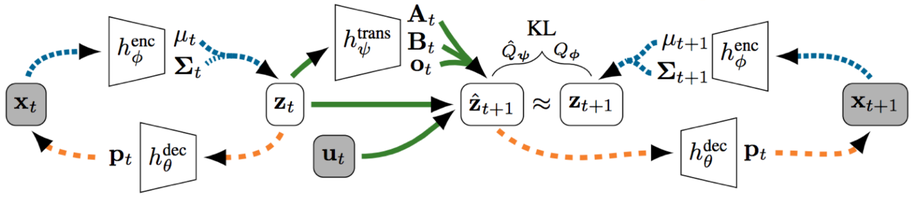
Watter et al. NeurIPS ‘15

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 $\mathcal{D}$

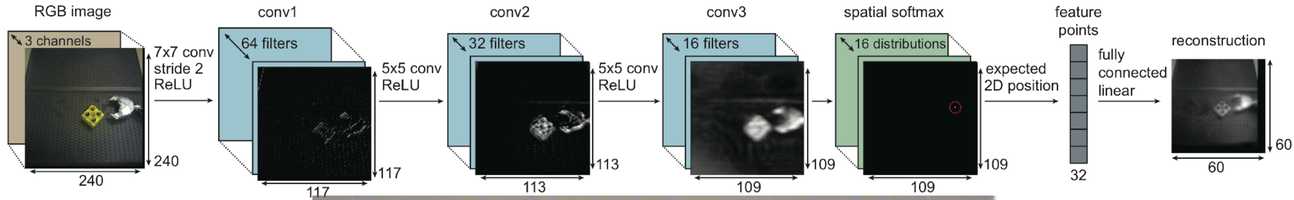
How to optimize latent embedding?

