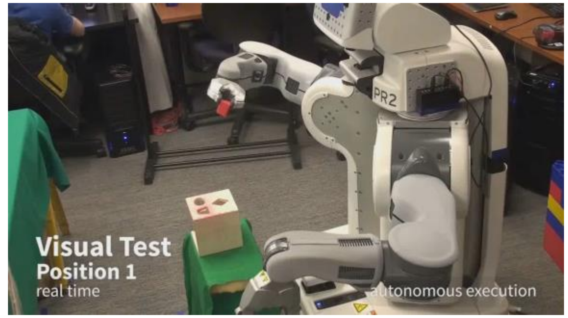
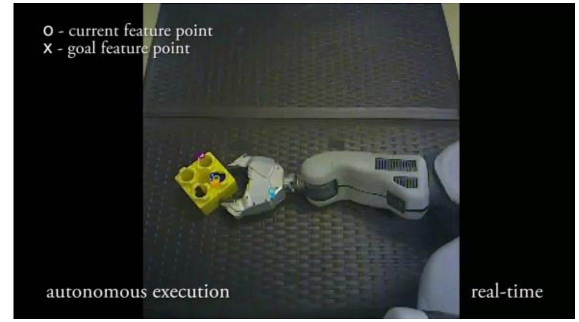
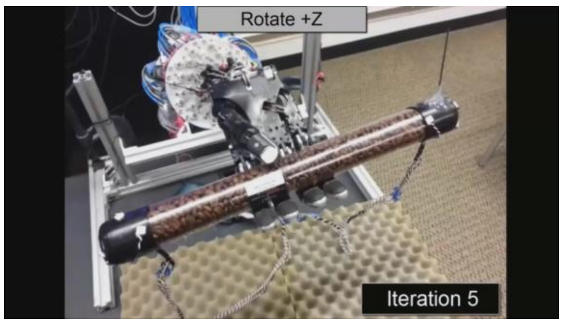
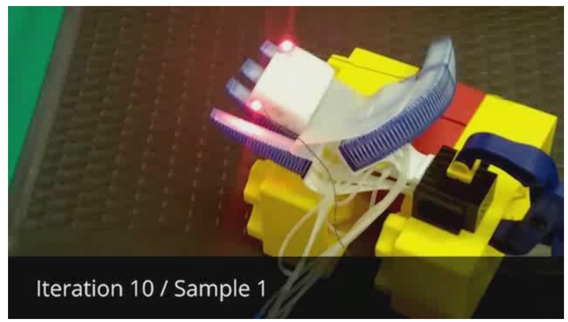
Deep Learning for Robotics

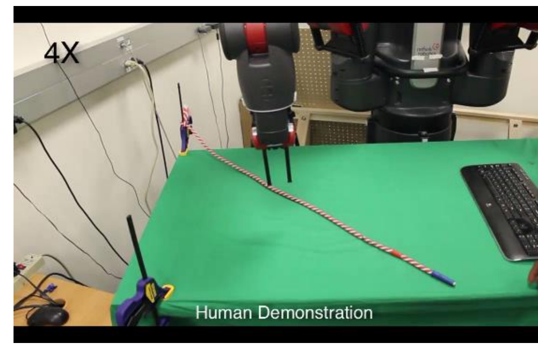
Sergey Levine

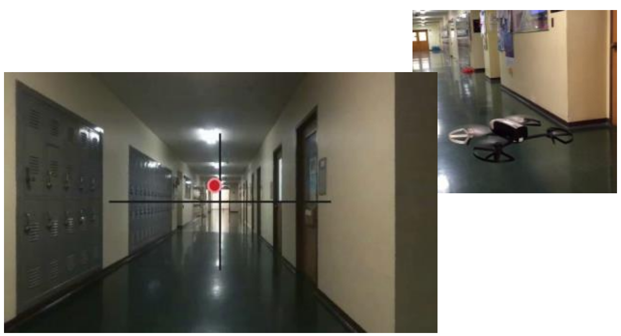










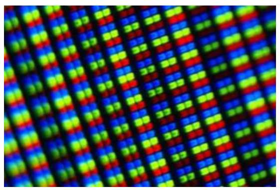


Real-World Experiments

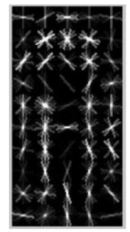
Not accounting for uncertainty
(higher-speed collisions)

Deep Learning: End-to-end vision

standard computer vision

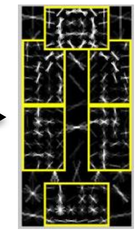


features
(e.g. HOG)



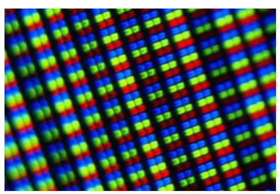
mid-level features
(e.g. DPM)

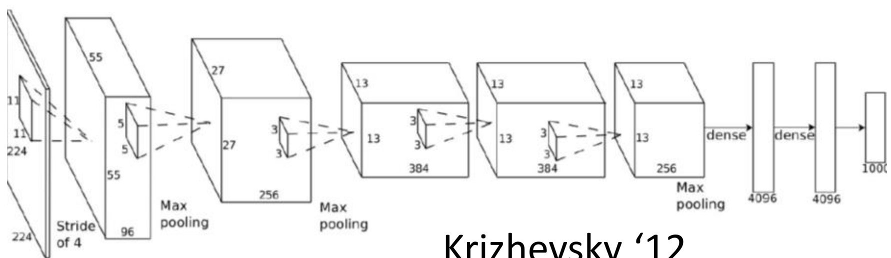
Felzenszwalb '08



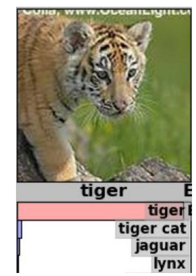
classifier
(e.g. SVM)

