

# Model-Based [Deep] Reinforcement Learning

Chelsea Finn

Deep RL Bootcamp



BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

# Outline

1. Why use model-based reinforcement learning?
2. Main model-based RL approaches
3. Using local models & guided policy search
4. Handling high-dimensional observations

# Outline

- 1. Why use model-based reinforcement learning?**
2. Main model-based RL approaches
3. Using local models & guided policy search
4. Handling high-dimensional observations

Why use model-based reinforcement learning?

- **sample efficiency**

gradient-free methods  
(e.g. NES, CMA, etc.)



fully online methods  
(e.g. A3C)



policy gradient methods  
(e.g. TRPO)



replay buffer value estimation methods  
(Q-learning, DDPG, NAF, etc.)



model-based deep RL  
(e.g. guided policy search)

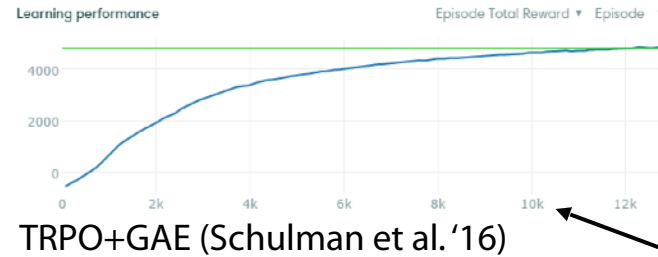


model-based "shallow" RL  
(e.g. PILCO)

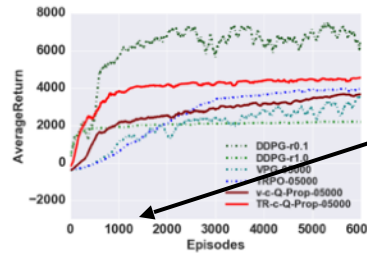
### Evolution Strategies as a Scalable Alternative to Reinforcement Learning

Tim Salimans<sup>1</sup> Jonathan Ho<sup>1</sup> Xi Chen<sup>1</sup> Ilya Sutskever<sup>1</sup>

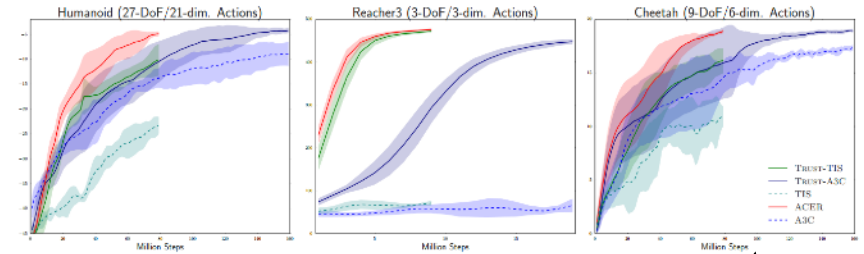
half-cheetah (slightly different version)



half-cheetah



	cart-pole	cart-double-pole	unicycle
state space	$\mathbb{R}^4$	$\mathbb{R}^6$	$\mathbb{R}^{12}$
# trials	$\leq 10$	20-30	$\approx 20$
experience	$\approx 20$ s	$\approx 60$ s-90 s	$\approx 20$ s-30 s
parameter space	$\mathbb{R}^{305}$	$\mathbb{R}^{1816}$	$\mathbb{R}^{28}$

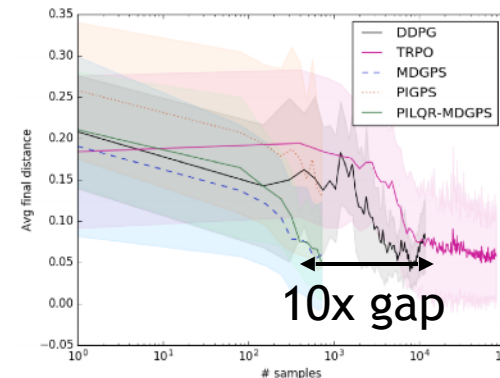
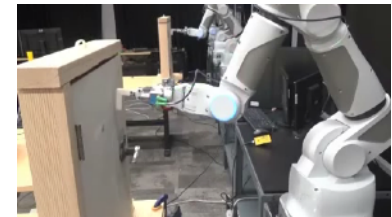


Wang et al. '17

10,000,000 steps  
(10,000 episodes)  
(~ 1.5 days real time)

1,000,000 steps  
(1,000 episodes)  
(~ 3 hours real time)

100,000,000 steps  
(100,000 episodes)  
(~ 15 days real time)



about 20 minutes of  
experience on a real  
robot

# Why use model-based reinforcement learning?

- sample efficiency
- transferability & generality

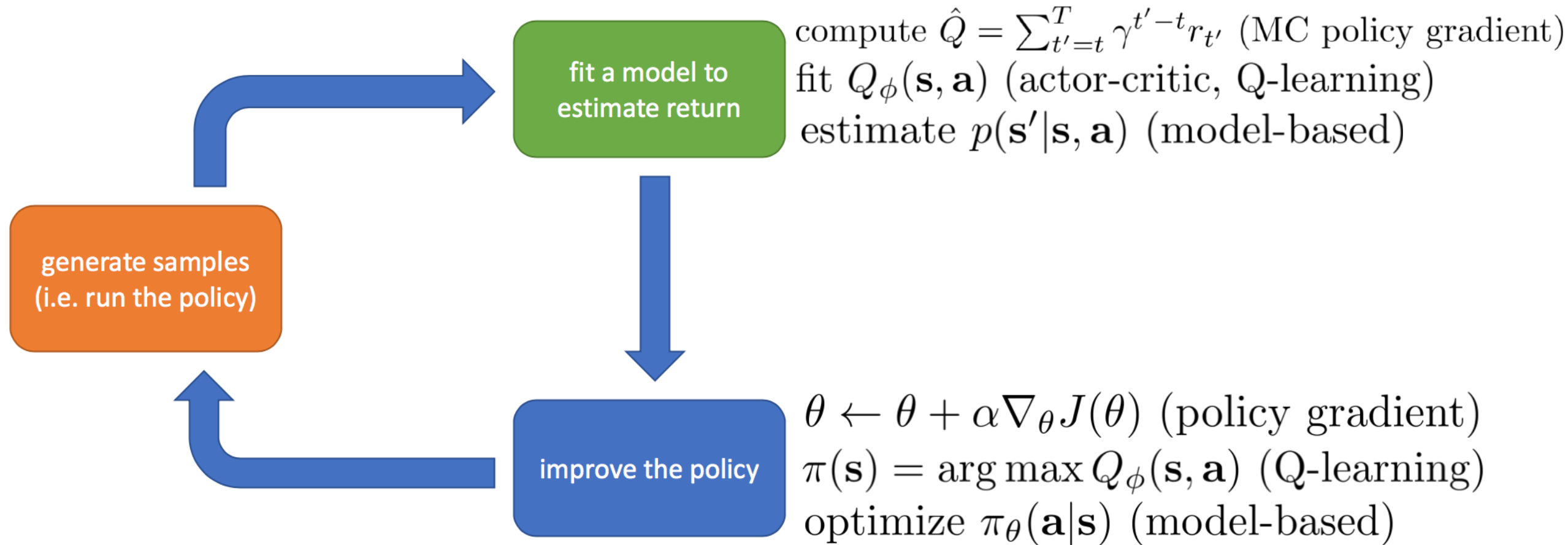
*A model can be reused for achieving different tasks.*

[more examples later]

# Outline

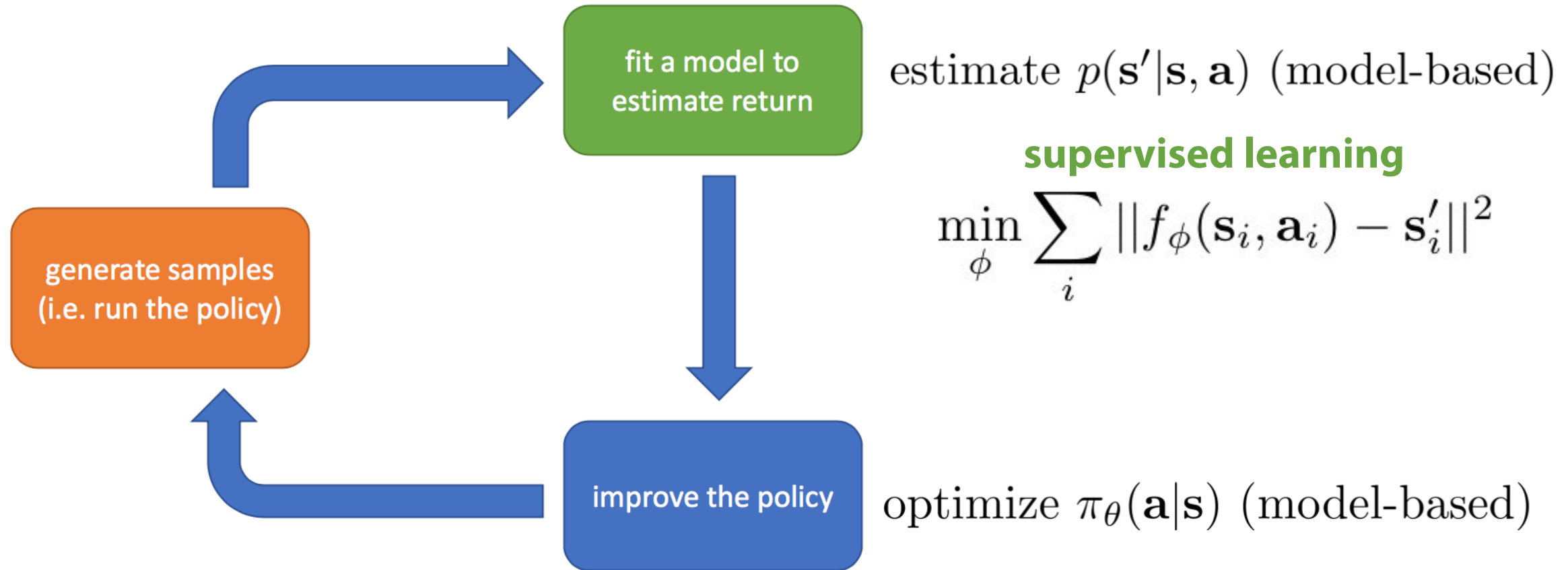
1. Why use model-based reinforcement learning?
- 2. Main model-based RL approaches**
3. Using local models & guided policy search
4. Handling high-dimensional observations

# The Anatomy of a Reinforcement Learning Problem

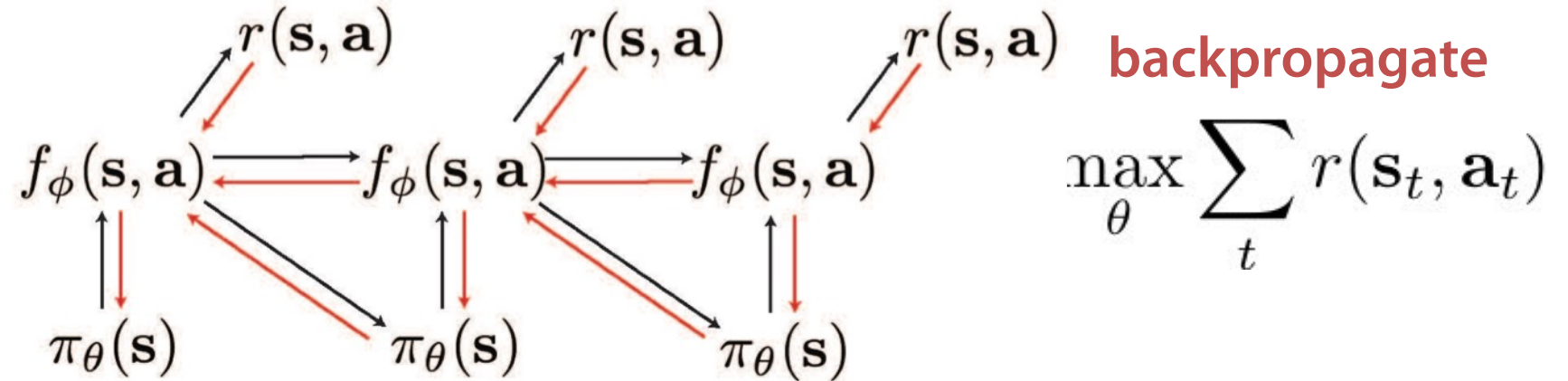




# Model-Based Reinforcement Learning



# Backprop through model to optimize policy



## Algorithm v0:

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. backpropagate through  $f_\phi(\mathbf{s}, \mathbf{a})$  into policy to optimize  $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