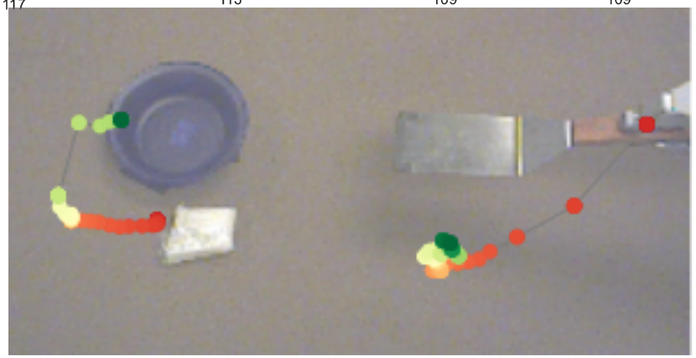
Watter et al.'15



learn embedding & model **jointly**

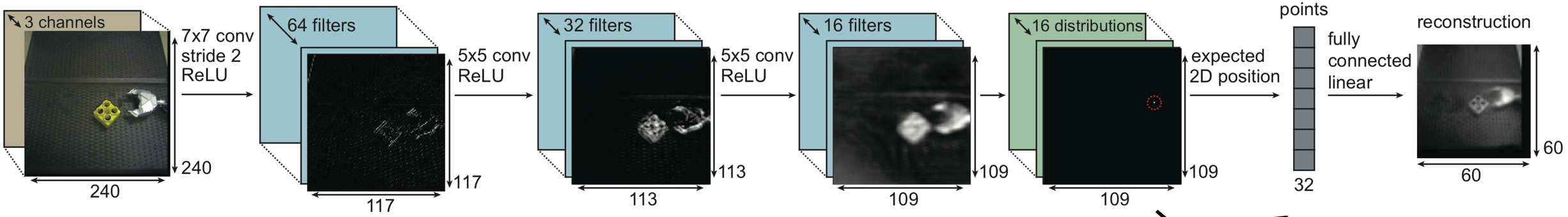
Finn et al.'16





embedding is **smooth** and **structured**

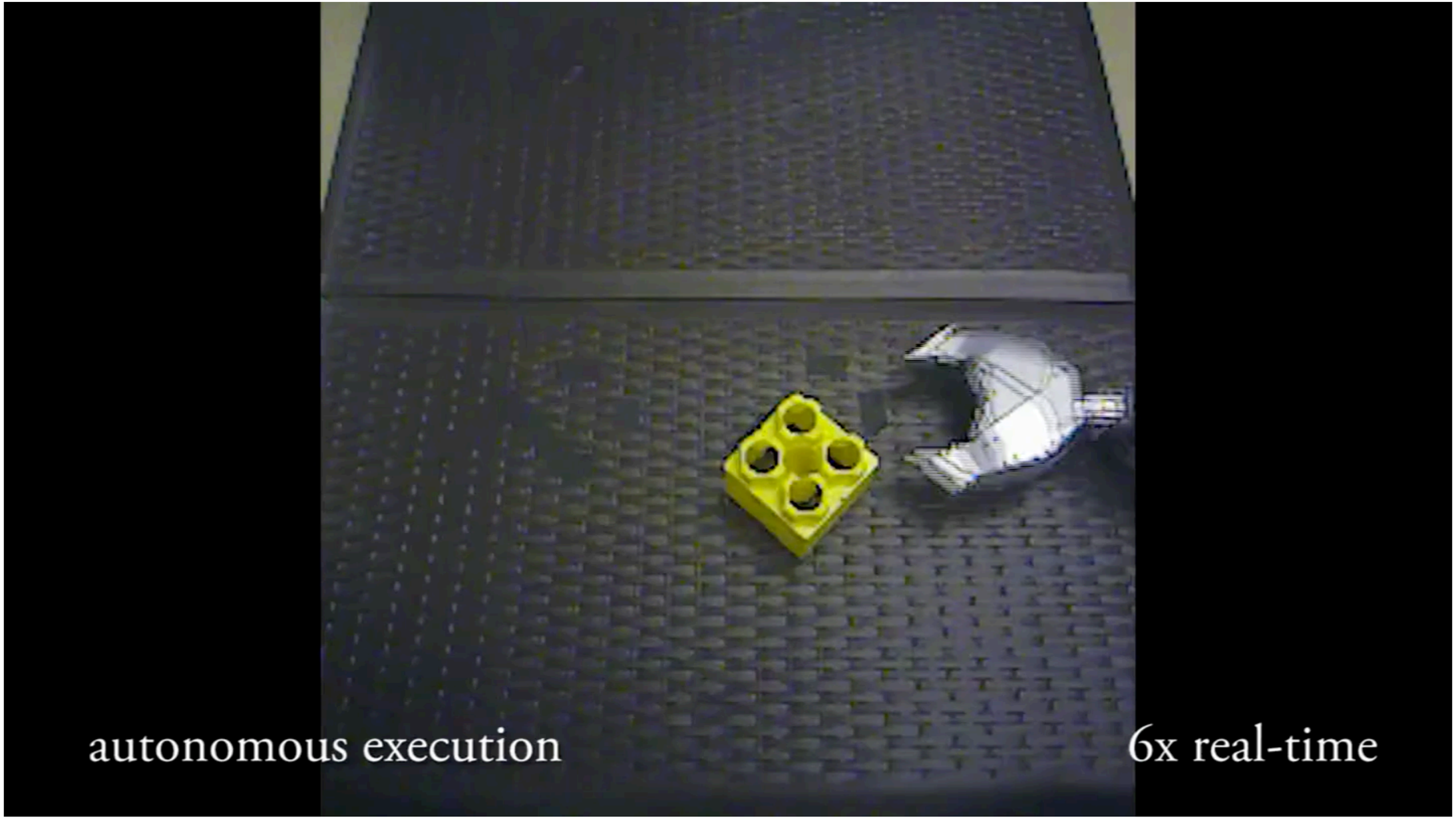
$$\mathbf{s}_{ij} = \frac{e^{\mathbf{z}_{ij}}}{\sum_{i',j'} e^{\mathbf{z}_{i'j'}}}$$