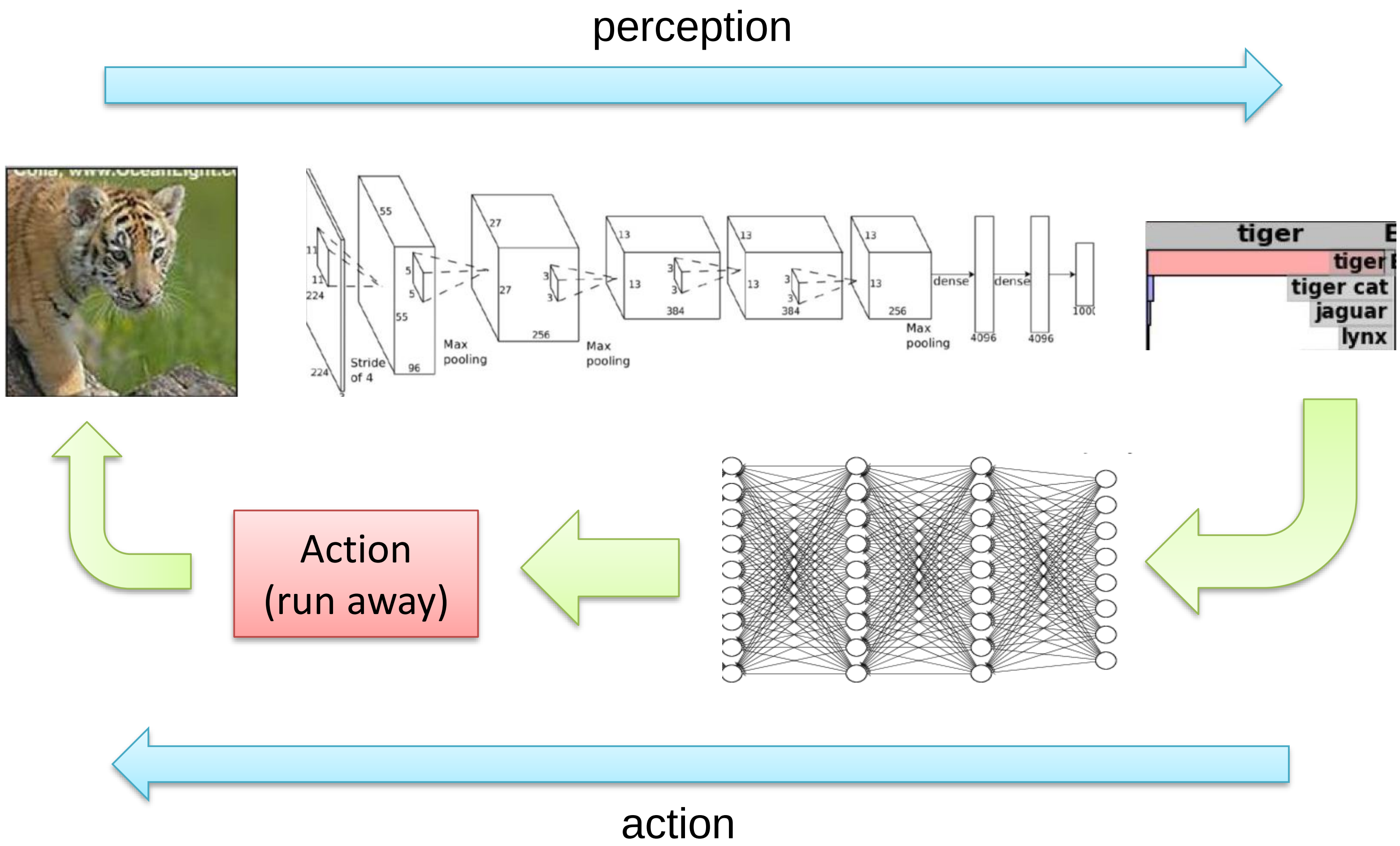
deep learning

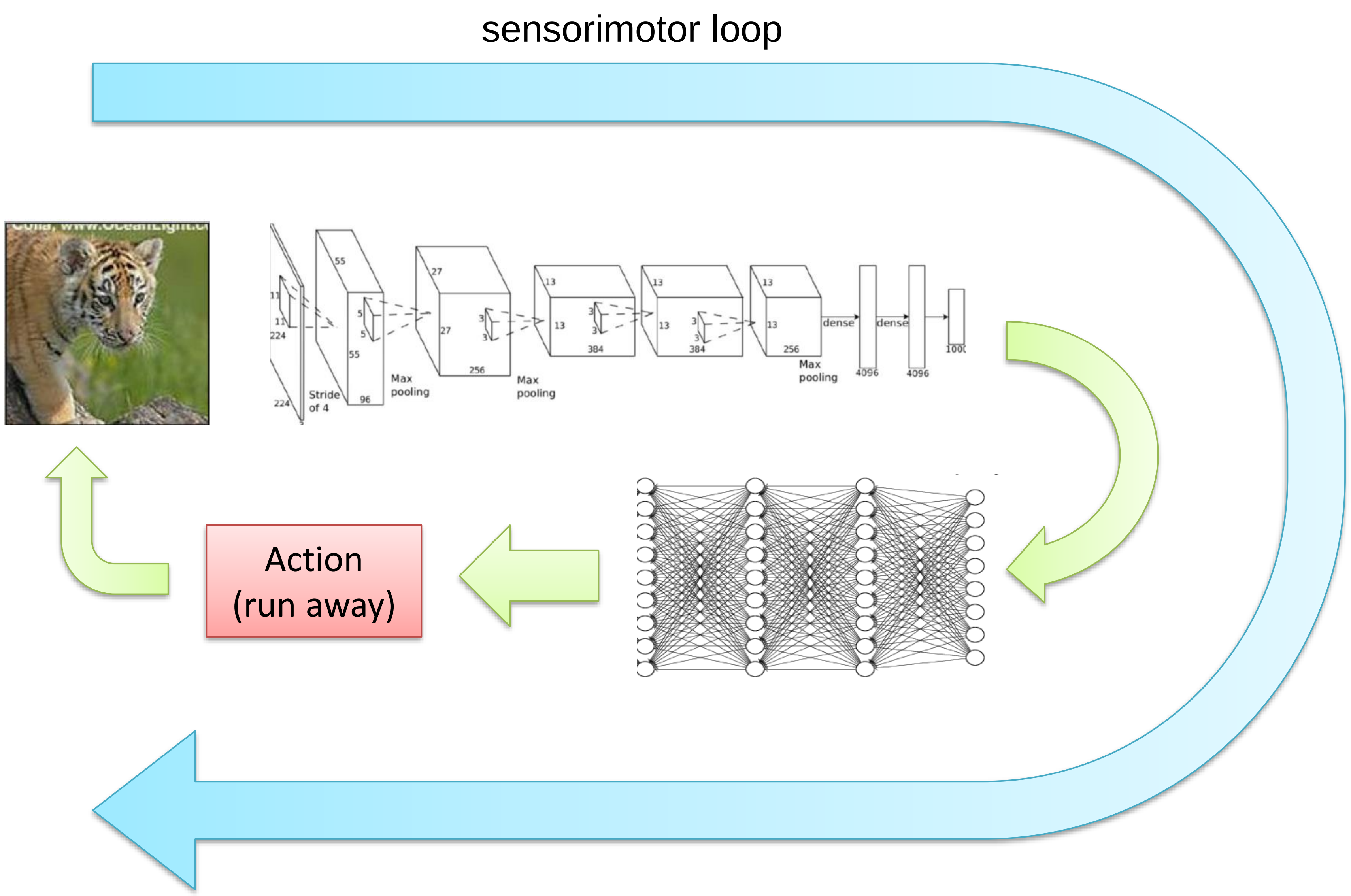


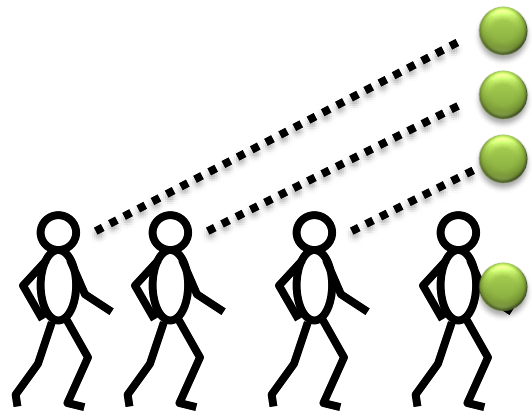


Krizhevsky '12




perception
tiger
tiger cat
jaguar
lynx
Action
(run away)
action


sensorimotor loop
Stride of 4
Max pooling
Max pooling
dense
dense
Max pooling
Action
(run away)



“When a man throws a ball high in the air and catches it again, he behaves as if he had solved a set of differential equations in predicting the trajectory of the ball … at some subconscious level, something functionally equivalent to the mathematical calculations is going on.”

-- Richard Dawkins

McLeod & Dienes. Do fielders know where to go to catch the ball or only how to get there? Journal of Experimental Psychology 1996, Vol. 22, No. 3, 531-543

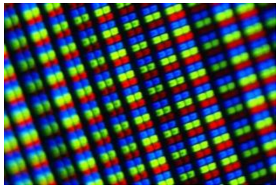
KAIST’s DRC-HUBO opening a door

DARPA Robotics Challenge 2015

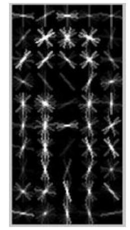
Philipp Krahenbuhl, Stanford University

End-to-end vision

standard computer vision

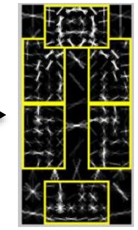


features (e.g. HOG)



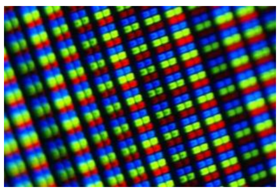
mid-level features (e.g. DPM)

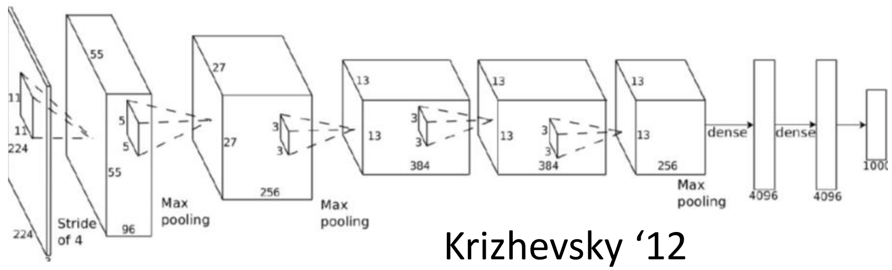
Felzenszwalb ‘08



classifier (e.g. SVM)

deep learning





Krizhevsky ‘12

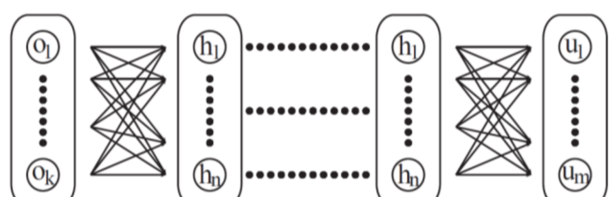
End-to-end robotic control

standard robotic control

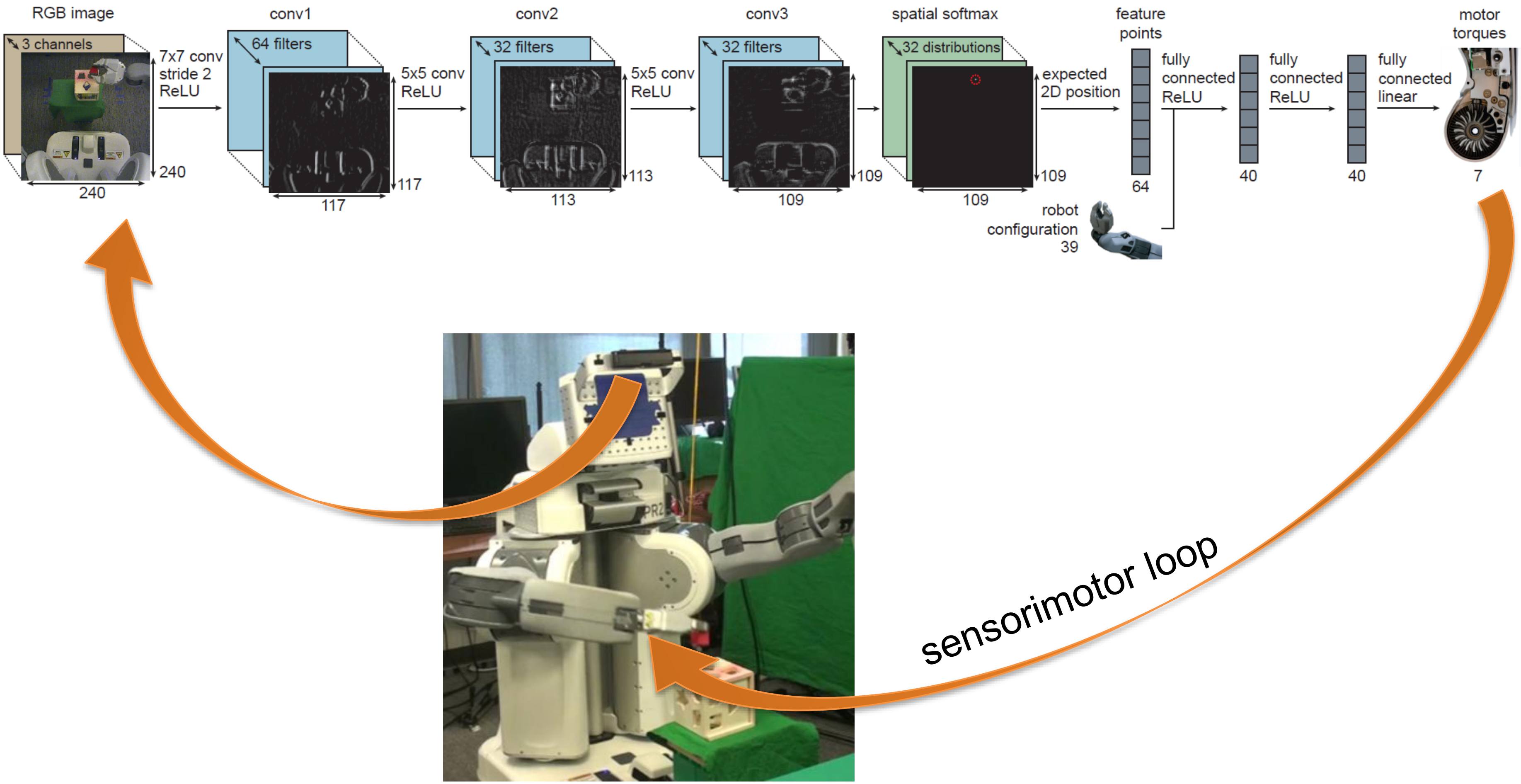
observations → state estimation (e.g. vision) → modeling & prediction → motion planning → low-level controller (e.g. PD) → motor torques

