

Model-Based Reinforcement Learning for Multi-Task Learning

CS 330

Logistics

Homework 3 out, due **Monday 10/26**.

Project milestone due **Monday 11/2**.

Plan for Today

Recap

goal-conditioned RL & relabeling

<- Topic of HW3

Model-based RL

and how it can be used for multi-task learning

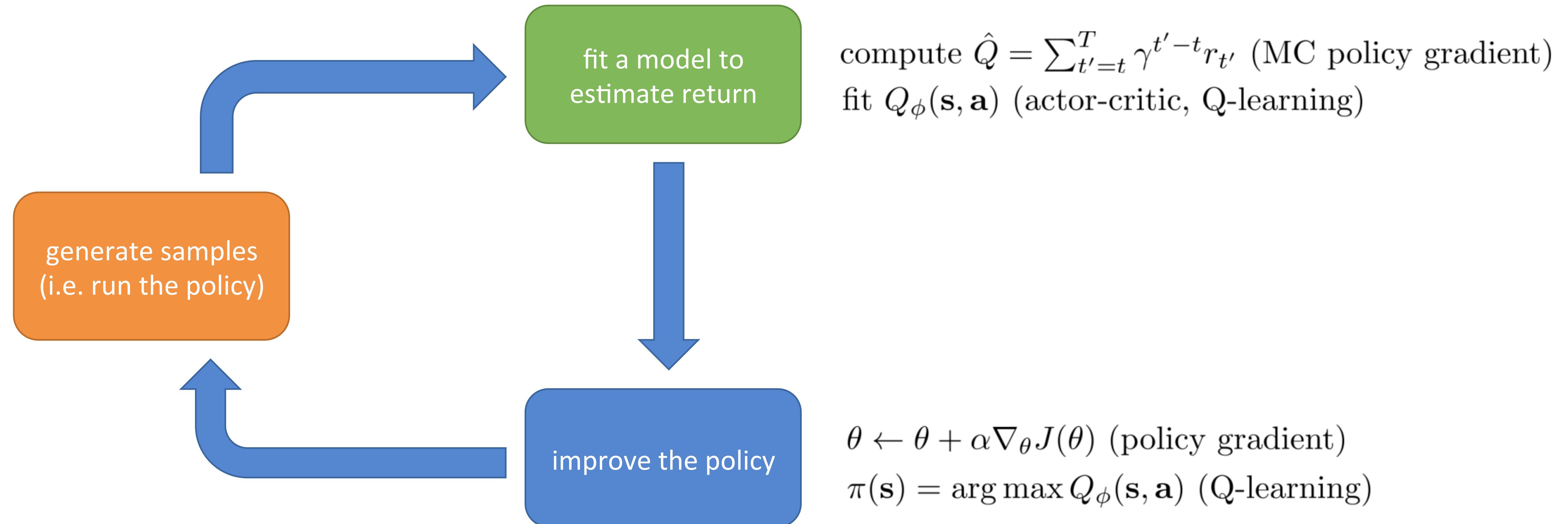
Model-based RL with image observations
or other high-dimensional inputs

- 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**

Lecture goals:

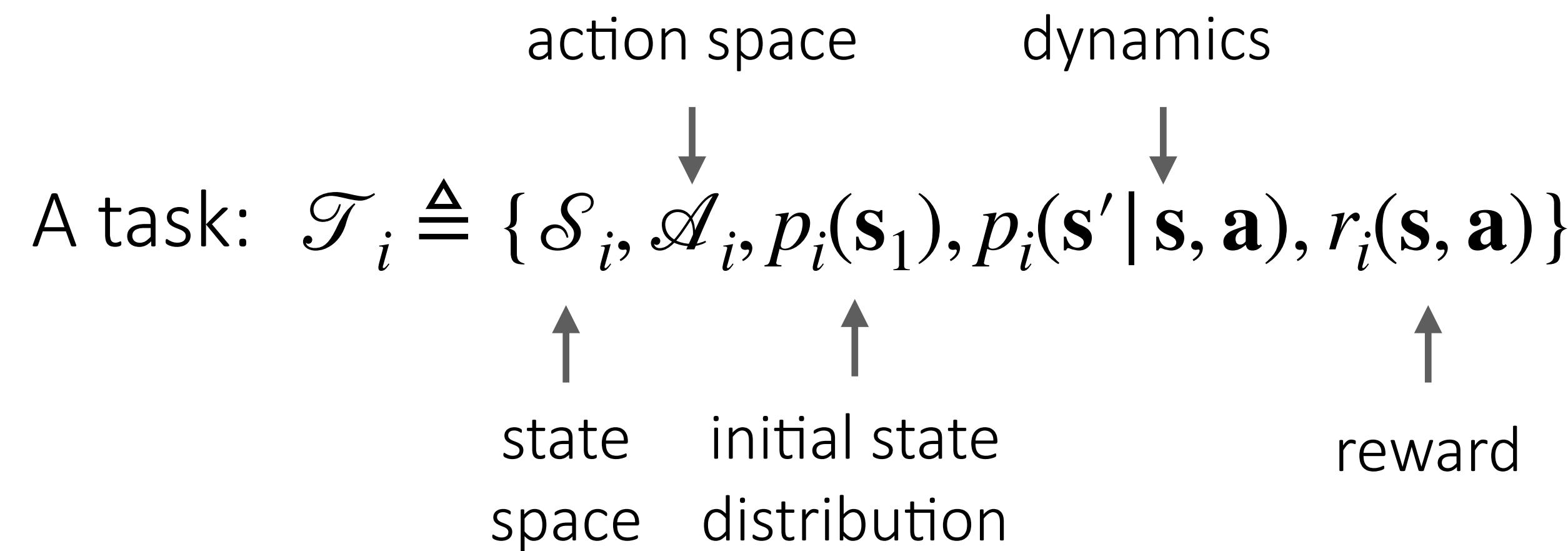
Recap: The anatomy of a reinforcement learning algorithm

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



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

Recap: What is a reinforcement learning task?



Alternatively: A task identifier is part of the state: $s = (\bar{s}, z_i)$

Policy: $\pi_\theta(a | \bar{s}) \rightarrow \pi_\theta(a | \bar{s}, z_i)$

original state

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

What is z_j ?

one-hot task ID

language description

desired goal state, $\mathbf{z}_i = \mathbf{s}_g$ \leftarrow “goal-conditioned RL”

Recap: Example multi-task RL problem

Task 1: passing



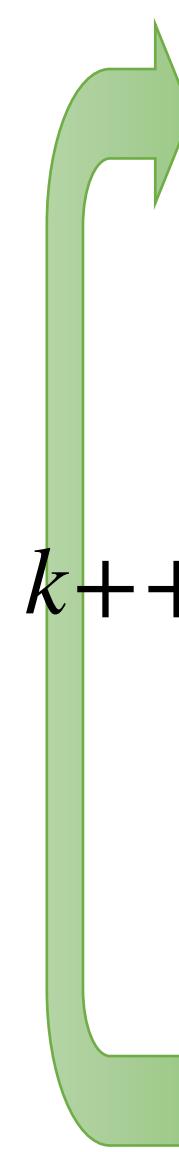
Task 2: shooting goals



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**:

- $k++$
- 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}
- ← 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!

Hindsight relabeling for goal-conditioned RL

Example: simulated robot manipulation

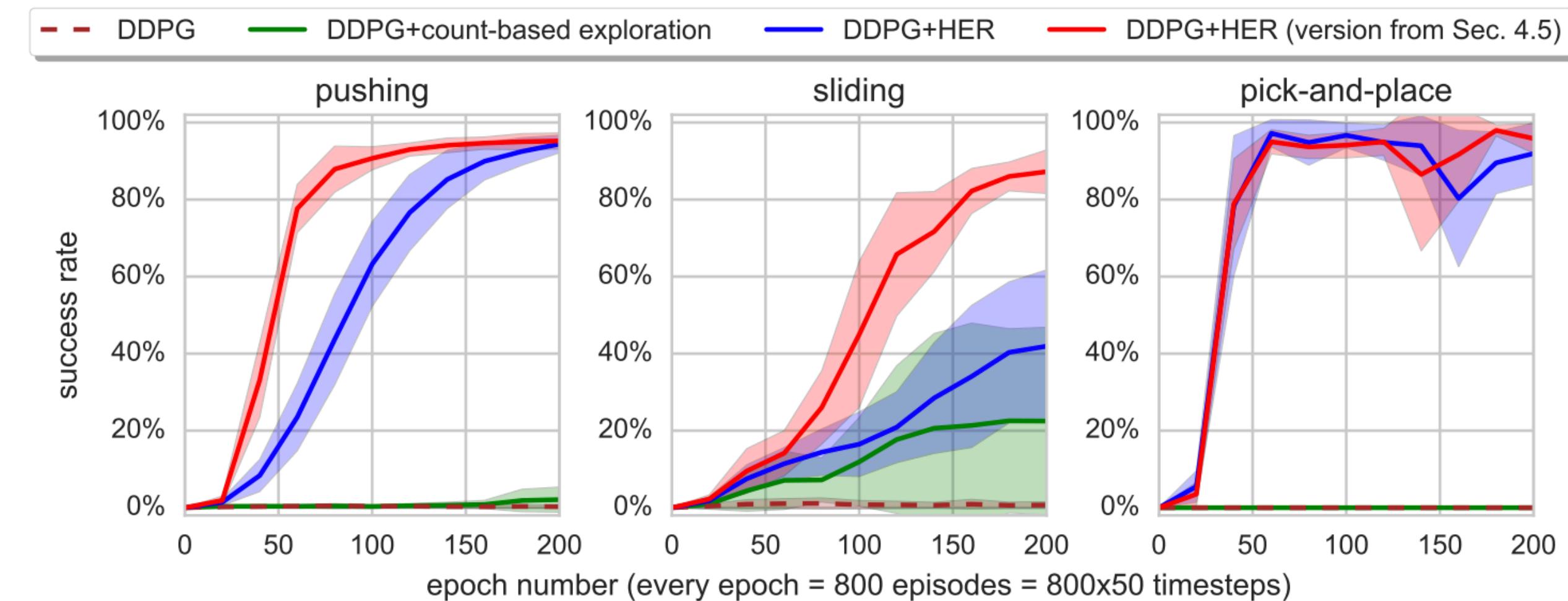
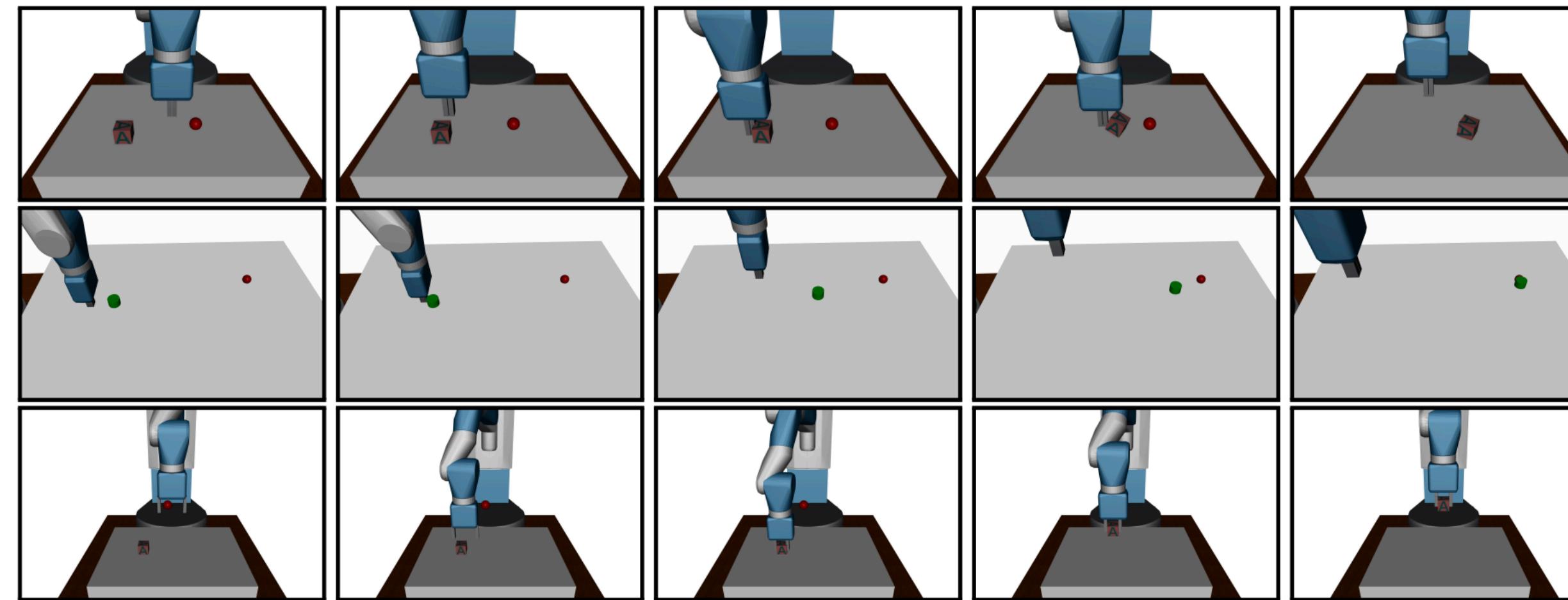


Figure 2: Different tasks: *pushing* (top row), *sliding* (middle row) and *pick-and-place* (bottom row). The red ball denotes the goal position.

The Plan

Recap

goal-conditioned RL & relabeling

<- Topic of HW3

Model-based RL

and how it can be used for multi-task learning

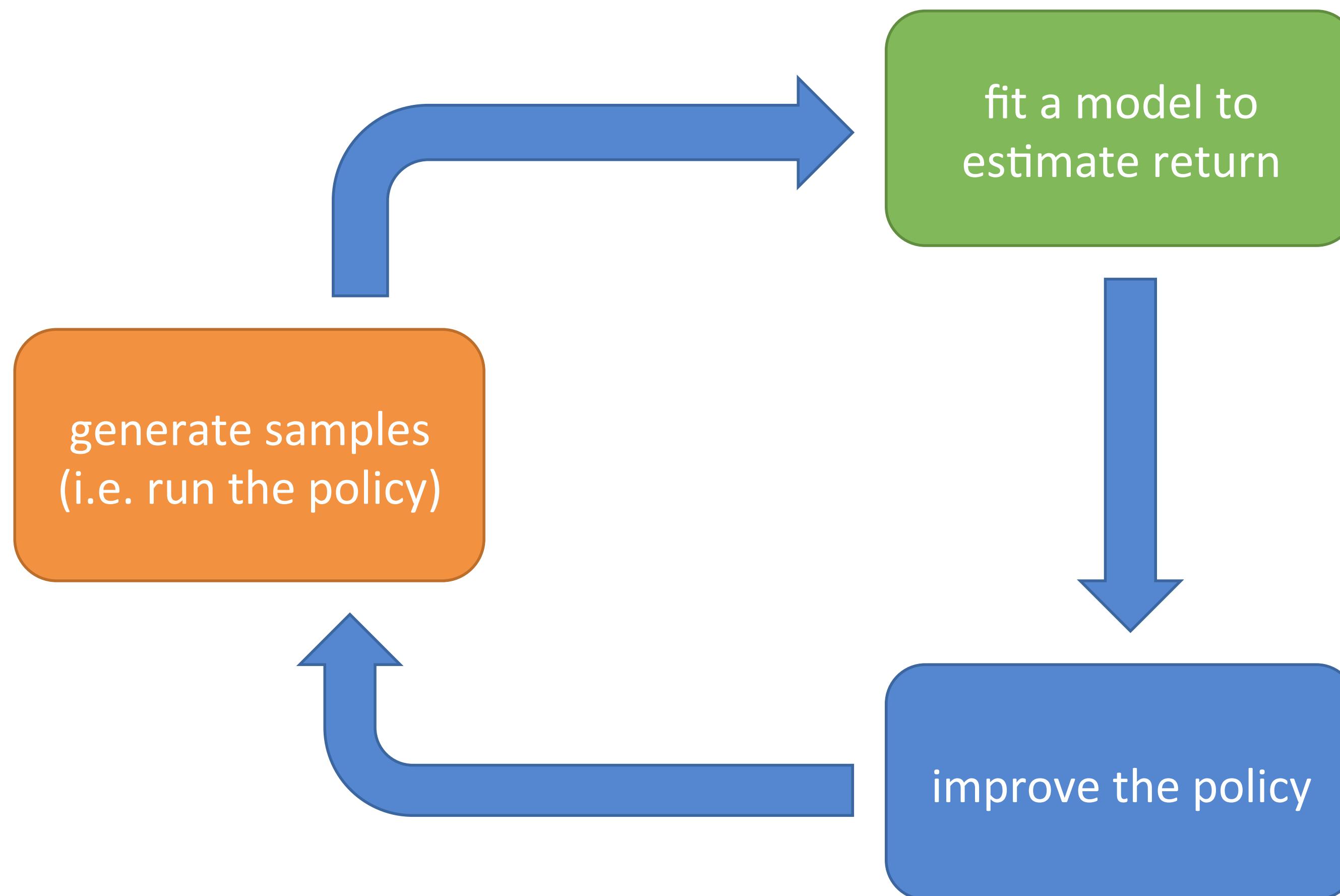
Model-based RL with image observations
or other high-dimensional inputs

Model-based Reinforcement Learning

Main idea: learn model of environment

Why?