$$\left(\sum_{i,j} i\mathbf{s}_{ij}, \sum_{i,j} j\mathbf{s}_{ij}\right)$$

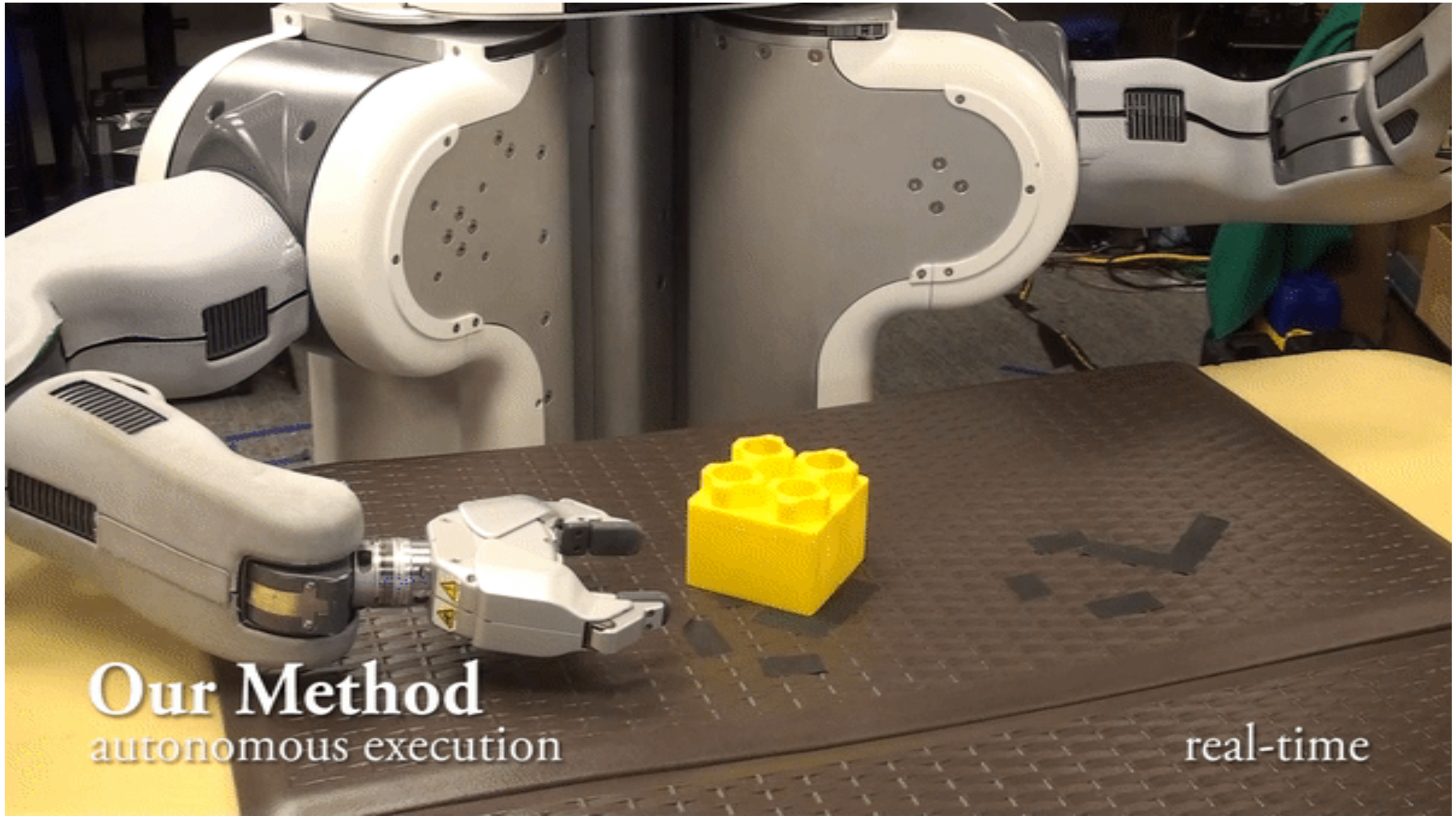
embedding is **structured** and **smooth**