deep sensorimotor learning

observations →



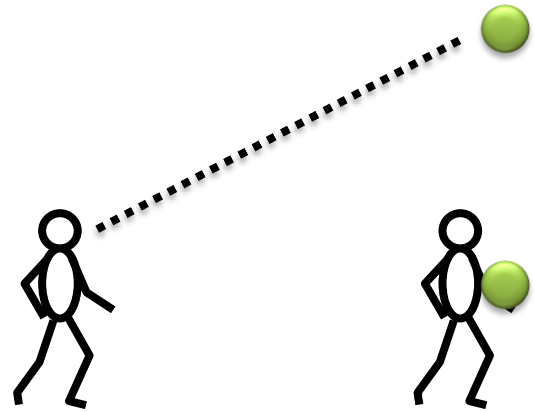
→ motor torques

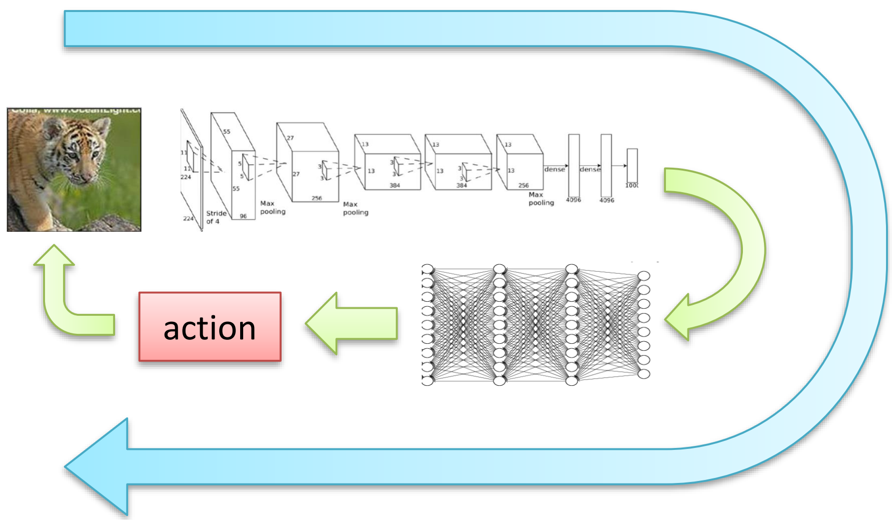


indirect supervision

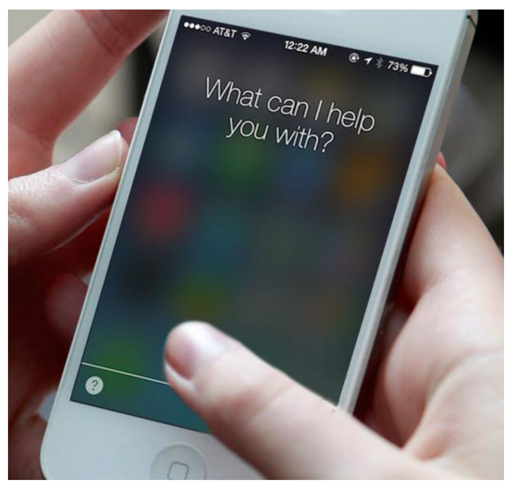
actions have consequences

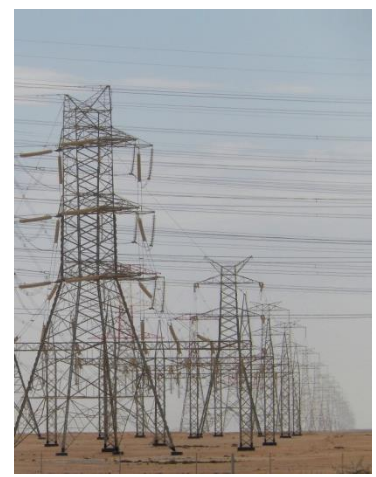
Why should we care?





Why should we care?

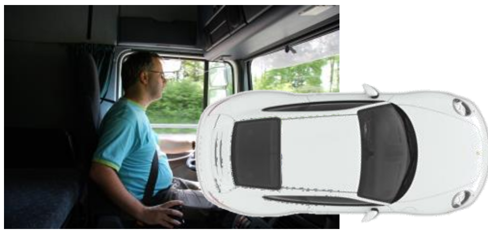




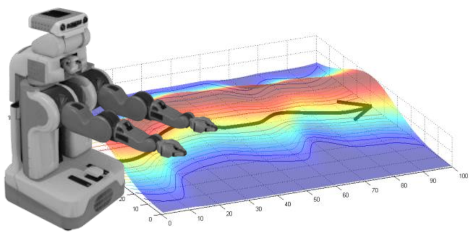
Goals of this lecture

- Introduce formalisms of decision making
  - states, actions, time, cost
  - Markov decision processes
- Survey recent research
  - imitation learning
  - reinforcement learning
- Outstanding research challenges
  - what is easy
  - what is hard
  - where could we go next?

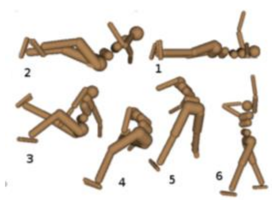
Contents



Imitation learning



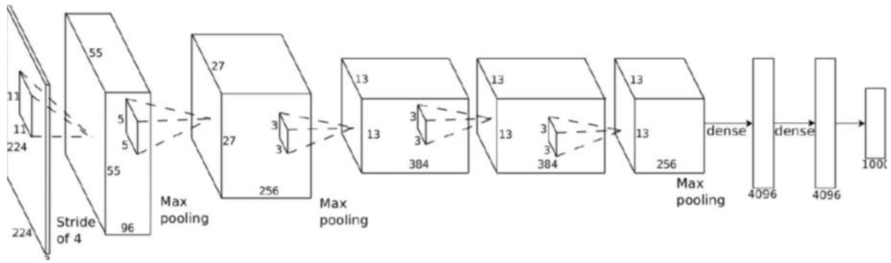
Imitation without a human

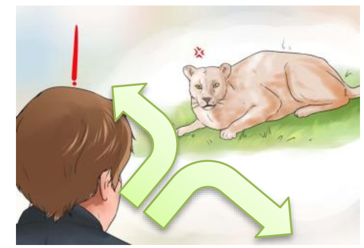


Reinforcement learning

Research frontiers

Terminology & notation



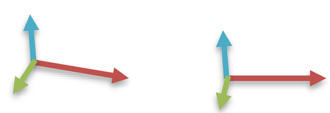


$\mathbf{o}_t$ $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ $\mathbf{u}_t$

$\mathbf{x}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{u}_t$ – action

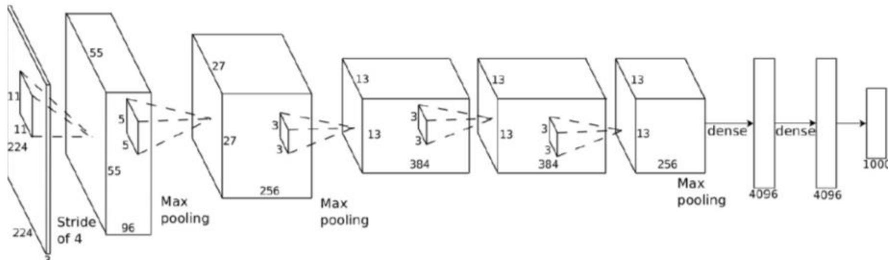
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ – policy

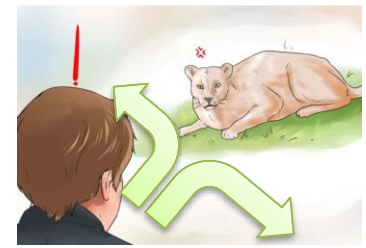
$\mathbf{o}_t$ – observation



$\mathbf{x}_t$ – state

Terminology & notation

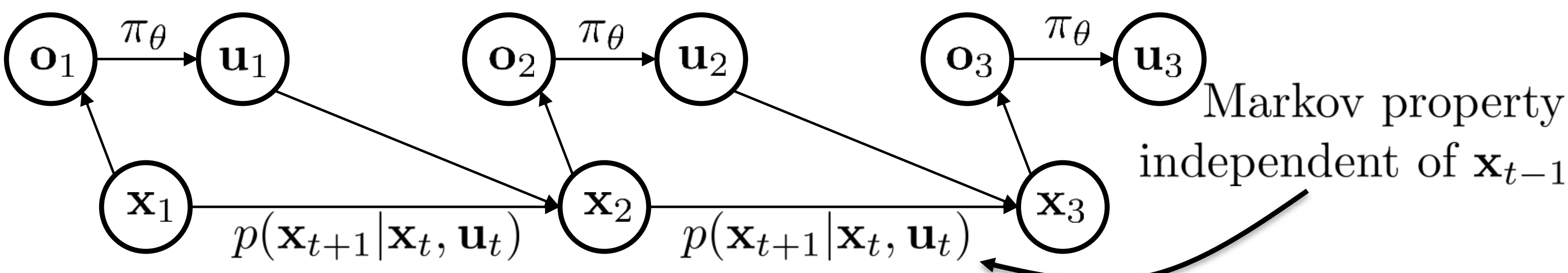




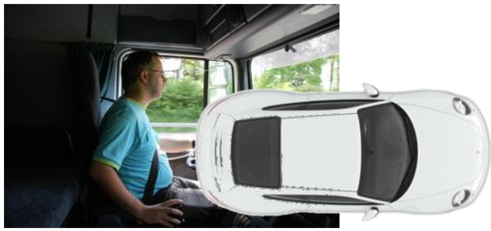
$\mathbf{o}_t$ $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ $\mathbf{u}_t$

$\mathbf{x}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{u}_t$ – action

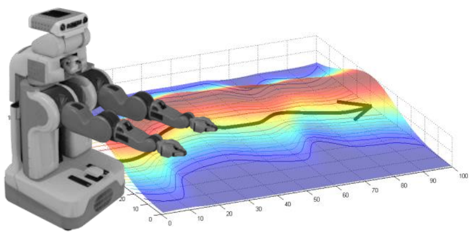
$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ – policy