- often leads to better efficiency
- model can be reused



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?

What is a reinforcement learning **task**?

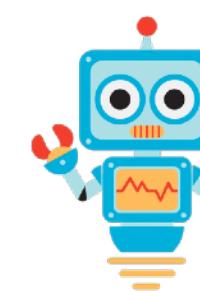
General case: A task: $\mathcal{T}_i \triangleq \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p_i(\mathbf{s}'|\mathbf{s}, \mathbf{a}), r_i(\mathbf{s}, \mathbf{a})\}$

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, ...)



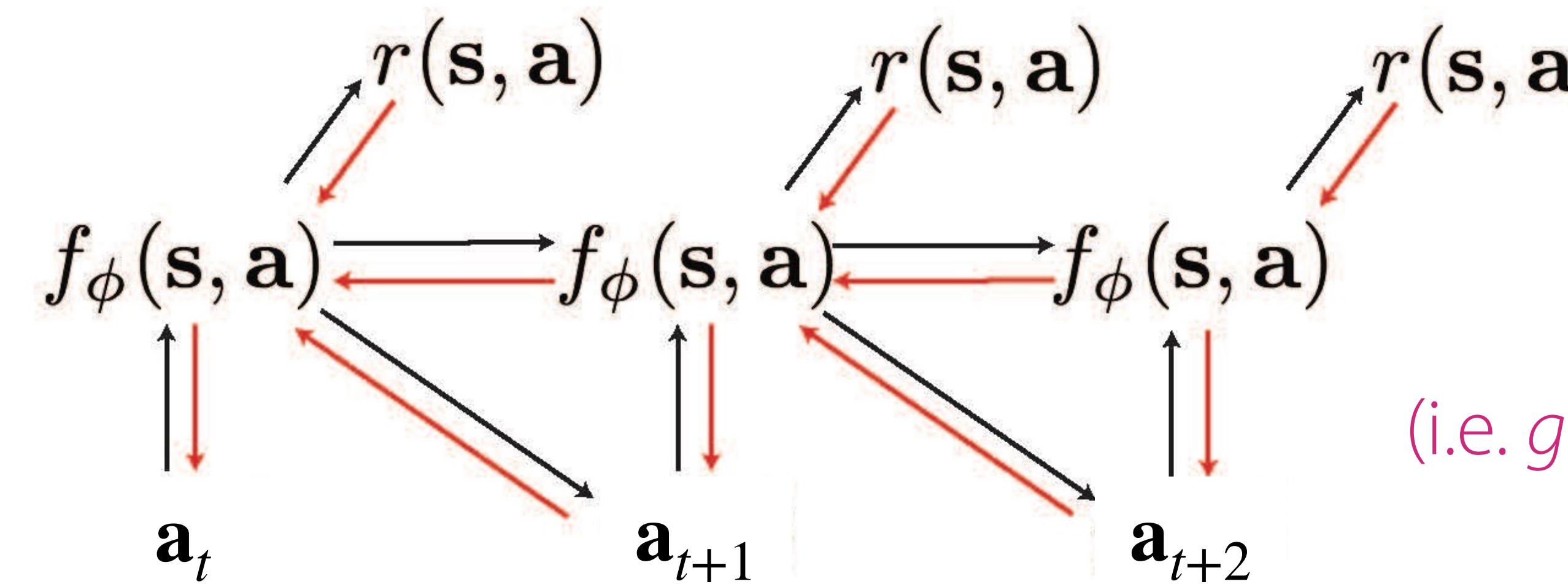
autonomous driving: rules of road constant, user preferences vary
(drive fast, drive smoothly, ...)

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)

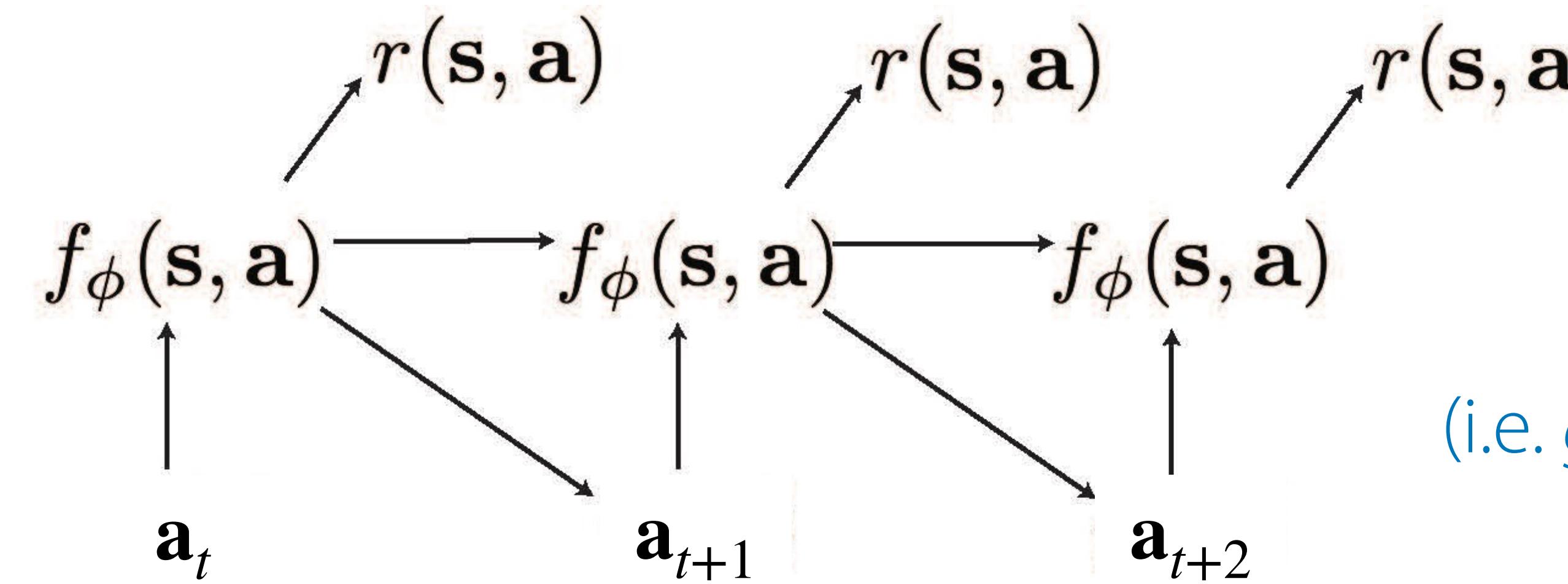
Algorithm:

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 (or pass gradients to policy $\pi_\theta(\mathbf{a} | \mathbf{s})$)

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*

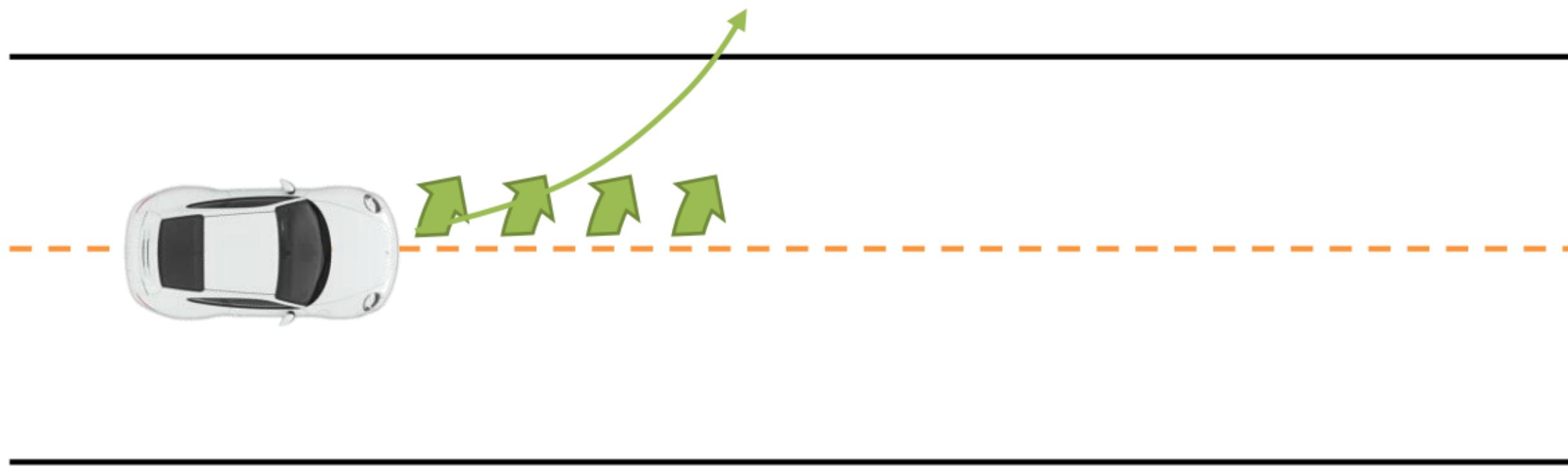
(i.e. *gradient-free* optimization)

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

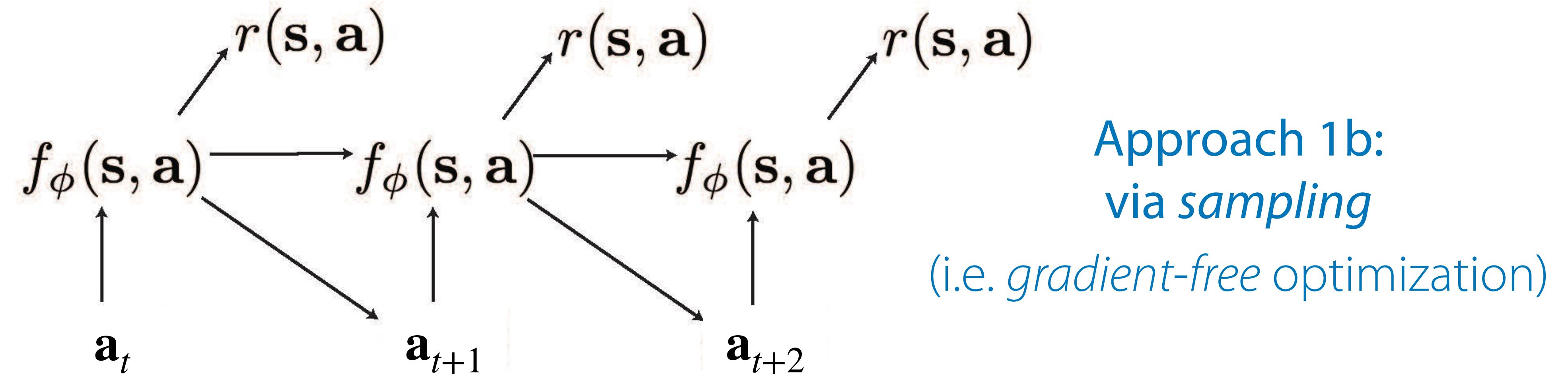
How can this approach fail?

Action optimization will exploit imprecisions in model



Refitting model using new data.

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

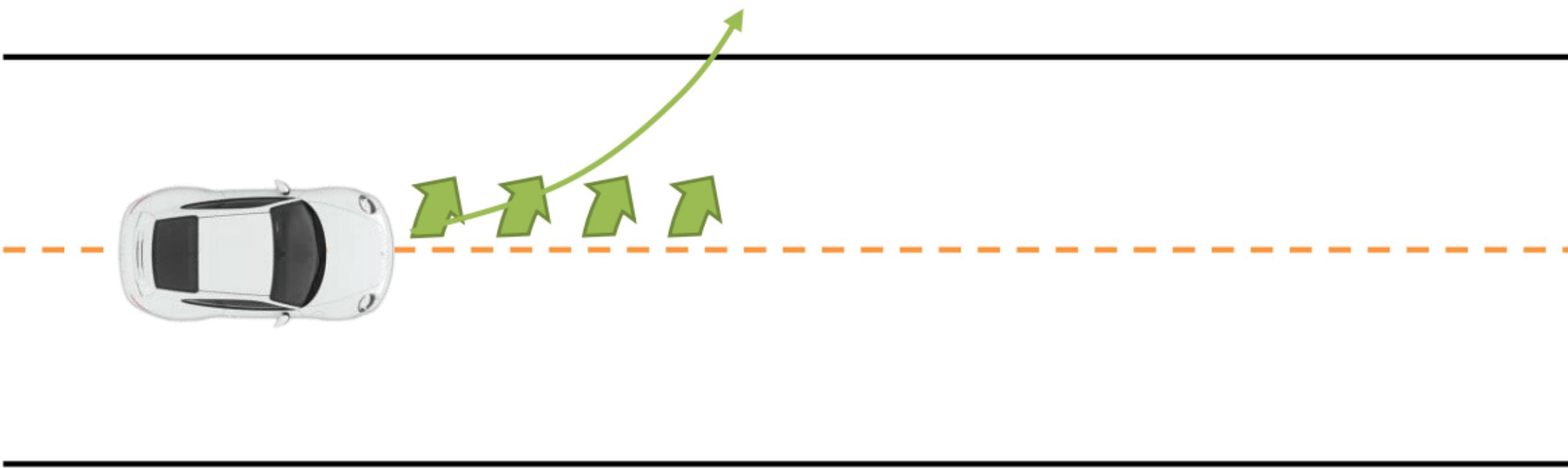


Algorithm:

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, learning a good global model is hard.