Learning in Latent Space



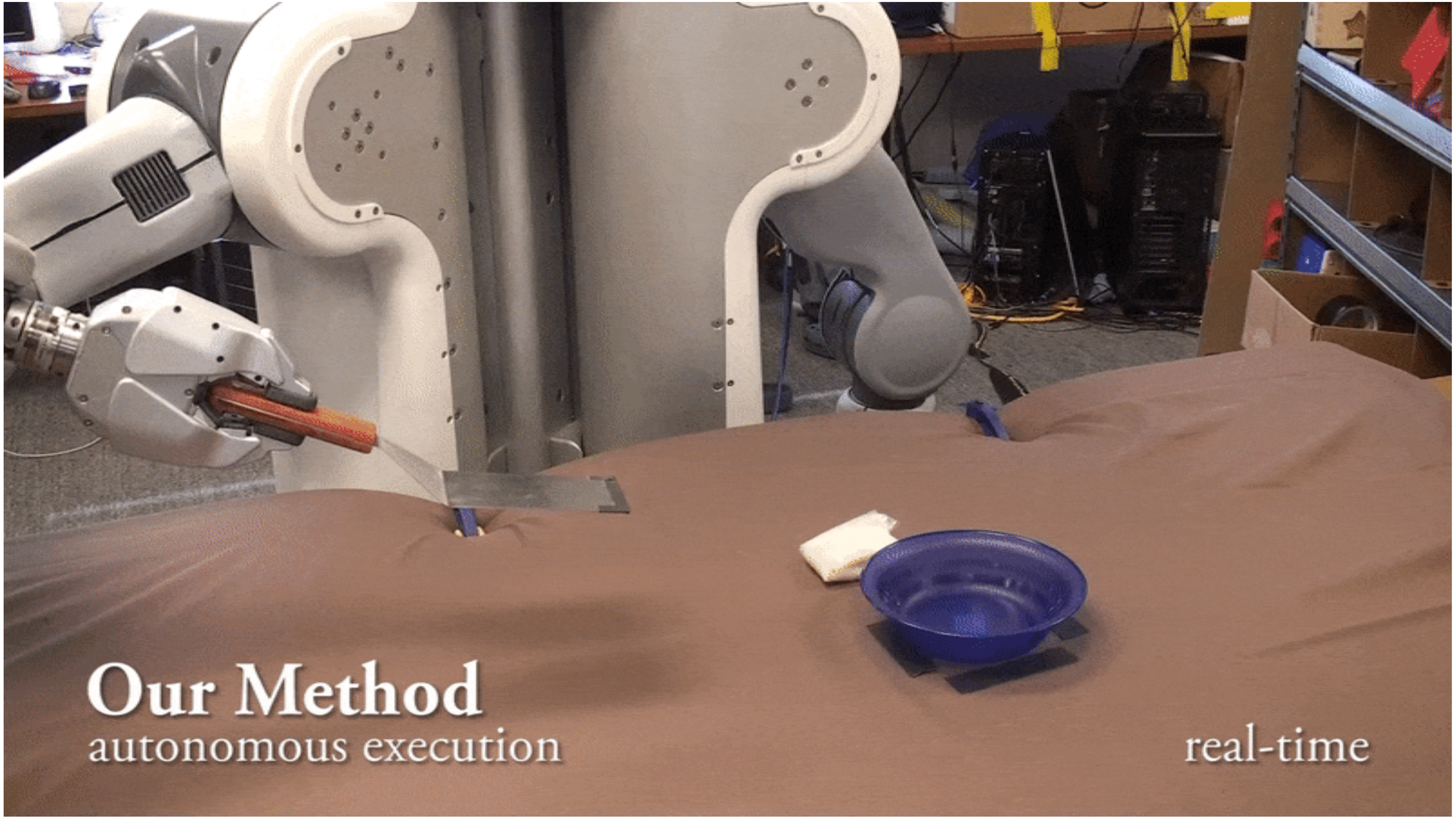
Finn et al. ICRA ‘16

Learning in Latent Space



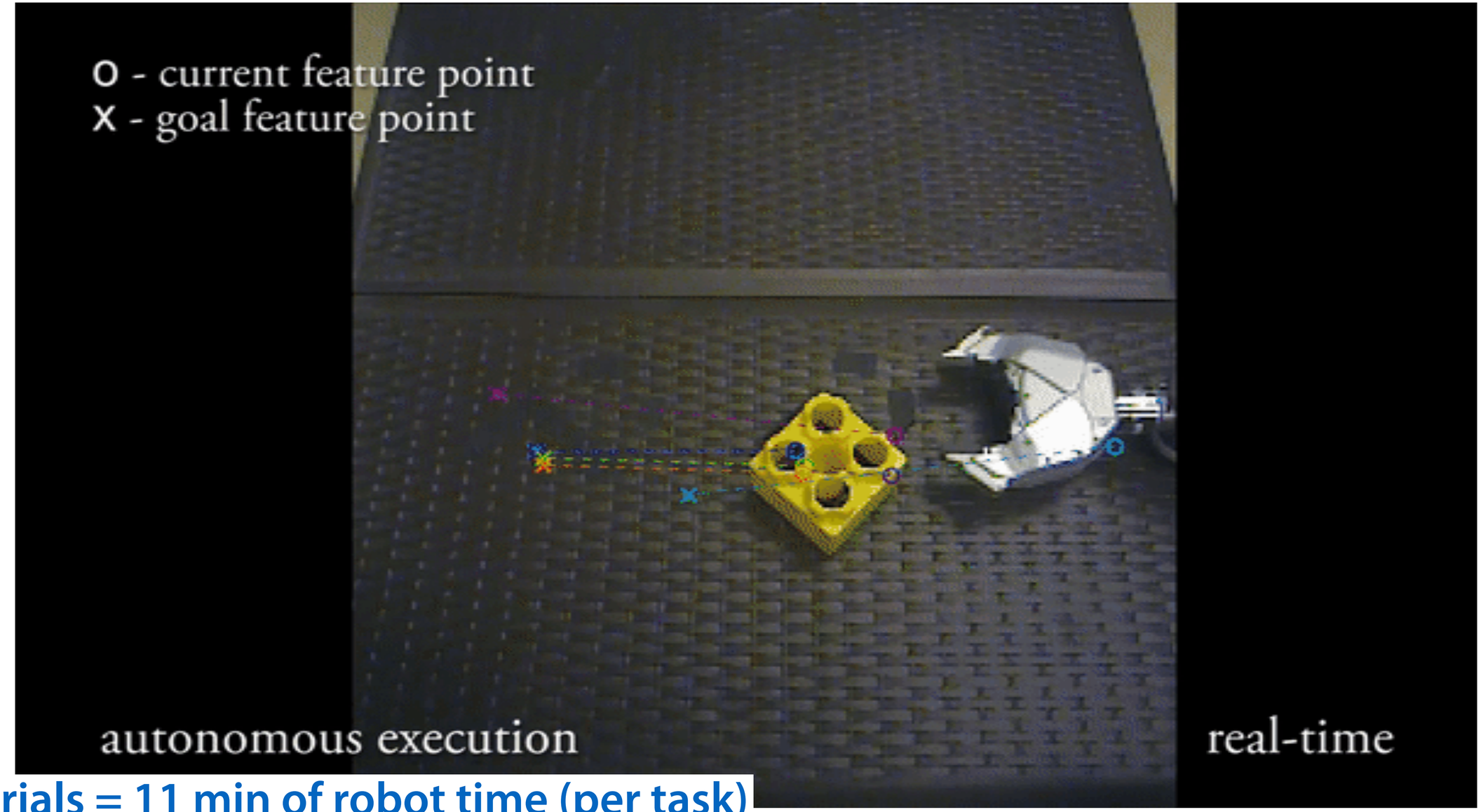
Finn et al. ICRA ‘16

Learning in Latent Space



Finn et al. ICRA ‘16

Learning in Latent Space



125 trials = 11 min of robot time (per task)

