Advanced Model-Based Reinforcement Learning

CS 294-112: Deep Reinforcement Learning

Sergey Levine

Class Notes

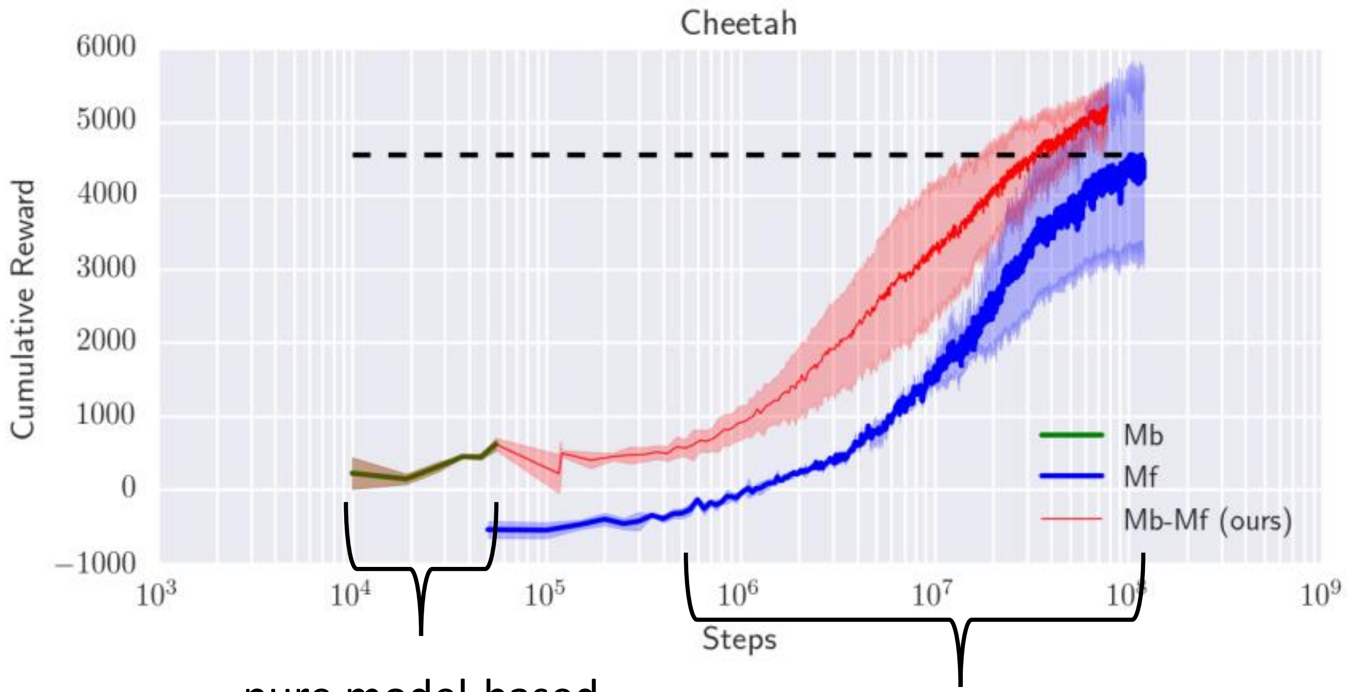
1. Homework 3 is extended by one week, to Wednesday after next

Today's Lecture

1. Managing overfitting in model-based RL
    - What's the problem?
    - How do we represent uncertainty?
2. Model-based RL with images
    - The POMDP model for model-based RL
    - Learning encodings
    - Learning dynamics-aware encoding

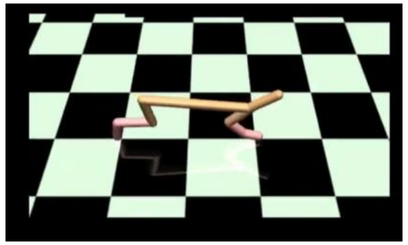
- Goals:
    - Understand the issue with overfitting and uncertainty in model-based RL
    - Understand how the POMDP model fits with model-based RL
    - Understand recent research on model-based RL with complex observations

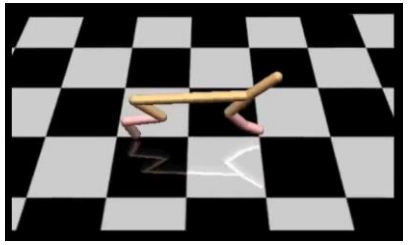
A performance gap in model-based RL



pure model-based
(about 10 minutes real time)

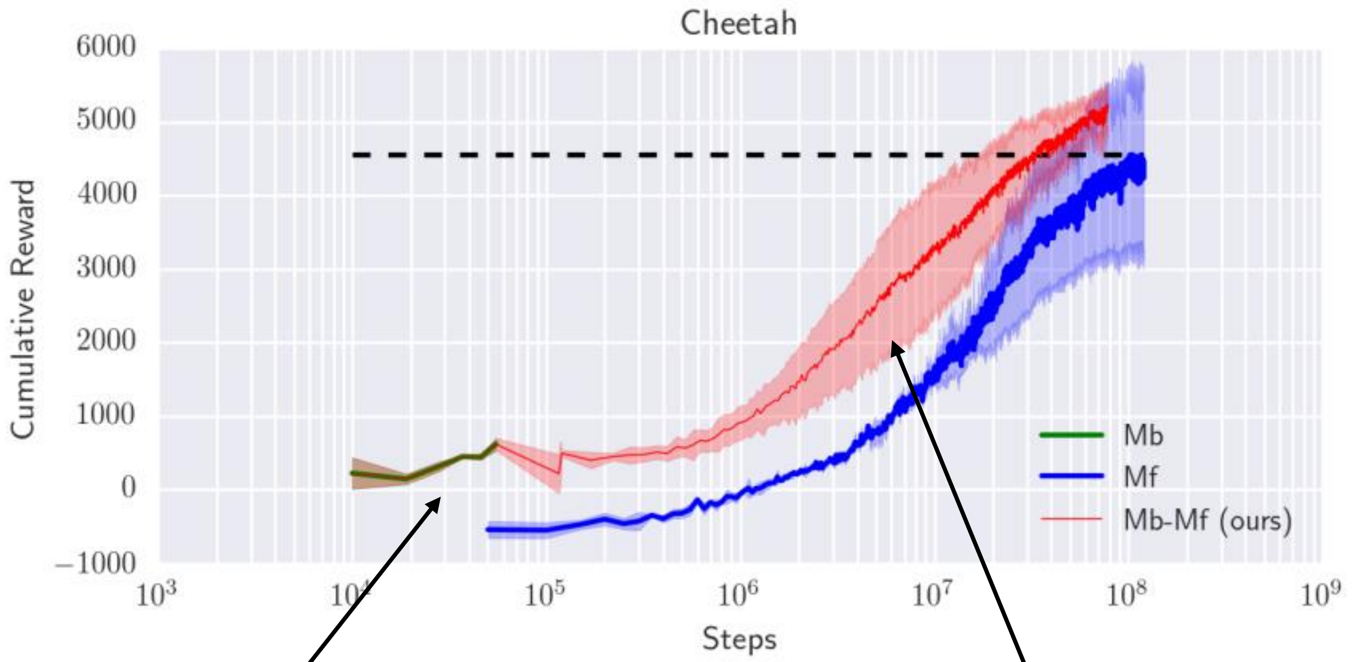
model-free training
(about 10 days…)





Nagabandi, Kahn, Fearing, L. ICRA 2018

Why the performance gap?



need to not overfit here…

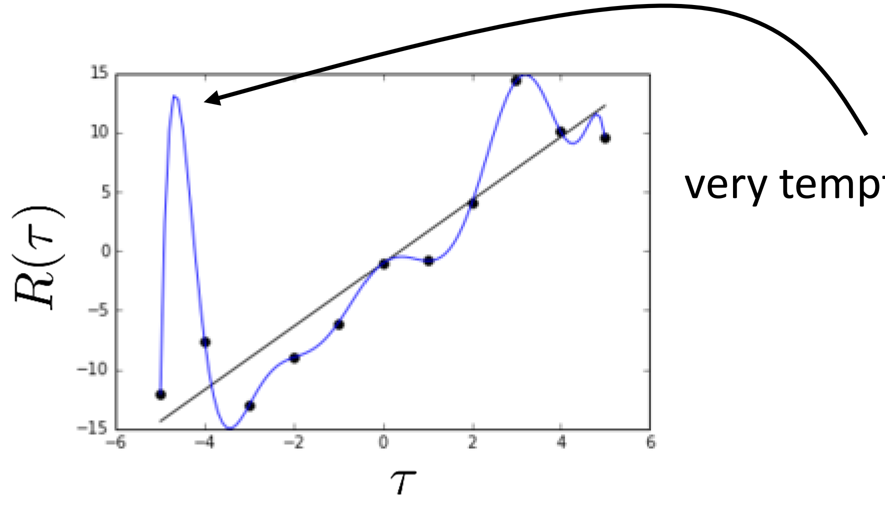
…but still have high capacity over here

Why the performance gap?

model-based reinforcement learning version 1.5:

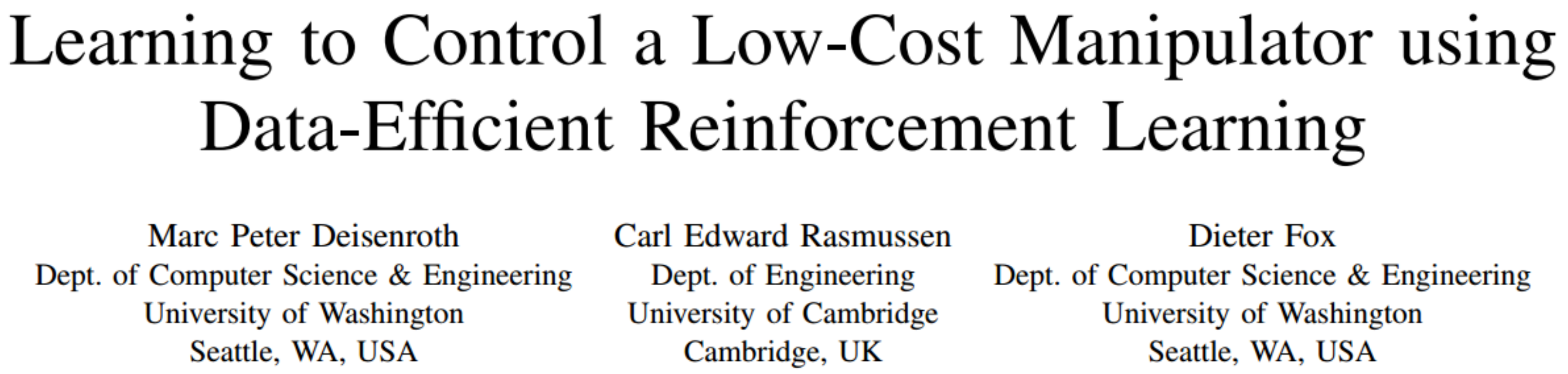
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s},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

every N steps



very tempting to go here…

Remember from last time...