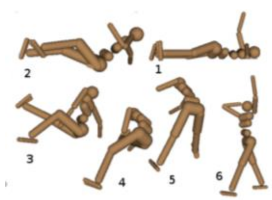
Contents



Imitation learning



Imitation without a human

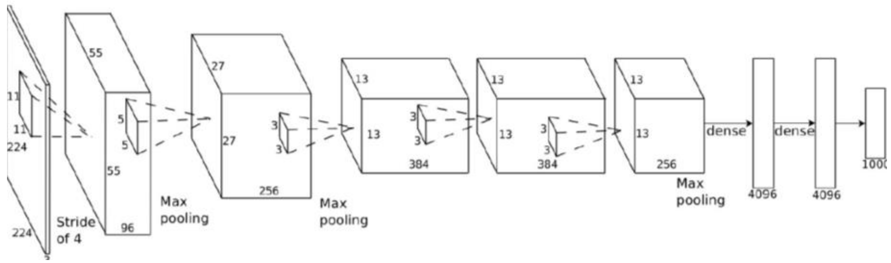


Reinforcement learning

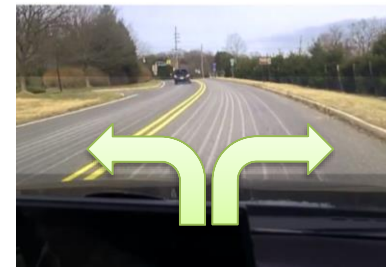
Research frontiers

Imitation Learning

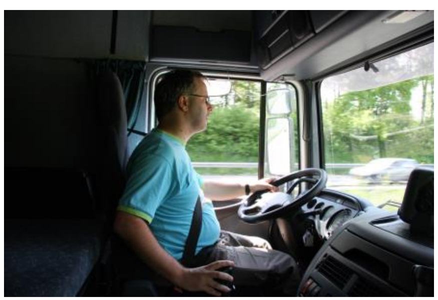
$\mathbf{o}_t$



$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$



$\mathbf{u}_t$



$\mathbf{o}_t$

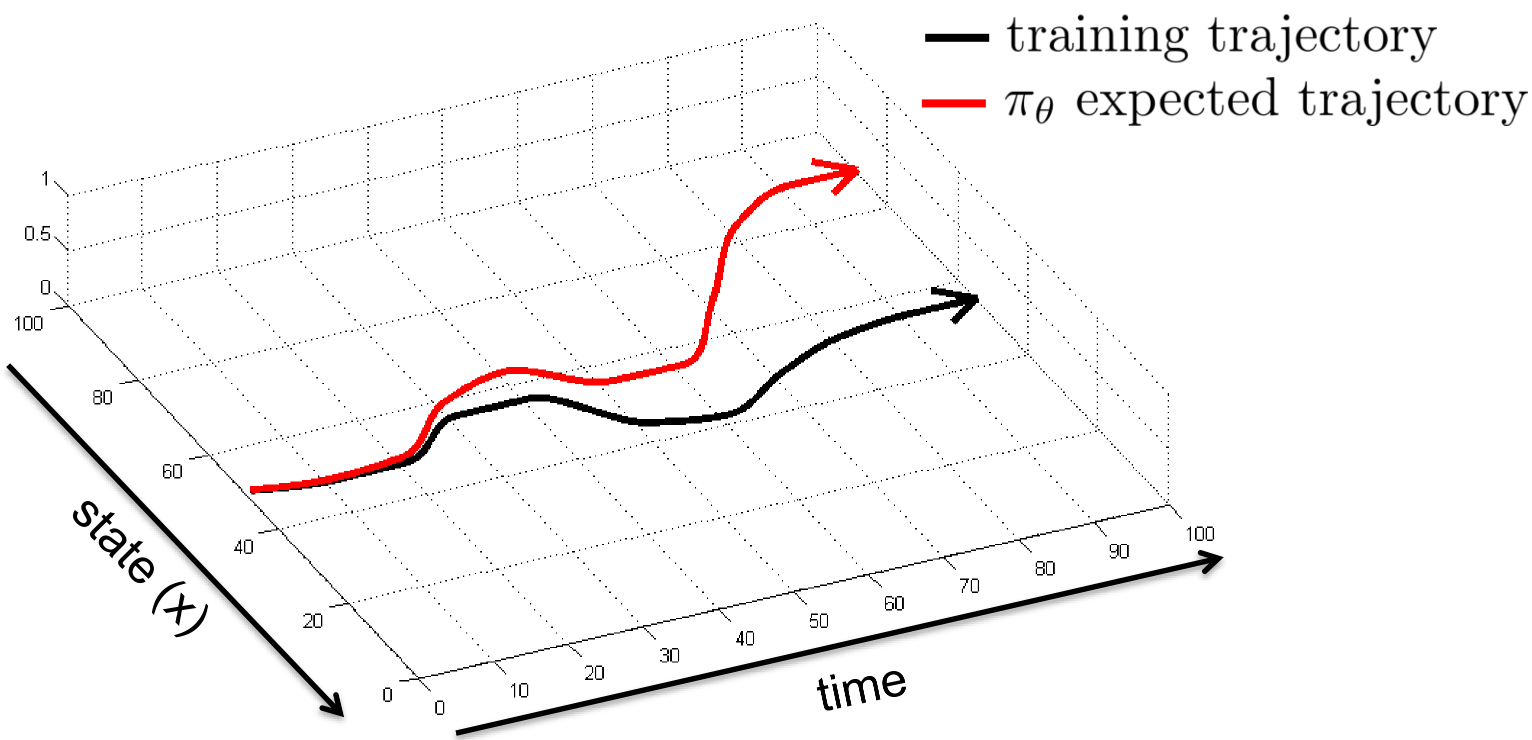
$\mathbf{u}_t$

→ training data → supervised learning

$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

Images: Bojarski et al. ‘16, NVIDIA

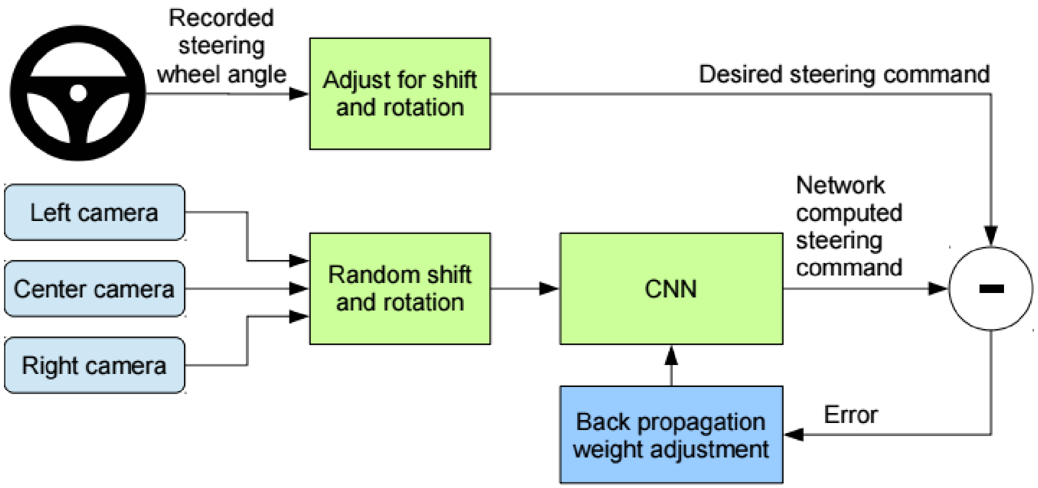
Does it work? No!

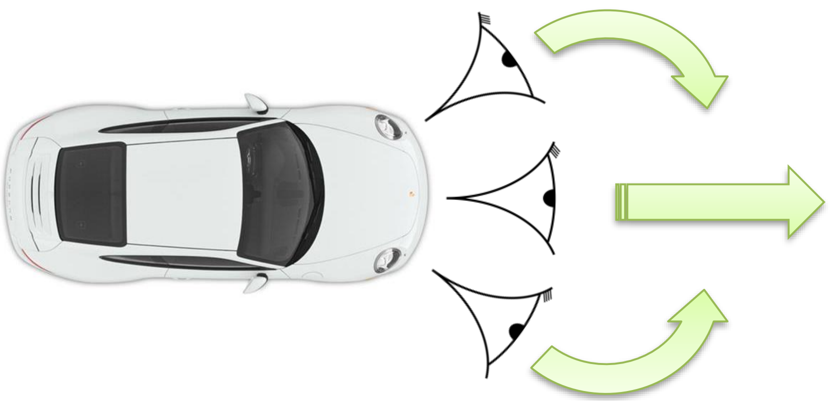


Does it work? Yes!

Video: Bojarski et al. ‘16, NVIDIA

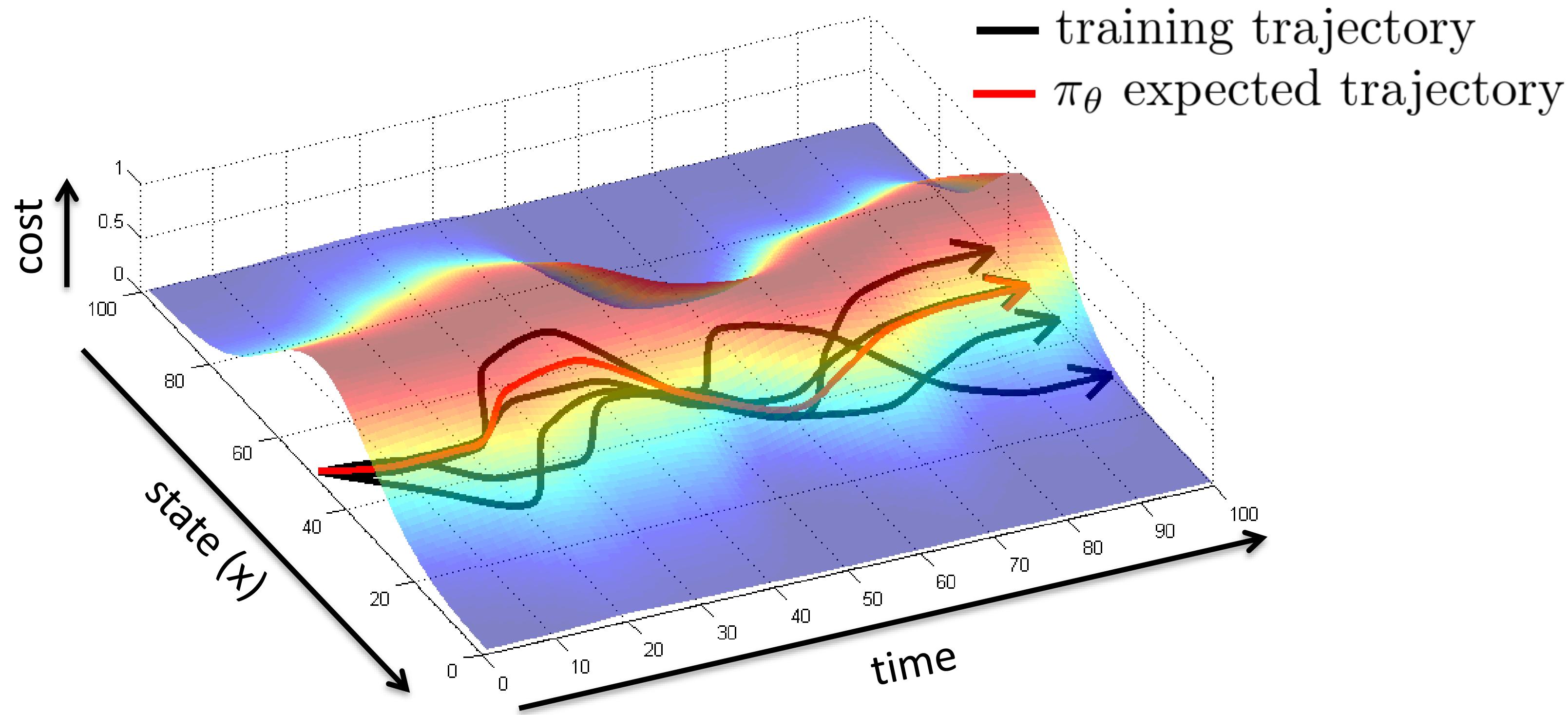
Why did that work?