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

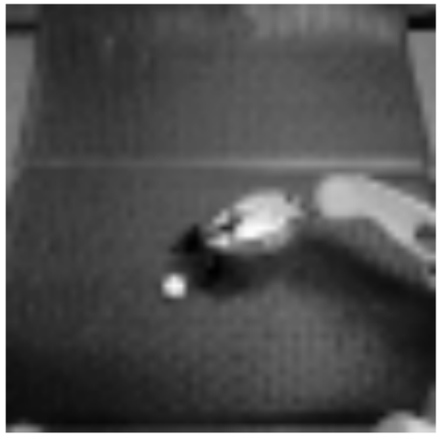
Pros:

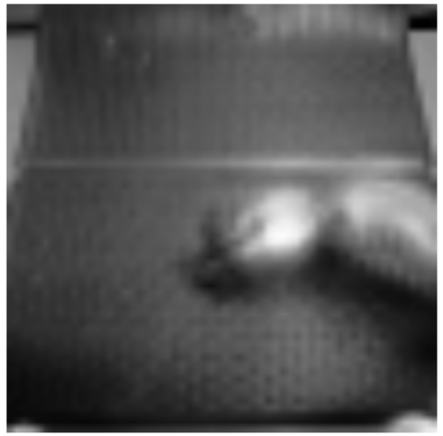
+ Learn complex visual skills very efficiently

+ Structured representation enables effective learning

Cons:

- Reconstruction objectives might not recover the right representation

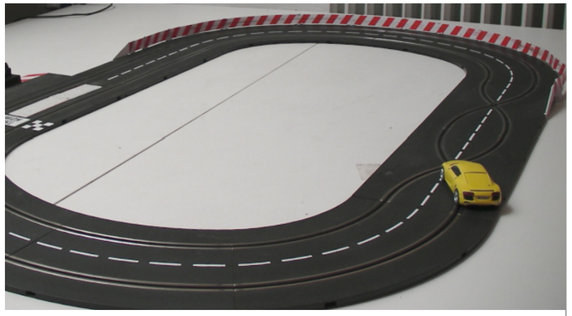




need better unsupervised representation learning methods

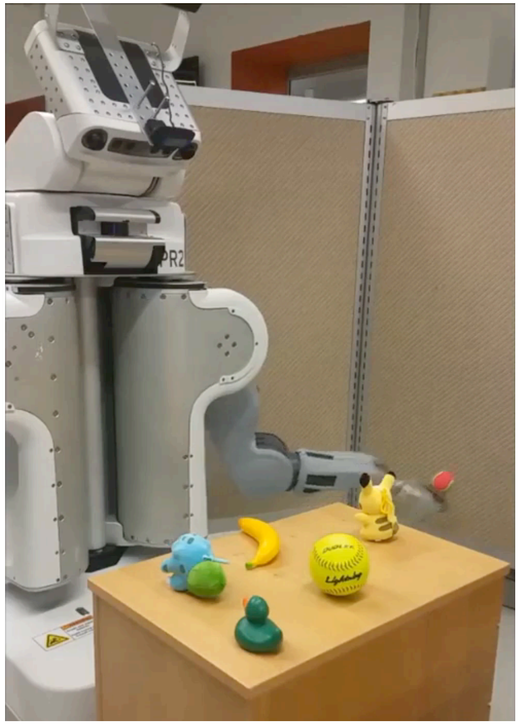
Aside: Low-dimensional embedding can also be useful for model-free approaches