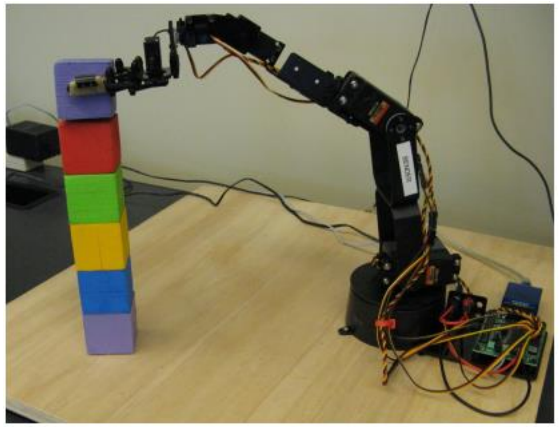


Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning

Marc Peter Deisenroth
Dept. of Computer Science & Engineering
University of Washington
Seattle, WA, USA

Carl Edward Rasmussen
Dept. of Engineering
University of Cambridge
Cambridge, UK

Dieter Fox
Dept. of Computer Science & Engineering
University of Washington
Seattle, WA, USA



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 GP dynamics model $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ to maximize $\sum_i \log p(\mathbf{s}'_i|\mathbf{s}_i, \mathbf{a}_i)$
3. backpropagate through $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ into the policy to optimize $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
4. run $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$, appending the visited tuples $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to $\mathcal{D}$

Remember from last time...

3. backpropagate through $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$ into the policy to optimize $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

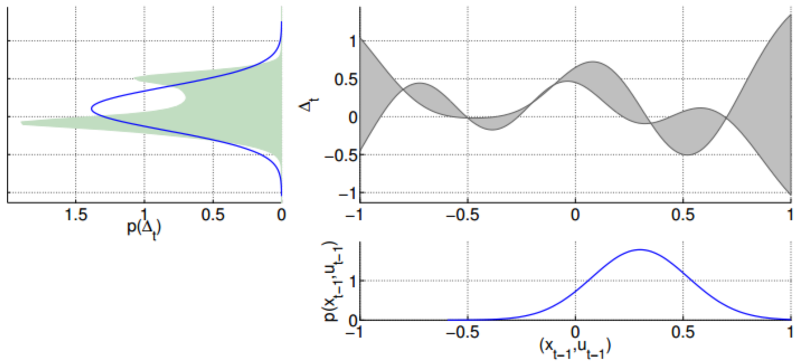
Given $p(\mathbf{s}_t)$, use $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$ to compute $p(\mathbf{s}_{t+1})$

If $p(\mathbf{s}_t)$ is Gaussian, we can get a (non-Gaussian) $\bar{p}(\mathbf{s}_{t+1})$ in closed form

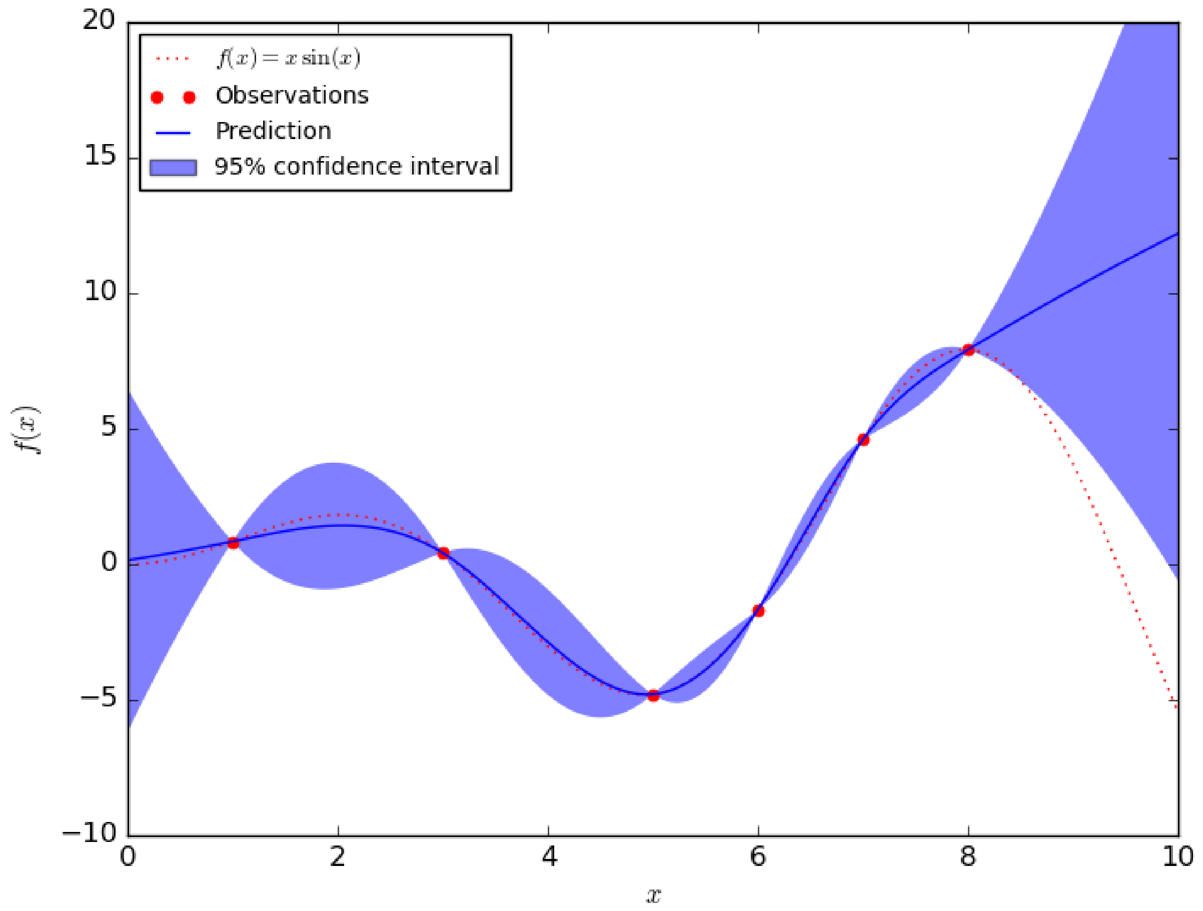
Project non-Gaussian $\bar{p}(\mathbf{s}_{t+1})$ to Gaussian $p(\mathbf{s}_{t+1})$ using moment matching

$E_{\mathbf{s}\sim p(\mathbf{s})}[r(\mathbf{s})]$ easy if r is nice and $p(\mathbf{s})$ Gaussian

Write $\sum_t E_{\mathbf{s}\sim p(\mathbf{s}_t)}[r(\mathbf{s}_t)]$ and differentiate



Why are GPs so popular for model-based RL?



expected reward under high-variance prediction
is **very** low, even though mean is the same!

Intuition behind uncertainty-aware RL

model-based reinforcement learning version 1.5:

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 dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute the first planned action, observe resulting state $\mathbf{s}'$ (MPC)
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$

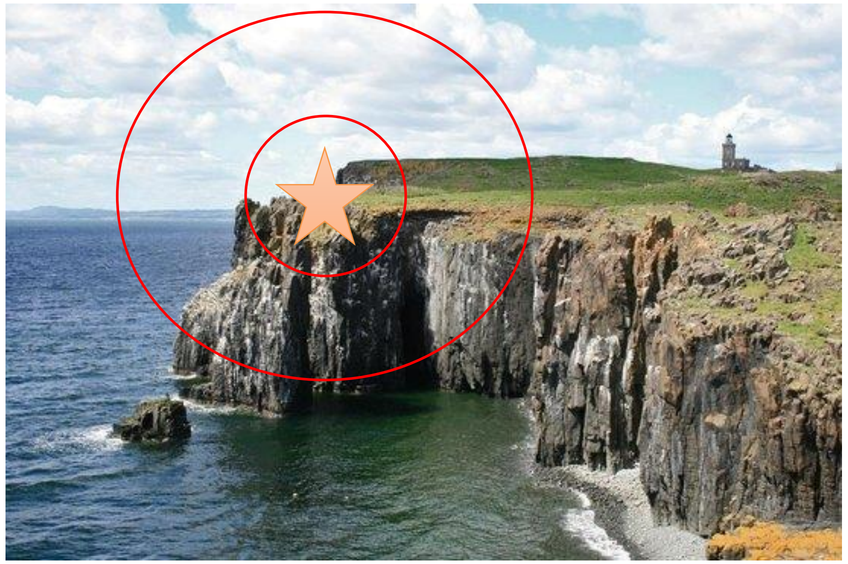
every N steps

only take actions for which we think we'll get high reward in expectation (w.r.t. uncertain dynamics)

This avoids "exploiting" the model

The model will then adapt and get better

There are a few caveats…



Need to explore to get better

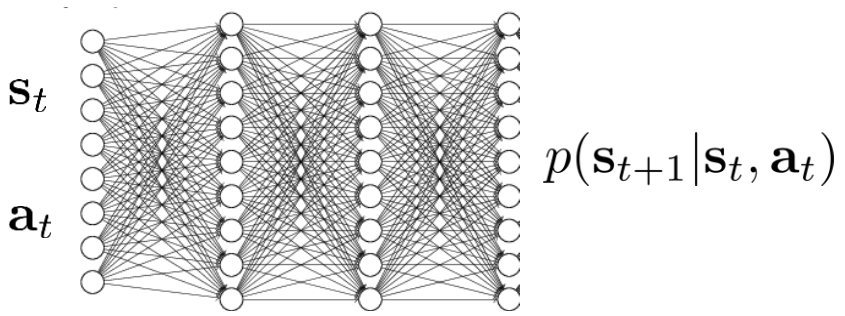
Expected value is not the same as pessimistic value

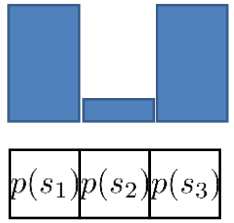
Expected value is not the same as optimistic value

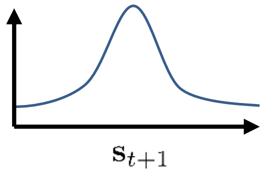
…but expected value is often a good start

How can we have uncertainty-aware models?

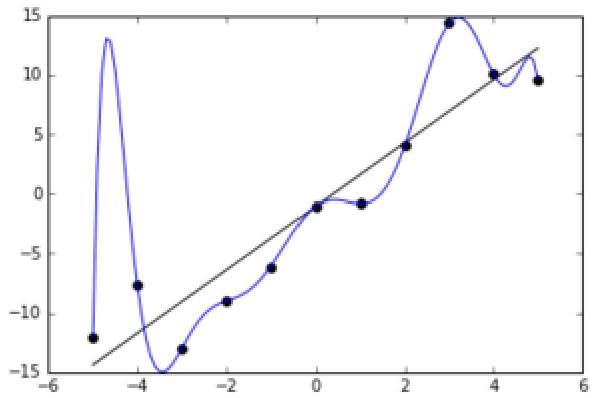
Idea 1: use output entropy





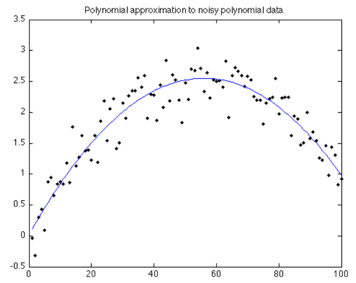


why is this not enough?