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

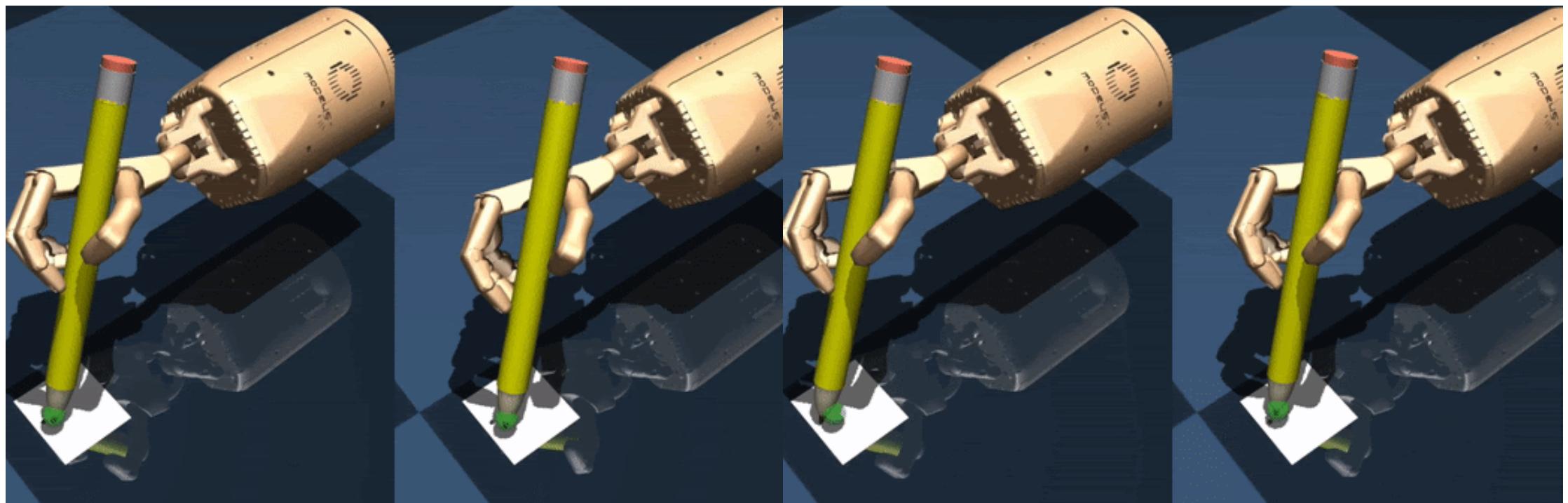
What does this have to do with multi-task RL?

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

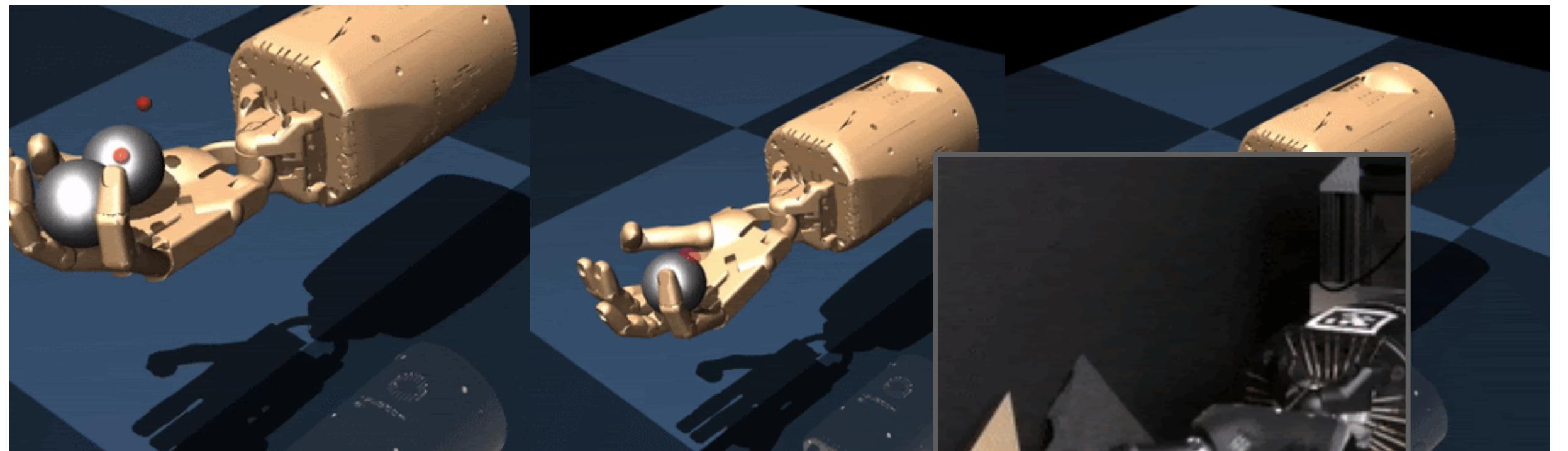
If yes: learn single model, plan w.r.t. each r_i at test time

*Caveat: reward will change how you collect data.

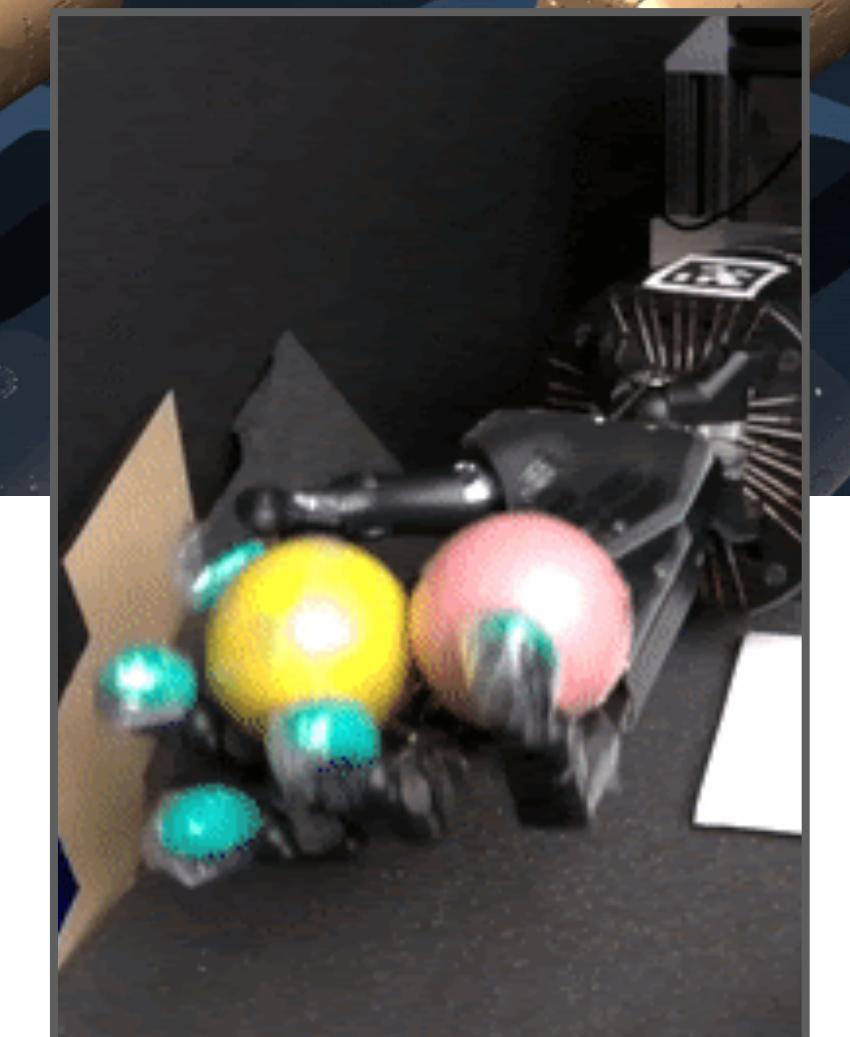
r_i corresponds to different pencil trajectories



r_i corresponds to different ball positions/trajectories



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



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



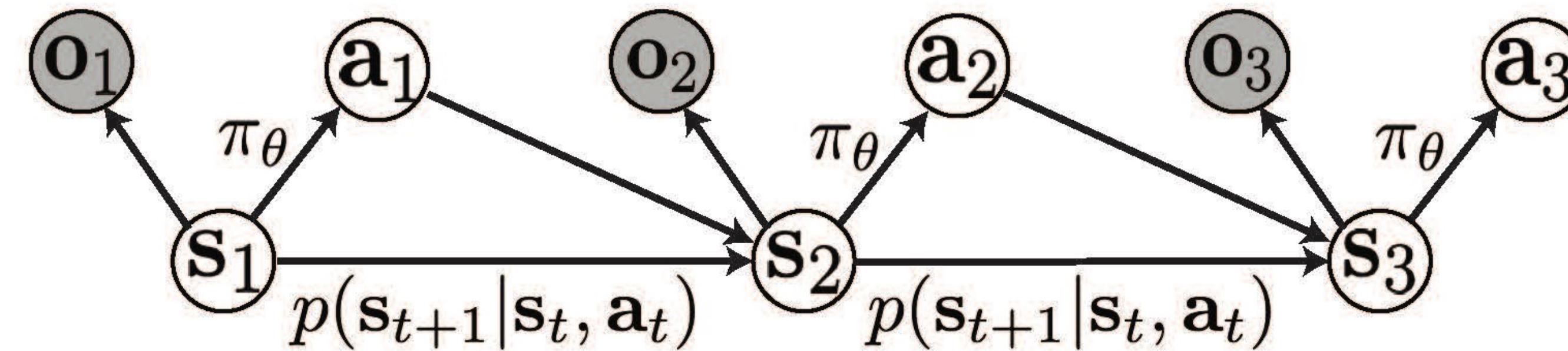
The Plan

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)?

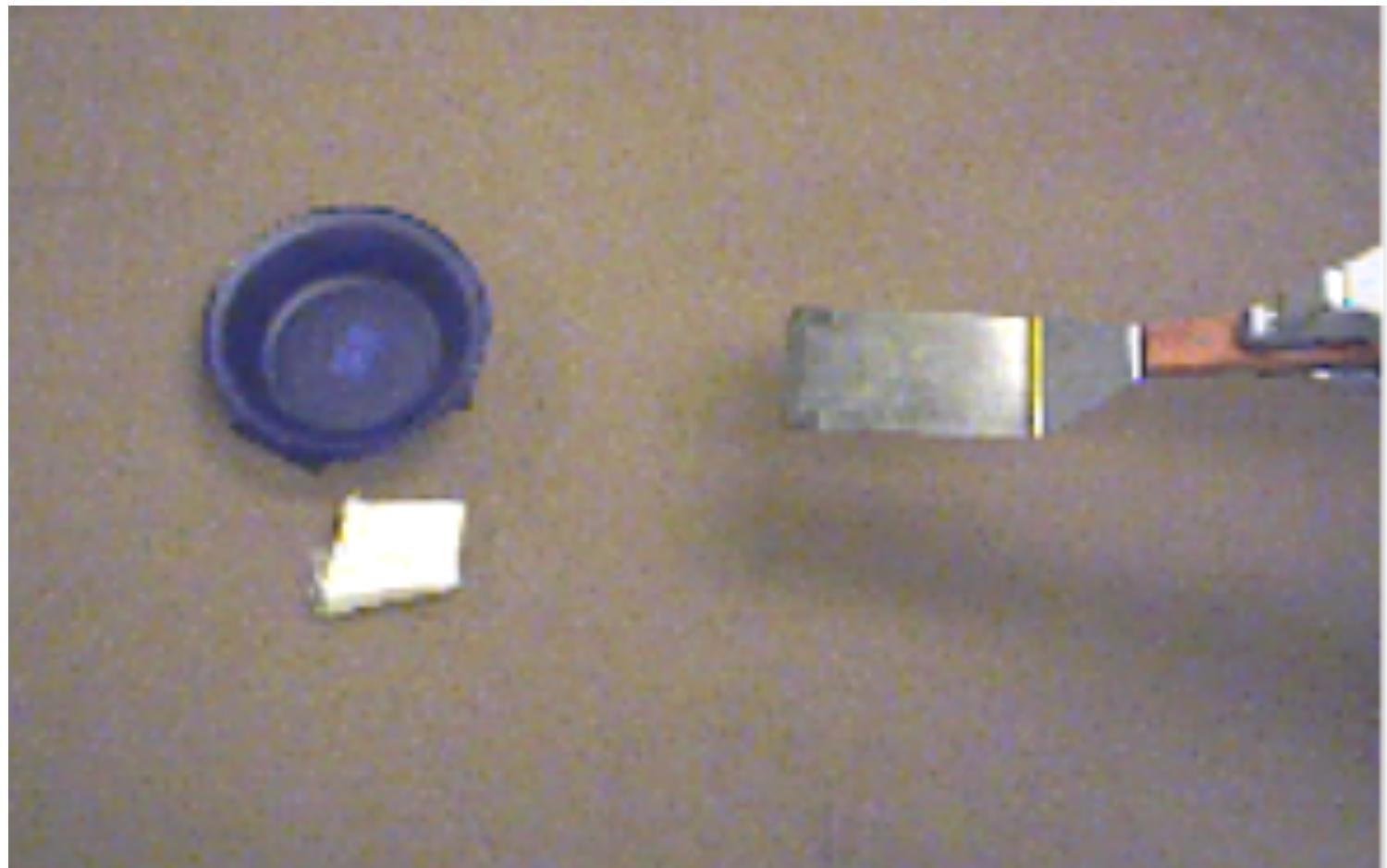


also: no reward signal with only observations

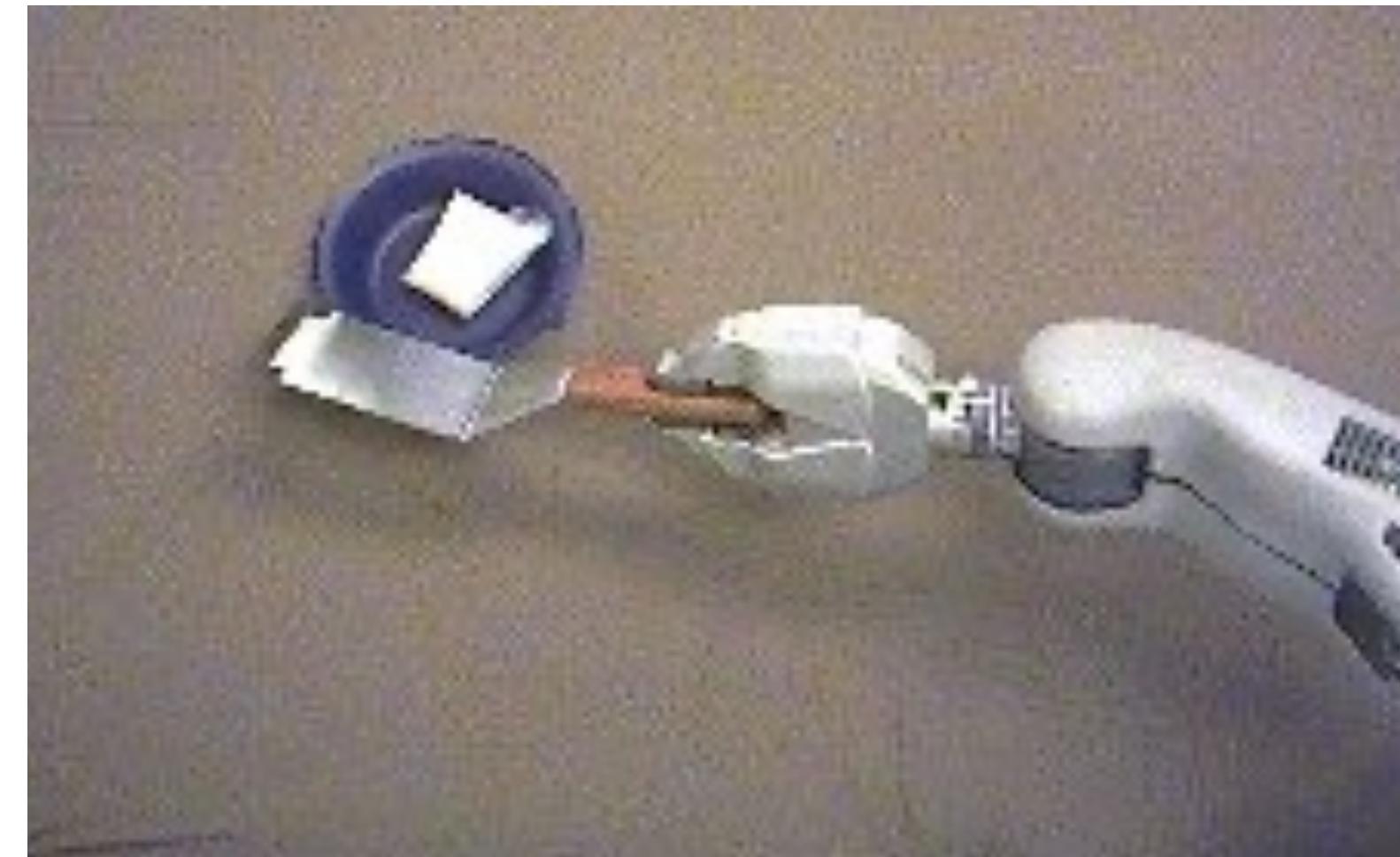
Only access to high-dimensional observations (i.e. images)?