Bojarski et al. '16, NVIDIA

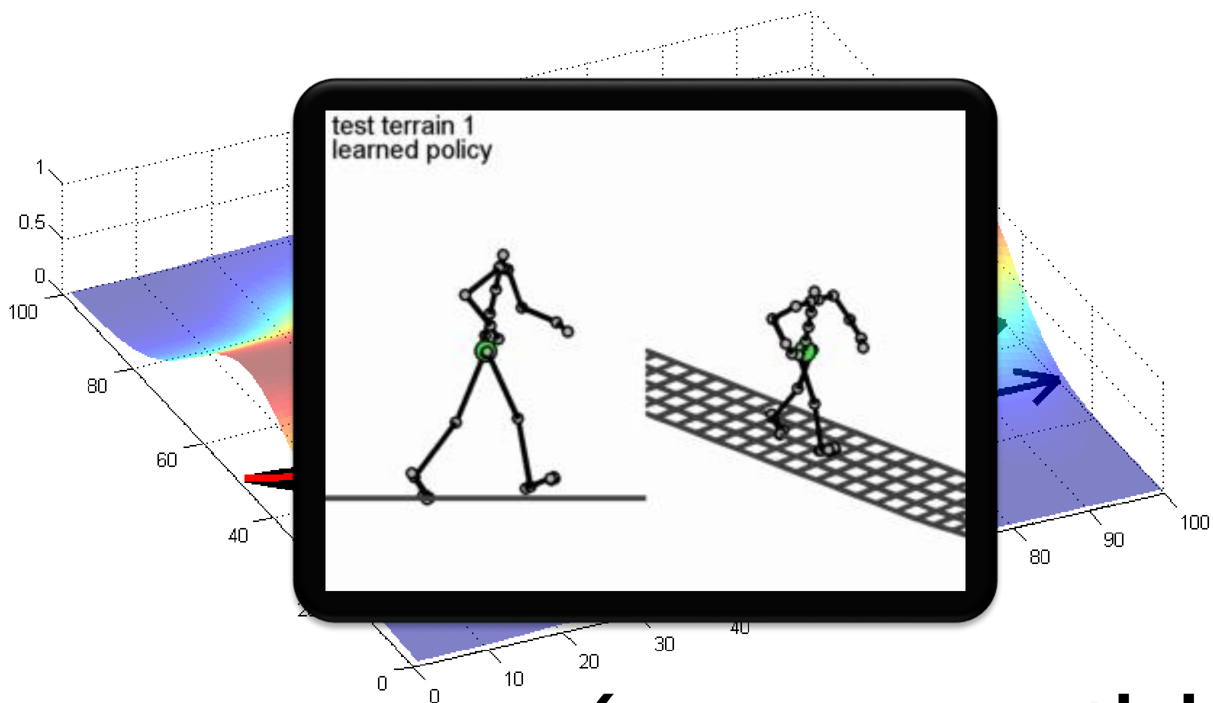
Can we make it work more often?



stability

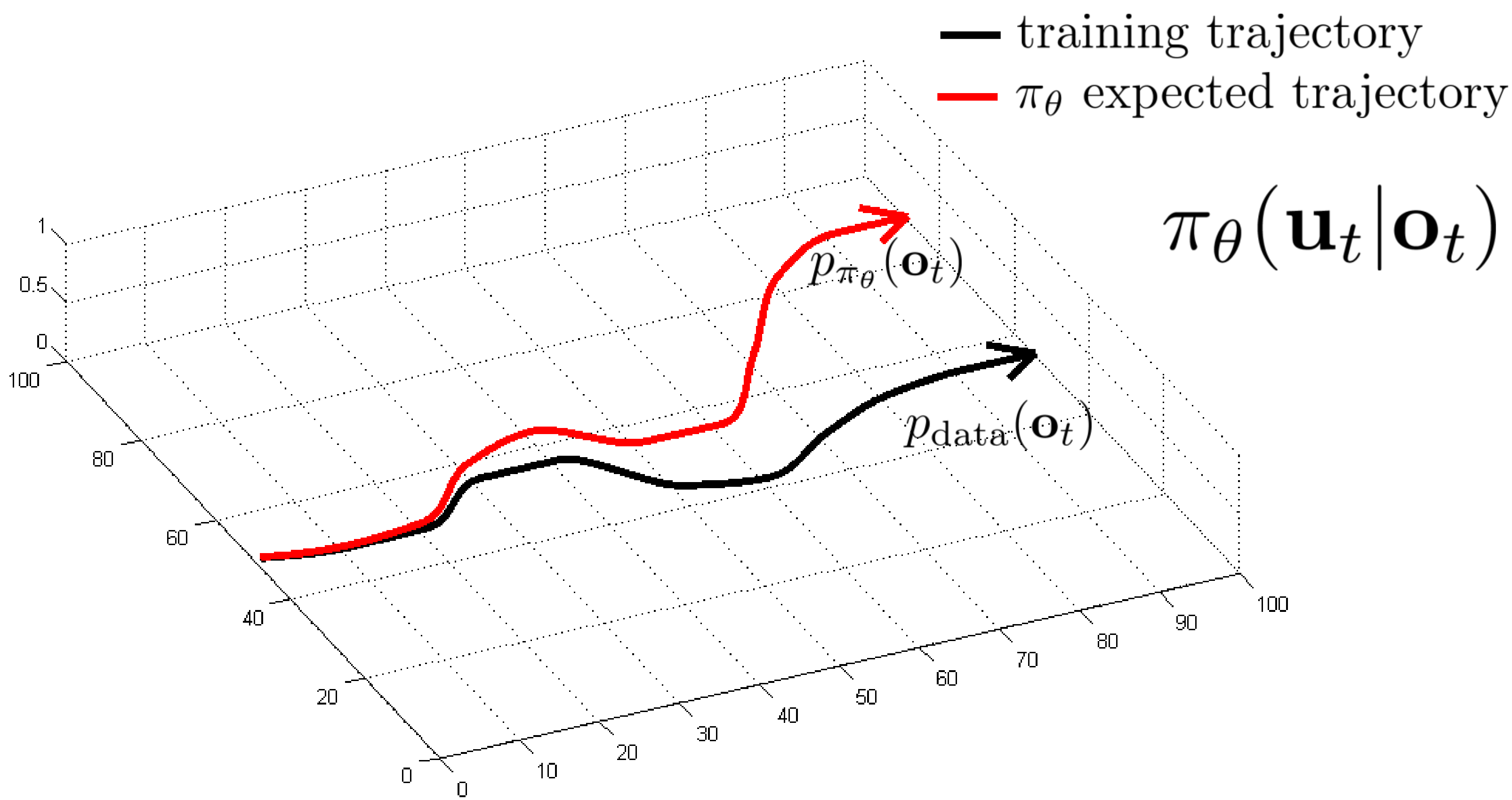
Learning from a stabilizing controller

$p(\underbrace{\mathbf{x}_1, \mathbf{u}_1, \mathbf{x}_2, \mathbf{u}_2, \ldots, \mathbf{x}_T, \mathbf{u}_T}_{\tau})$ Gaussian distribution obtained using variant of iterative LQR



(more on this later)

Can we make it work more often?



$$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$$

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

Can we make it work more often?

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

idea: instead of being clever about $p_{\pi_\theta}(\mathbf{o}_t)$, be clever about $p_{\text{data}}(\mathbf{o}_t)$!

DAgger: **D**ataset **Agg**regation

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

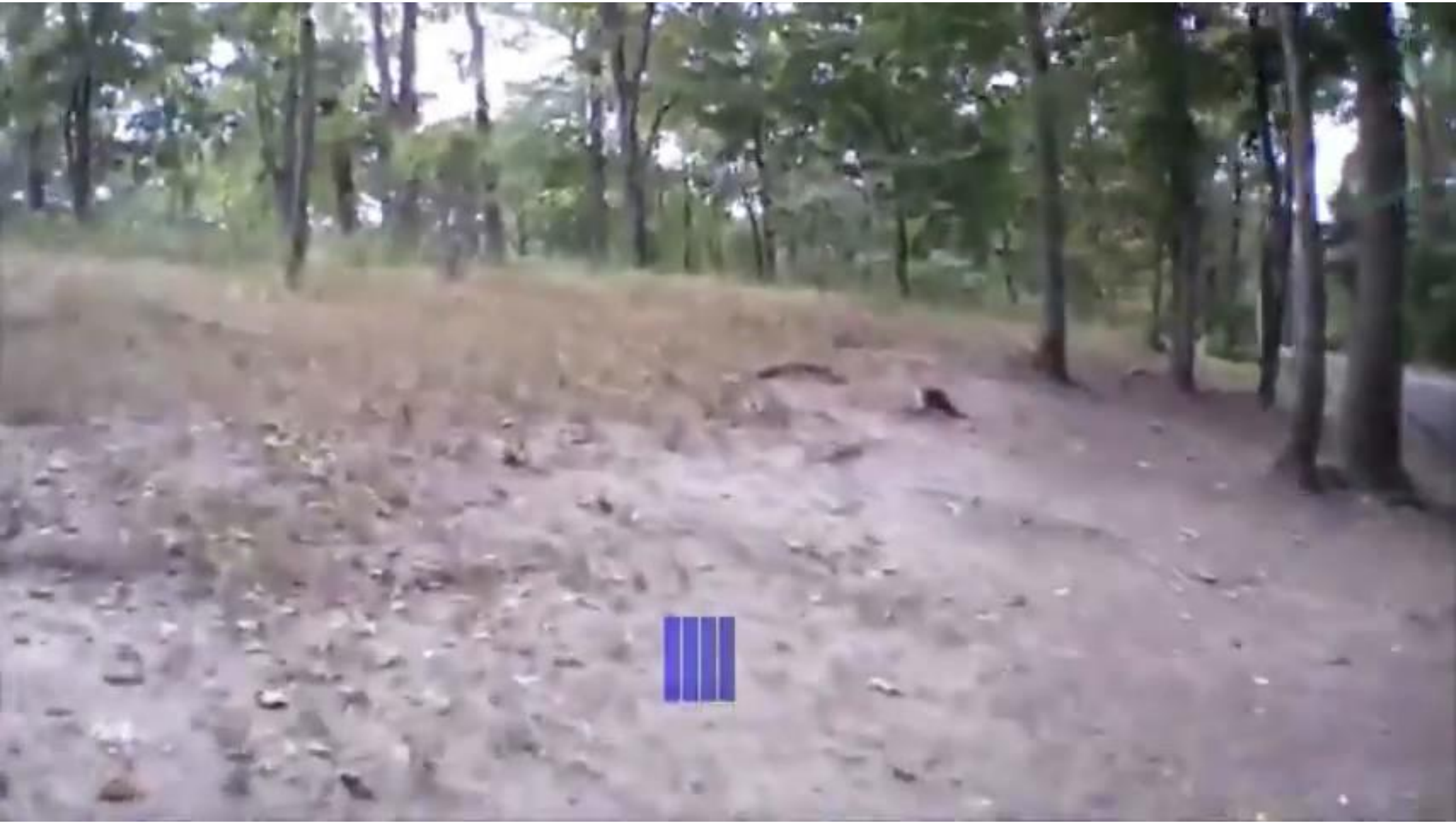
how? just run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

but need labels $\mathbf{u}_t$!