Two types of uncertainty:

aleatoric or *statistical* uncertainty $\longrightarrow$



$\longleftarrow$ *epistemic* or *model* uncertainty

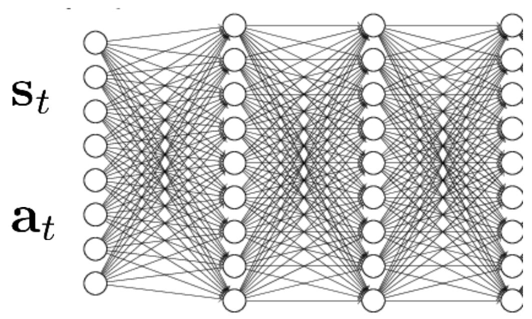
"the model is certain about the data, but we are not certain about the model"

what is the variance here?

How can we have uncertainty-aware models?

Idea 2: estimate mode uncertainty

"the model is certain about the data, but we are not certain about the model"



$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$

parameters θ

usually, we estimate

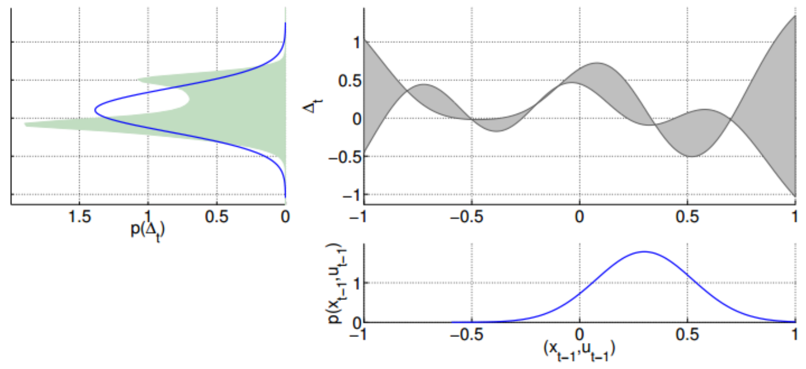
$$\arg\max_\theta \log p(\theta|\mathcal{D}) = \arg\max_\theta \log p(\mathcal{D}|\theta)$$

can we instead estimate $p(\theta|\mathcal{D})$?

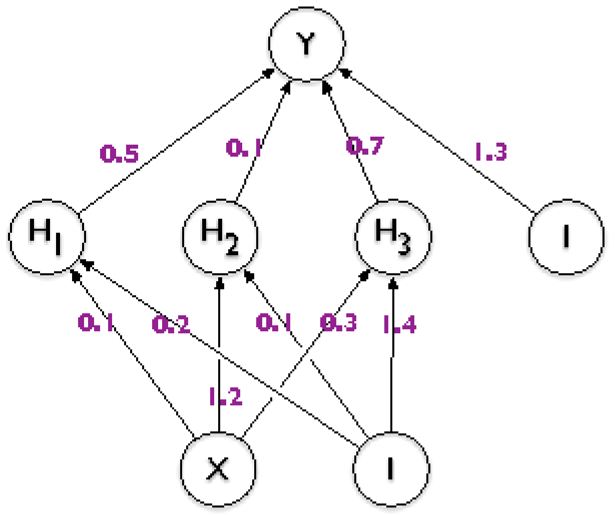
the entropy of this tells us the model uncertainty!

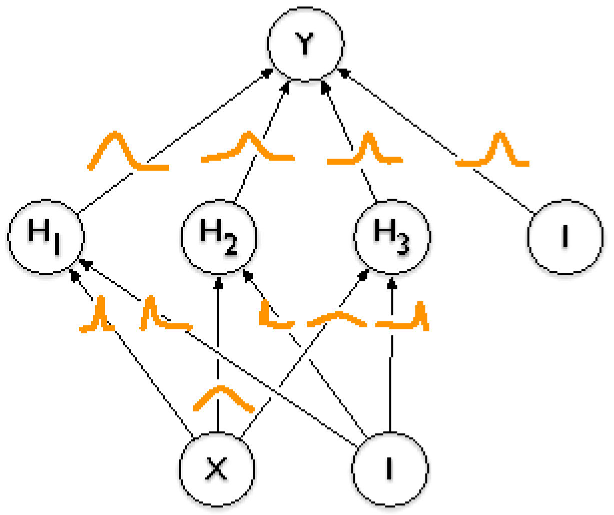
predict according to:

$$\int p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta) p(\theta|\mathcal{D}) d\theta$$



Quick overview of Bayesian neural networks





common approximation:

$$p(\theta|\mathcal{D}) = \prod_i p(\theta_i|\mathcal{D})$$

$$p(\theta_i|\mathcal{D}) = \mathcal{N}(\mu_i, \sigma_i)$$

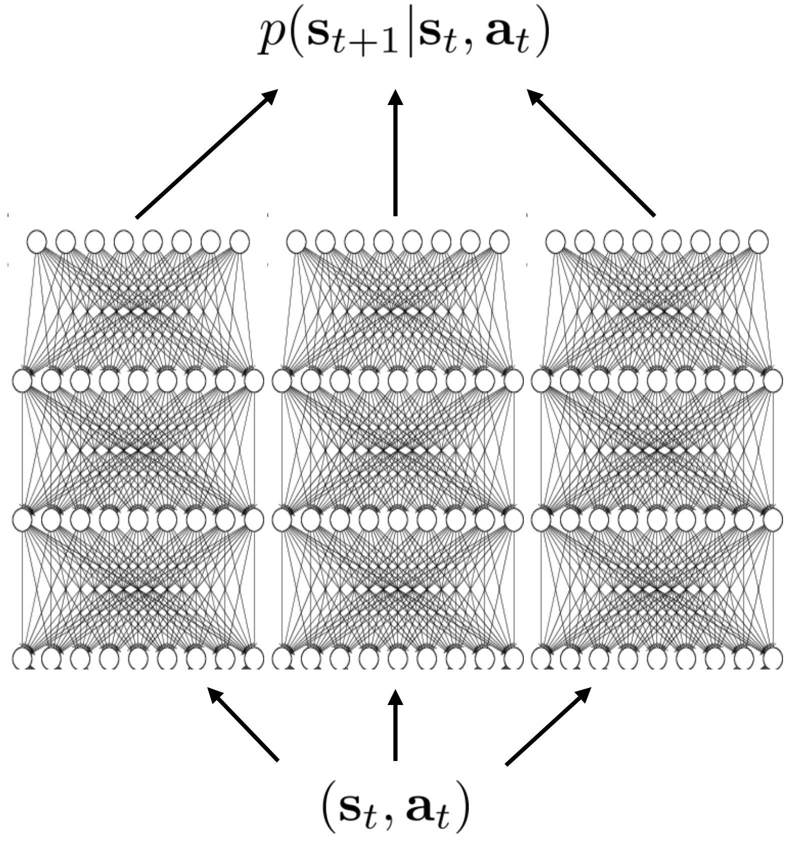
expected weight

uncertainty
about the weight

For more, see:
Blundell et al., Weight Uncertainty in Neural Networks
Gal et al., Concrete Dropout

We'll learn more about variational inference later!

Bootstrap ensembles


$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
$(\mathbf{s}_t, \mathbf{a}_t)$

Train multiple models and see if they agree!

$$\text{formally:} \quad p(\theta|\mathcal{D}) \approx \frac{1}{N}\sum_i \delta(\theta_i)$$

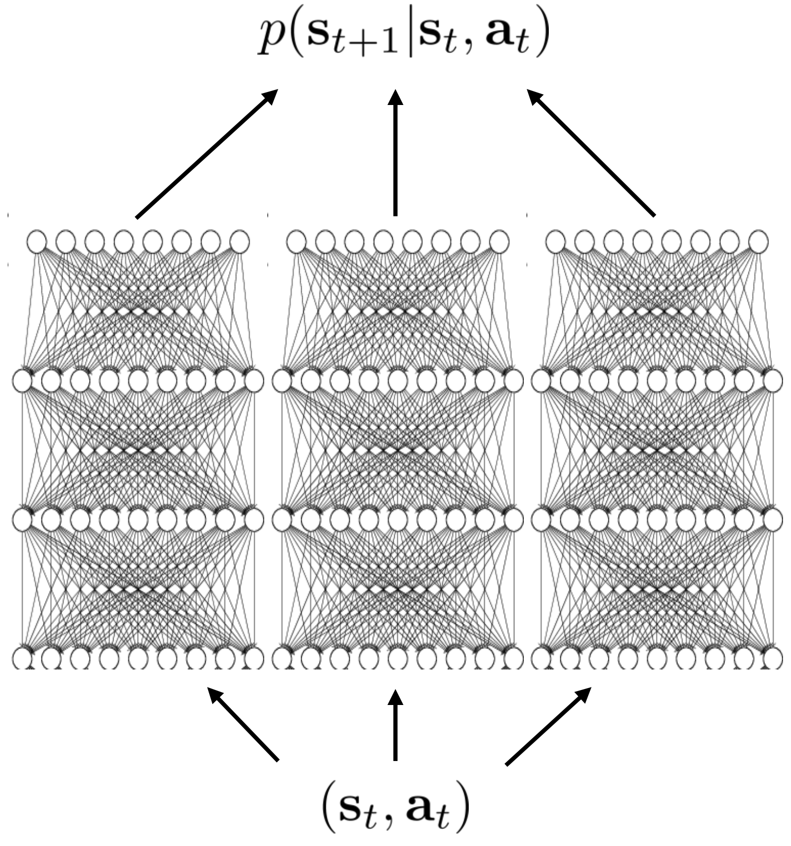
$$\int p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta)p(\theta|\mathcal{D})d\theta \approx \frac{1}{N}\sum_i p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

How to train?