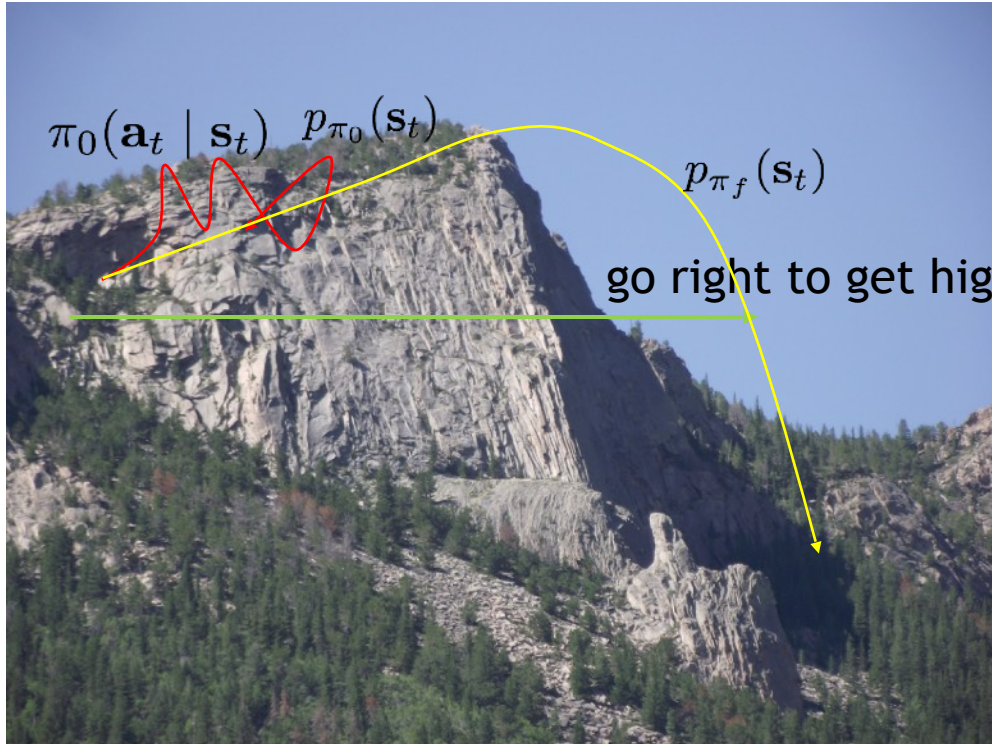
Does it work?

Yes!

- Essentially how system identification works in classical robotics
- Some care should be taken to design a good base policy
- Particularly effective if we can hand-engineer a dynamics representation using our knowledge of physics, and fit just a few parameters

# Does it work?

# No!



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  $\sum \|f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. backpropagate through  $f_\phi(\mathbf{s}, \mathbf{a})$  into policy to optimize  $\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)$

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


- Distribution mismatch problem becomes exacerbated as we use more expressive model classes

# Can we do better?

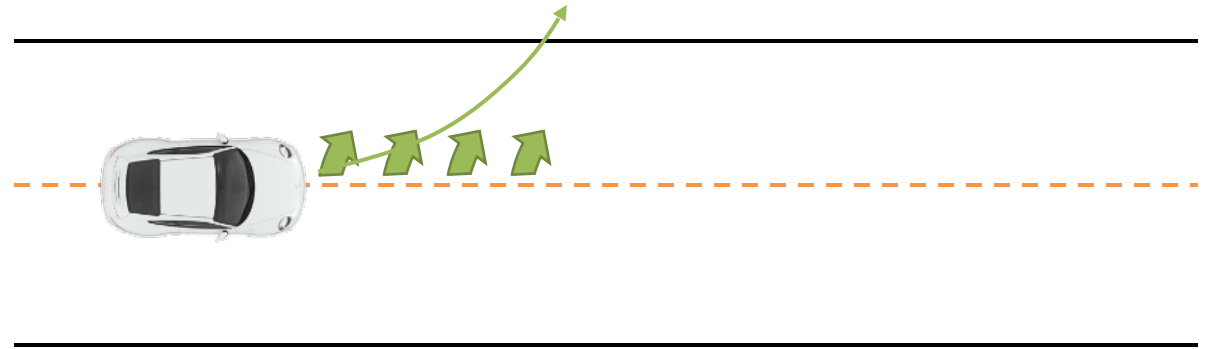
can we make  $p_{\pi_0}(\mathbf{s}_t) = p_{\pi_f}(\mathbf{s}_t)$  ?

need to collect data from  $p_{\pi_f}(\mathbf{s}_t)$

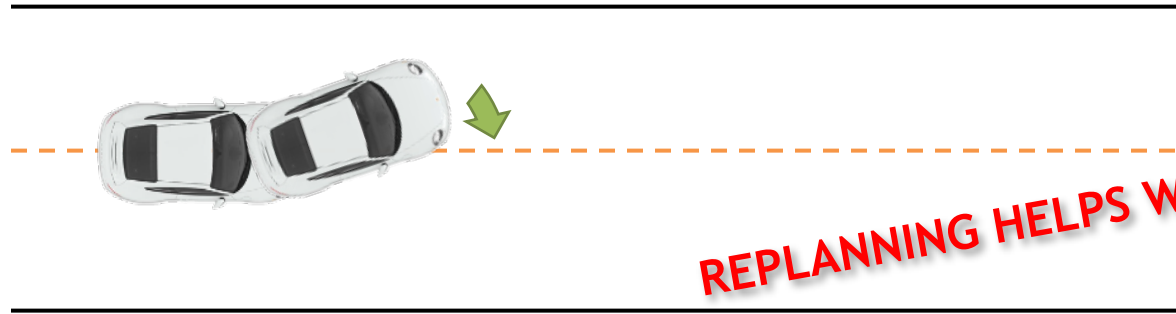
## Algorithm v1:

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. backpropagate through  $f_\phi(\mathbf{s}, \mathbf{a})$  into policy to optimize  $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
  4. run  $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ , appending visited tuples  $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$  to  $\mathcal{D}$
- 

# What if we make a mistake?



# Can we do better?



## Algorithm v2a:

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. backpropagate through  $f_\phi(\mathbf{s}, \mathbf{a})$  to choose actions.
4. execute the first planned action, observe resulting state  $\mathbf{s}'$
5. append  $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$  to dataset  $\mathcal{D}$

every N steps

*model-predictive control (MPC)*

# An alternative way to choose actions

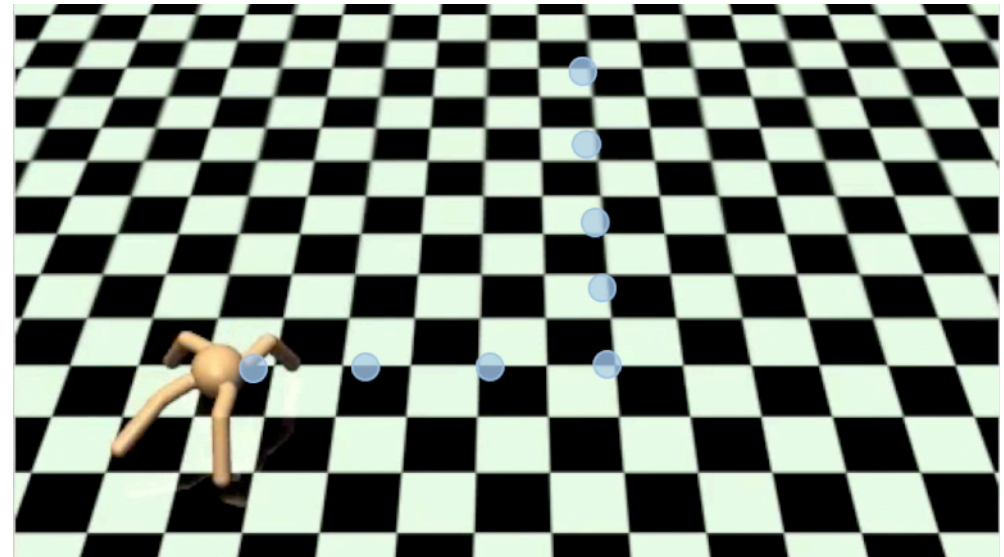
1. run base policy  $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$  (e.g., random policy) to collect  $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn model  $f_\phi(\mathbf{s}, \mathbf{a})$  to minimize  $\sum_i \|f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. backpropagate through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

every N steps



## *Can instead sample to choose actions:*

- A. Sample action sequences from some distribution (e.g. uniformly at random)
- B. Run actions through model to prediction future
- C. Choose action leading to the best future





# Summary so far

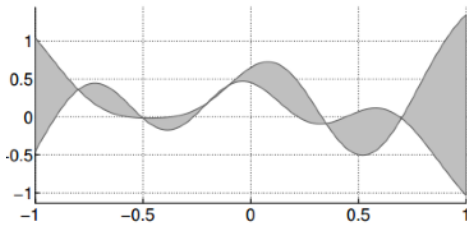
- Version 0: collect random samples, train dynamics, plan
  - Pro: simple, no iterative procedure
  - Con: distribution mismatch problem
- Version 1: iteratively collect data, refit model
  - Pro: simple, solves distribution mismatch
  - Con: might make mistakes with imperfect model
- Version 2: iteratively collect data using MPC (replan at each step)
  - Pro: robust to small model errors
  - Con: computationally expensive, but have a planning algorithm available

Two ways to optimize policy w.r.t. model:

- backprop through model into policy
- sampling-based optimization

# What kind of models can we use?

## Gaussian process



GP with input  $(\mathbf{s}, \mathbf{a})$  and output  $\mathbf{s}'$

Pro: very data-efficient

Con: not great with non-smooth dynamics

Con: very slow when dataset is big

## neural network

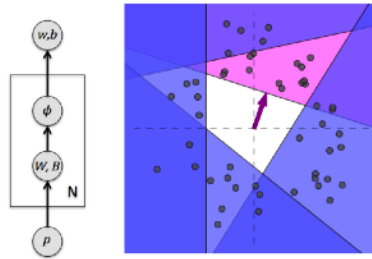


image: Punjani & Abbeel '14

Input is  $(\mathbf{s}, \mathbf{a})$  and output is  $\mathbf{s}'$

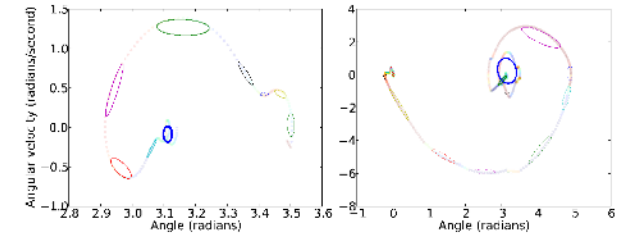
Euclidean training loss corresponds to Gaussian  $p(\mathbf{s}' | \mathbf{s}, \mathbf{a})$

More complex losses, e.g. output parameters of Gaussian mixture

Pro: very expressive, can use lots of data

Con: not so great in low data regimes

## other



GMM over  $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$  tuples

Train on  $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ , condition to get  $p(\mathbf{s}' | \mathbf{s}, \mathbf{a})$

For  $i^{\text{th}}$  mixture element,  $p_i(\mathbf{s}, \mathbf{a})$  gives region

where the mode  $p_i(\mathbf{s}' | \mathbf{s}, \mathbf{a})$  holds

other classes: domain-specific models (e.g. physics parameters)



video prediction?  
more on this later

# Outline

1. Why use model-based reinforcement learning?
2. Main model-based RL approaches
- 3. Using local models & guided policy search**
4. Handling high-dimensional observations