model-free RL in latent space



FQI in latent space

Lange et al. '12

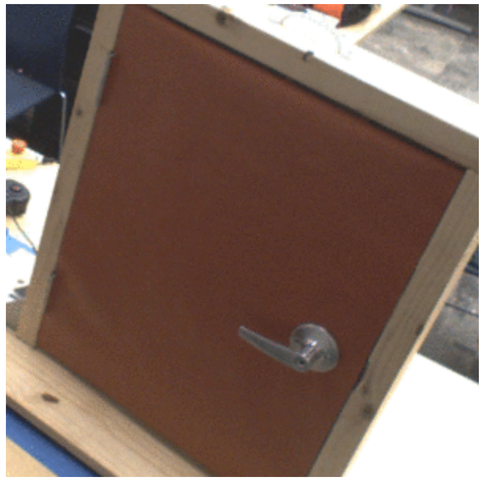


TRPO in latent space

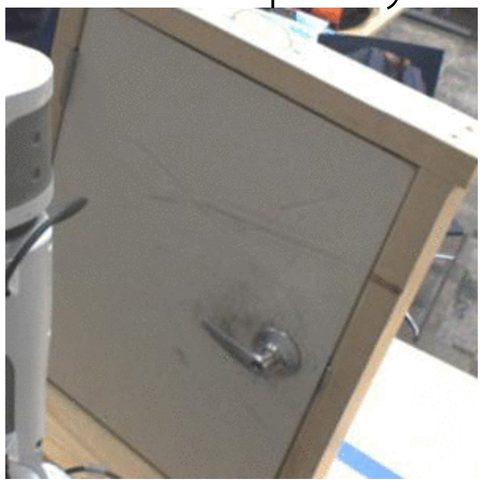
Ghadirzadeh et al. '17

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)

Learning with Image Observations

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

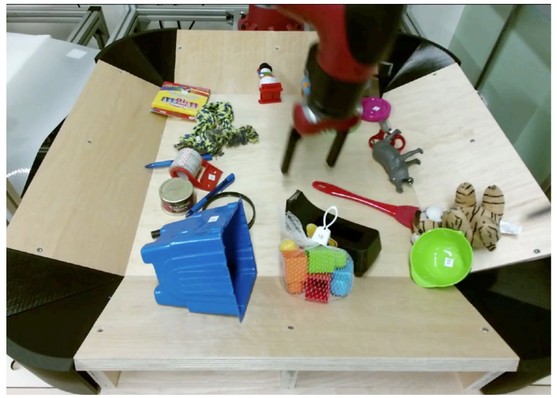
Modeling directly in observation space

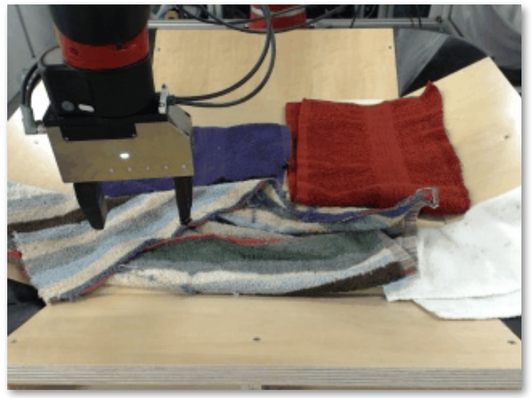
Recall MPC

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 $\mathcal{D}$

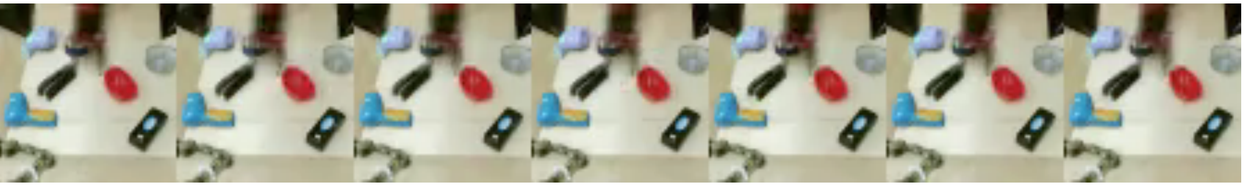
What's new? Model operates directly on $\mathbf{o}_t$

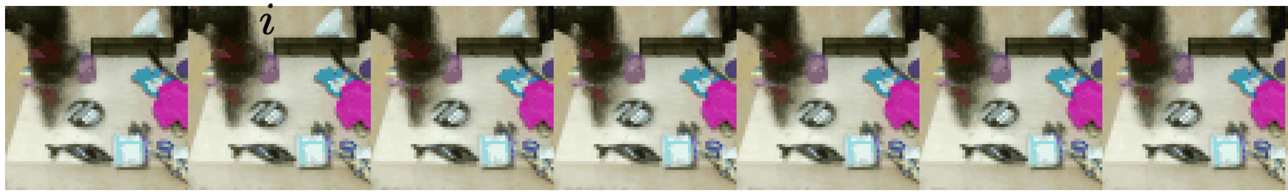
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{o}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$

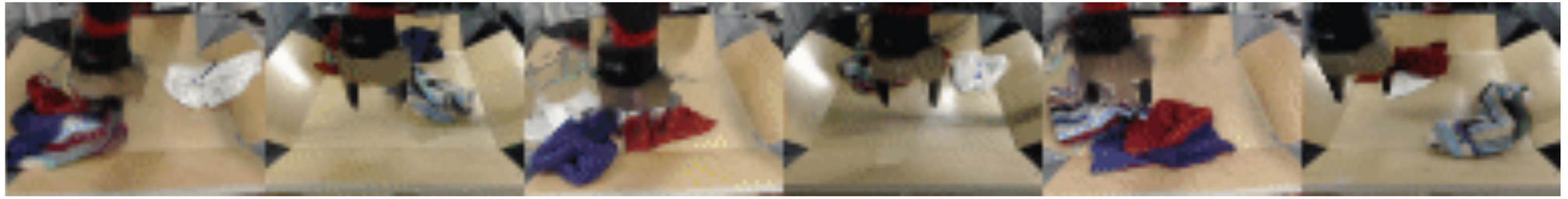




2. learn model $f_\phi(\mathbf{o}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{o}_i, \mathbf{a}_i) - \mathbf{o}'_i||^2$