Main idea: need to generate "independent" datasets to get "independent" models

θ_i is trained on $\mathcal{D}_i$, sampled *with replacement* from $\mathcal{D}$

Bootstrap ensembles in deep learning


$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
$(\mathbf{s}_t, \mathbf{a}_t)$

This basically works

Very crude approximation, because the number of models is usually small (< 10)

Resampling with replacement is usually unnecessary, because SGD and random initialization usually makes the models sufficiently independent

How to plan with uncertainty

Before: $J(\mathbf{a}_1, \ldots, \mathbf{a}_H) = \sum_{t=1}^{H} r(\mathbf{s}_t, \mathbf{a}_t)$, where $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$

Now: $J(\mathbf{a}_1, \ldots, \mathbf{a}_H) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{H} r(\mathbf{s}_{t,i}, \mathbf{a}_t)$, where $\mathbf{s}_{t+1,i} = f_i(\mathbf{s}_{t,i}, \mathbf{a}_t)$

distribution over deterministic models

In general, for candidate action sequence $\mathbf{a}_1, \ldots, \mathbf{a}_H$:

Step 1: sample $\theta \sim p(\theta|\mathcal{D})$

Step 2: at each time step t, sample $\mathbf{s}_{t+1} \sim p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$

Step 3: calculate $R = \sum_t r(\mathbf{s}_t, \mathbf{a}_t)$

Step 4: repeat steps 1 to 3 and accumulate the average reward

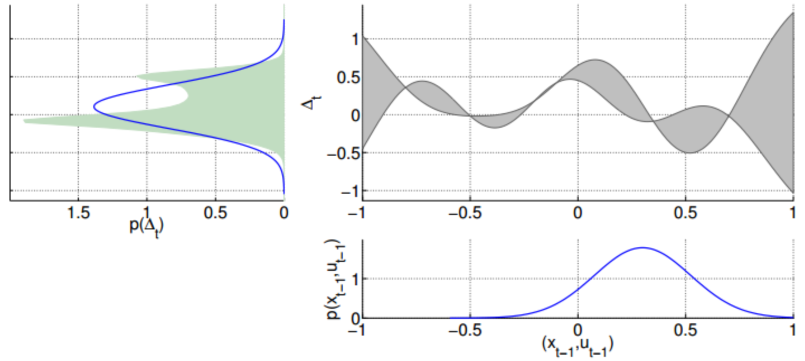
Slightly more complex option: moment matching

Given $p(\mathbf{s}_t)$, use $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$ to estimate $p(\mathbf{s}_{t+1})$

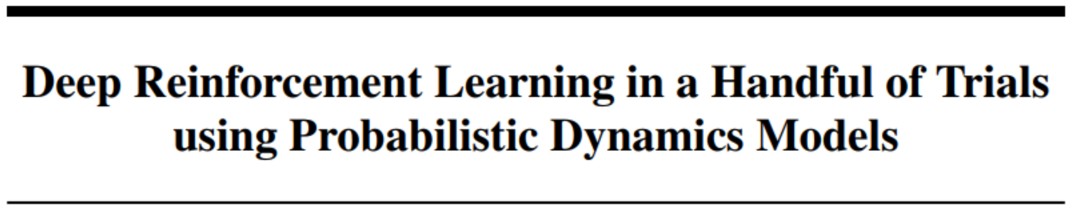
We can get $\bar{p}(\mathbf{s}_{t+1})$, potentially represented with samples

Project non-Gaussian $\bar{p}(\mathbf{s}_{t+1})$ to Gaussian $p(\mathbf{s}_{t+1})$ using moment matching

Write $\sum_t E_{\mathbf{s}\sim p(\mathbf{s}_t)}[r(\mathbf{s}_t)]$ and differentiate

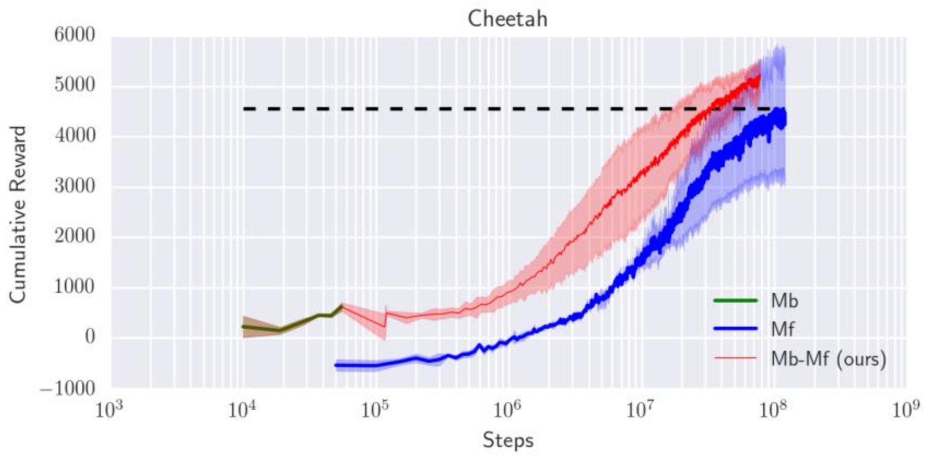


Example: model-based RL with ensembles

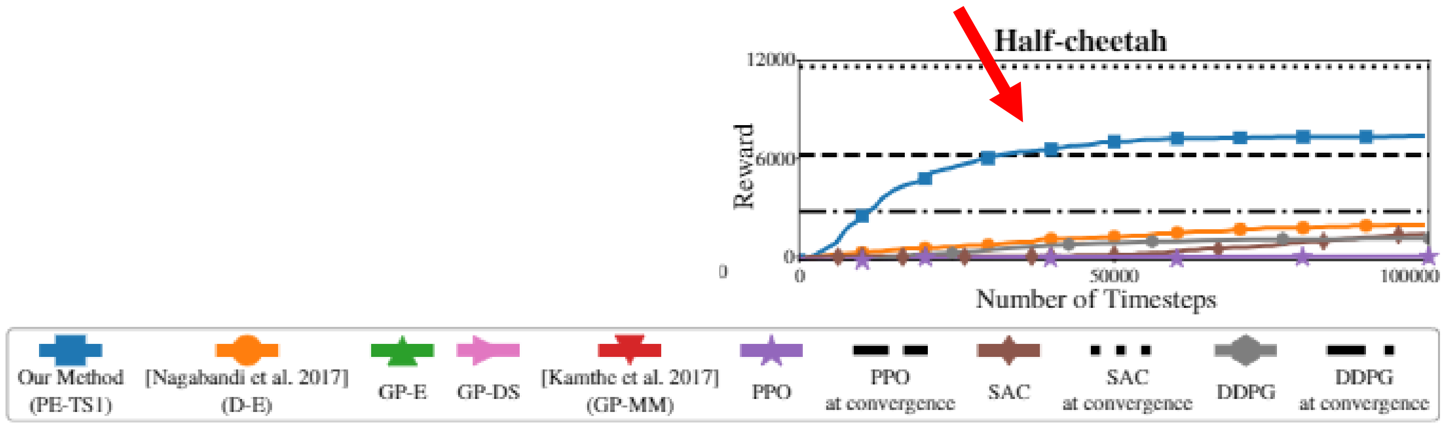


Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models

exceeds performance of model-free after 40k steps
(about 10 minutes of real time)



before



after

Further readings

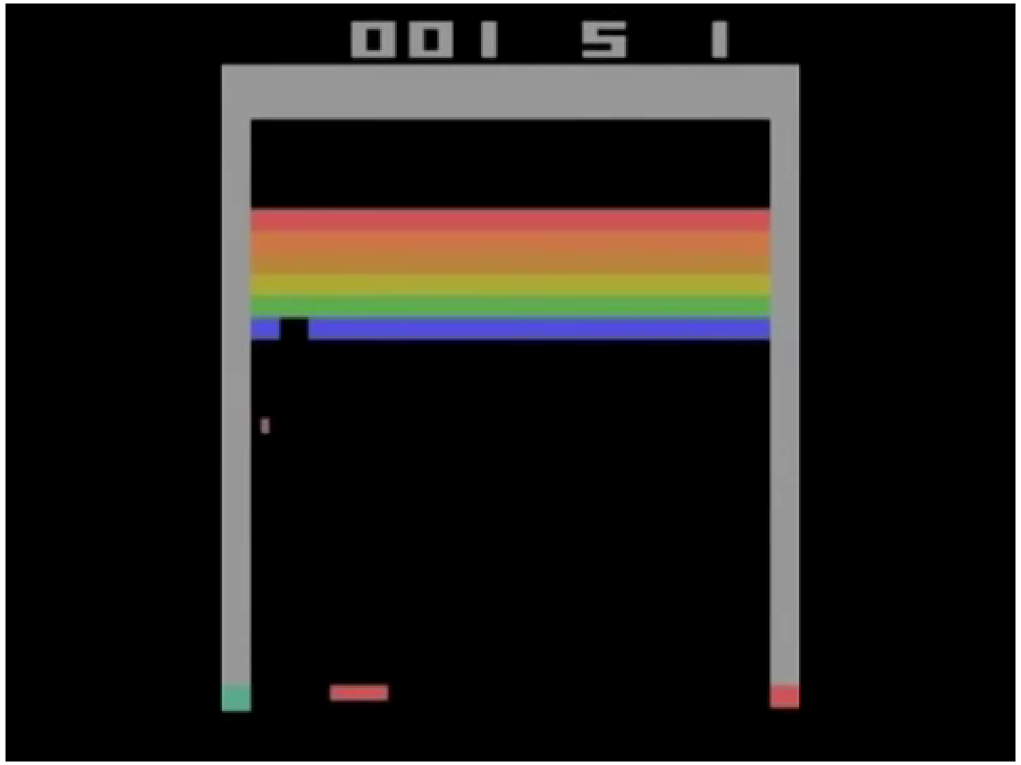
- Deisenroth et al. PILCO: A Model-Based and Data-Efficient Approach to Policy Search.

Recent papers:

- Chua et al. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models.
- Feinberg et al. Model-Based Value Expansion for Efficient Model-Free Reinforcement Learning.
- Buckman et al. Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion.

Break

Previously: model-free RL with images



This lecture: Can we use model-based methods with images?

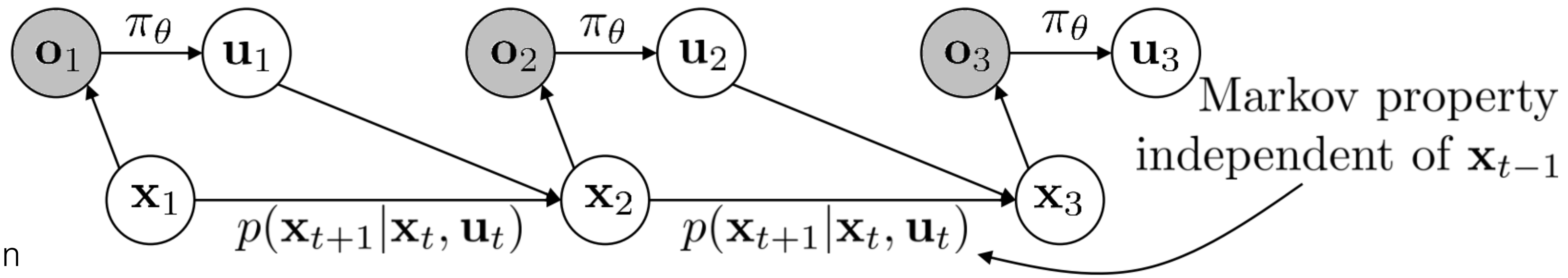
slides from C. Finn

Recap: Model-based RL

model-based reinforcement learning version 1.0:

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 dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute those actions and add the resulting data $\{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_j\}$ to $\mathcal{D}$

What about POMDPs?