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

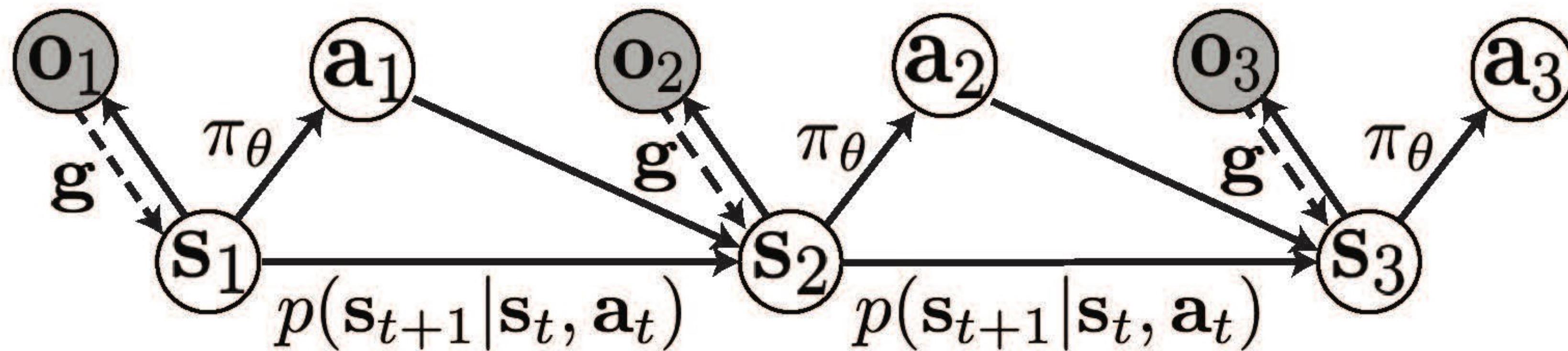
1. Learn model in latent space
2. Learn model of observations (e.g. video)
3. Predict alternative quantities

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

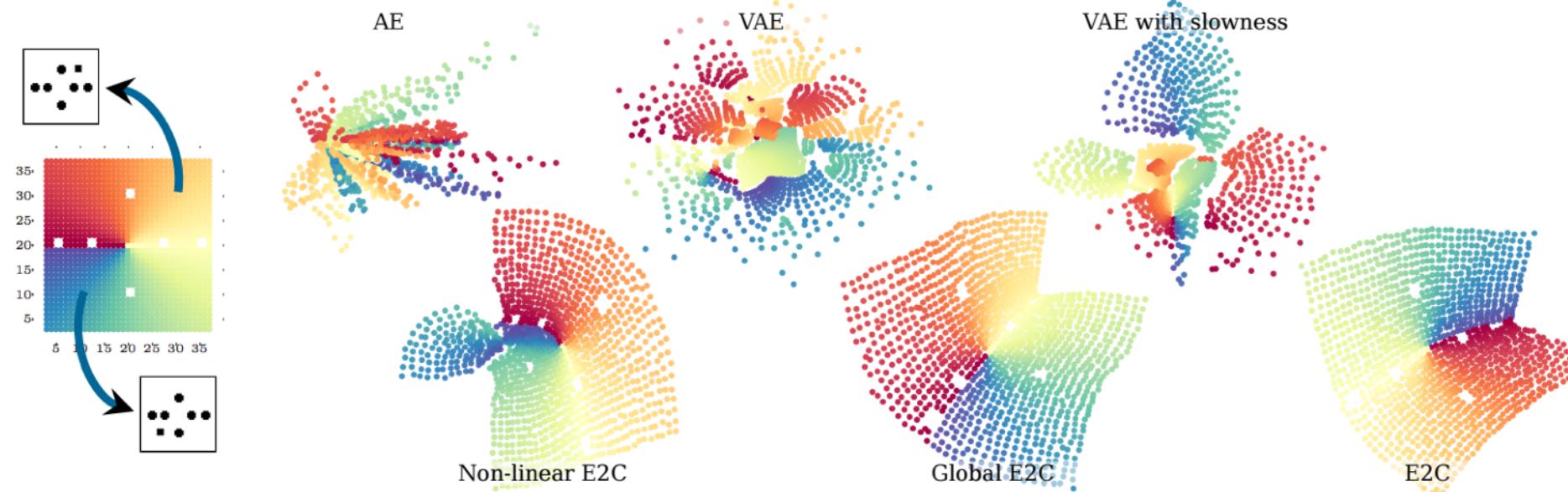


Learning in Latent Space

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`

Deep Spatial Autoencoders for Visuomotor Learning

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

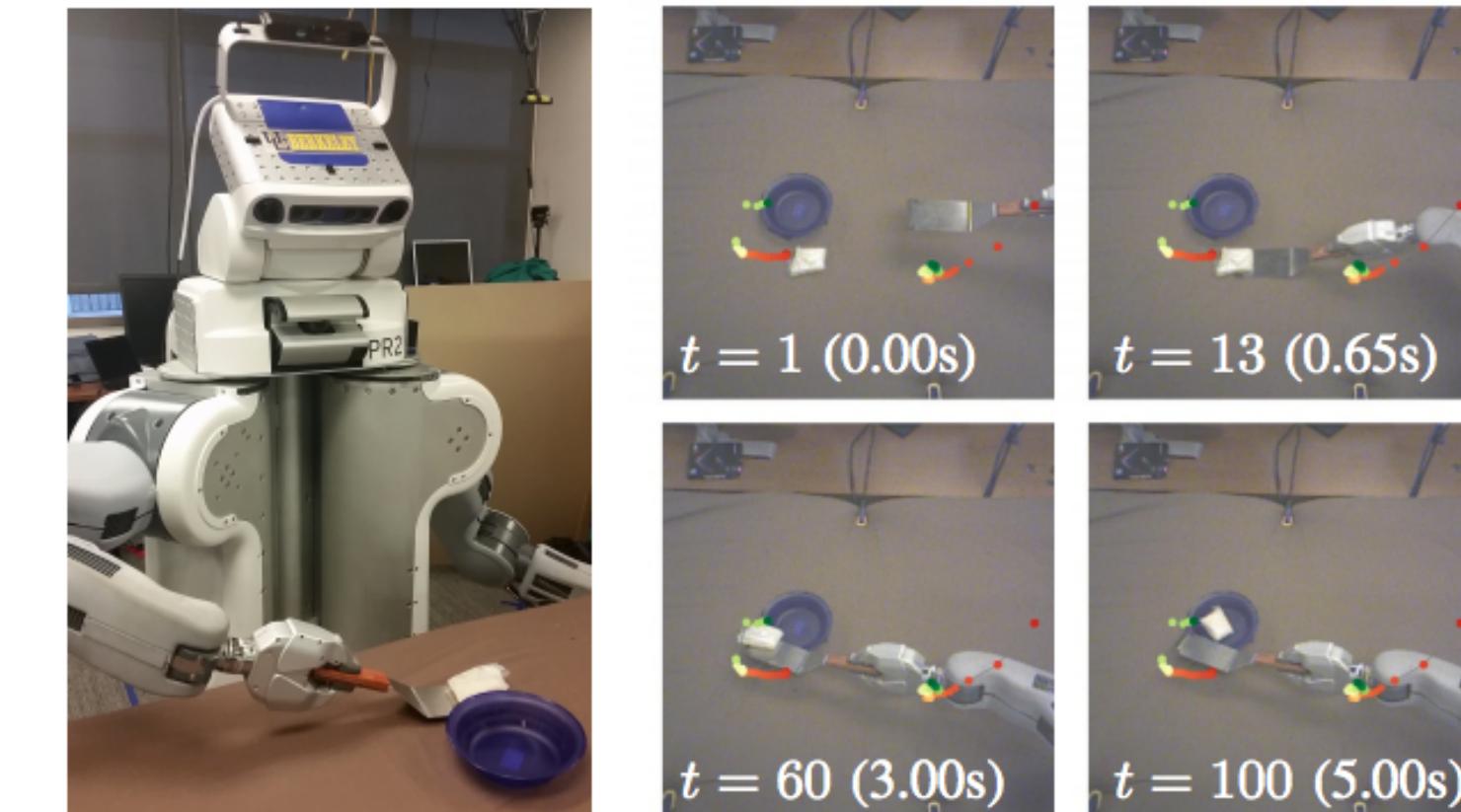
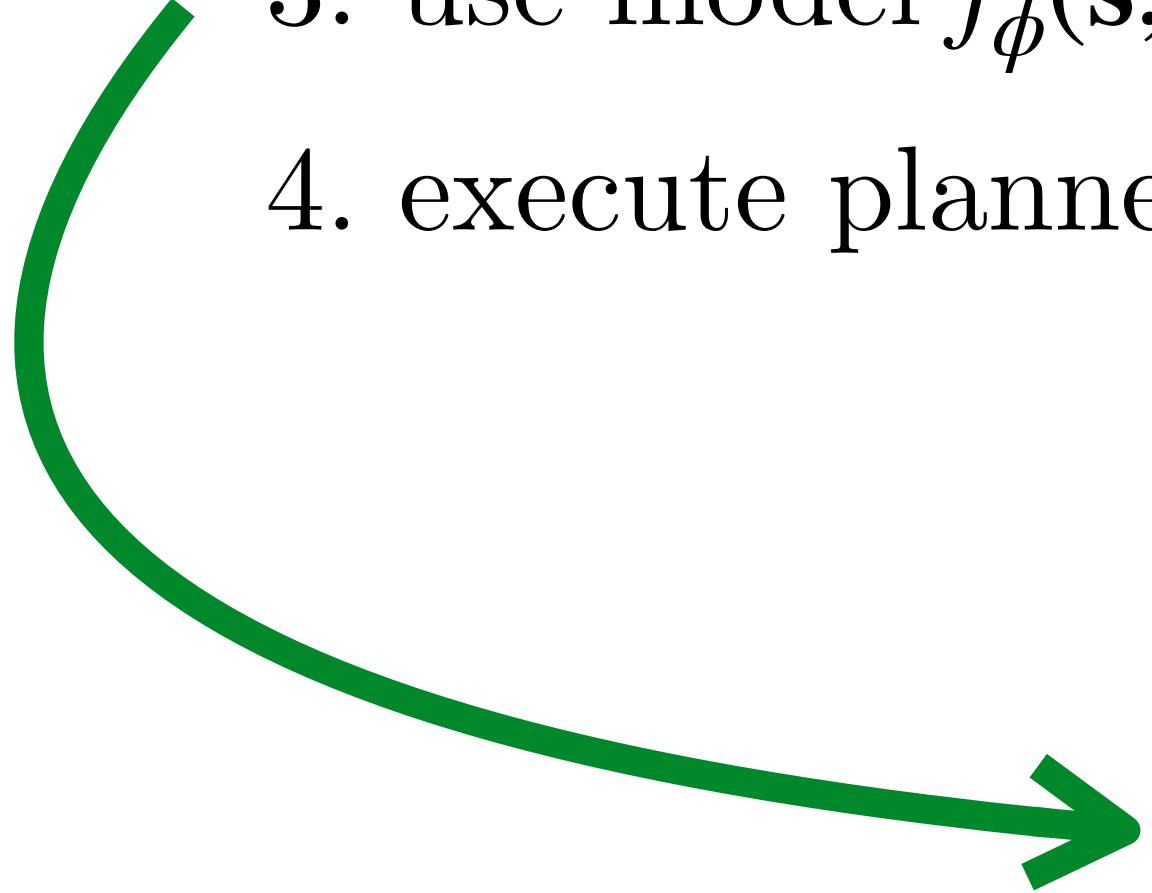


Fig. 1: PR2 learning to scoop a bag of rice into a bowl with a spatula (left) using a learned visual state representation (right).

Learning in Latent Space

Algorithm

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}



What is the reward for optimizing actions?

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

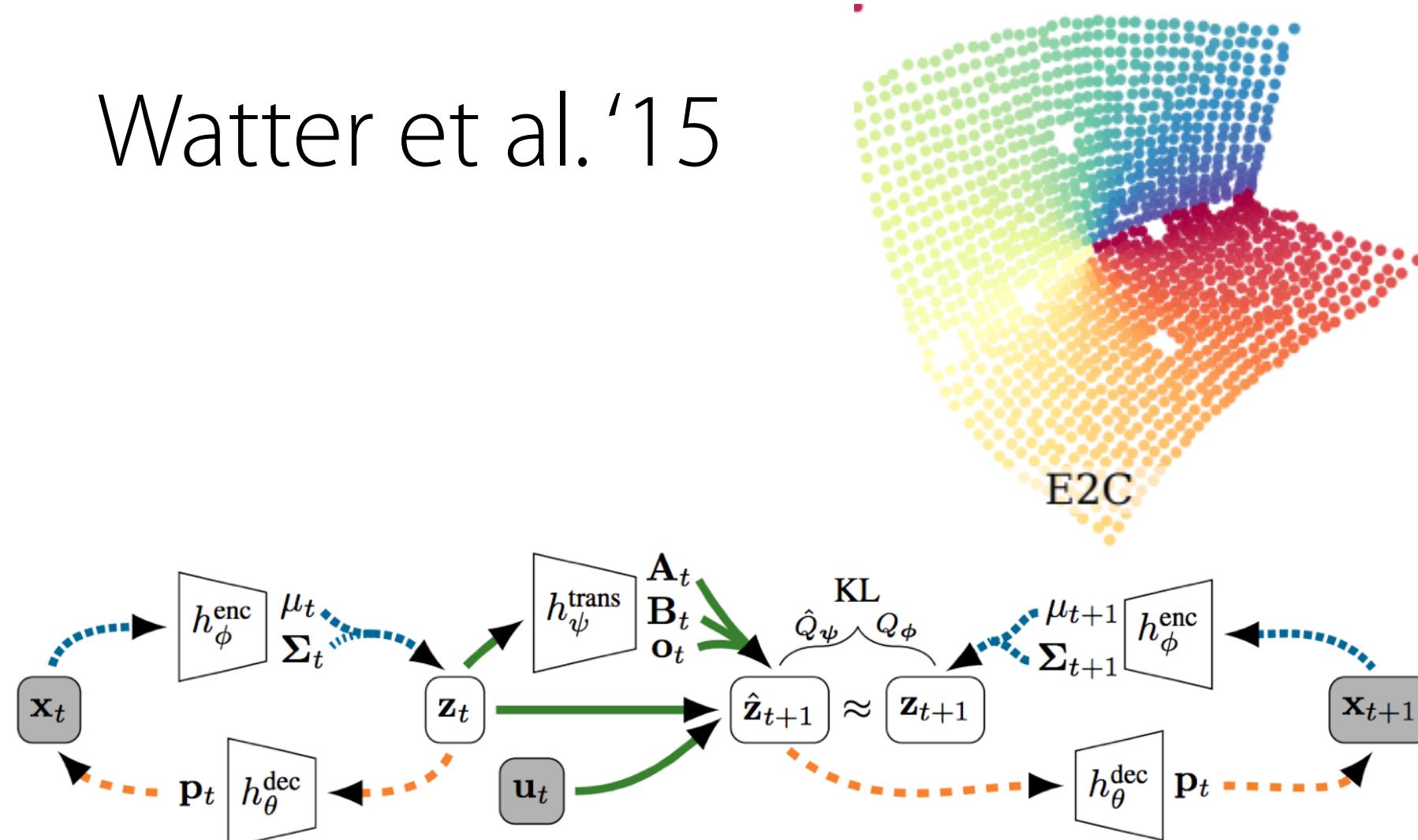
Assumption: distance in latent space is an accurate metric.