# The trouble with global models

Global model:  $f_\phi(\mathbf{s}_t, \mathbf{a}_t)$  represented by a big neural network

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. backpropagate through  $f_\phi(\mathbf{s}, \mathbf{a})$  into policy to optimize  $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

4. run  $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ , appending visited tuples  $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$  to  $\mathcal{D}$

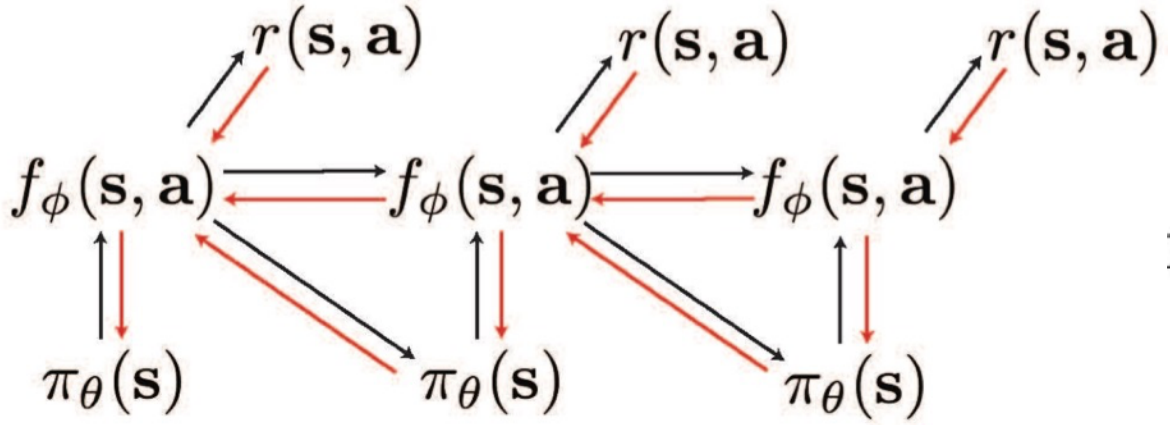
- Planner will seek out regions where the model is erroneously optimistic
- Need to find a very good model in most of the state space to converge on a good solution

# The trouble with global models

- Planner will seek out regions where the model is erroneously optimistic
- Need to find a very good model in most of the state space to converge on a good solution
- In some tasks, the model is much more complex than the policy



# Local models



backpropagate

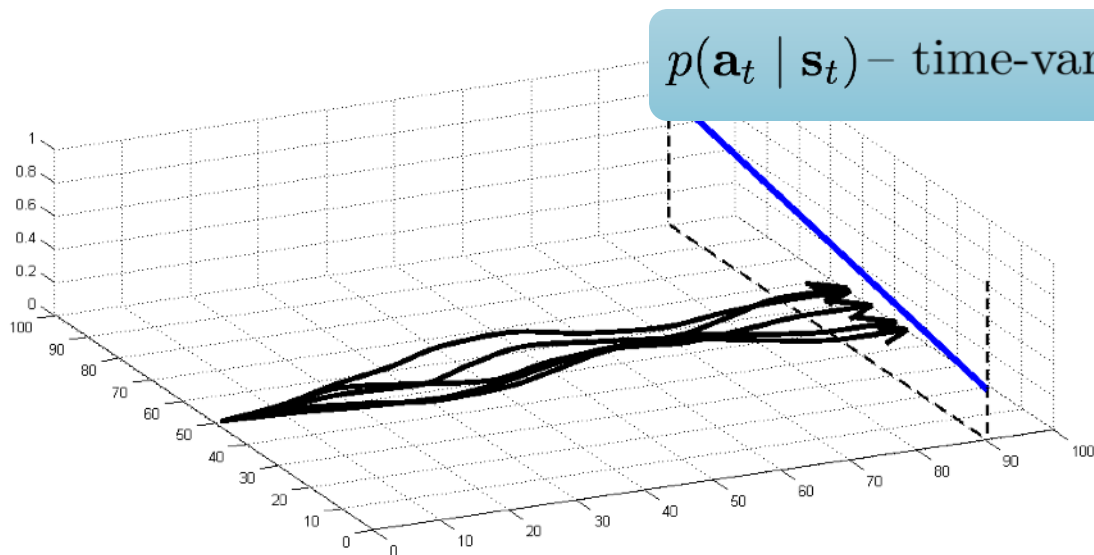
$$\max_{\theta} \sum_t r(\mathbf{s}_t, \mathbf{a}_t)$$

need  $\frac{df}{ds_t}, \frac{df}{da_t}, \frac{dr}{ds_t}, \frac{dr}{da_t}$

# Local models

need  $\frac{df}{ds_t}, \frac{df}{da_t}, \frac{dr}{ds_t}, \frac{dr}{da_t}$

idea: just fit  $\frac{df}{ds_t}, \frac{df}{da_t}$ , around current trajectory or policy!



$p(\mathbf{a}_t | \mathbf{s}_t)$  – time-varying linear-Gaussian controller

can **execute** on the robot!

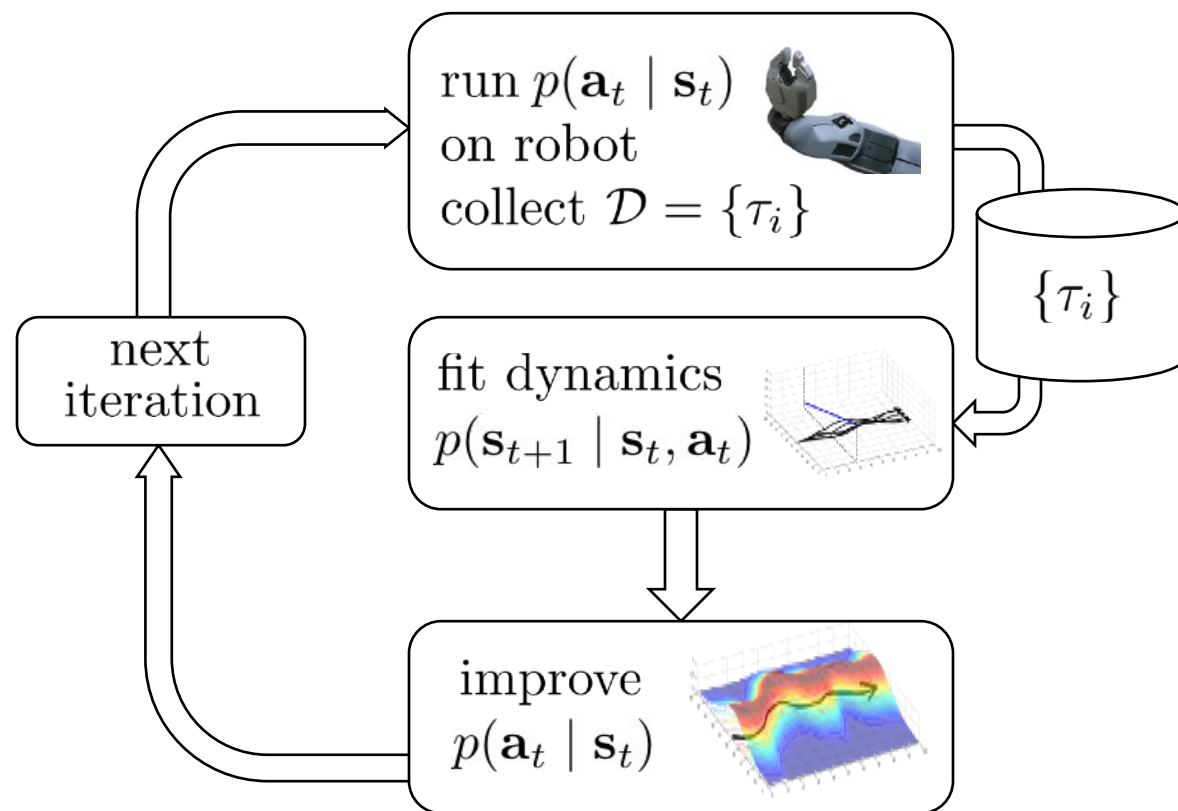
“local policy”

# Local models

$$p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) = \mathcal{N}(f(\mathbf{s}_t, \mathbf{a}_t), \Sigma)$$

$$f(\mathbf{s}_t, \mathbf{a}_t) \approx \mathbf{A}_t \mathbf{s}_t + \mathbf{B}_t \mathbf{a}_t$$

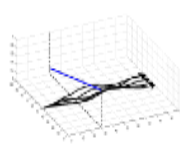
$$\mathbf{A}_t = \frac{df}{d\mathbf{s}_t} \quad \mathbf{B}_t = \frac{df}{d\mathbf{a}_t}$$





# How to fit the dynamics?

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



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

Version 1.0: fit  $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$  at each time step using linear regression

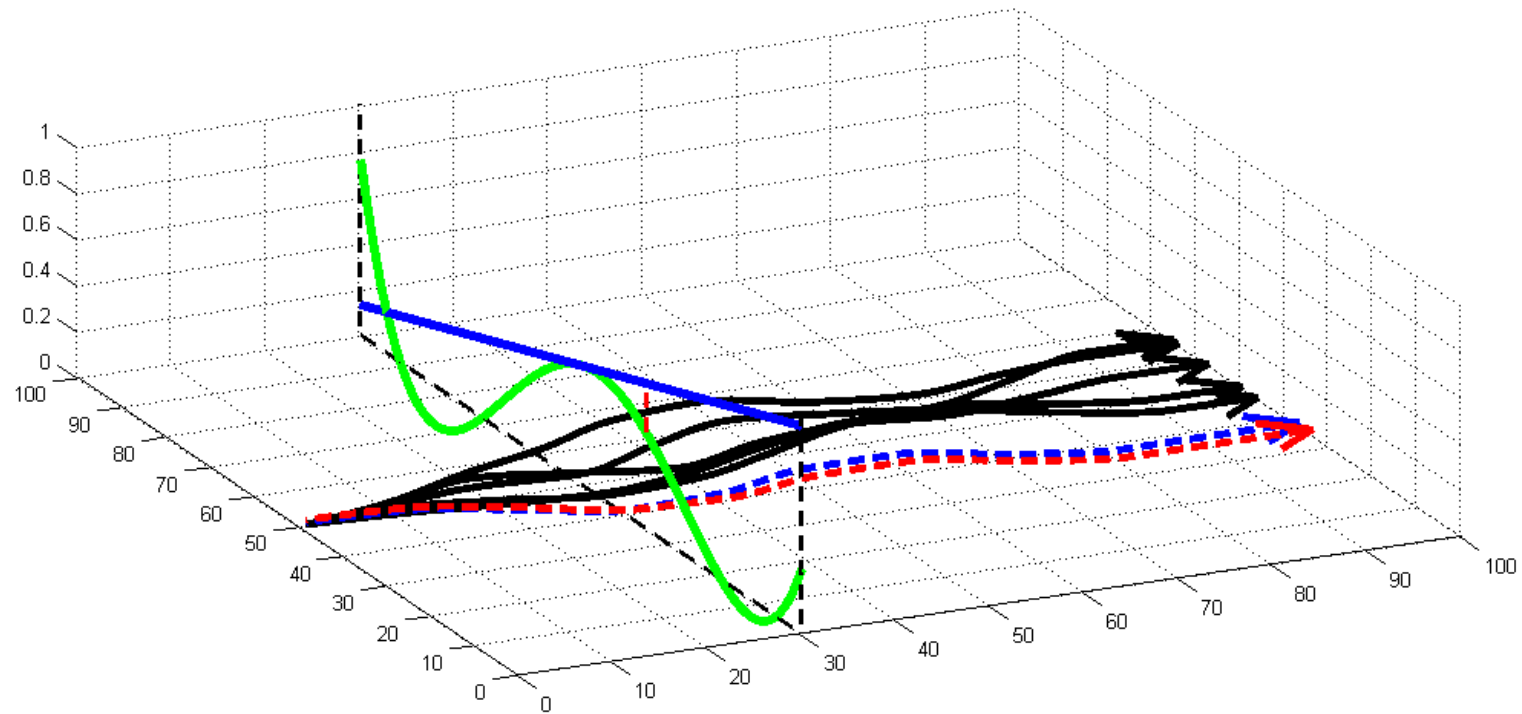
$$p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t) = \mathcal{N}(\mathbf{A}_t \mathbf{s}_t + \mathbf{B}_t \mathbf{a}_t + \mathbf{c}_t, \mathbf{N}_t) \quad \mathbf{A}_t \approx \frac{df}{d\mathbf{s}_t} \quad \mathbf{B}_t \approx \frac{df}{d\mathbf{a}_t}$$