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

Ross et al. '11

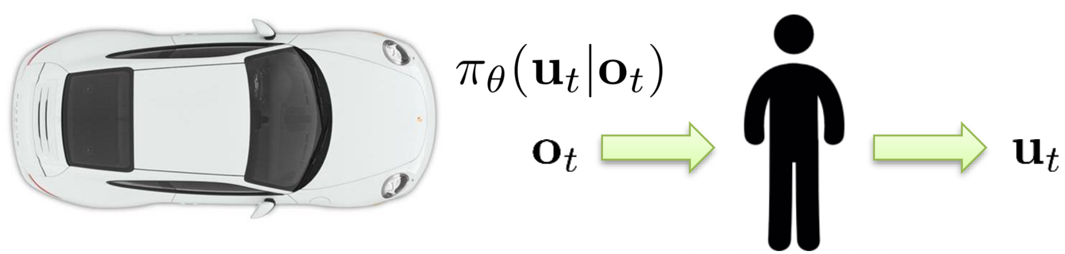
DAgger Example



Ross et al. '11

What's the problem?

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

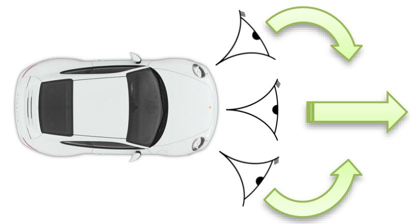


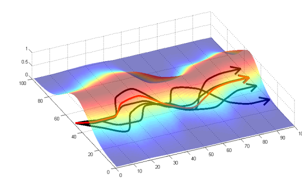
Ross et al. '11

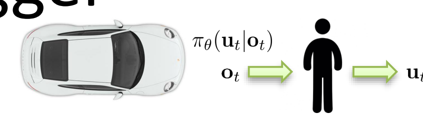
Imitation learning: recap

$\mathbf{o}_t$ $\mathbf{u}_t$ → training data → supervised learning $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

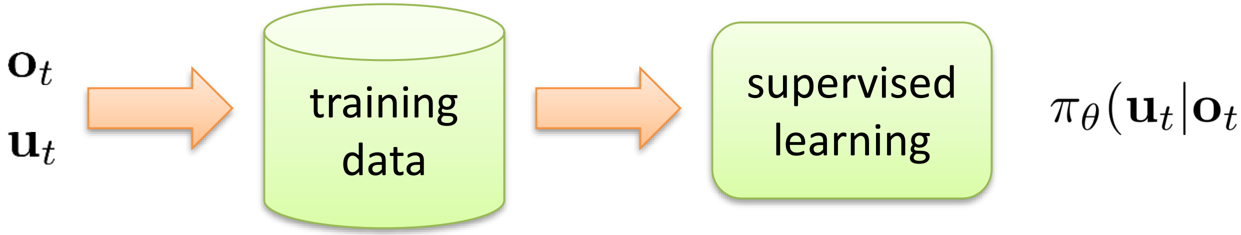
- Usually (but not always) insufficient by itself
  - Distribution mismatch problem
- Sometimes works well
  - Hacks (e.g. left/right images)
  - Samples from a stable trajectory distribution
  - Add more **on-policy** data, e.g. using DAgger



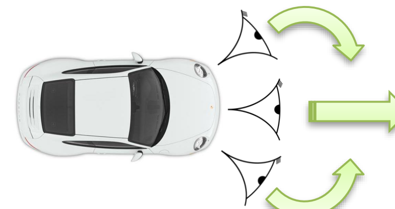




Imitation learning: questions

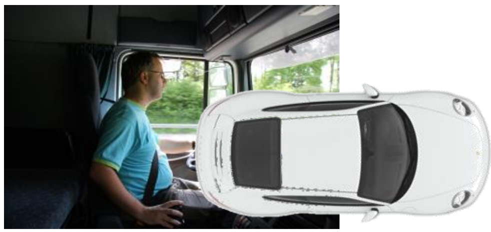


- Distribution mismatch does not seem to be the whole story
  - Imitation often works without Dagger
  - Can we think about how stability factors in?

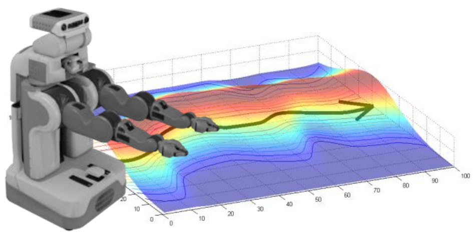


- Do we need to **add** data for imitation to work?
  - Can we just use existing data more cleverly?

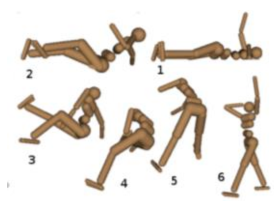
Contents



Imitation learning



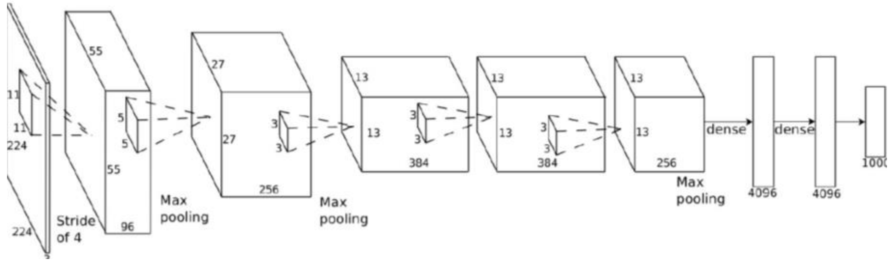
Imitation without a human

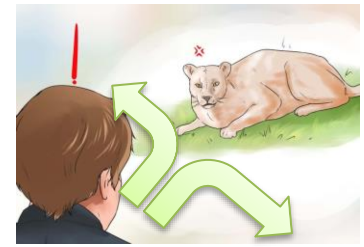


Reinforcement learning

Research frontiers

Terminology & notation





$\mathbf{o}_t$ $\qquad\qquad$ $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ $\qquad\qquad$ $\mathbf{u}_t$

$\mathbf{x}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{u}_t$ – action

$c(\mathbf{x}_t, \mathbf{u}_t)$ – cost function

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \text{ s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} \log p(\text{eaten by tiger}|\mathbf{u}_1, \ldots, \mathbf{u}_T)$$

Trajectory optimization

$$\min_{\mathbf{u}_1,\dots,\mathbf{u}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \text{ s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$

$$\min_{\mathbf{u}_1,\dots,\mathbf{u}_T} c(\mathbf{x}_1, \mathbf{u}_1) + c(f(\mathbf{x}_1, \mathbf{u}_1), \mathbf{u}_2) + \dots + c(f(f(\dots)\dots), \mathbf{u}_T)$$

usual story: differentiate via backpropagation and optimize!

need $\frac{df}{d\mathbf{x}_t}, \frac{df}{d\mathbf{u}_t}, \frac{dc}{d\mathbf{x}_t}, \frac{dc}{d\mathbf{u}_t}$

in practice, it really helps to use a 2^{nd} order method!

see differential dynamic programming (DDP) and iterative LQR (iLQR)

Probabilistic version

deterministic dynamics: $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$

stochastic dynamics: $\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$

simple stochastic dynamics: $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f(\mathbf{x}_t, \mathbf{u}_t), \Sigma)$

simple stochastic policy: $p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\min_{\mathbf{u}_1,\dots,\mathbf{u}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t)$$

$$\min_{\mathbf{K}_1,\mathbf{k}_1,\Sigma_{\mathbf{u}_1},\dots,\mathbf{K}_T,\mathbf{k}_T,\Sigma_{\mathbf{u}_T}} \sum_{t=1}^{T} E_{(\mathbf{x}_t,\mathbf{u}_t)\sim p(\mathbf{x}_t,\mathbf{u}_t)}[c(\mathbf{x}_t, \mathbf{u}_t)]$$

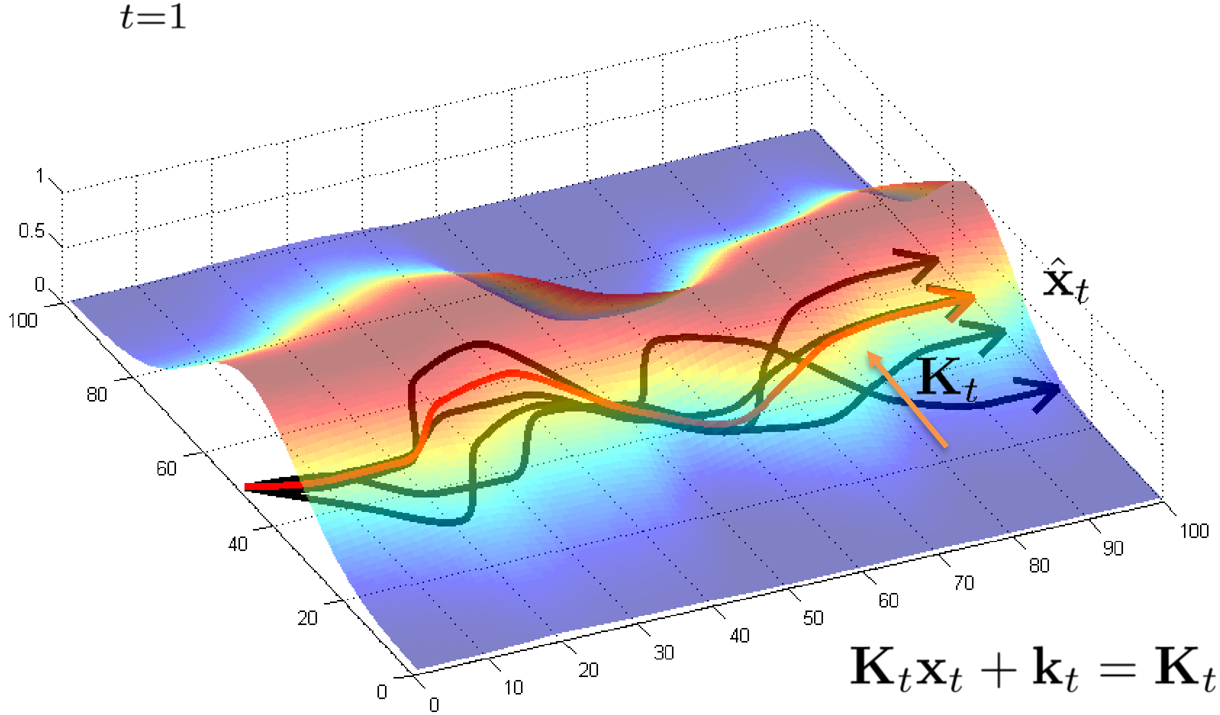
$$p(\tau) = p(\mathbf{x}_1) \prod_{t=1}^{T} p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) p(\mathbf{u}_t|\mathbf{x}_t)$$