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)

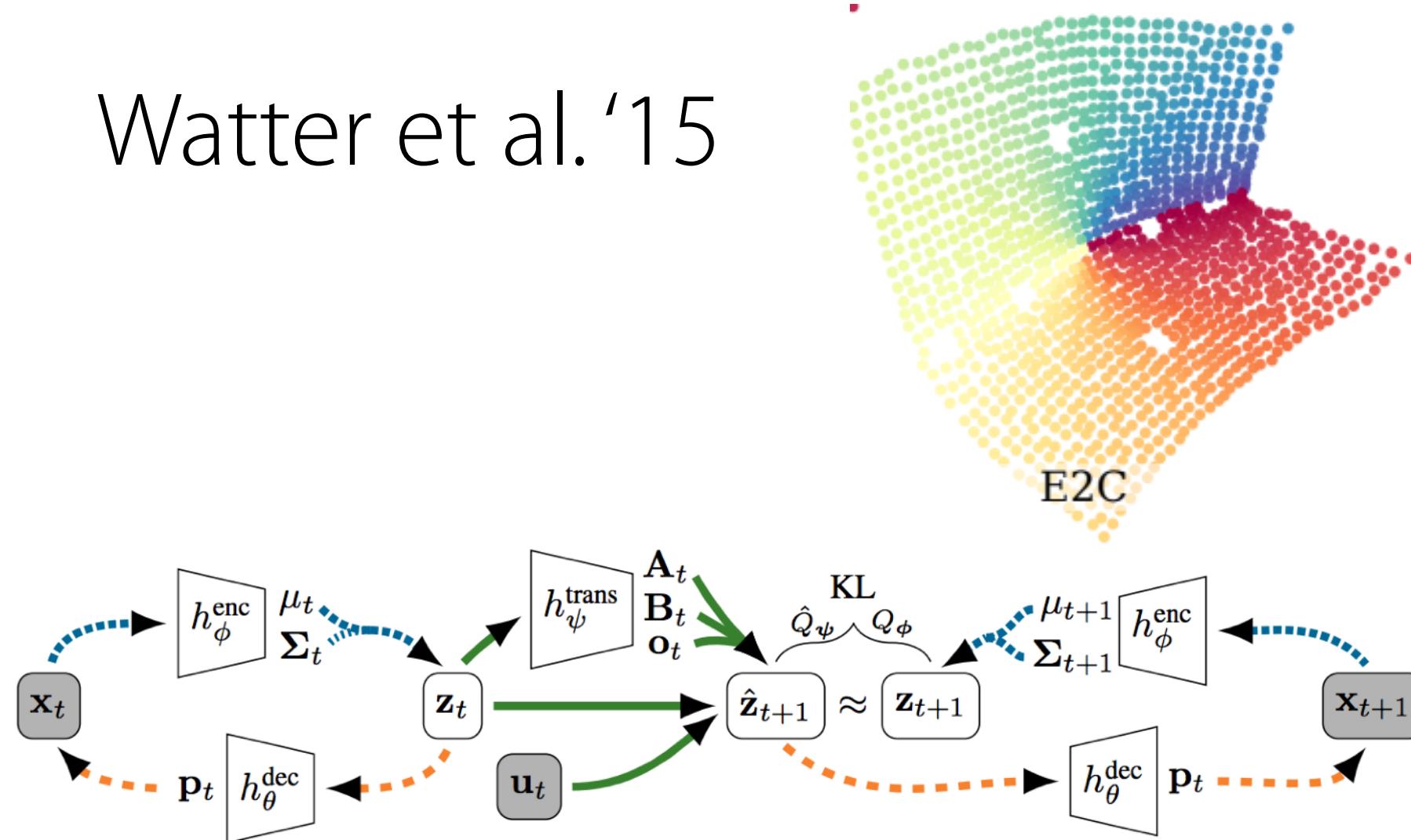
Watter et al. NeurIPS'15

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}

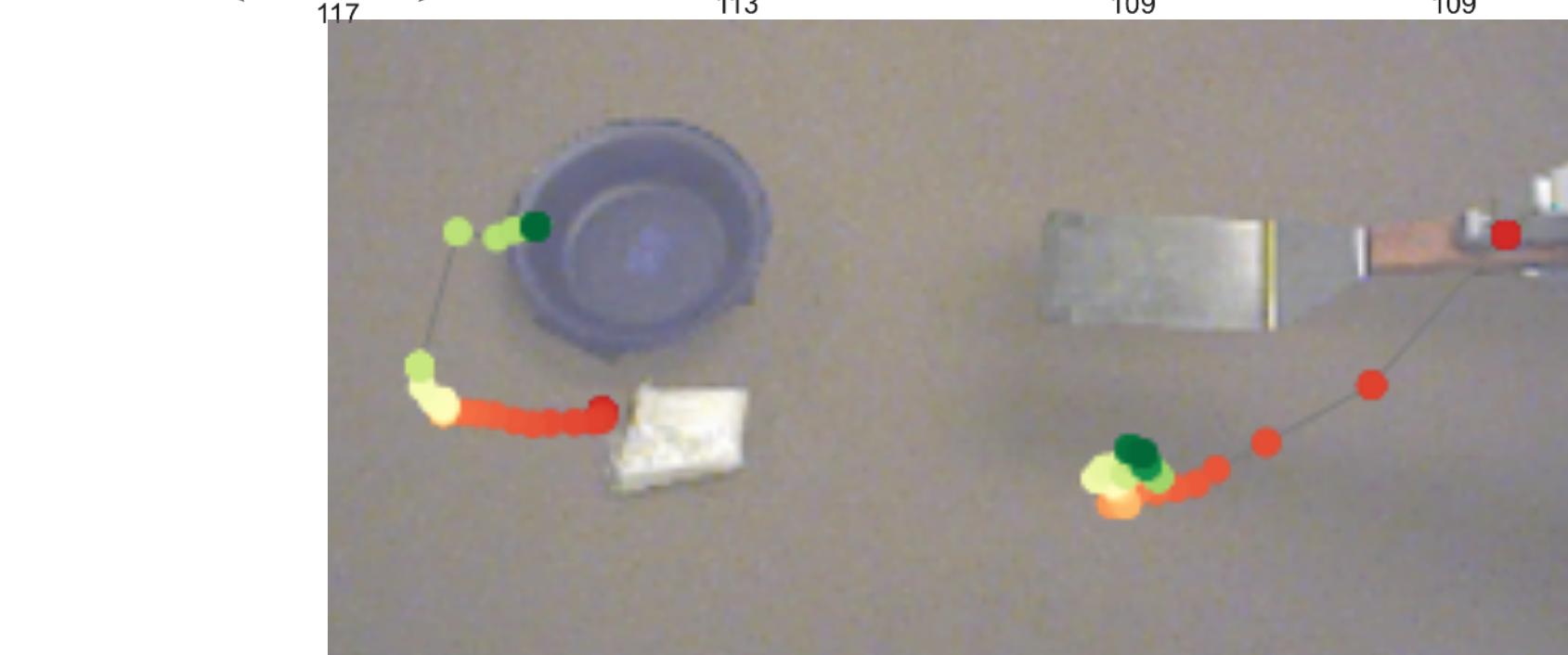
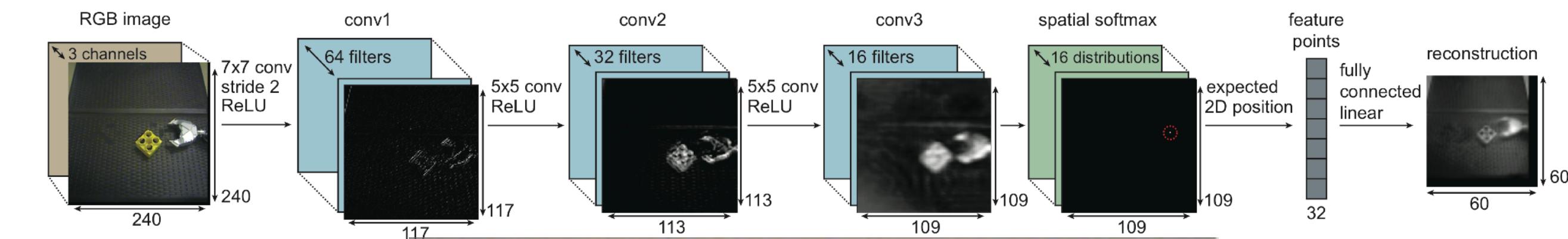
How to optimize latent embedding?