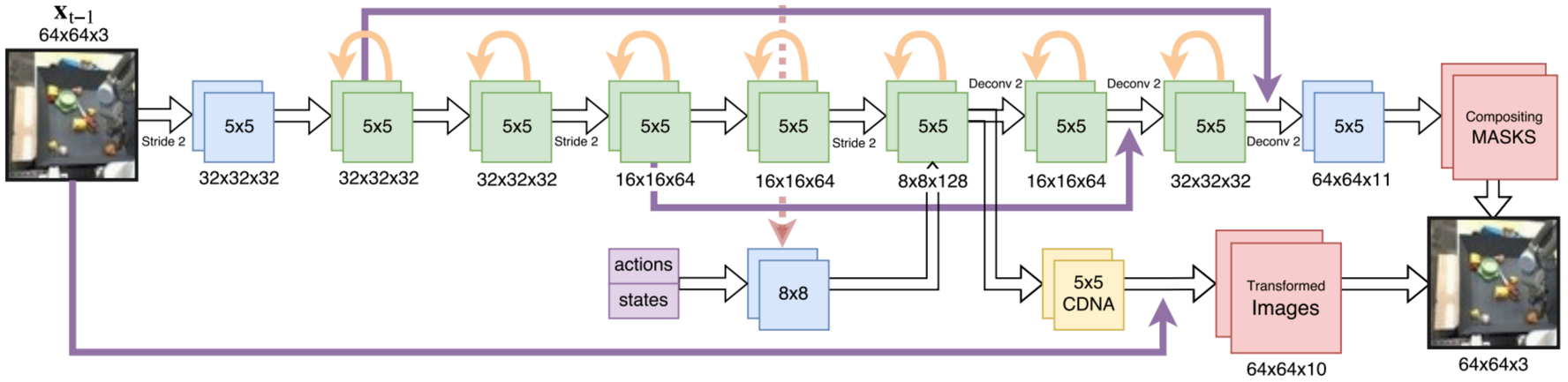


3. use model $f_\phi(\mathbf{o}, \mathbf{a})$ to optimize action sequence

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

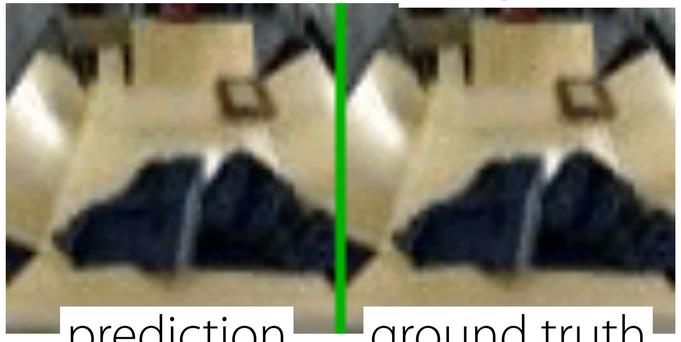
How to predict video?

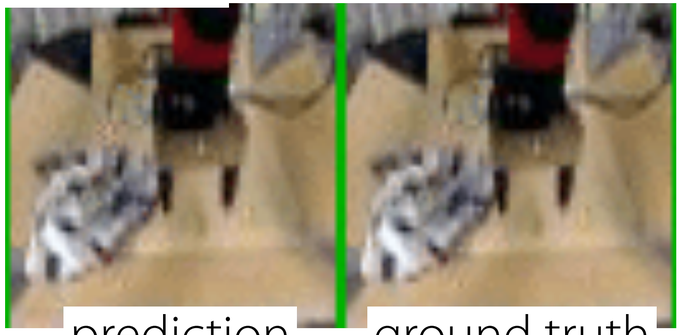
2. learn model $f_\phi(\mathbf{o}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{o}_i, \mathbf{a}_i) - \mathbf{o}'_i||^2$



- deep recurrent network
- multi-frame prediction
- action-conditioned
- often *stochastic*
(to capture uncertainty)