Probabilistic version (in pictures)

simple stochastic dynamics: $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f(\mathbf{x}_t, \mathbf{u}_t), \Sigma)$

simple stochastic policy: $p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

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



$$\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t = \mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t) + \hat{\mathbf{u}}_t$$

DAgger without Humans

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

$\mathbf{o}_t$

$\mathbf{u}_t$

Ross et al. '11

Another problem

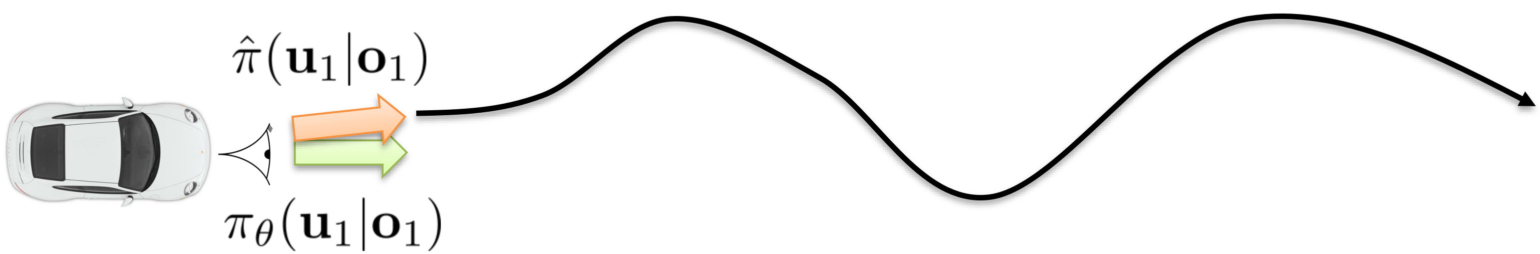
1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



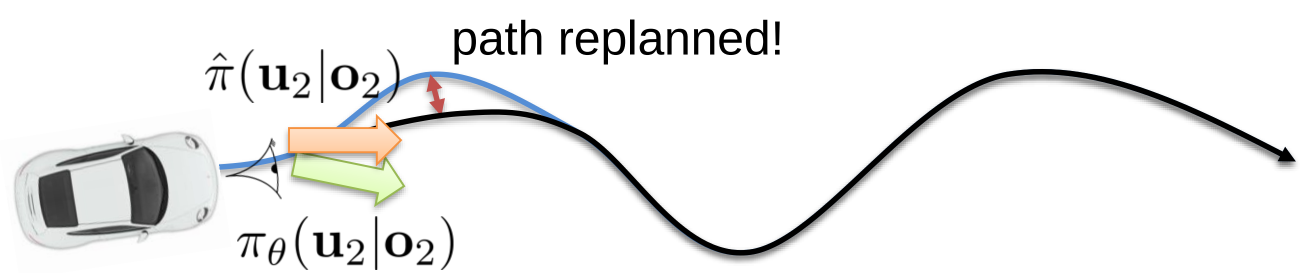
Kahn, Zhang, Levine, Abbeel '16

PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



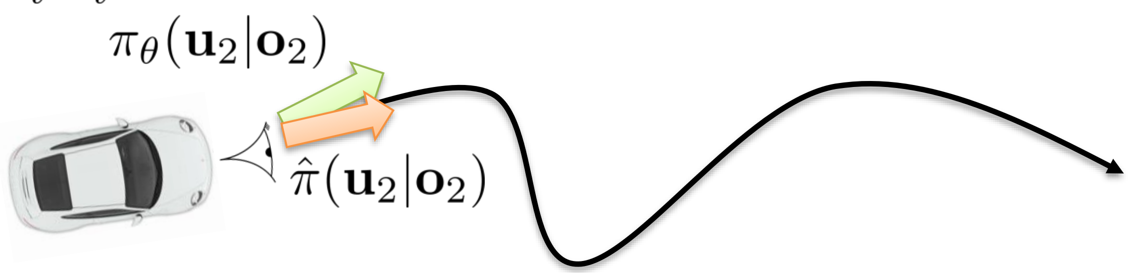
Kahn, Zhang, Levine, Abbeel ‘16

PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



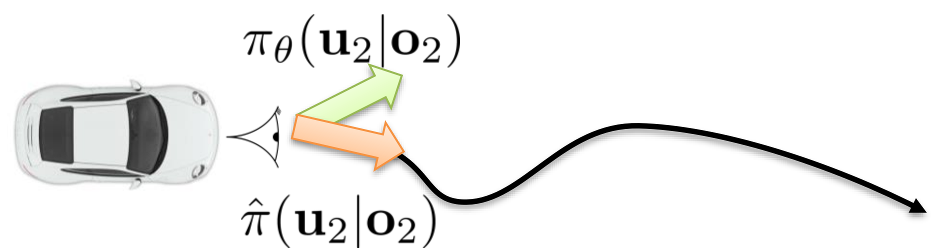
Kahn, Zhang, Levine, Abbeel '16

PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



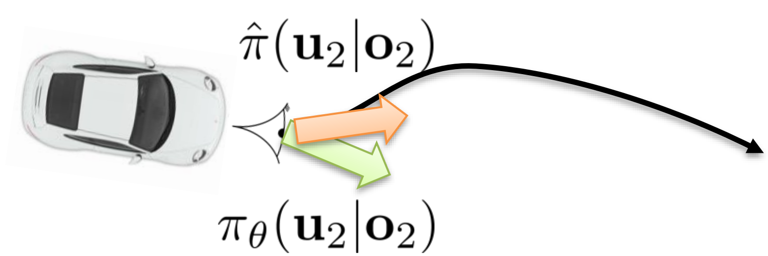
Kahn, Zhang, Levine, Abbeel ‘16

PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



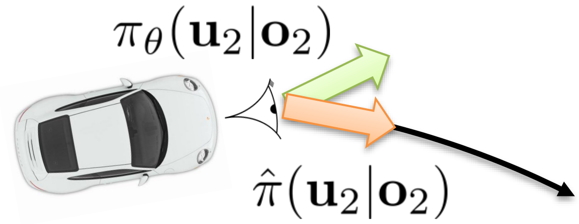
Kahn, Zhang, Levine, Abbeel ‘16

PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



Kahn, Zhang, Levine, Abbeel '16

PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

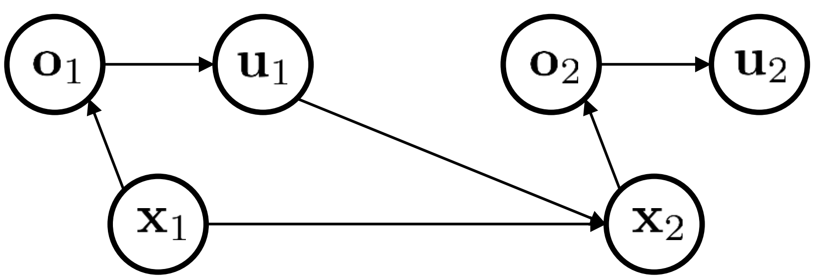
simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$

replanning = **M**odel **P**redictive **C**ontrol (MPC)

$\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ – control from **images**

$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t)$ – control from **states**