## Can we do better?

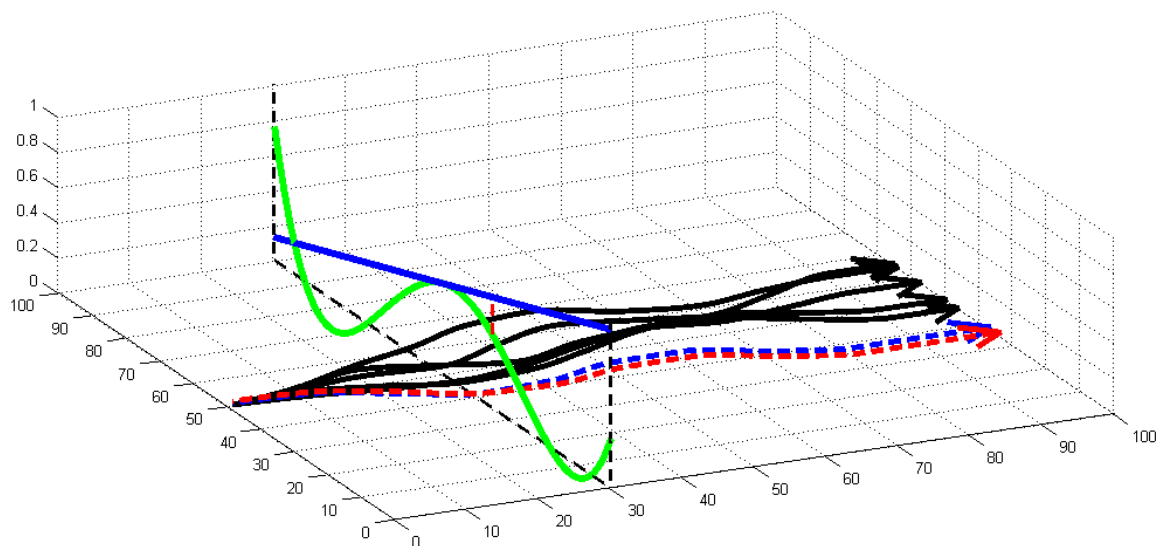
Version 2.0: fit  $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$  using *Bayesian* linear regression

Use your favorite *global* model as prior (GP, deep net, GMM)

# What if we go too far?



# How to stay close to old controller?



improve  
 $p(\mathbf{a}_t | \mathbf{s}_t)$

$$p(\tau) = p(\mathbf{s}_1) \prod_{t=1}^T p(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

What if the new  $p(\tau)$  is “close” to the old one  $\bar{p}(\tau)$ ?

If trajectory distribution is close, then dynamics will be close too!

What does “close” mean?  $D_{\text{KL}}(p(\tau) || \bar{p}(\tau)) \leq \epsilon$

# Local Models Approach Summary

Levine & Abbeel NIPS '14

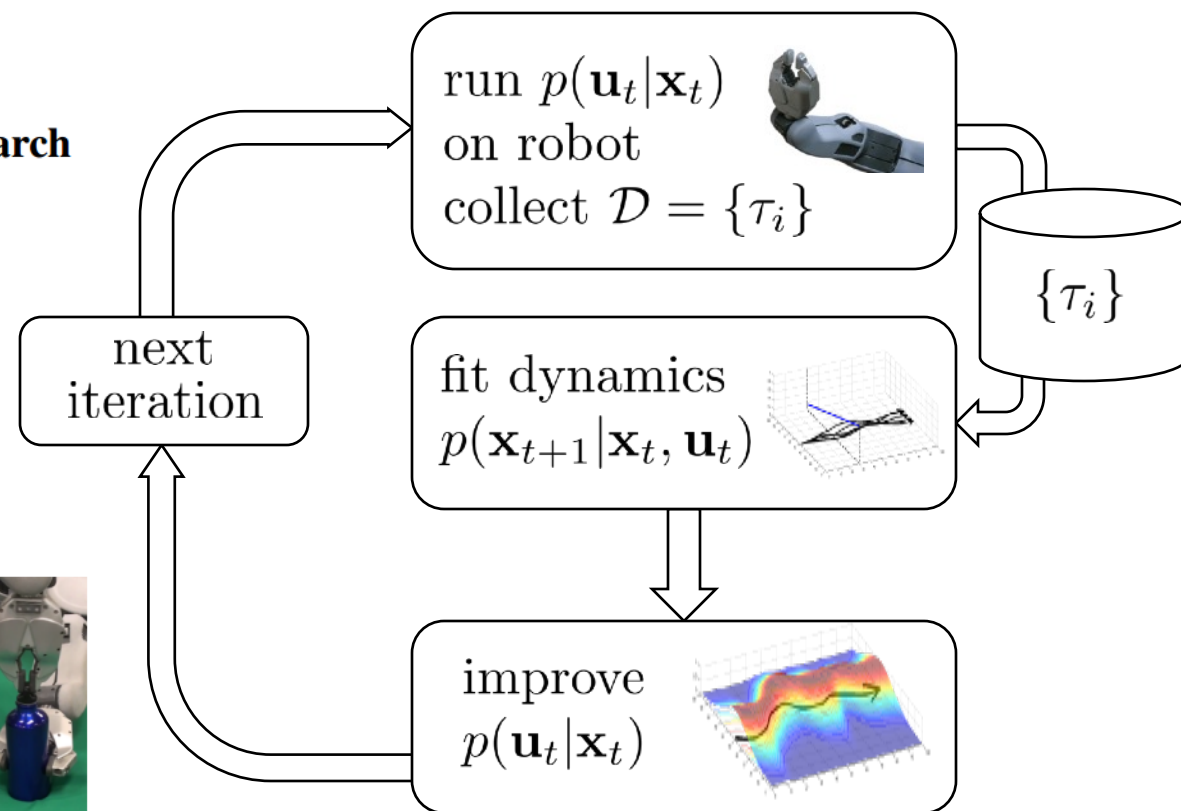
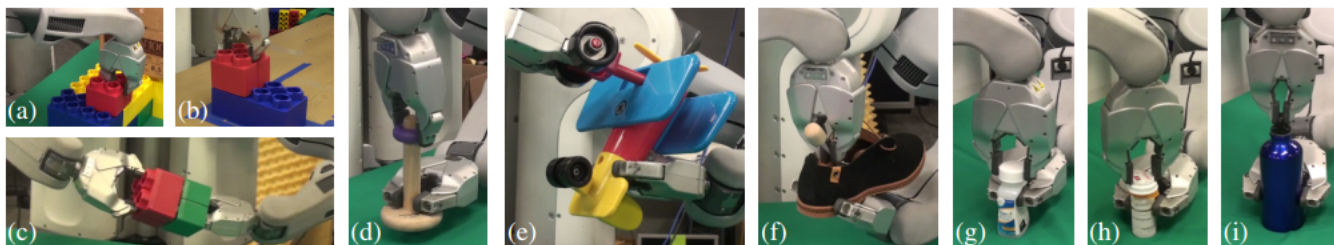
1. run base policy  $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$  to collect  $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn local model  $f(\mathbf{s}, \mathbf{a})$  to minimize  $\sum_i \|f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$   
**e.g. using linear regression**
3. update local policy  $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$  using local model  $f_\phi$  with KL constraint.
4. run  $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ , putting visited tuples  $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$  in  $\mathcal{D}$

**e.g. using iterative LQR**

# Case study: local models & iterative LQR

## Learning Contact-Rich Manipulation Skills with Guided Policy Search

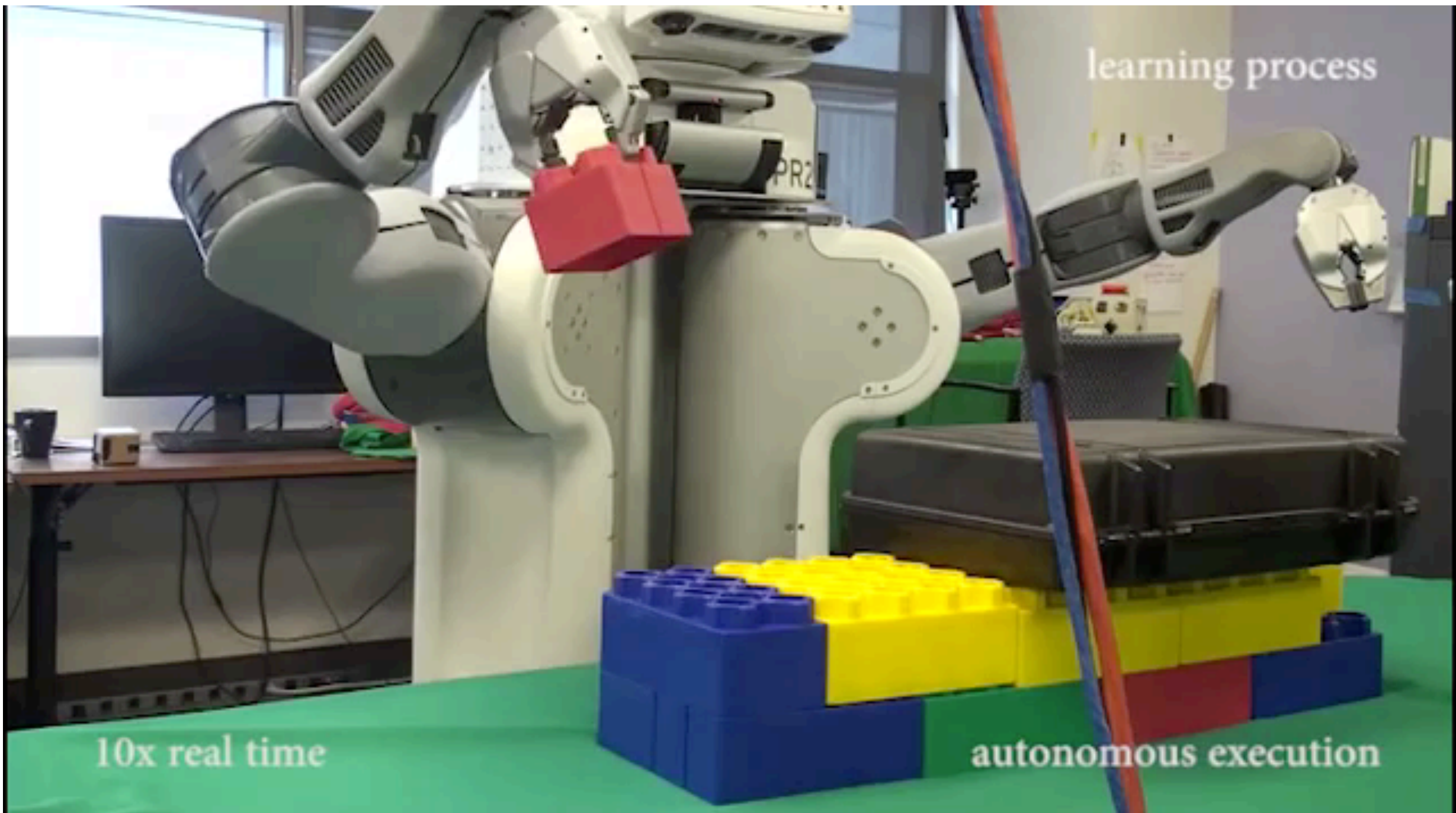
Sergey Levine, Nolan Wagener, Pieter Abbeel