Watter et al. '15



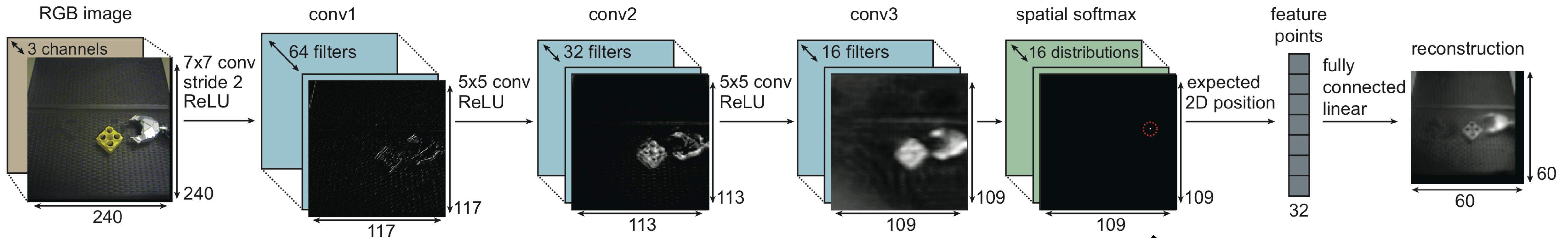
learn embedding & model jointly

Finn et al. '16

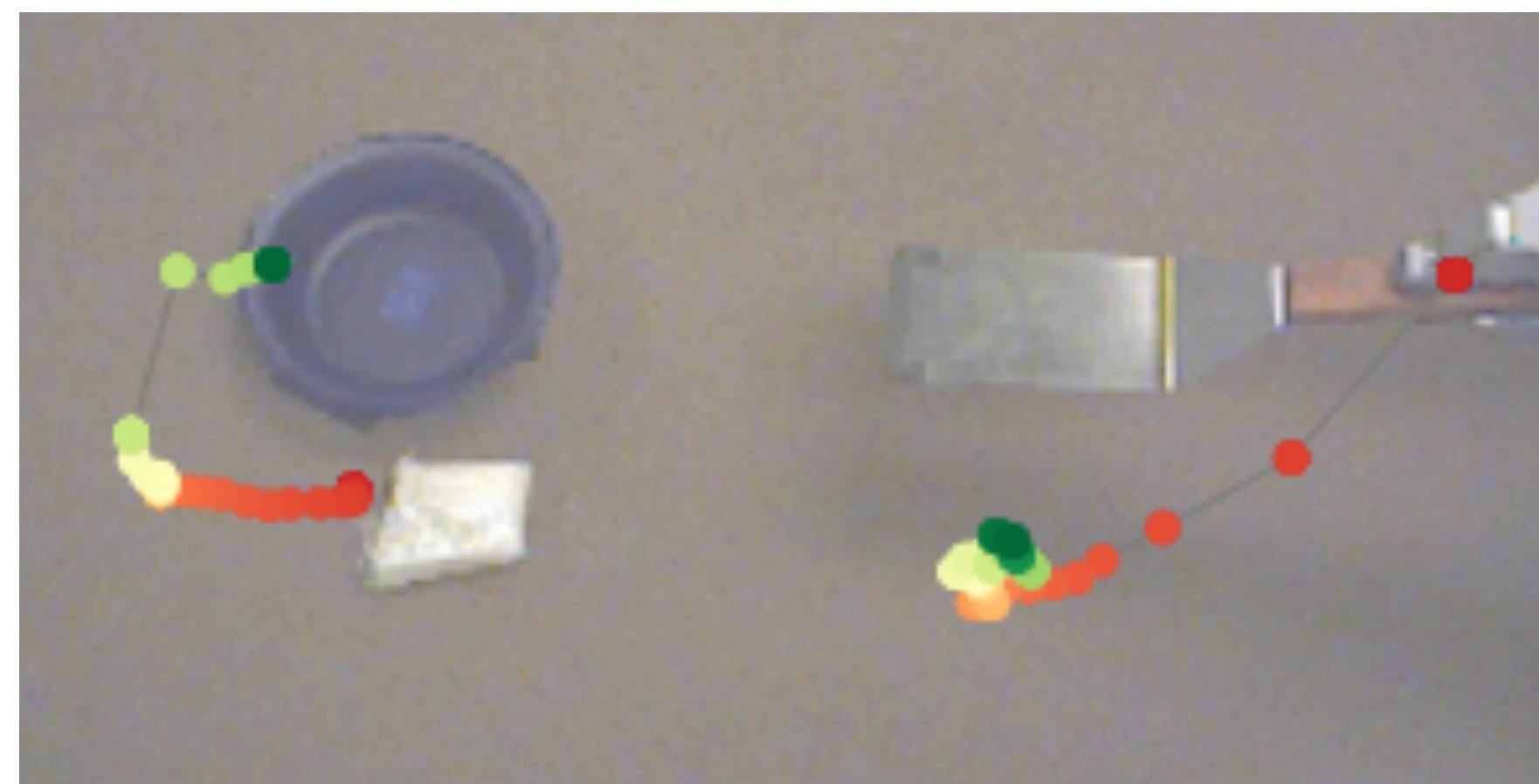


embedding is **smooth** and **structured**

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

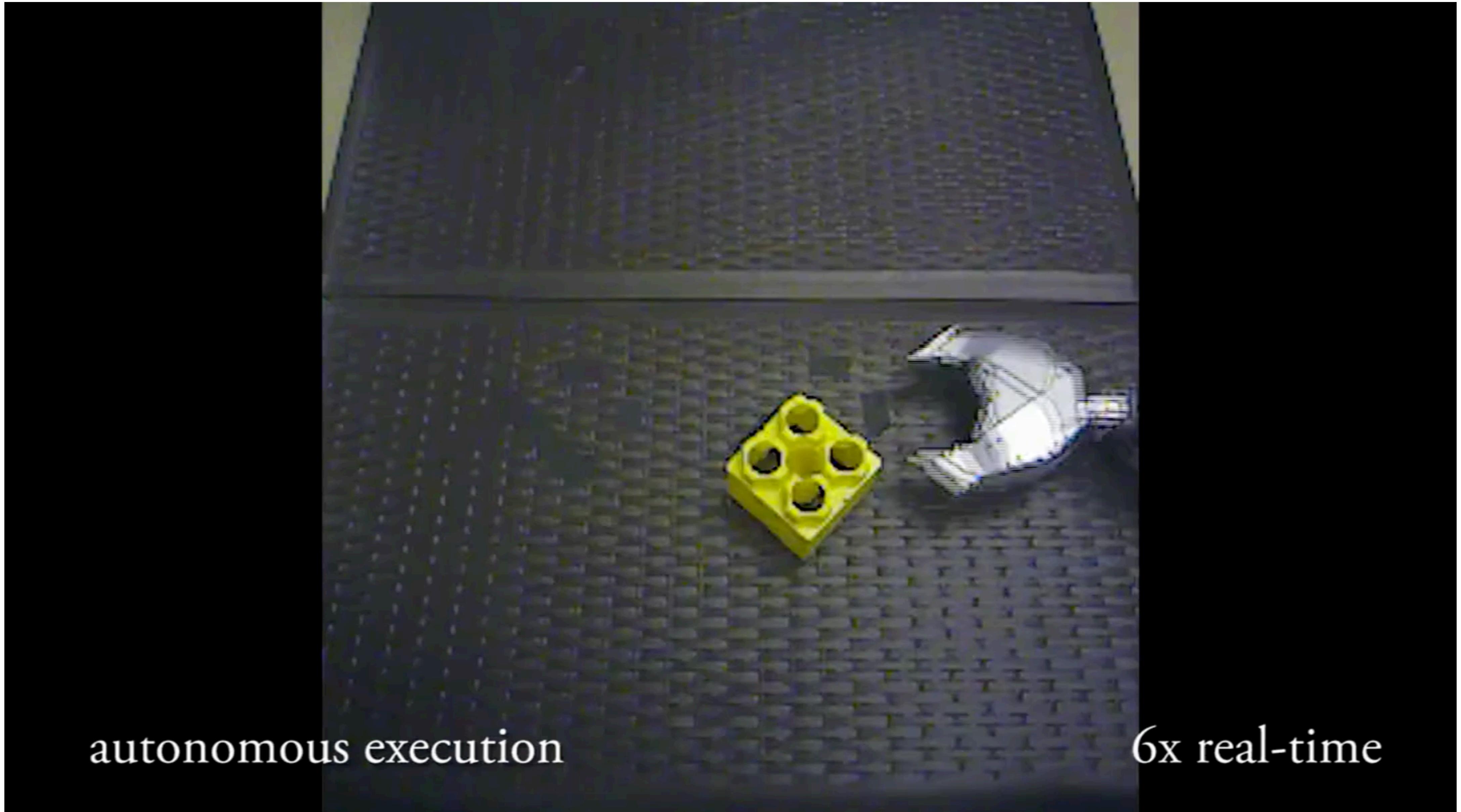


$$\left(\sum_{i,j} i S_{ij}, \sum_{i,j} j S_{ij} \right)$$



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

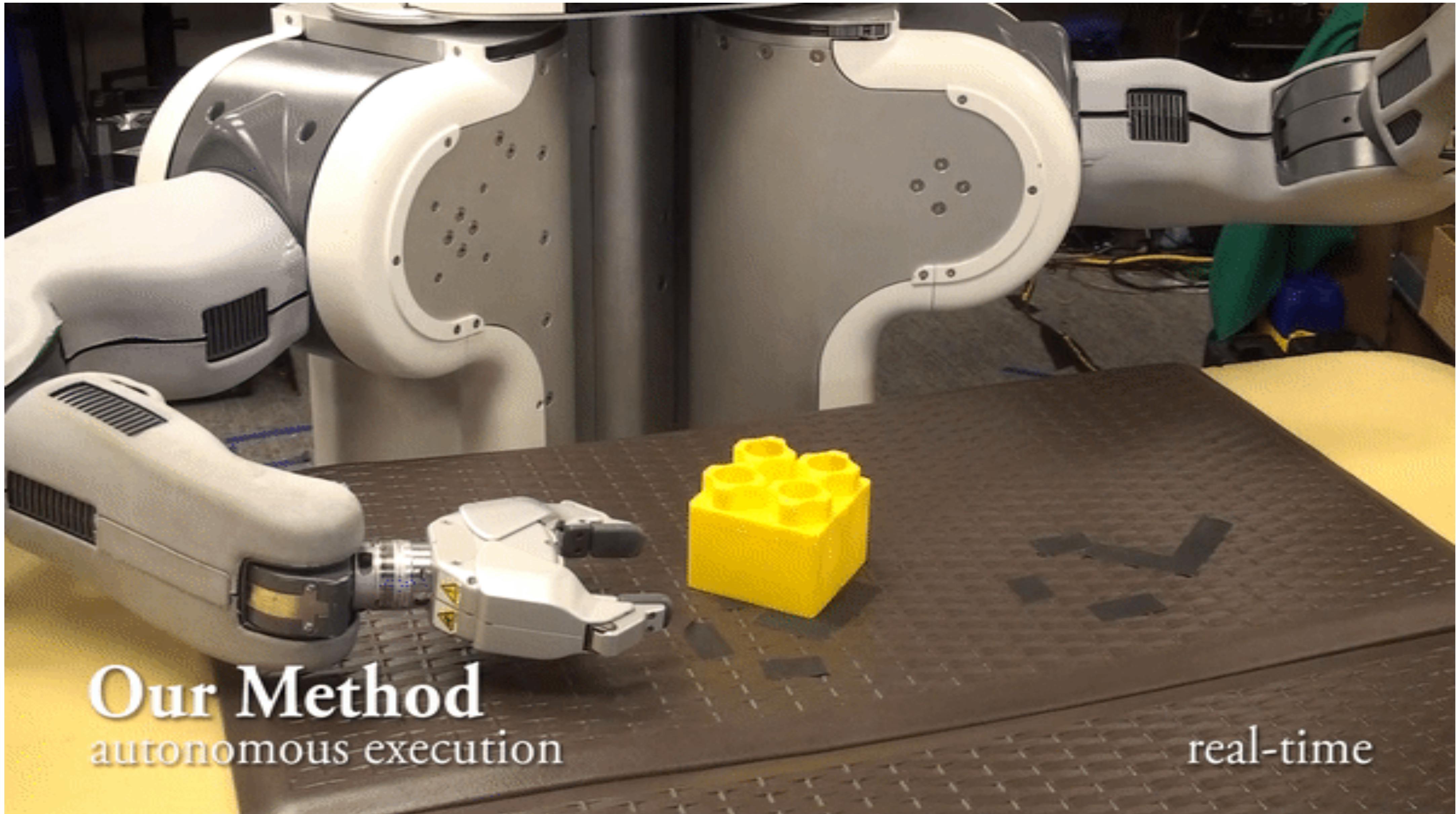
Learning in Latent Space



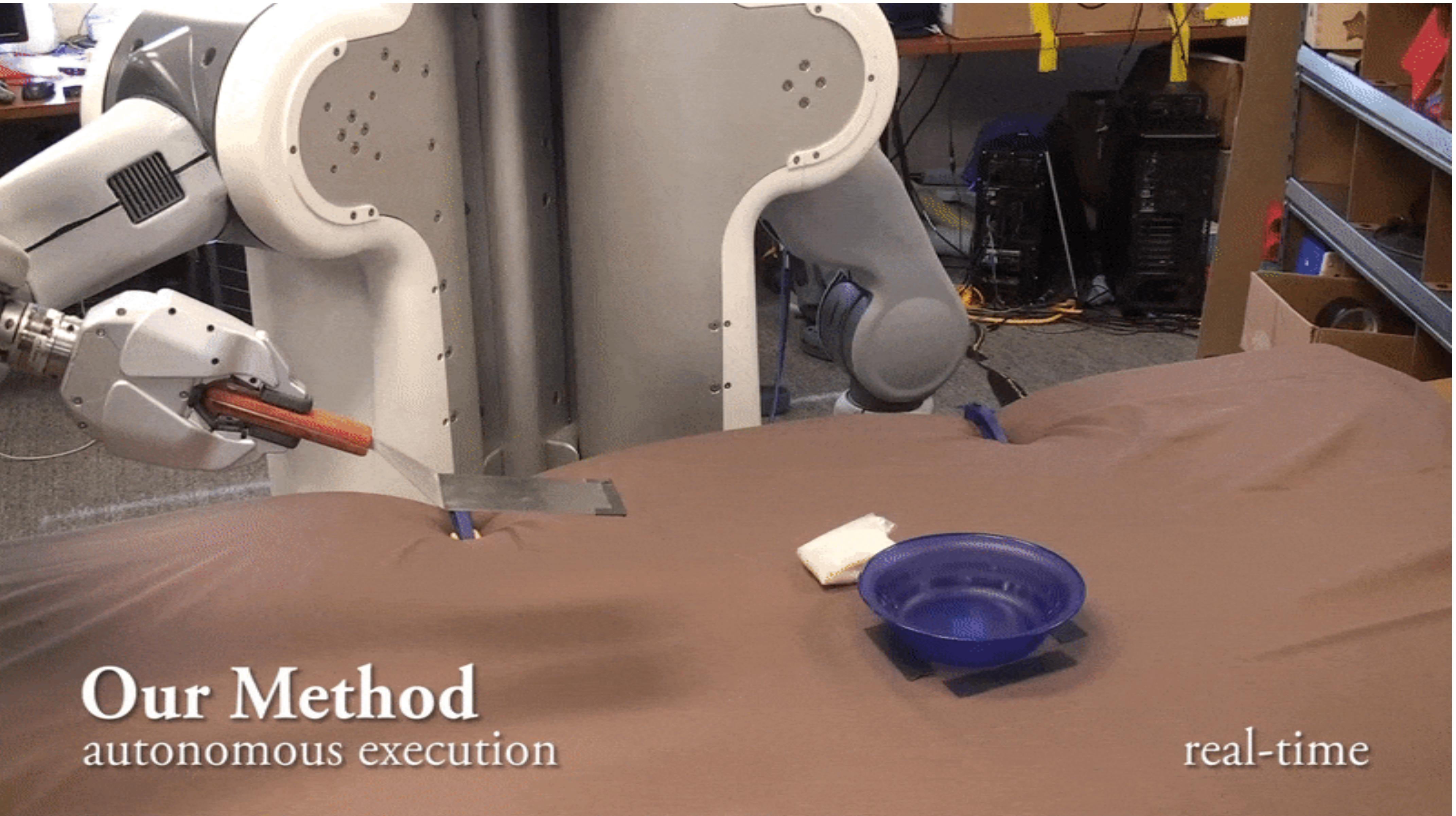
autonomous execution

6x real-time

Learning in Latent Space



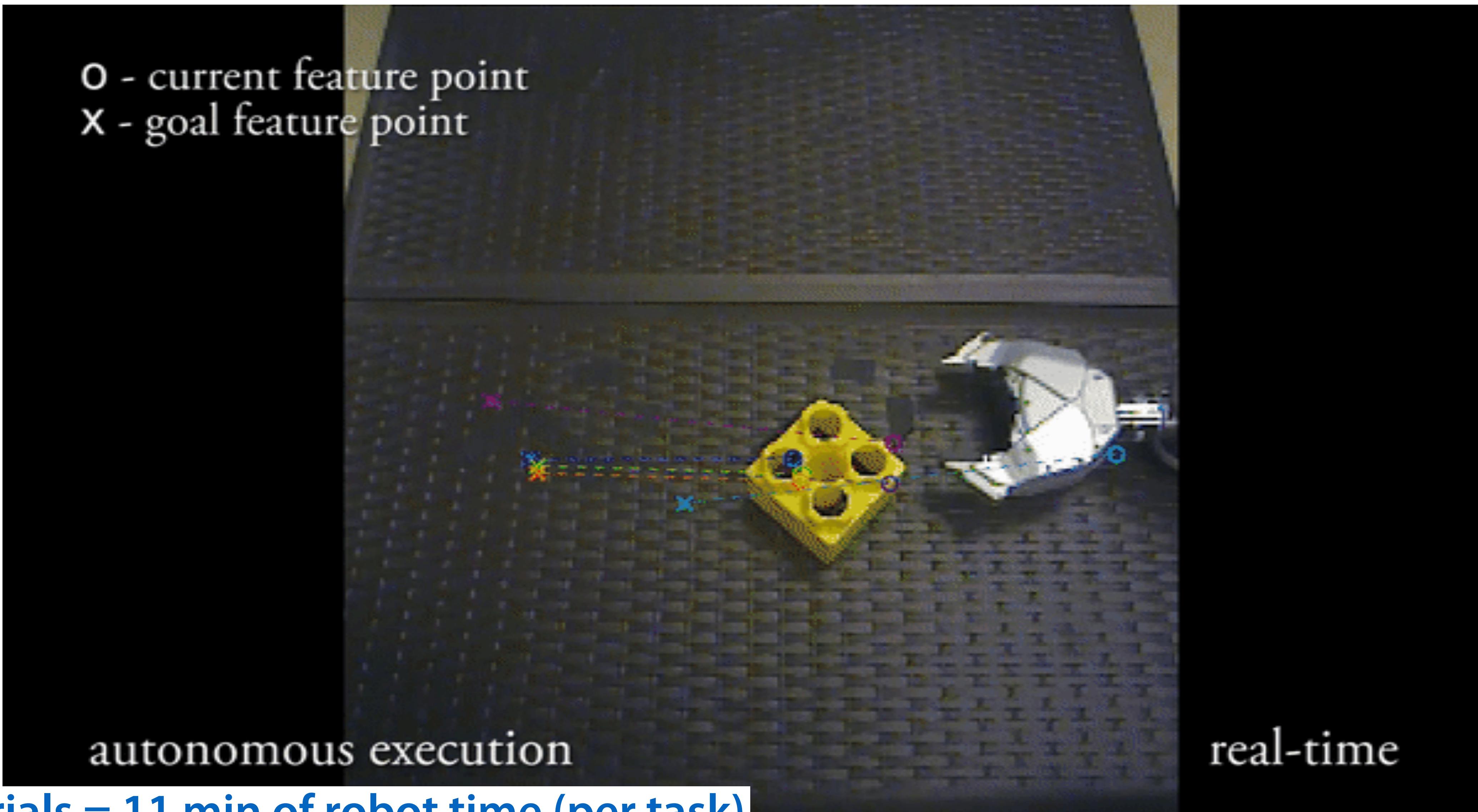
Learning in Latent Space



Our Method
autonomous execution

real-time

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?