Villegas et al. NeurIPS '19


prediction ground truth


prediction ground truth

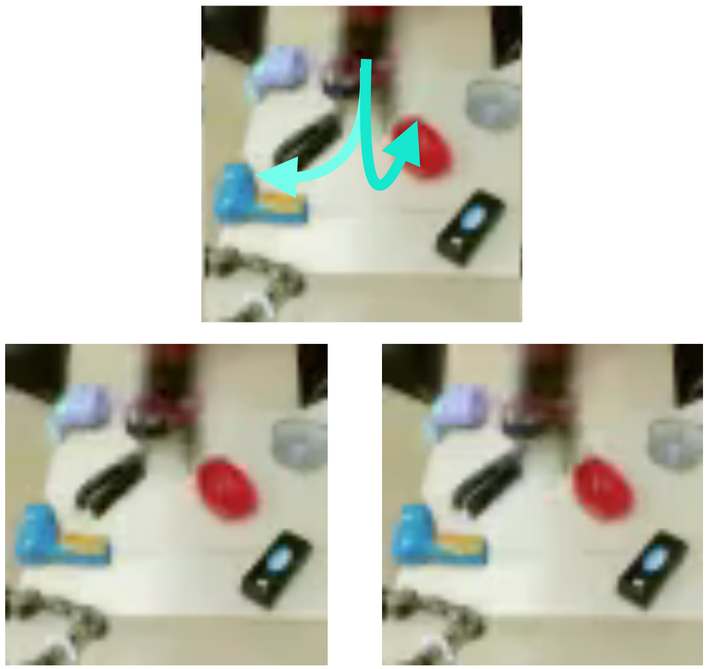
How to plan?

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

Sampling-based optimization:

1. Sample 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

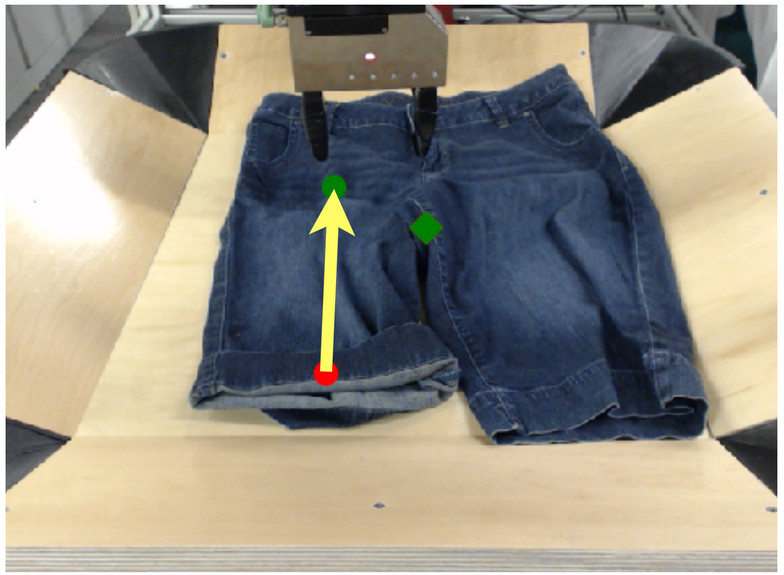
visual "model-predictive control" (MPC)



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

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

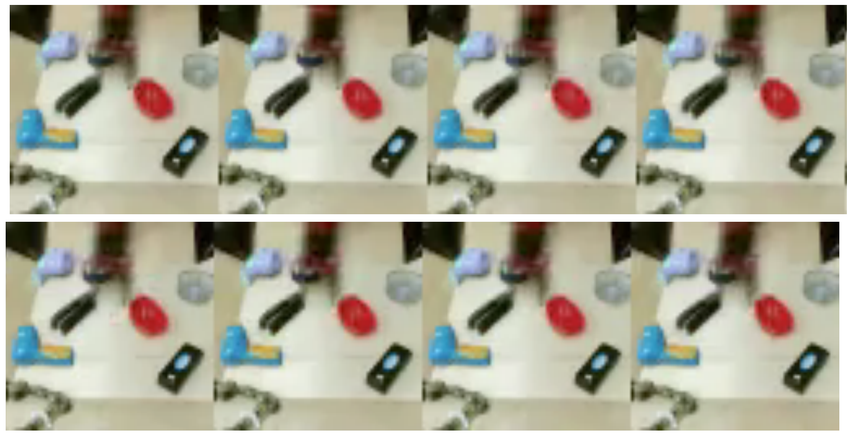
Modeling directly in observation space

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



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:

Will I successfully grasp?

Will I collide?

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 directly observe them

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!

Plan for Today

Recap
goal-conditioned RL & relabeling <- Topic of HW3

Model-based RL
and how it can be used for multi-task learning

Model-based RL with image observations
or other high-dimensional inputs

Lecture goals:

- Understand how to use & implement **model-based RL**
- Understand how **model-based RL** can be used for **multi-task** RL
- Challenges and strategies for **model-based RL** in **high-dimensional spaces**

Next time

So far: multi-task RL

Next: meta RL

Reminders

Midquarter survey is out.

Homework 3 out, due **Wednesday 10/27**