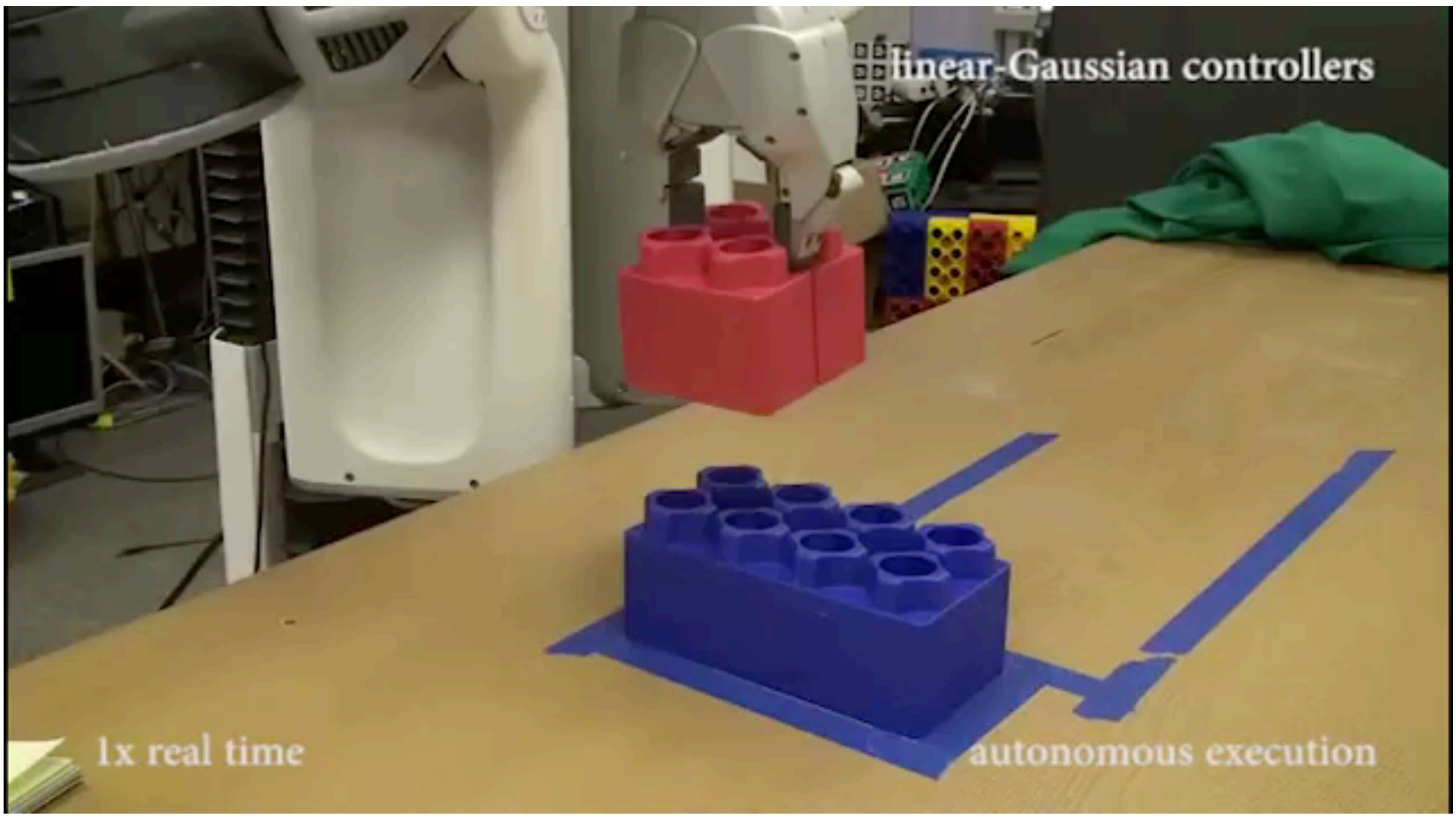
learning process

10x real time

autonomous execution



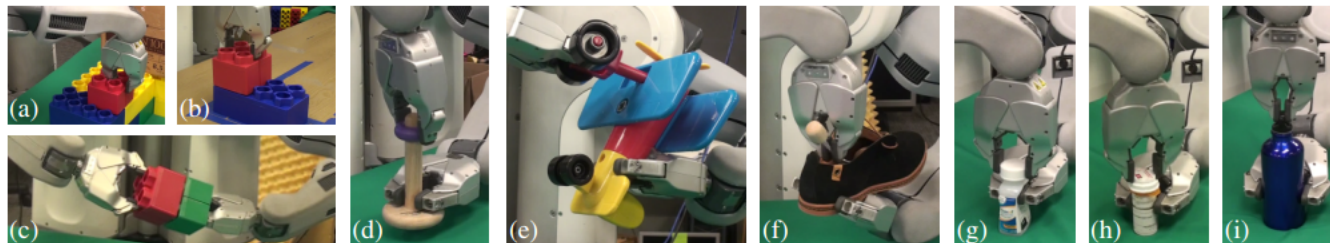
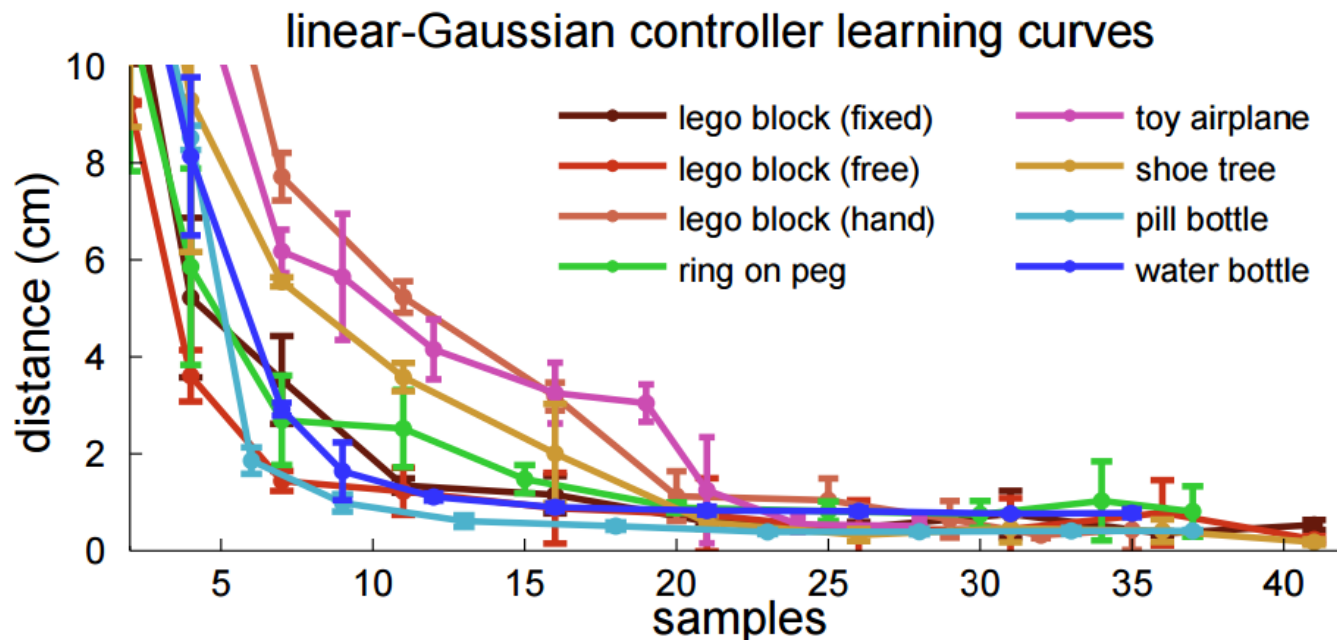
linear-Gaussian controllers



1x real time

autonomous execution

# Case study: local models & iterative LQR





# Local Models Approach Summary

Levine & Abbeel NIPS '14

1. run base policy  $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$  to collect  $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn local model  $f(\mathbf{s}, \mathbf{a})$  to minimize  $\sum_i \|f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$   
**e.g. using linear regression**
3. update local policy  $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$  using local model  $f_\phi$  with KL constraint.
4. run  $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ , putting visited tuples  $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$  in  $\mathcal{D}$