(answer in chat or by raising hand)

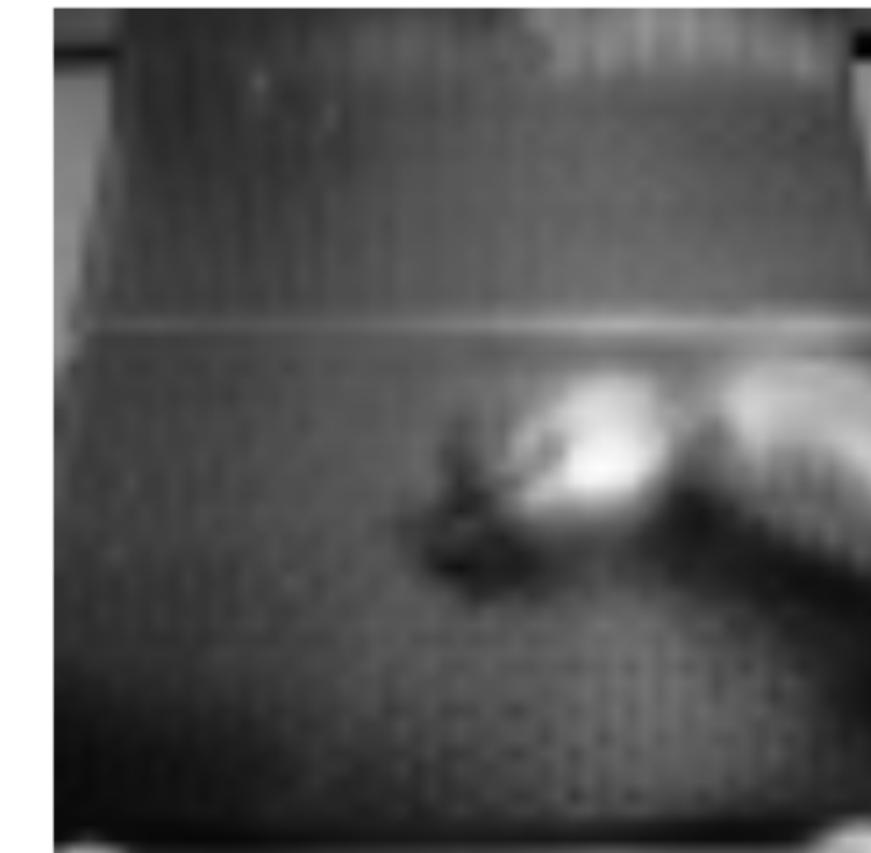
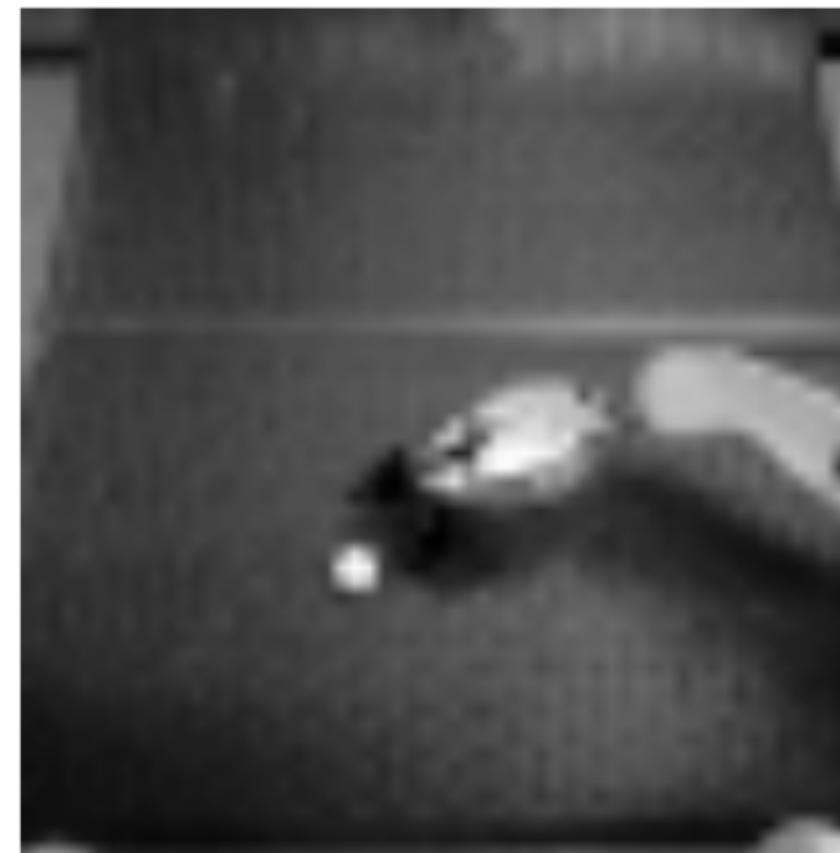
Learning in Latent Space

Pros:

- + Learn complex visual skills very efficiently
- + Structured representation enables effective learning

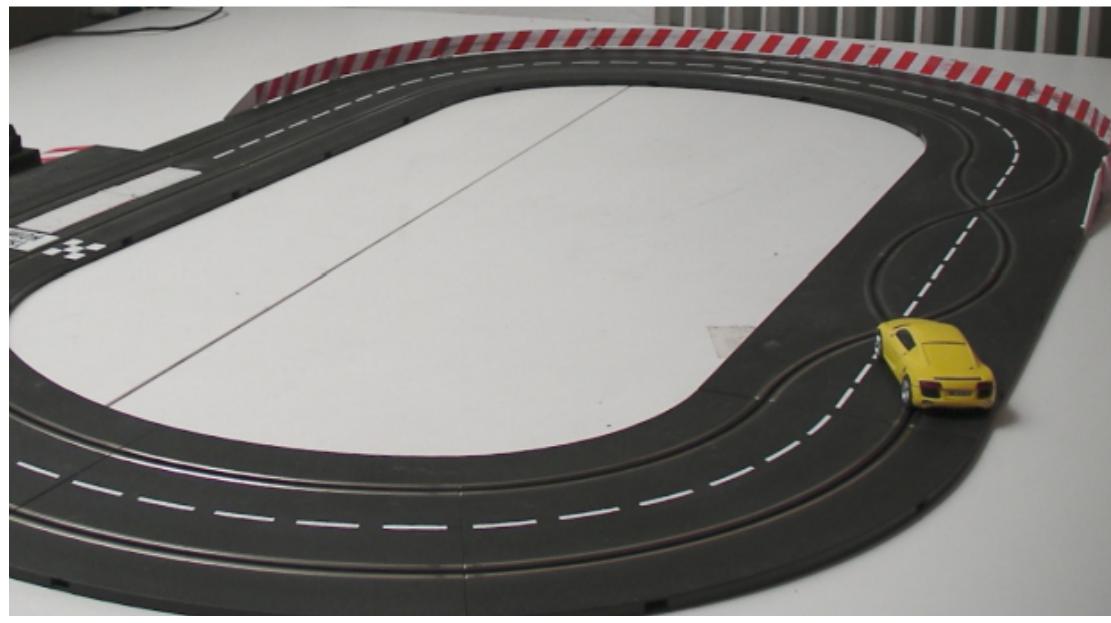
Cons:

- 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



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

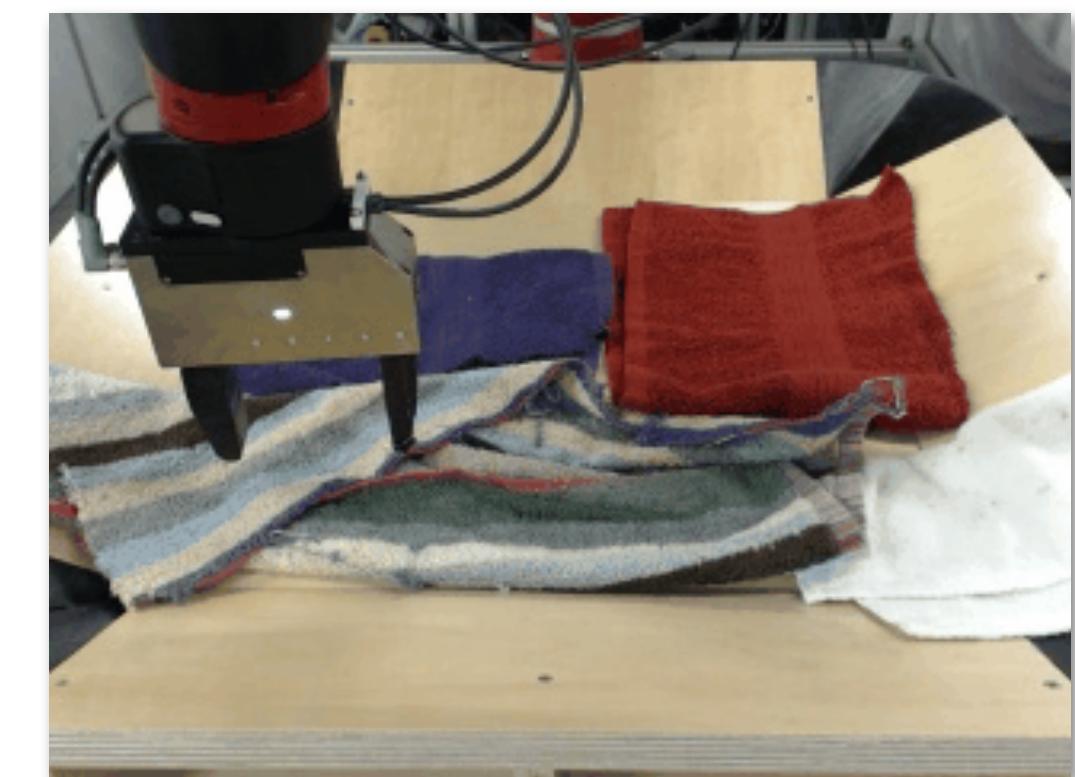
1. run base policy $\pi_0(\mathbf{a}_t | \mathbf{o}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$
2. learn model $f_\phi(\mathbf{o}, \mathbf{a})$ to minimize $\sum_i \|f_\phi(\mathbf{o}_i, \mathbf{a}_i) - \mathbf{o}'_i\|^2$
3. use model $f_\phi(\mathbf{o}, \mathbf{a})$ 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}



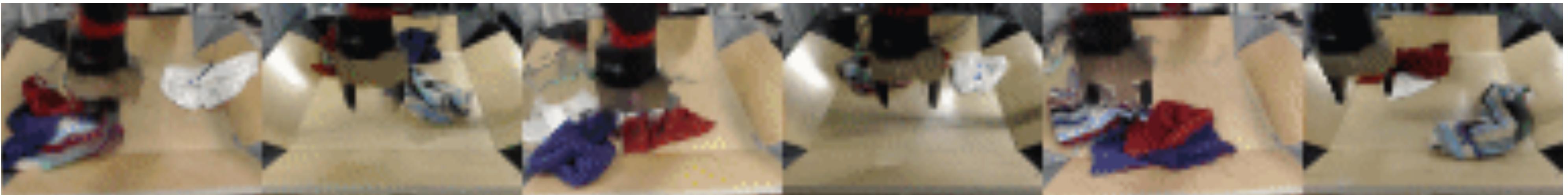
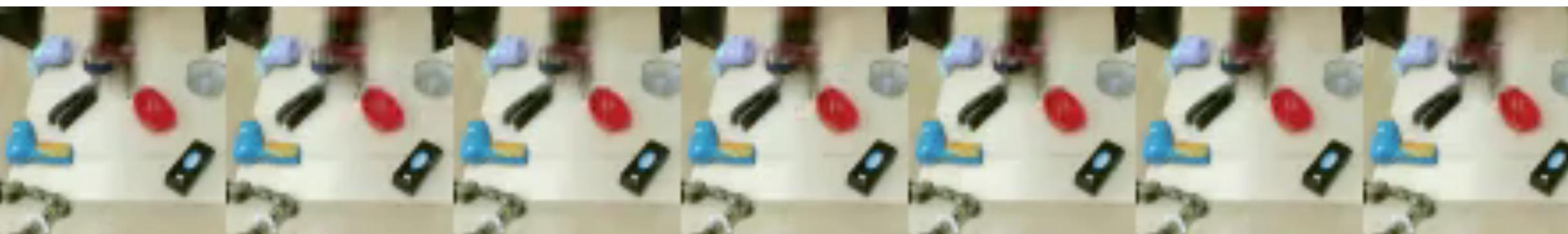
What's new?

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 $\mathcal{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$



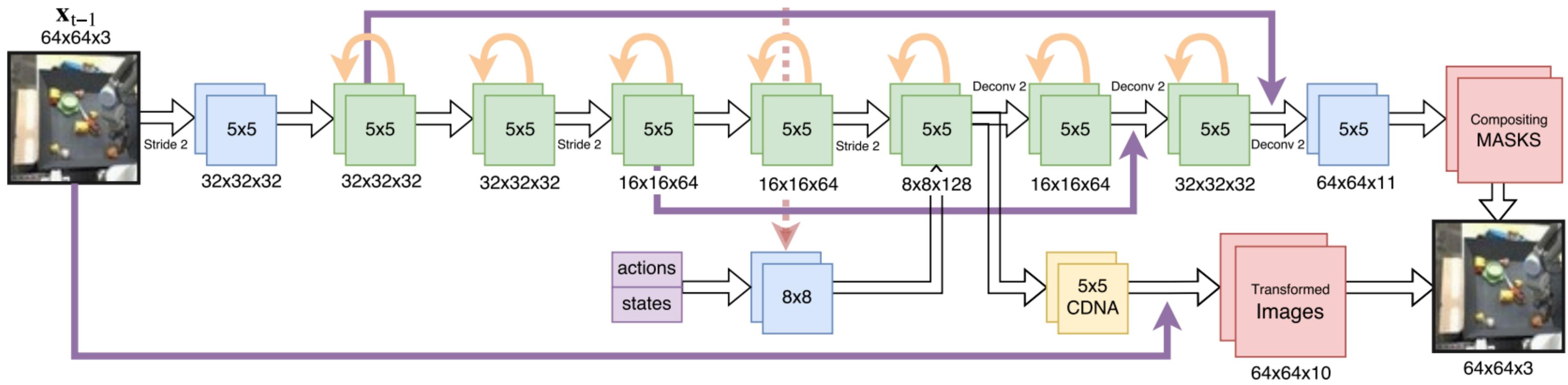
2. learn model $f_\phi(\mathbf{o}, \mathbf{a})$ to minimize $\sum \|f_\phi(\mathbf{o}_i, \mathbf{a}_i) - \mathbf{o}'_i\|^2$



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

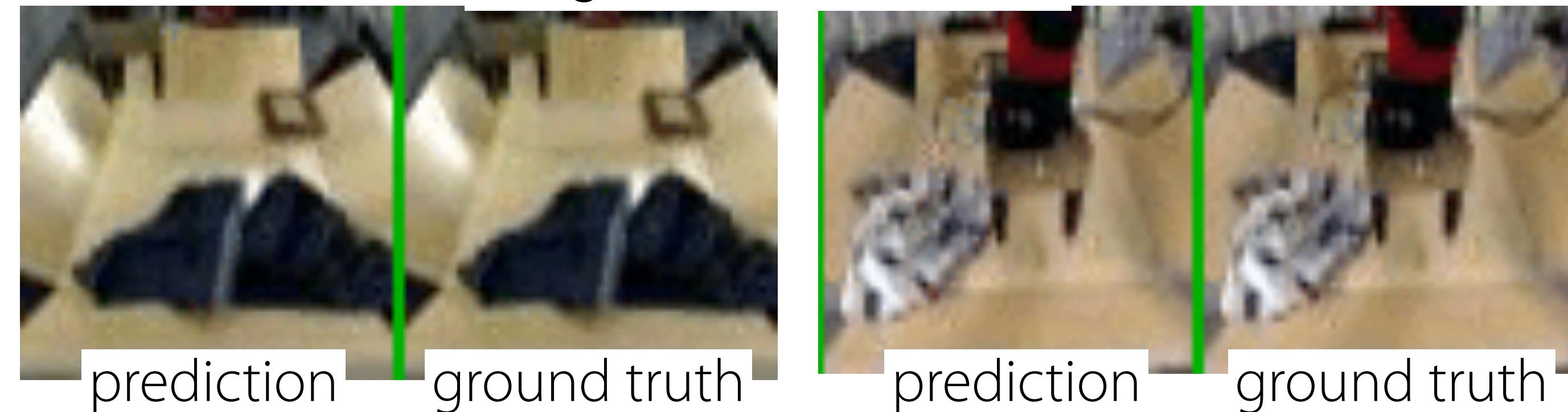
How to predict video?