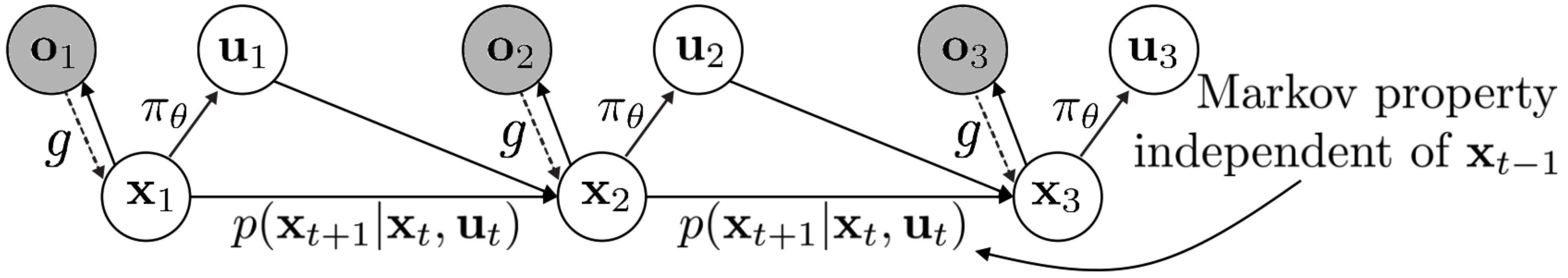


slides from C. Finn

Learning in Latent Space

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

(model-based or model-free)



What do we want g to be?

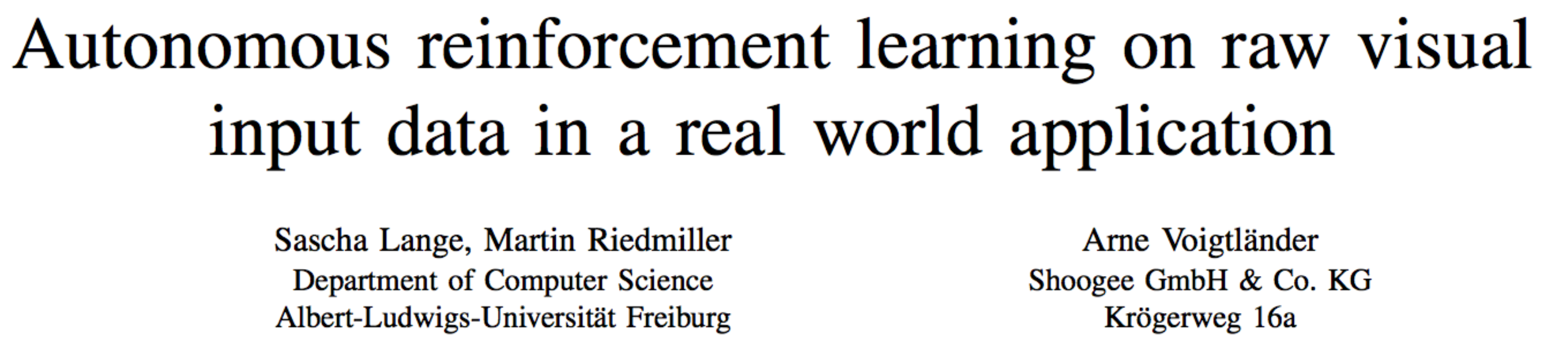
It depends on the method — we'll see.

slides from C. Finn

Learning in Latent Space

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

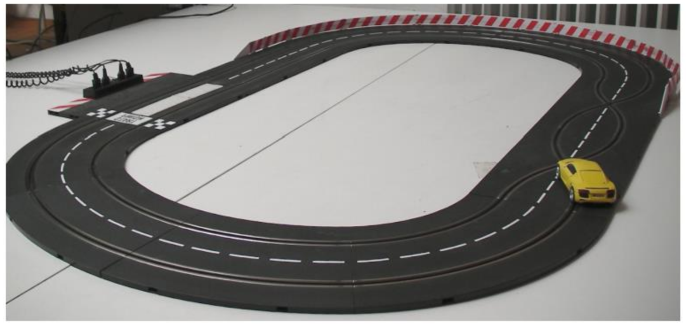
(model-based or **model-free**)

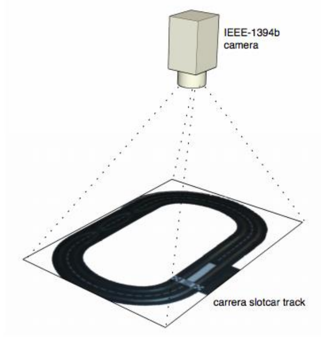


Autonomous reinforcement learning on raw visual input data in a real world application

Sascha Lange, Martin Riedmiller
Department of Computer Science
Albert-Ludwigs-Universität Freiburg

Arne Voigtländer
Shoogee GmbH & Co. KG
Krögerweg 16a

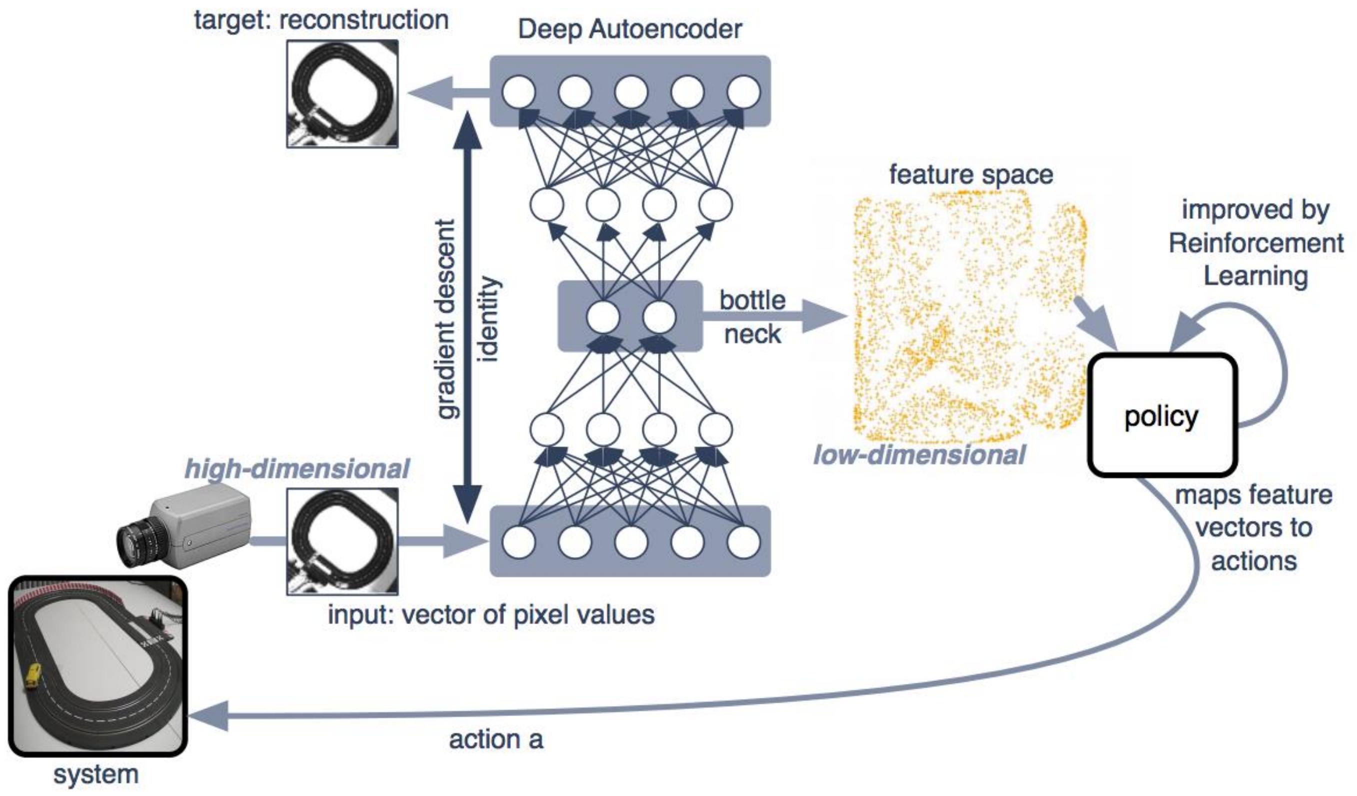




controlling a slot-car

slides from C. Finn

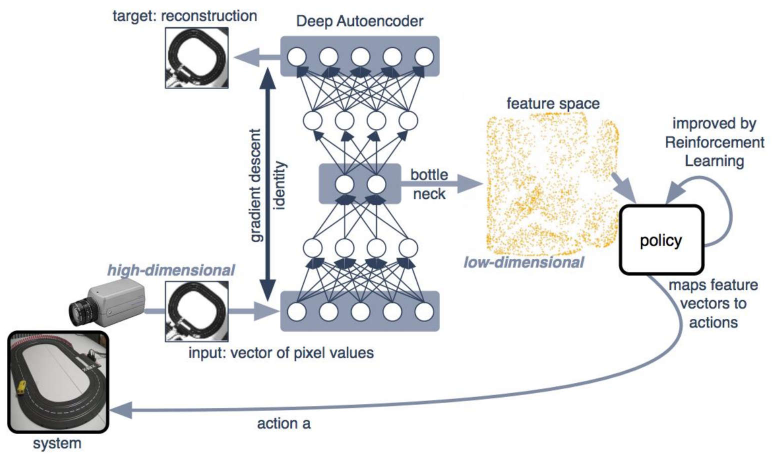
1. collect data with exploratory policy
2. learn **low-dimensional** embedding of image (how?)
3. run q-learning with function approximation with embedding



embedding is **low-dimensional** and summarizes the image

slides from C. Finn

1. collect data with exploratory policy
2. learn **low-dimensional** embedding of image (how?)
3. run q-learning with function approximation with embedding



Pros:

+ Learn visual skill very efficiently

Cons:

- Autoencoder might not recover the right representation
- Not necessarily suitable for model-based methods

slides from C. Finn

Learning in Latent Space

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

(**model-based** or model-free)

Deep Spatial Autoencoders for Visuomotor Learning

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

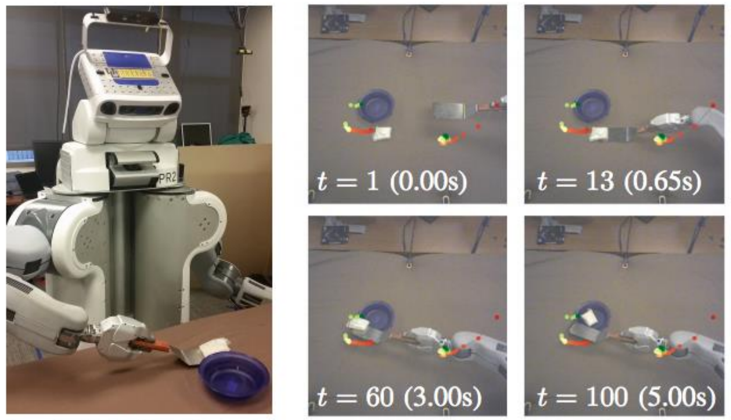
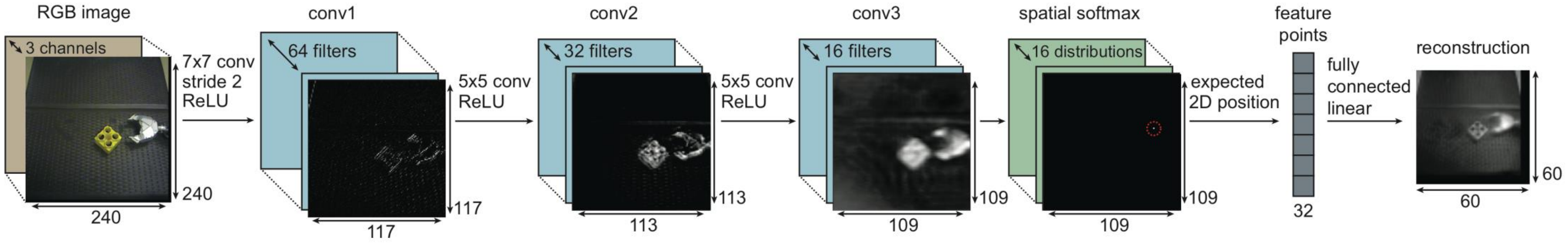