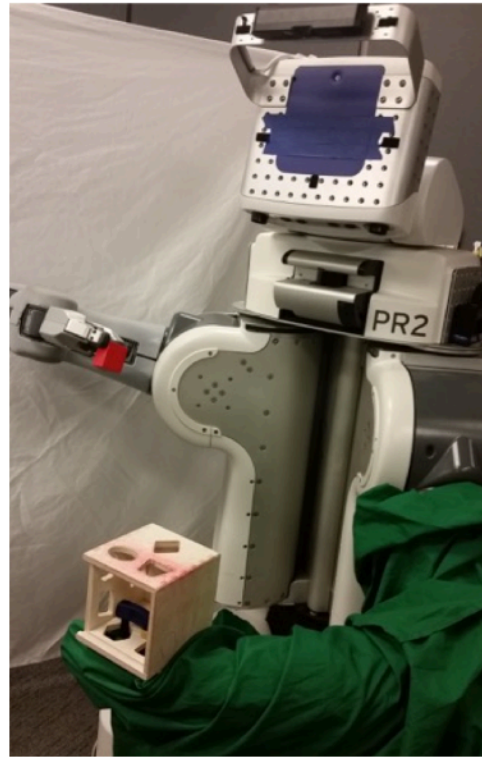
**e.g. using iterative LQR**

**end result: single local policy**

**Guided policy search: supervise one global policy using multiple local policies**

# Case study: guided policy search

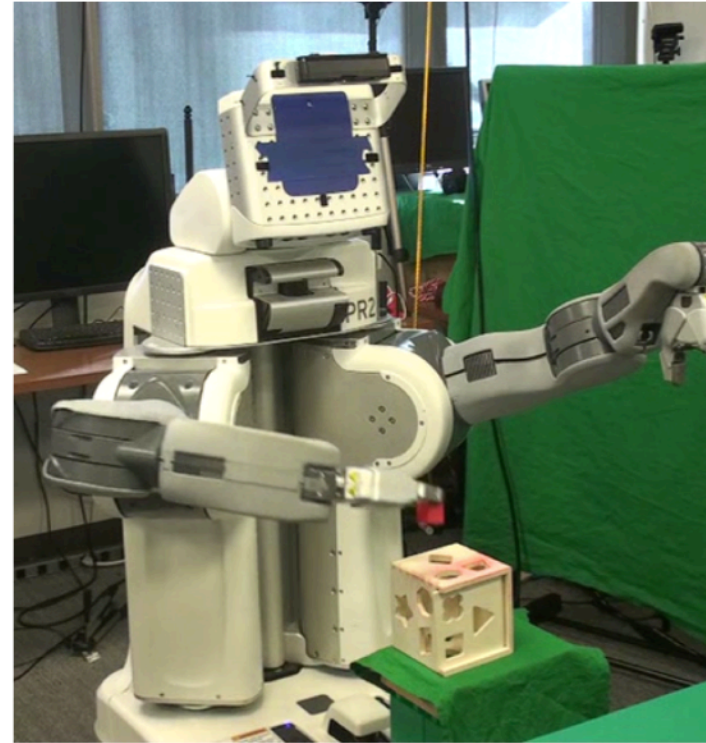
**Training time**



$$\mathbf{s}_t \longrightarrow \mathbf{a}_t$$

target pose known

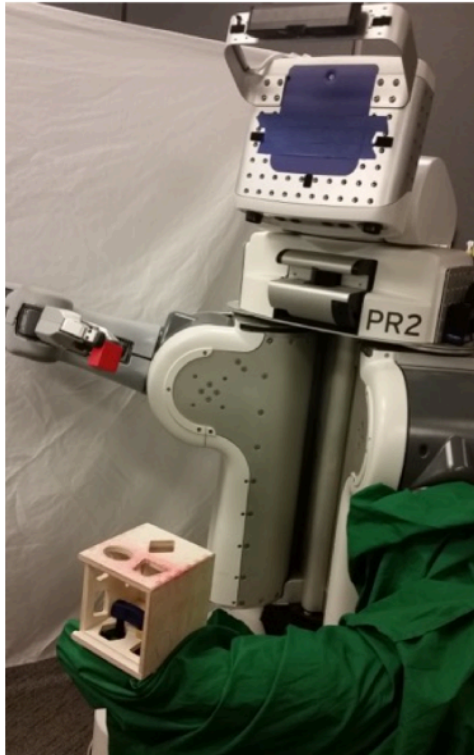
**Test time**



$$\mathbf{o}(\mathbf{s}_t) \longrightarrow \mathbf{a}_t$$

# Case study: guided policy search

## Training time

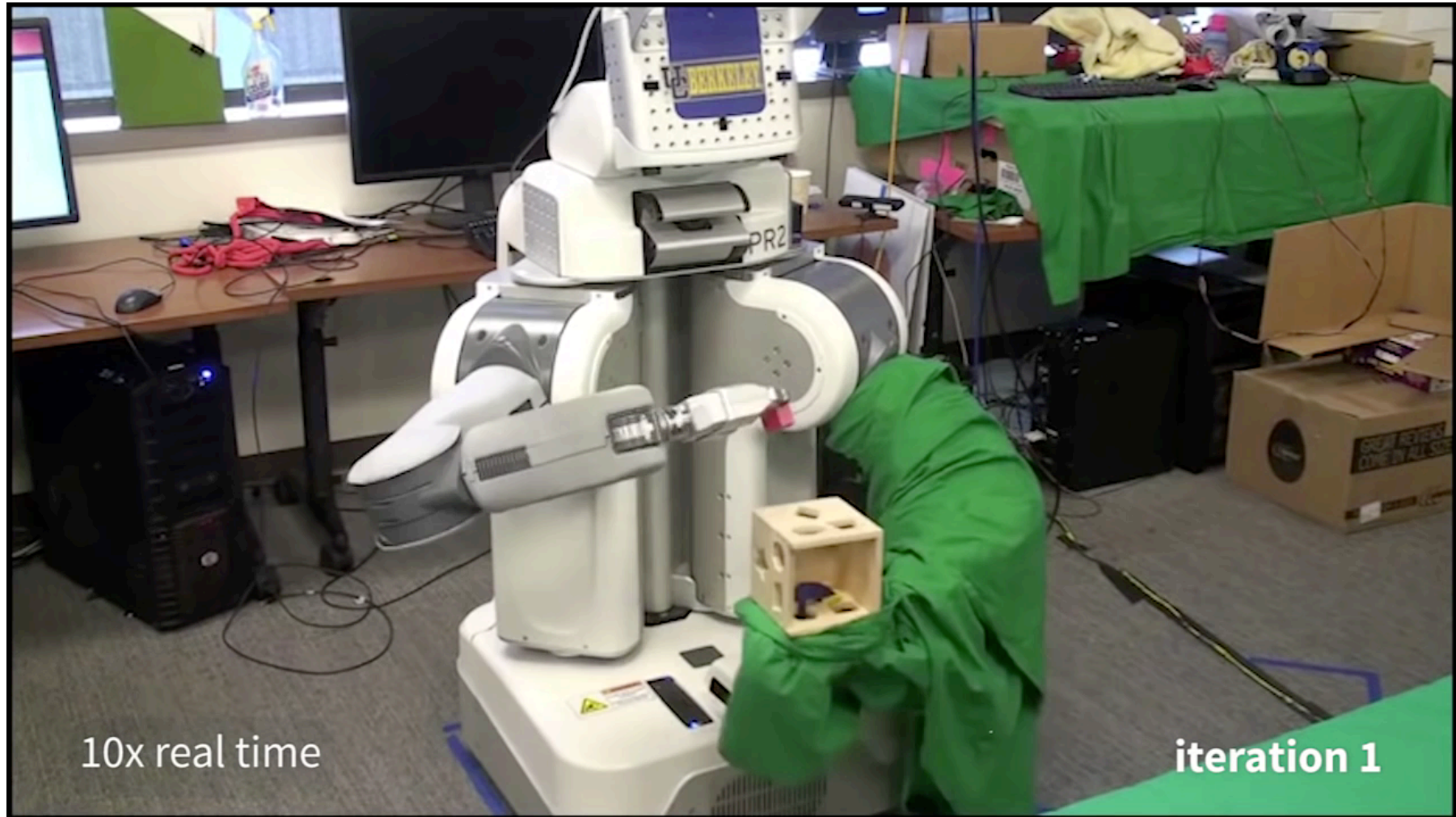


- take samples for each target position
- fit local model and solve for local policy for each target position
- use supervision from local policies to train global neural network policy w/ vision

$$\mathbf{s}_t \longrightarrow \mathbf{a}_t$$

# Guided Policy Search: learning

(Levine\*, Finn\*, et al. JMLR '16)



# Guided Policy Search: learned behaviors

(Levine\*, Finn\*, et al. JMLR '16)



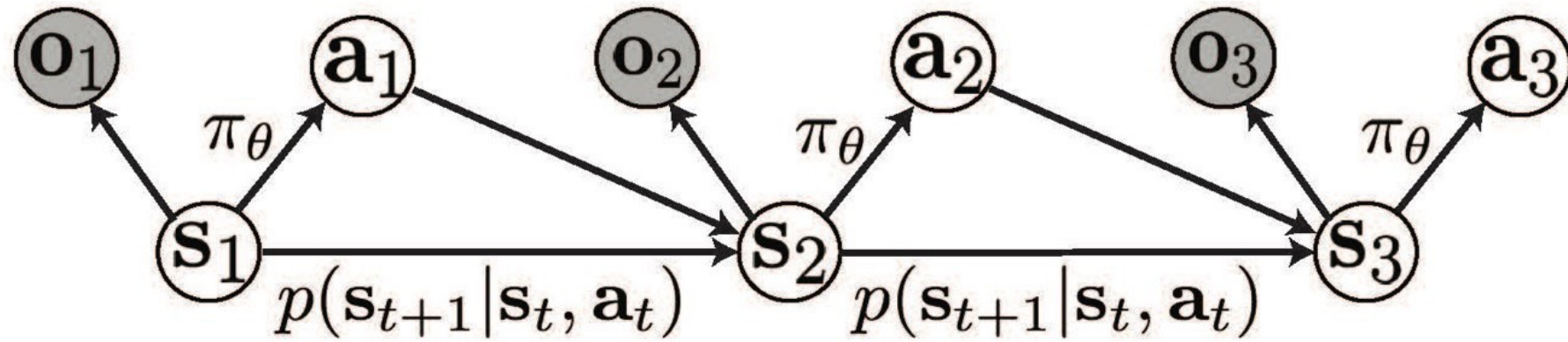
**< 300 trials = 25 min of robot time (per task)**

+ efficiently learn complex vision-based skills - requires state during training

# Outline

1. Why use model-based reinforcement learning?
2. Main model-based RL approaches
3. Using local models & guided policy search
- 4. Handling high-dimensional observations**

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



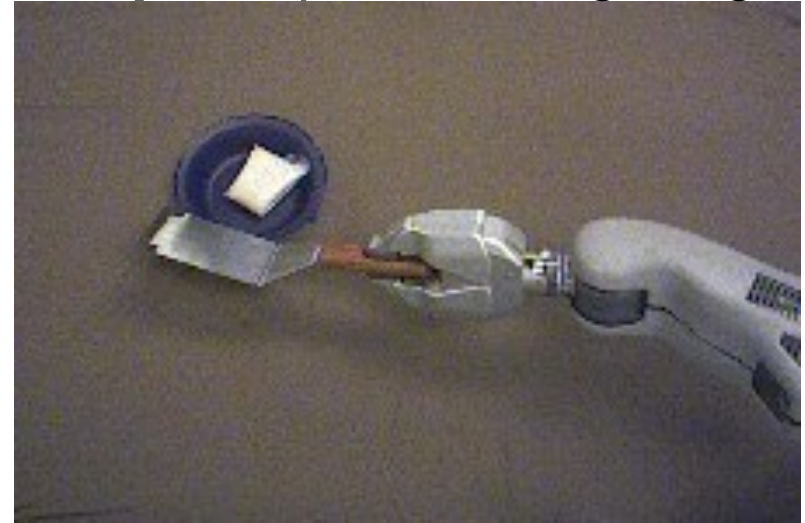
**also:** no reward signal with only observations

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



**also:** no reward signal with only observations

**one option:** provide image of goal



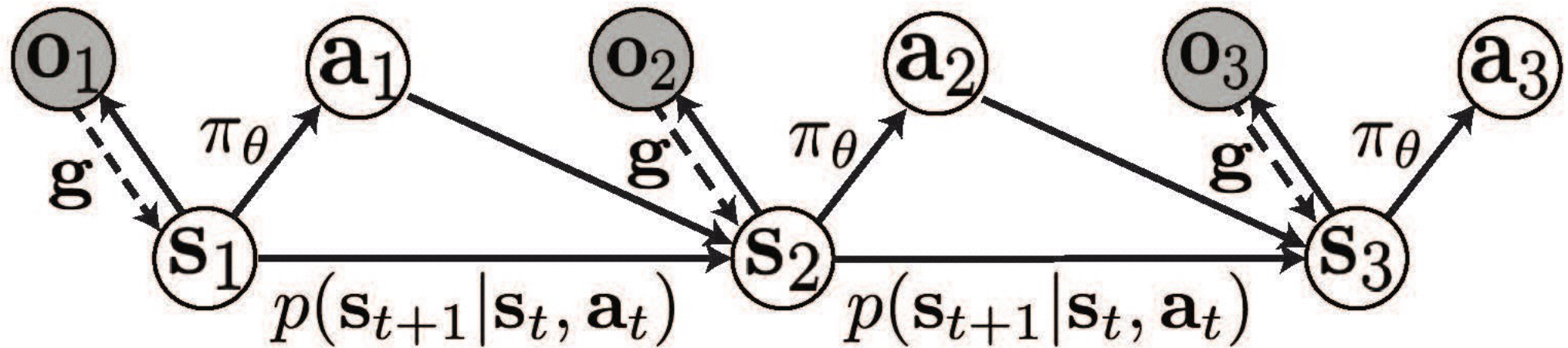
## Approaches

1. Learn model in latent space
2. Learn model of observations (e.g. video)
3. Inverse models [won't cover]



# 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

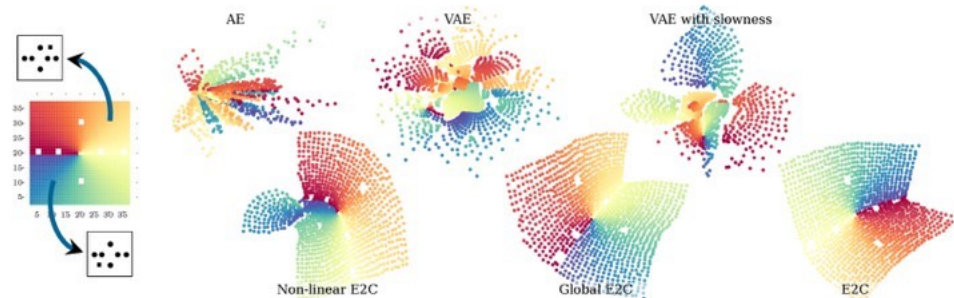
---

## Deep Spatial Autoencoders for Visuomotor Learning

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

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

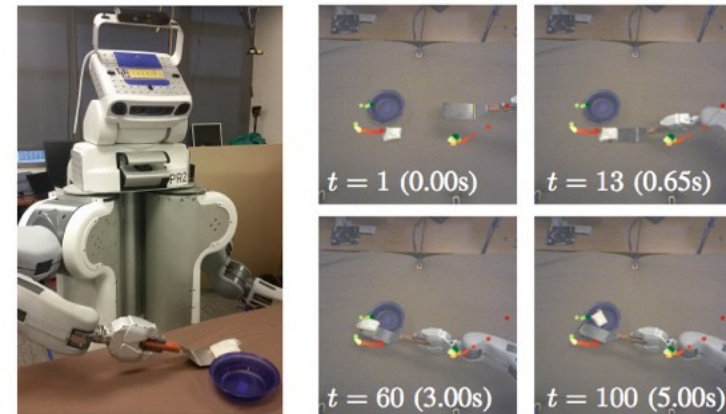


Fig. 1: PR2 learning to scoop a bag of rice into a bowl with a spatula (left) using a learned visual state representation (right).