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



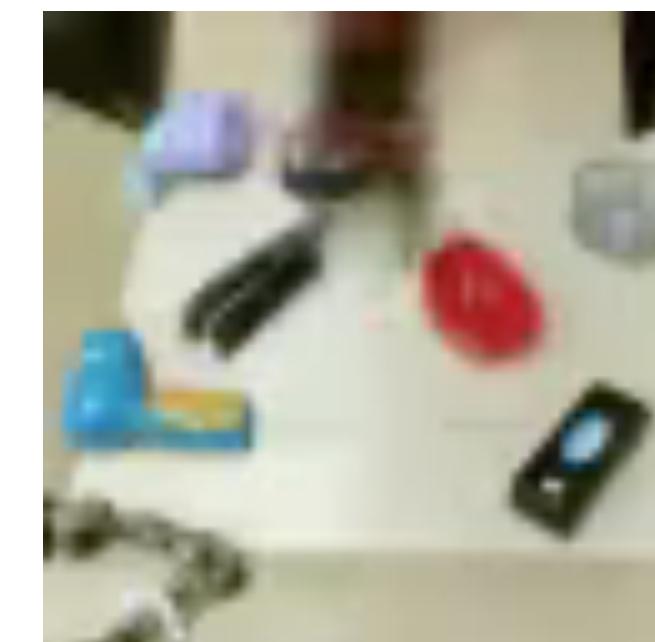
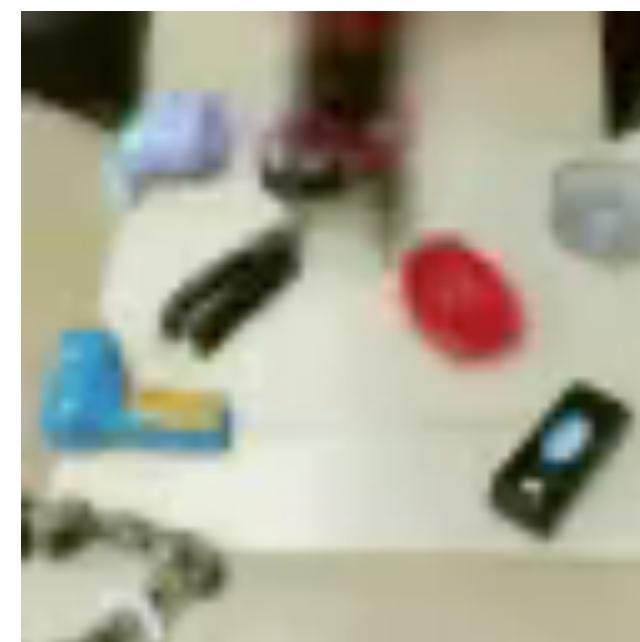
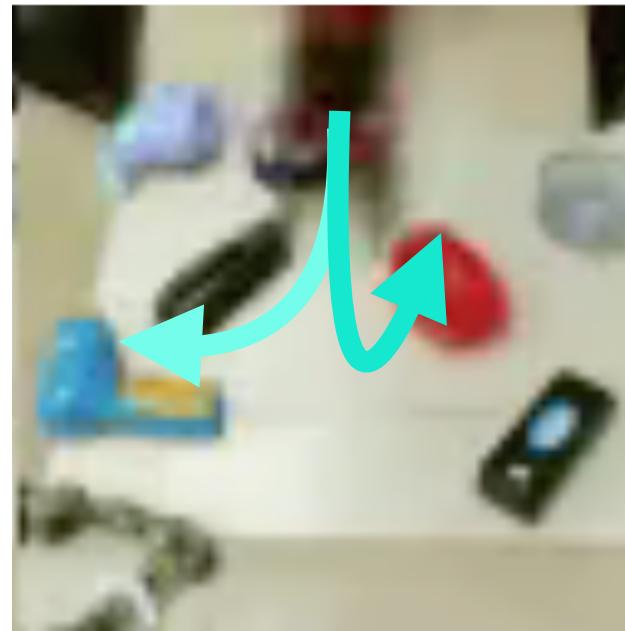
How to plan?

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

Gradient-free 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)



How it works

Specify goal



Visual MPC execution

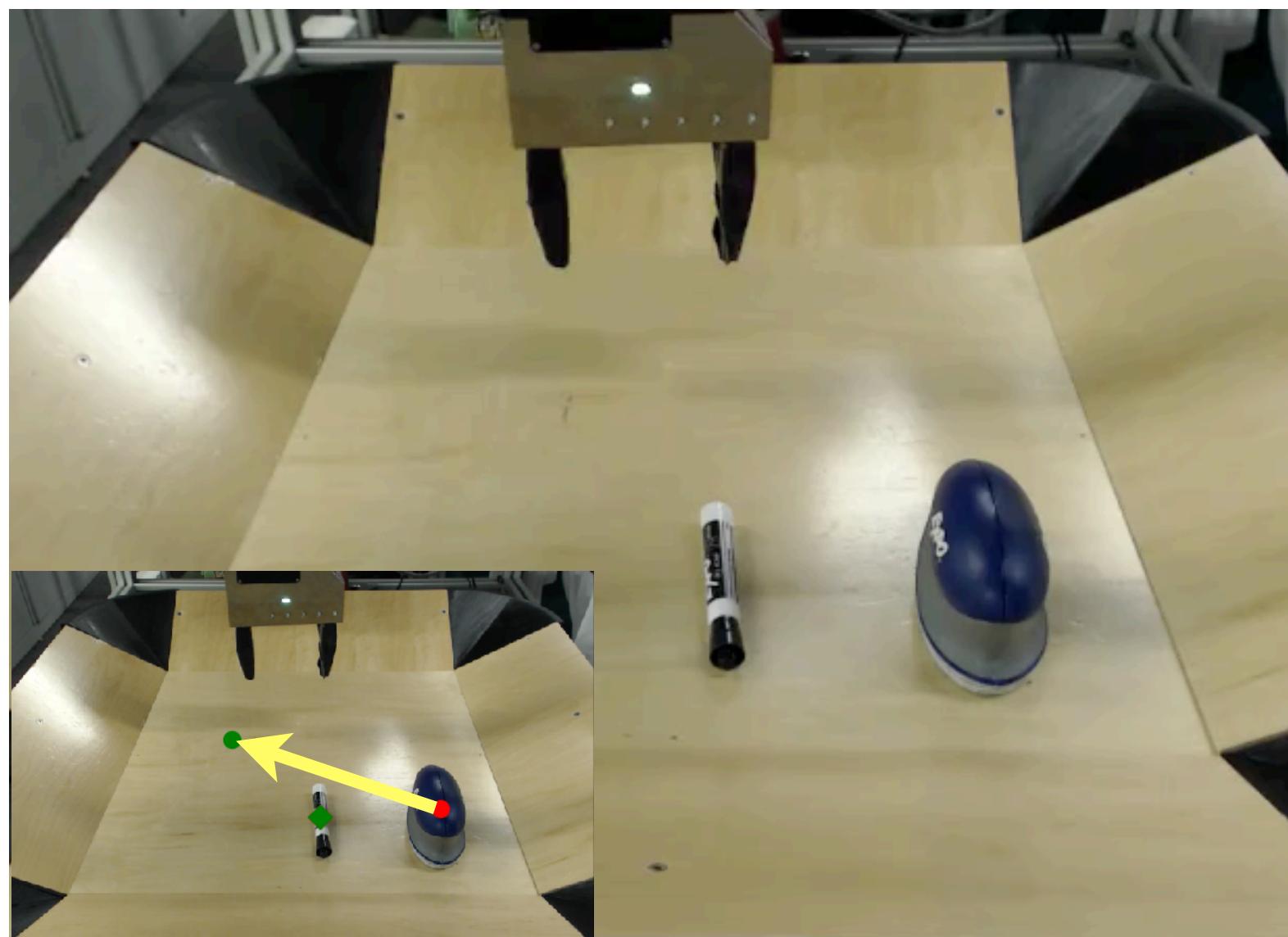
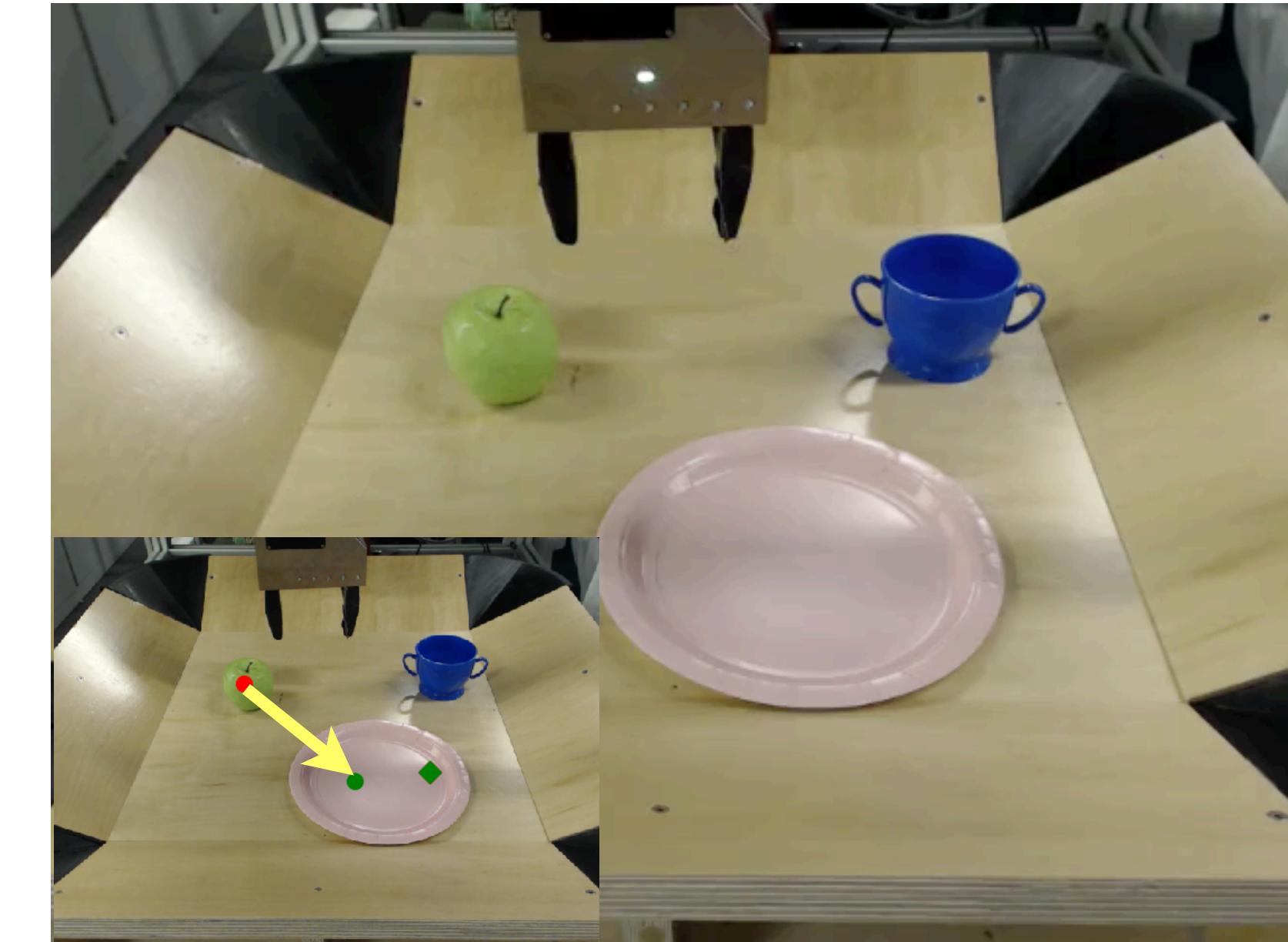


Visual MPC
w.r.t. goal



Planning with a single model for many tasks

Video speed: 2x



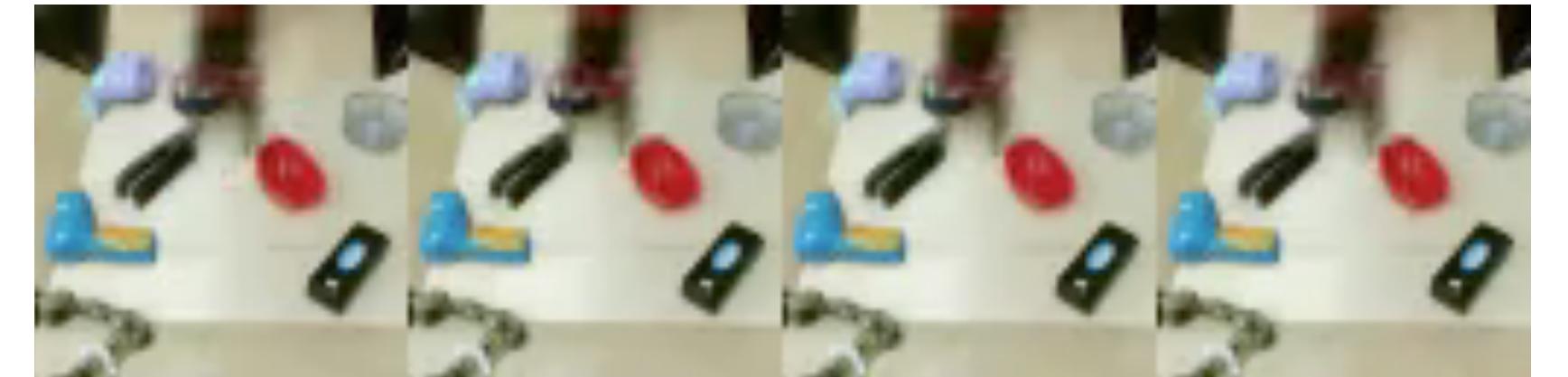
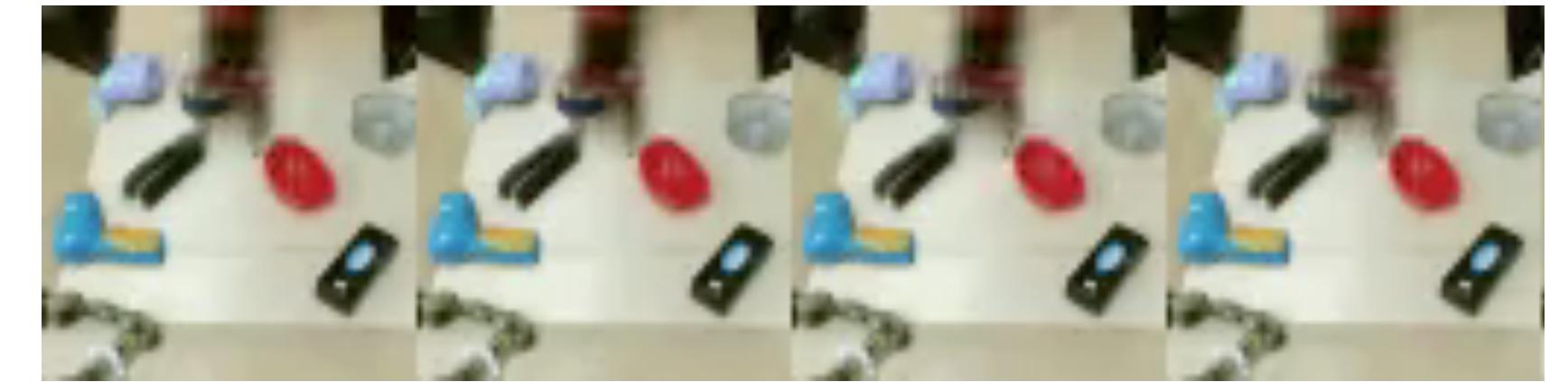
Modeling directly in observation space

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 = $p(\text{event})$)

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!

Next time

So far: multi-task RL
Next: meta RL

Reminders

Homework 3 out, due **Monday 10/26**.
Project milestone due **Monday 11/2**.