Fig. 1: PR2 learning to scoop a bag of rice into a bowl with a spatula (left) using a learned visual state representation (right).

slides from C. Finn

1. collect data with exploratory policy
2. learn **smooth, structured** embedding of image
3. learn local-linear model with embedding
4. run iLQG with local models to learn to reach goal image

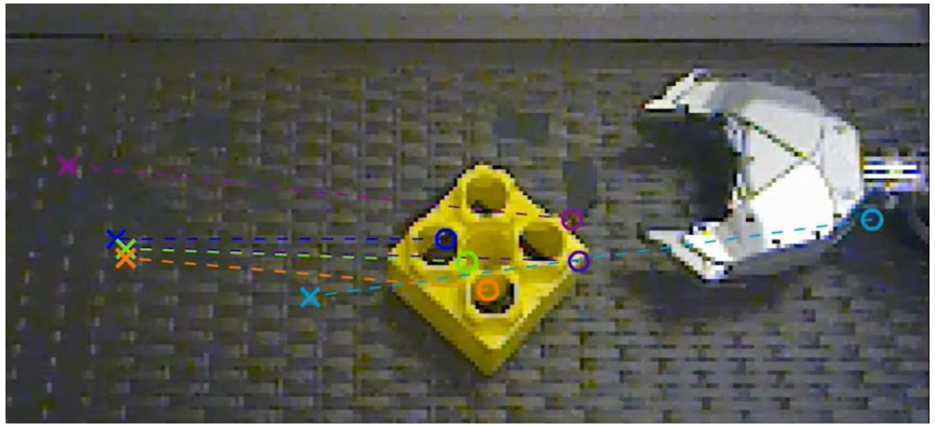


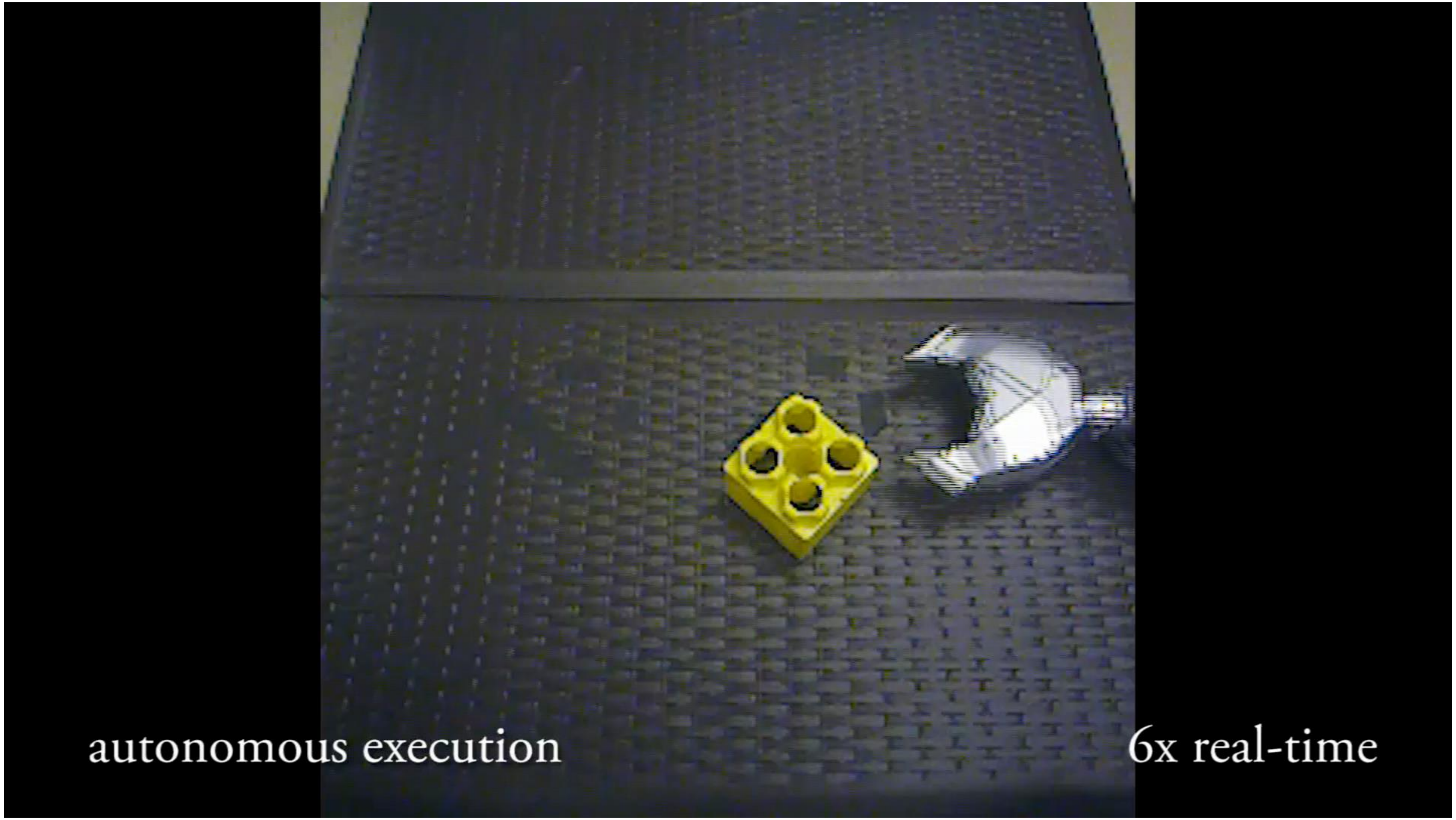
embedding is **smooth and structured**

slides from C. Finn

1. collect data with exploratory policy
2. learn **smooth, structured** embedding of image
3. learn local-linear model with embedding
4. run iLQG with local models to learn to reach goal image

Because we aren't using states, we need a reward.



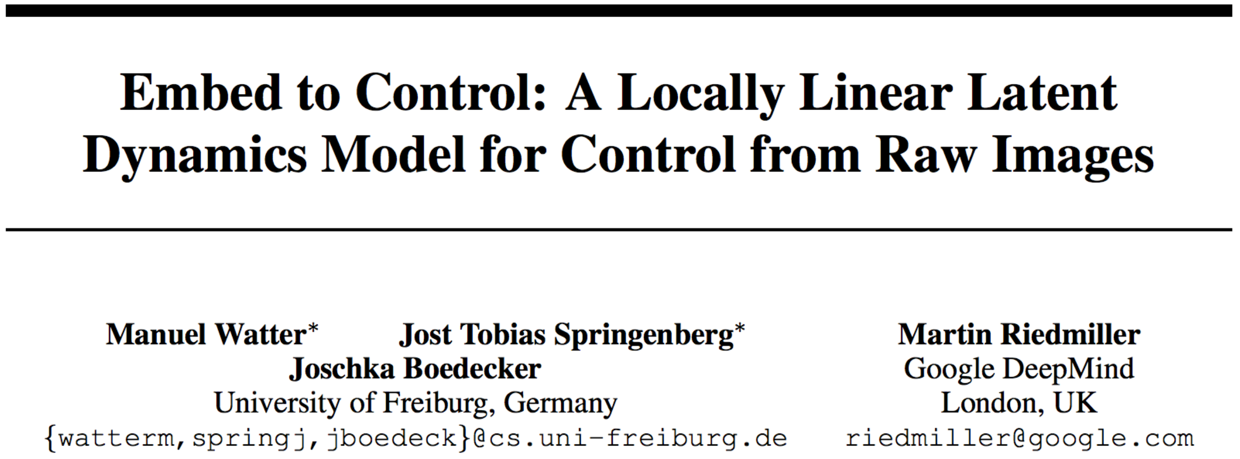


slides from C. Finn

Learning in Latent Space

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

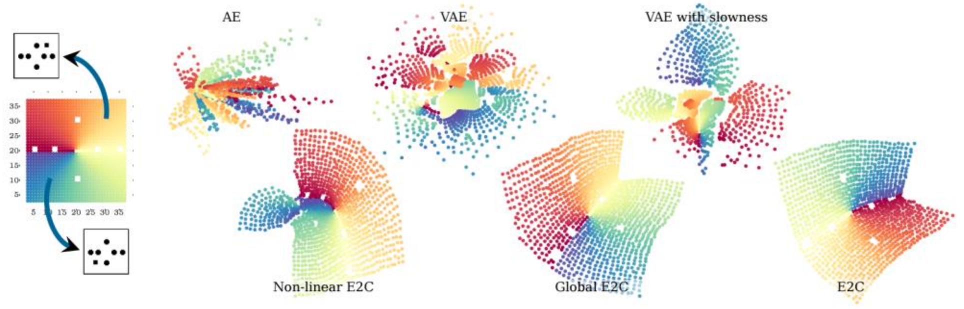
(**model-based** or model-free)



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

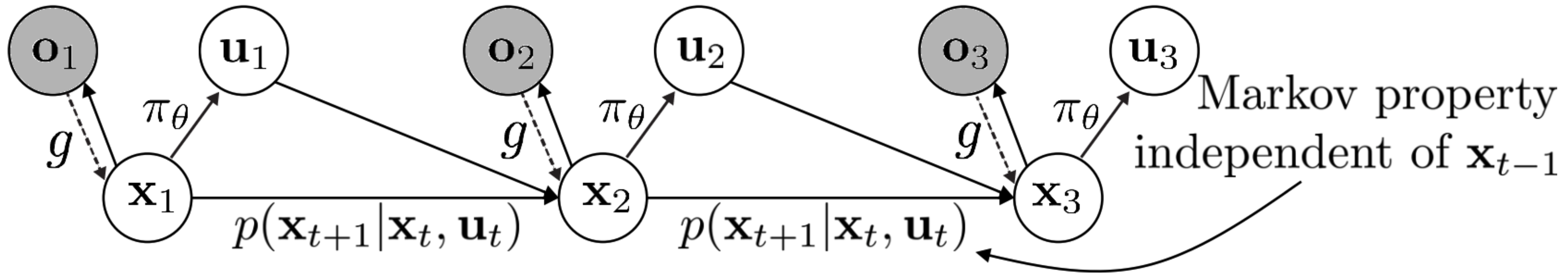
Manuel Watter* **Jost Tobias Springenberg*** **Joschka Boedecker**
University of Freiburg, Germany
{watterm, springj, jboedeck}@cs.uni-freiburg.de