ICRA 2016

# Learning in Latent Space

1. run base policy  $\pi_0(\mathbf{a}_t|\mathbf{o}_t)$  (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  $\mathcal{D}$

What is reward for optimizing policy?

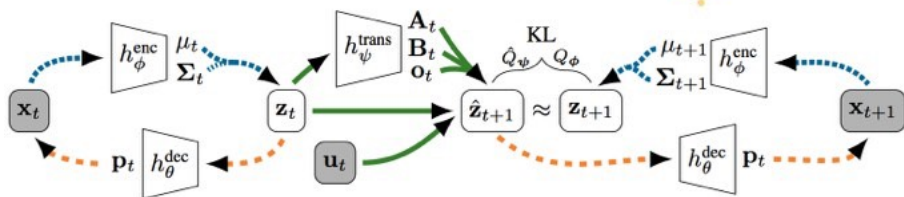
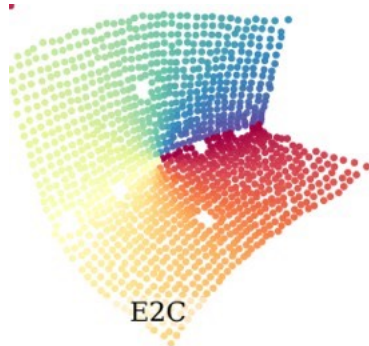
**reward signal:**  $r(\mathbf{o}, \mathbf{a}) = r(\mathbf{a}) + \|g(\mathbf{o}) - g(\mathbf{o}_{\text{goal}})\|$

# 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  $\mathcal{D}$

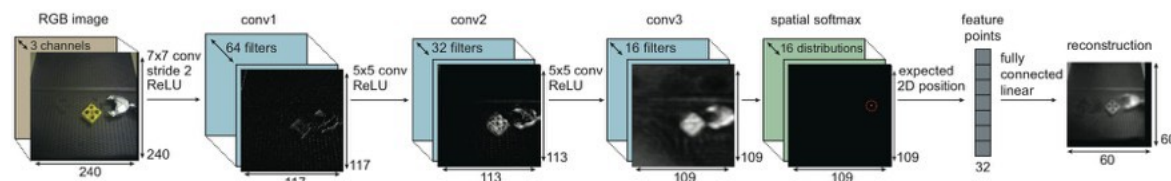
How to optimize latent embedding  $g$ ?

Watter et al. '15



learn embedding & model jointly

Finn et al. '16



embedding is smooth and structured

# Learning in Latent Space

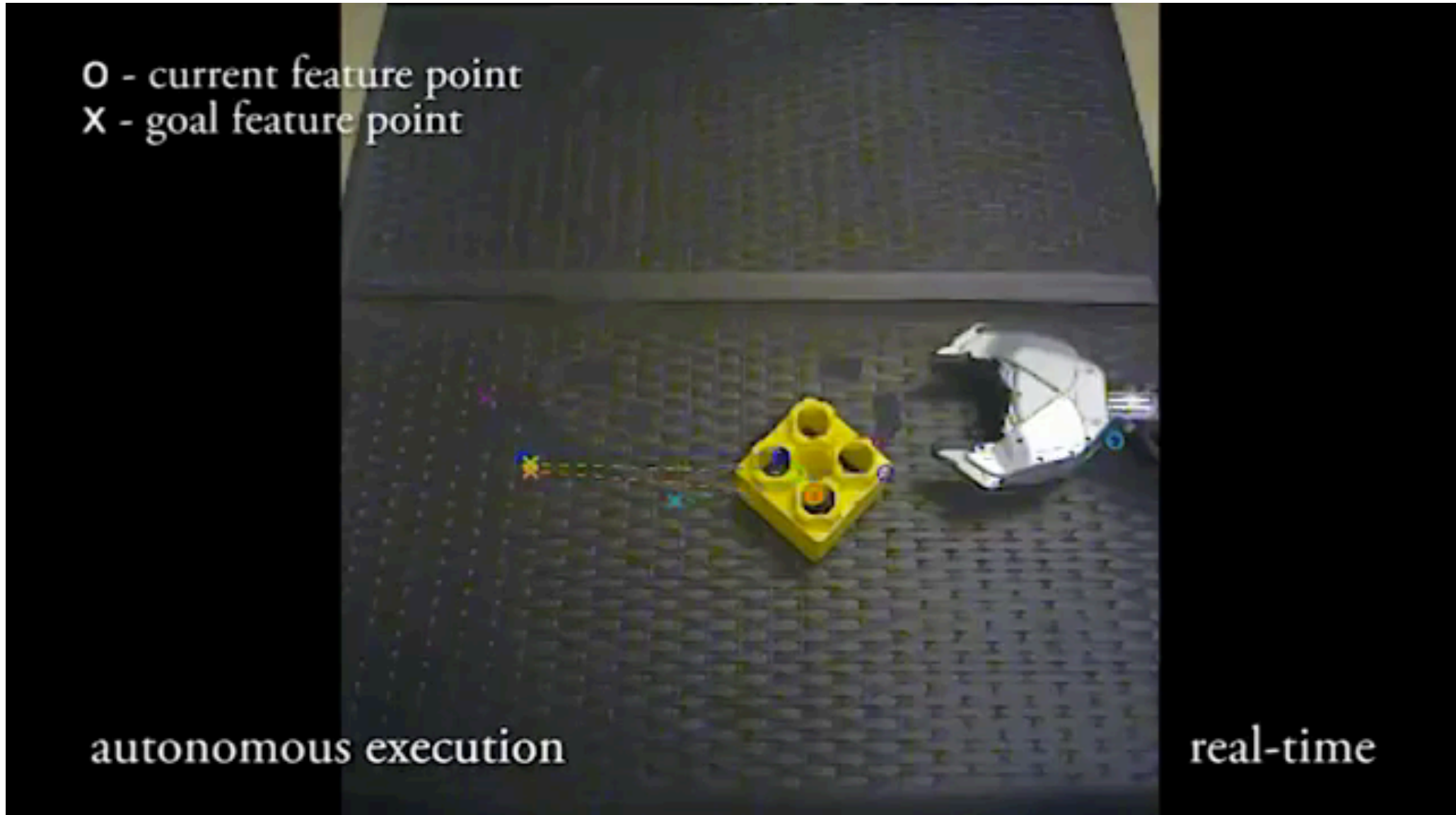


Goal state for trajectory optimization

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

Watter et al. NIPS '15

# Learning in Latent Space



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

Finn et al. ICRA'16

# Learning in Latent Space

## **Pros:**

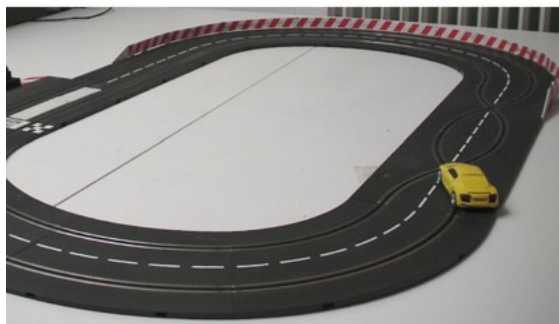
- + Learn complex visual skills very efficiently
- + Structured representation enables effective learning

## **Cons:**

- Reconstruction objectives might not recover the right representation

**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

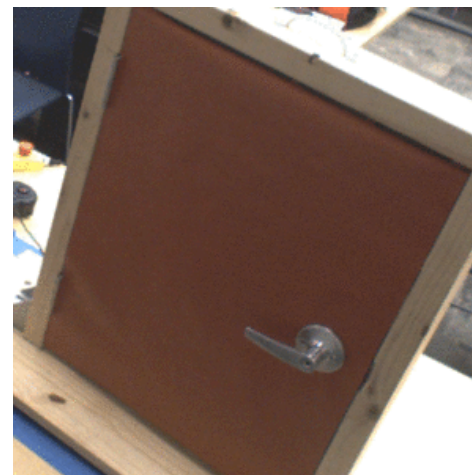


TRPO in latent space  
Ghadirzadeh et al. '17

**use embedding for reward function**

Sermanet et al. RSS '17

video demonstration



acquire reward using  
ImageNet features

learned policy



+ model-free RL

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

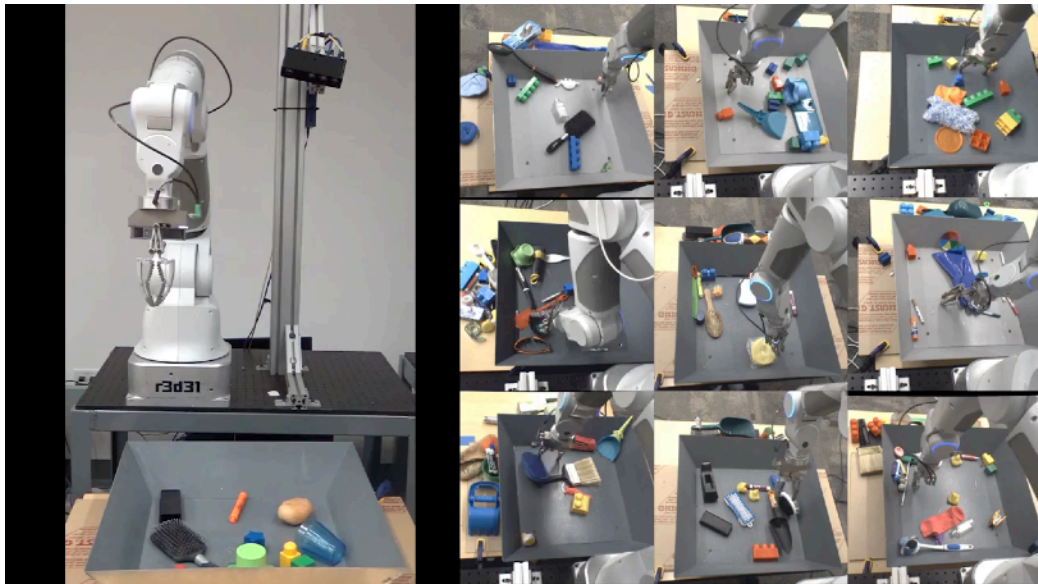


# Modeling directly in observation space

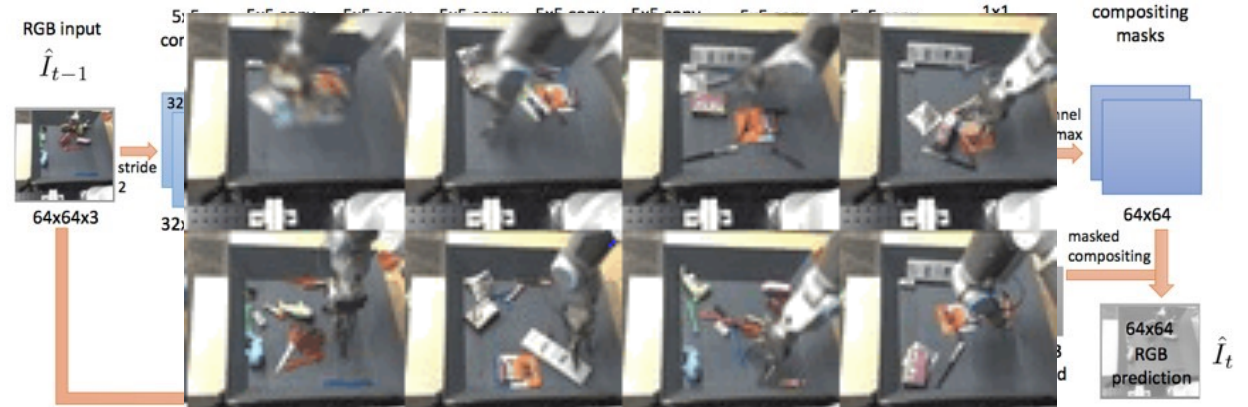
## Recall MPC

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. backpropagate through  $f_\phi(\mathbf{o}, \mathbf{a})$  to choose actions.
4. execute the first planned action, observe resulting state  $\mathbf{o}'$
5. append  $(\mathbf{o}, \mathbf{a}, \mathbf{o}')$  to dataset  $\mathcal{D}$

every N steps

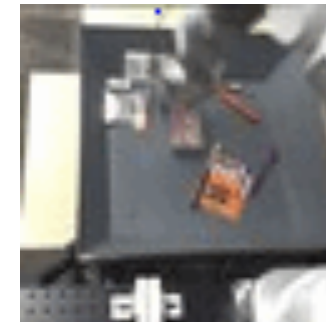
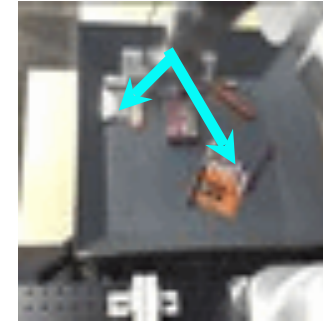


## action-conditioned video prediction



# Planning with Visual Foresight

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



# Modeling directly in observation space

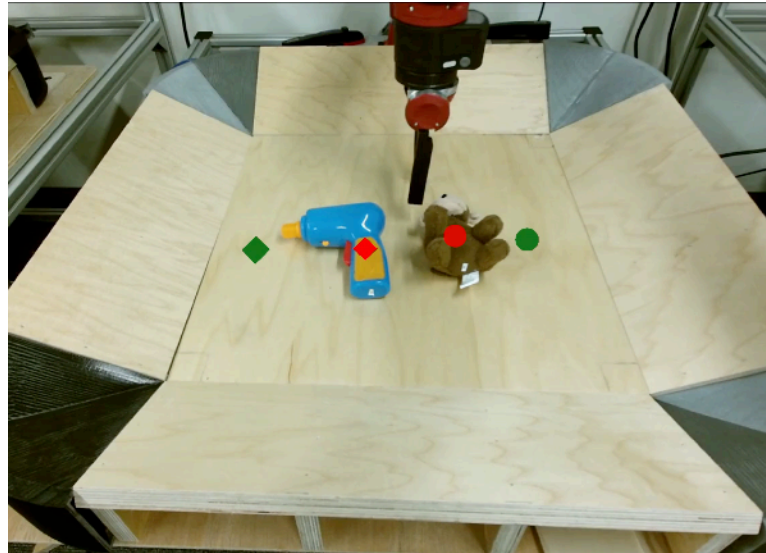
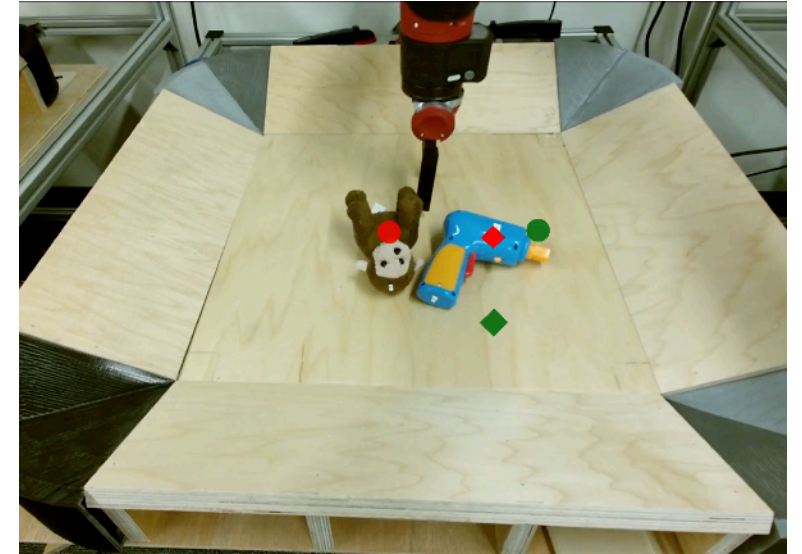
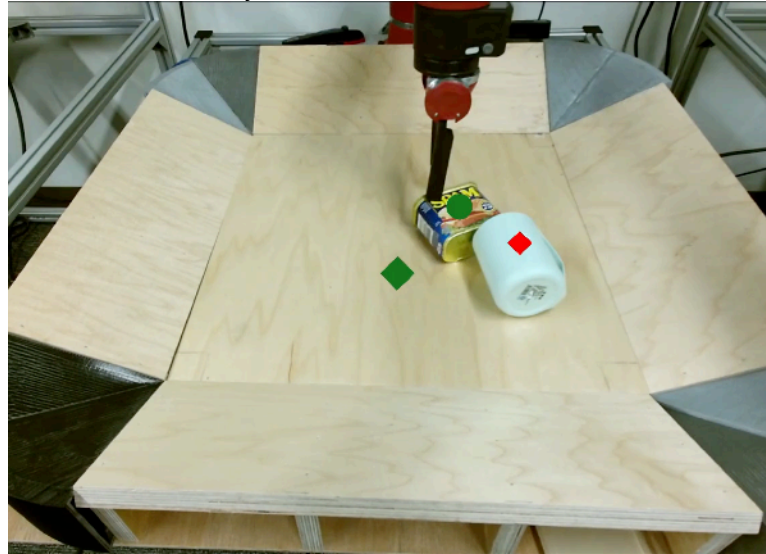
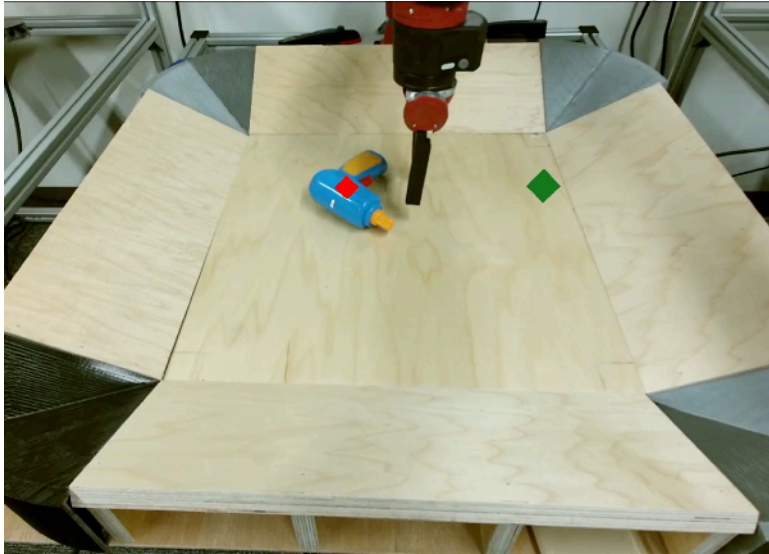


Finn & Levine ICRA'17

**<2 days of *unsupervised* robot time**

**Only human involvement:** programming initial motions and providing objects to play with.

# Modeling directly in observation space



**model can be reused for different tasks**

Ebert et al.'17

# Modeling directly in observation space

## **Pros:**

- + Entirely self-supervised
- + Learn for a variety of tasks
- + More efficient than single-task model-free learning

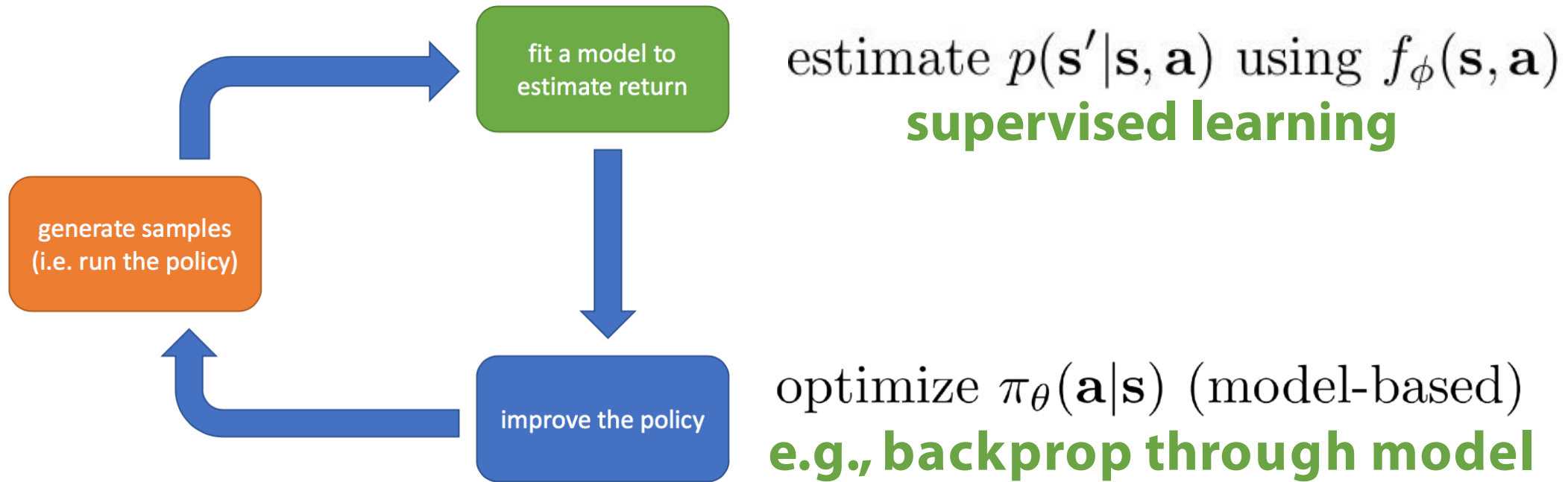
## **Cons:**

- Can't [yet] handle as complex skills as model-free methods

# Outline

1. Why use model-based reinforcement learning?
2. Main model-based RL approaches
3. Using local models & guided policy search
4. Handling high-dimensional observations

# Model-based RL Review



## Correcting for model errors:

refitting model with new data, replanning with MPC, using local models

## Model-based RL from raw observations:

learn latent space, typically with unsupervised learning, or  
model & plan directly in observational space

# Suggested Reading on Model-based RL

**Tassa et al. IROS '12.** *Synthesis and Stabilization of Complex Behaviors.* Good introduction to MPC with a known model

**Levine\*, Finn\* et al. JMLR '16.** *End-to-End Learning of Deep Visuomotor Policies.* Thorough paper on guided policy search for learning real robotic vision-based skills

**Heess et al. NIPS '15.** *Stochastic Value Gradients.* Backdrop through dynamics to assist model-free learner

**Watter et al. NIPS '15.** *Embed-to-Control,* Learn latent space and use model-based RL in learned latent space to reach image of goal

**Finn & Levine ICRA '17.** *Deep Visual Foresight for Planning Robot Motion.* Plan using learned action-conditioned video prediction model



# Further Reading on Model-based RL

**Use known model:** Tassa et al. IROS '12, Tan et al. TOG '14, Mordatch et al. TOG '14

**Guided policy search:** Levine\*, Finn\* et al. JMLR '16, Mordatch et al. RSS '14, NIPS '15

**Backprop through model:** Deisenroth et al. ICML '11, Heess et al. NIPS '15, Mishra et al. ICML '17, Degraeve et al. '17, Henaff et al. '17

**Inverse models:** Agrawal et al. NIPS '16

**MBRL in latent space:** Watter et al. NIPS '15, Finn et al. ICRA '16

**MPC with deep models:** Lenz et al. RSS '15, Finn & Levine ICRA '17

**Combining Model-Based & Model-Free:**

- use roll-outs from model as experience: Sutton '90, Gu et al. ICML '16
- use model as baseline: Chebotar et al. ICML '17
- use model for exploration: Stadie et al. arXiv '15, Oh et al. NIPS '16
- model-free policy with planning capabilities: Tamar et al. NIPS '16, Pascanu et al. '17
- model-based look-ahead: Guo et al. NIPS '14, Silver et al. Nature '16

# Model-Based vs. Model-Free Algorithms

## **Models:**

- + Easy to collect data in a scalable way (self-supervised)
- + Possibility to transfer across tasks
- + Typically require a smaller quantity of 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)
- Not transferable across tasks

**Ultimately we will want both!**

# Closing Remarks

Model-based RL is an under-explored area of research

**Two active, exciting areas:**

- model-based approaches with high-dimensional observations
- combining elements of model-based planning & model-free policies

Questions?