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



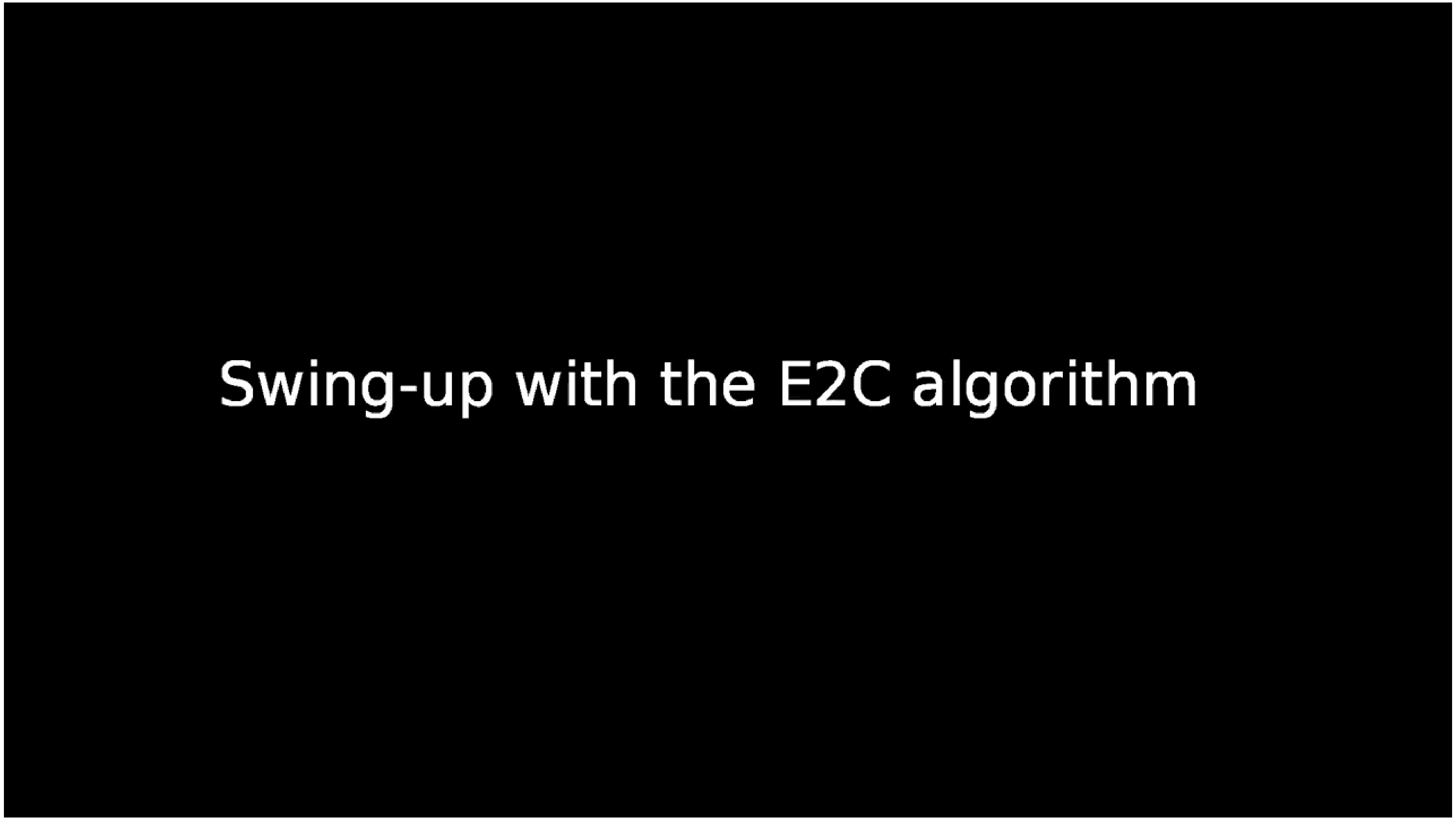
slides from C. Finn

1. collect data
2. learn embedding of image & dynamics model (**jointly**)
3. run iLQG to learn to reach image of goal



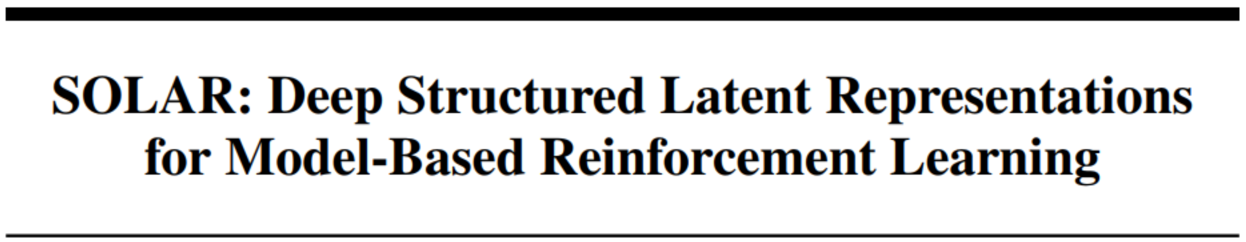
embedding that can be **modeled**

slides from C. Finn

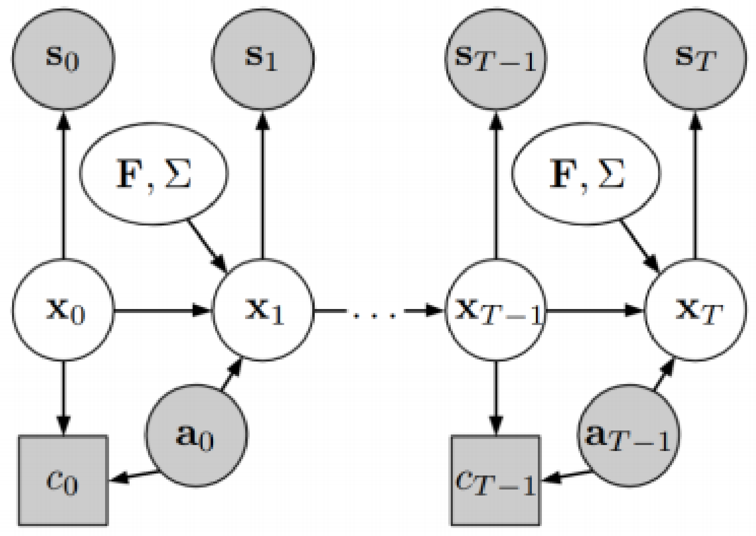


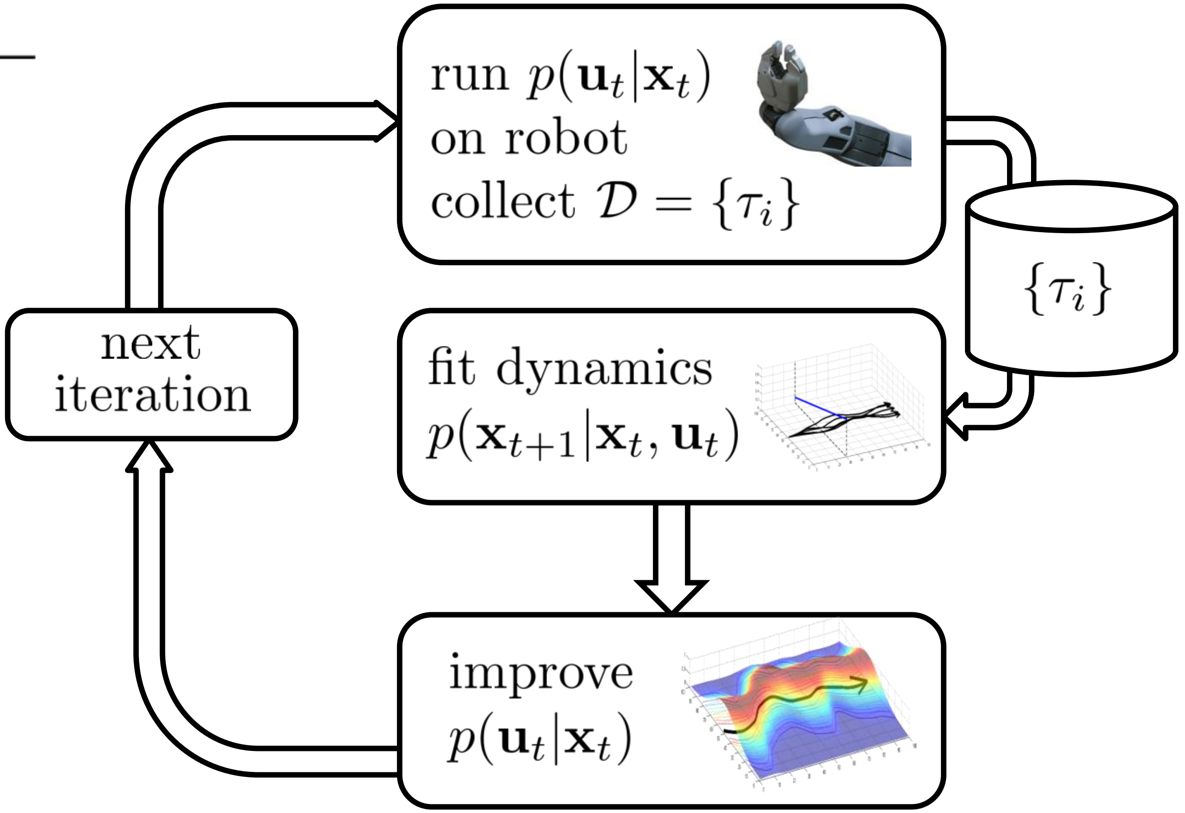
Swing-up with the E2C algorithm

Local models with images



SOLAR: Deep Structured Latent Representations for Model-Based Reinforcement Learning



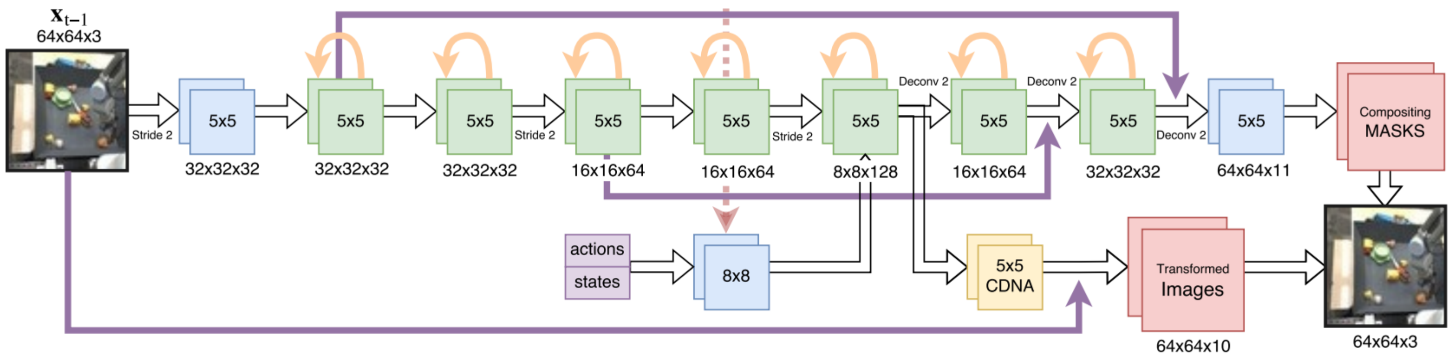


run $p(\mathbf{u}_t|\mathbf{x}_t)$ on robot collect $\mathcal{D} = \{\tau_i\}$
$\{\tau_i\}$
fit dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$
improve $p(\mathbf{u}_t|\mathbf{x}_t)$
next iteration

Learn directly in observation space

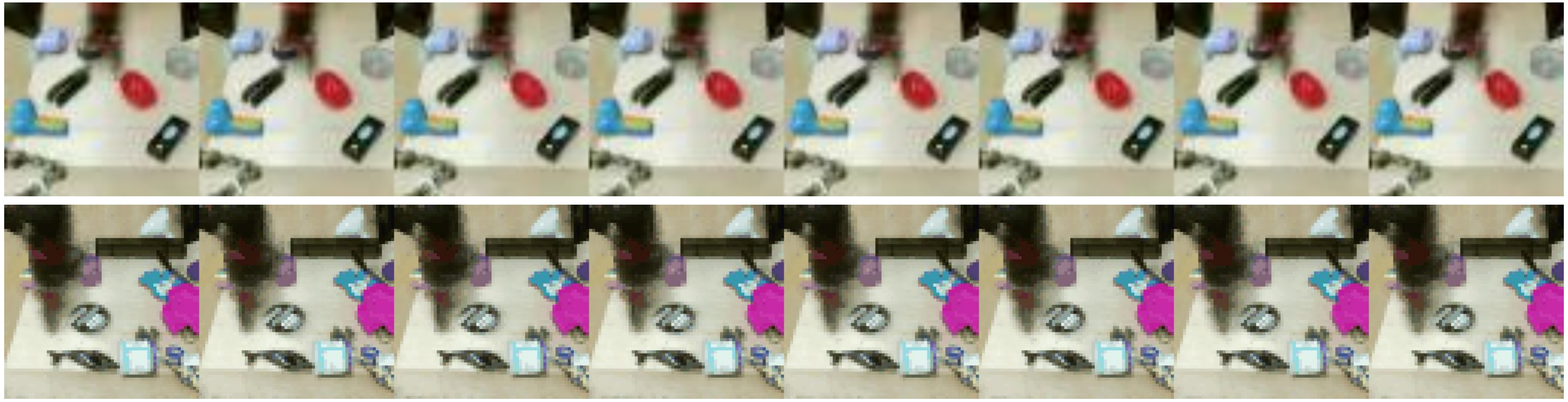
~~**Key idea**: learn embedding $g(\mathbf{o}_t) = \mathbf{x}_t$~~

directly learn $p(\mathbf{o}_{t+1} | \mathbf{o}_t, \mathbf{a}_t)$

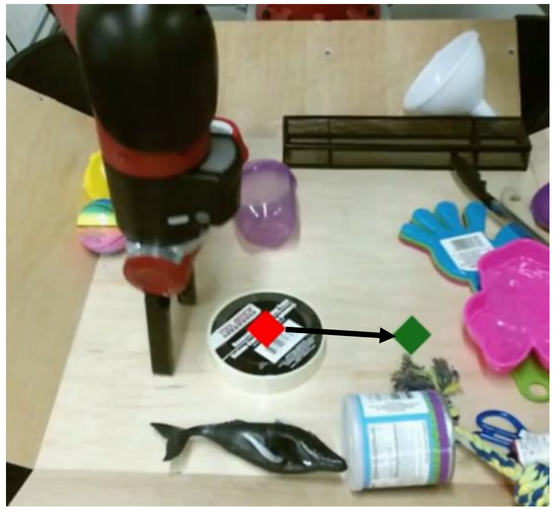


Finn, L. **Deep Visual Foresight for Planning Robot Motion.** ICRA 2017.

Ebert, Finn, Lee, L. **Self-Supervised Visual Planning with Temporal Skip Connections.** CoRL 2017.



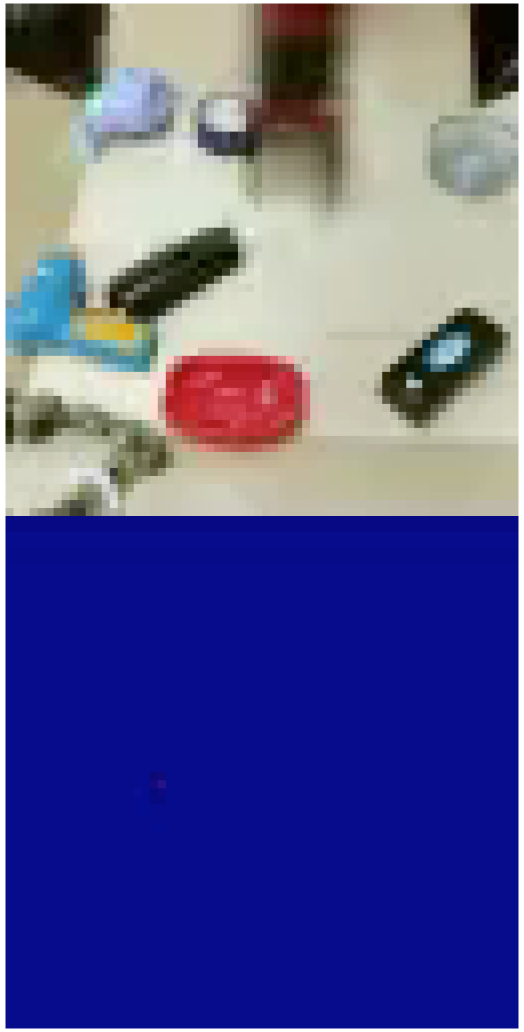
Use predictions to complete tasks

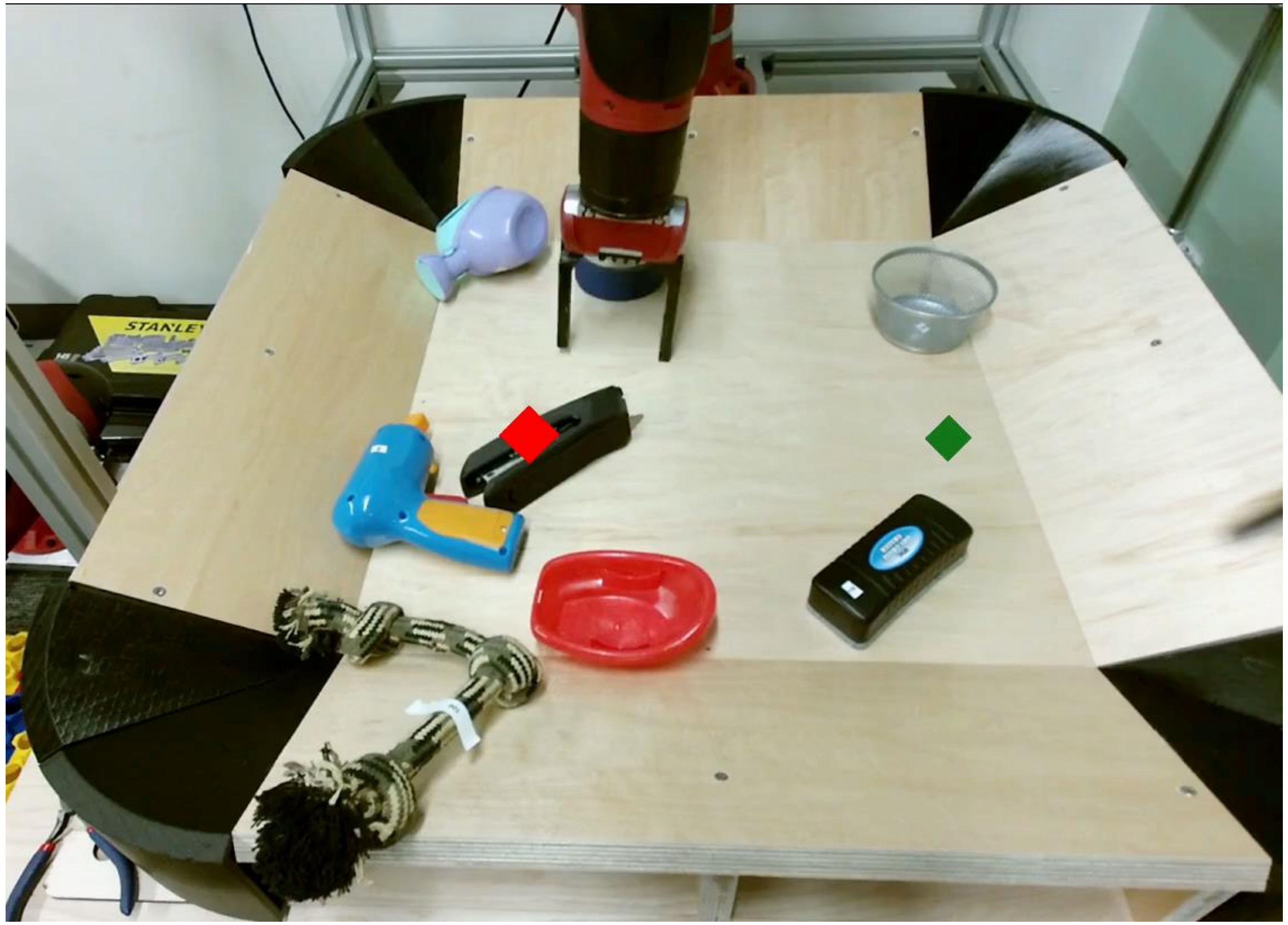


Designated Pixel ♦

Goal Pixel ♦

Task execution





Predict alternative quantities

If I take a set of actions:

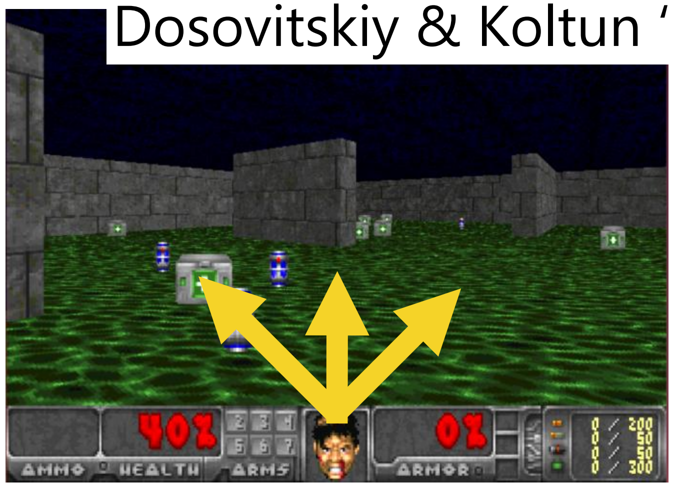
Pinto et al. '16

Will I successfully grasp?

Will I collide?

Kahn et al. '17

Dosovitskiy & Koltun '17



What will health/damage/etc. be?

Pros:

+ Only predict task-relevant quantities!

Cons:

- Need to manually pick quantities, must be able to directly observe them