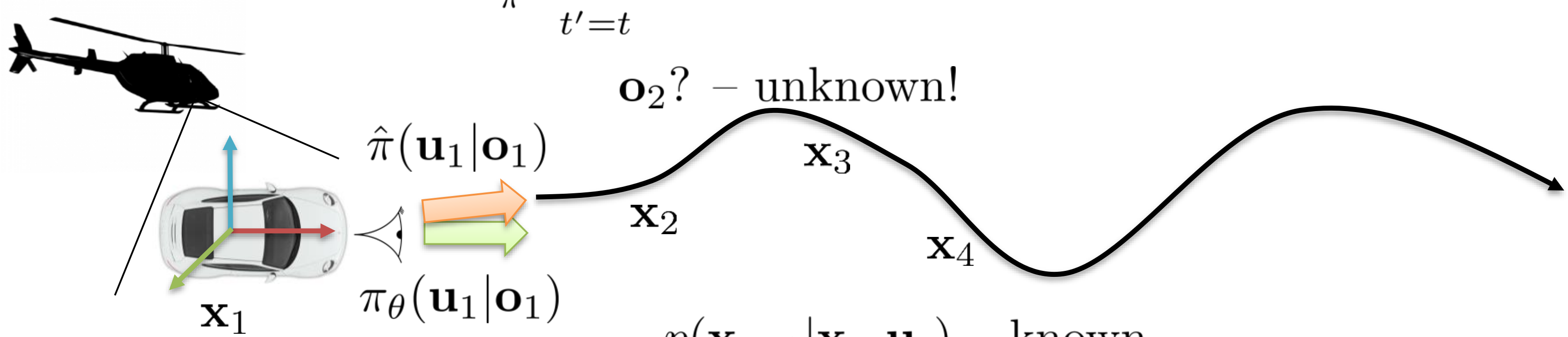
Kahn, Zhang, Levine, Abbeel '16

PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$



$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ – known

$p(\mathbf{o}_t|\mathbf{x}_t)$ – unknown

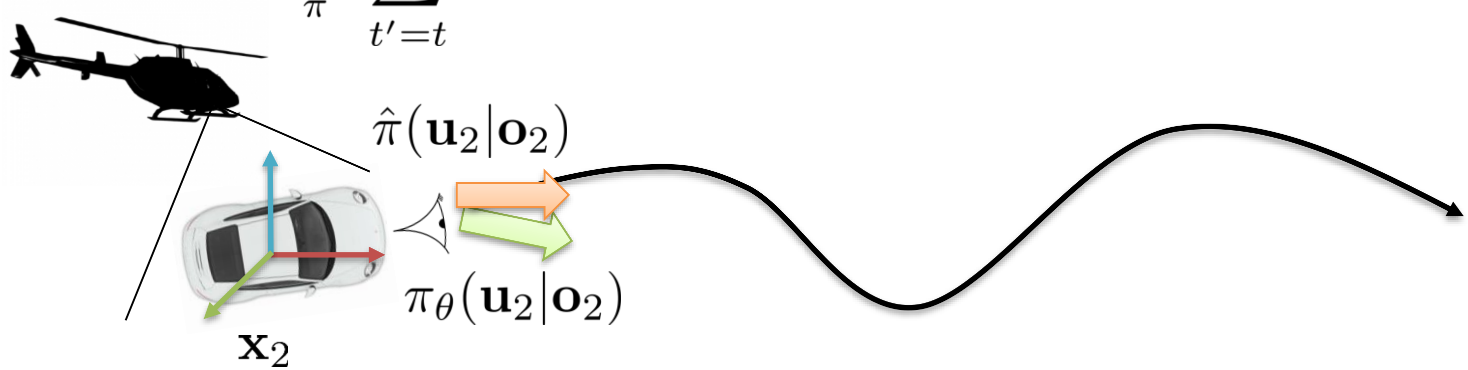
Kahn, Zhang, Levine, Abbeel '16

PLATO: Policy Learning with Adaptive Trajectory Optimization

1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \ldots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\hat{\pi}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \ldots, \mathbf{o}_M\}$
3. Ask computer to label $\mathcal{D}_\pi$ with actions $\mathbf{u}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

simple stochastic policy: $\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \Sigma_{\mathbf{u}_t})$

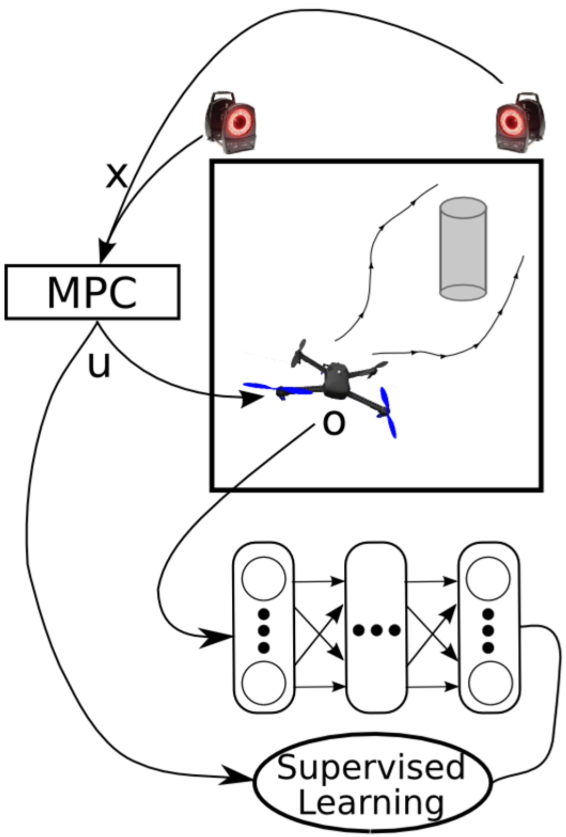
$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})] + \lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))$$

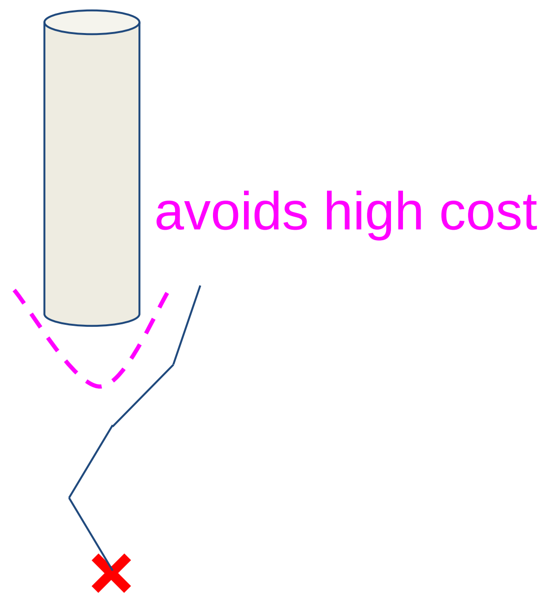


Kahn, Zhang, Levine, Abbeel ‘16

PLATO: Policy Learning with Adaptive Trajectory Optimization

$$\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) = \arg\min_{\hat{\pi}} \sum_{t'=t}^{T} \underline{E_{\hat{\pi}}[c(\mathbf{x}_{t'}, \mathbf{u}_{t'})]} + \underline{\lambda D_{\mathrm{KL}}(\hat{\pi}(\mathbf{u}_t|\mathbf{x}_t) \| \pi_\theta(\mathbf{u}_t|\mathbf{o}_t))}$$

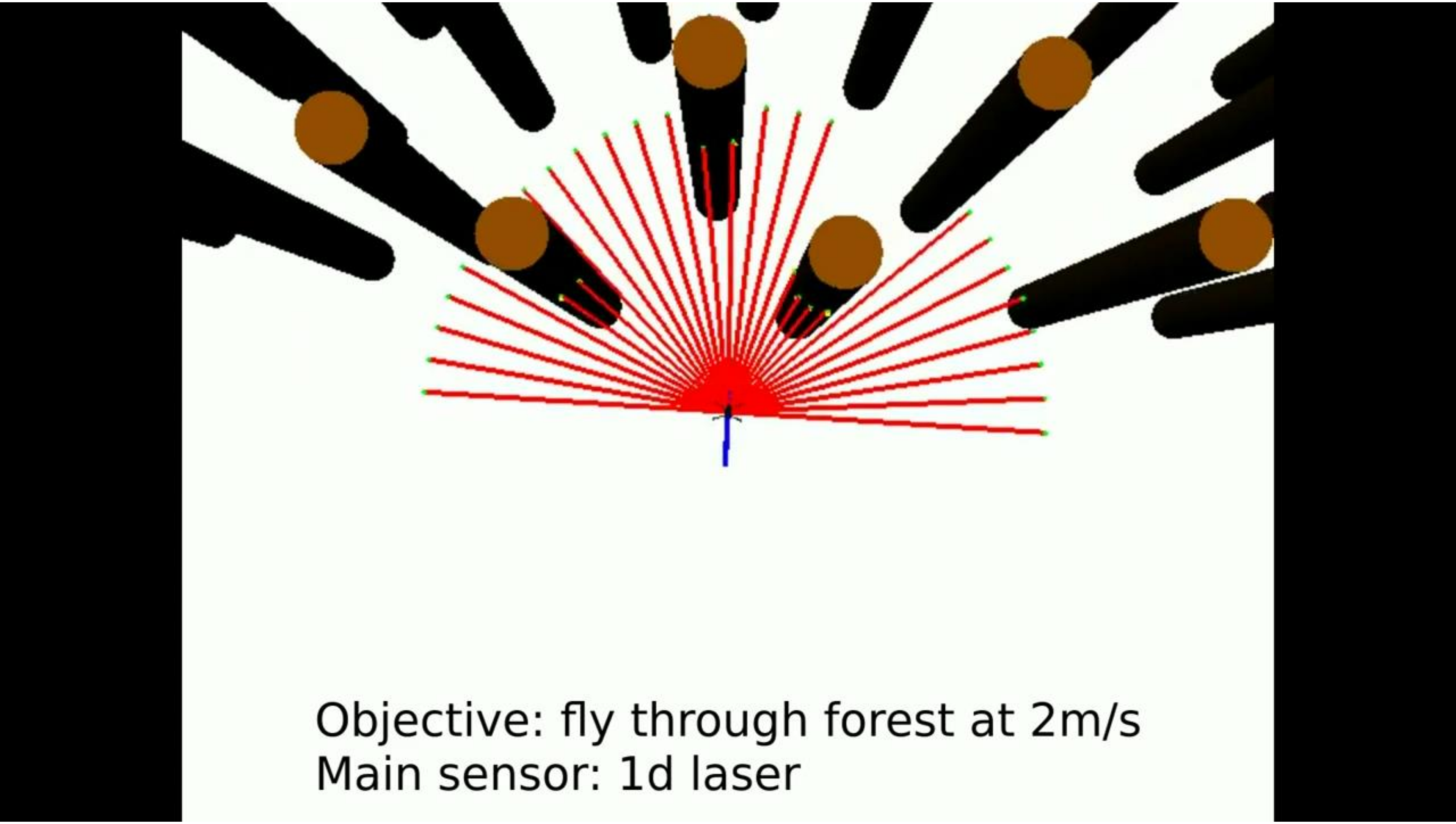




input substitution trick
need state at training time
but not at test time!

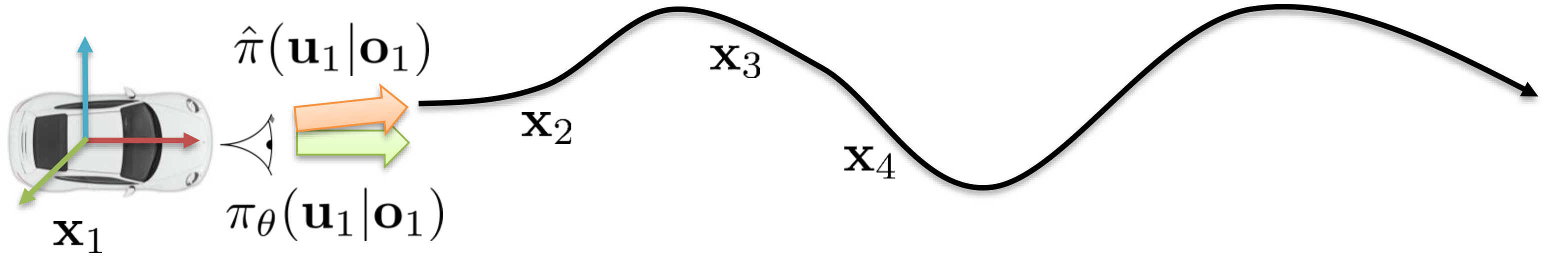
Kahn, Zhang, Levine, Abbeel '16

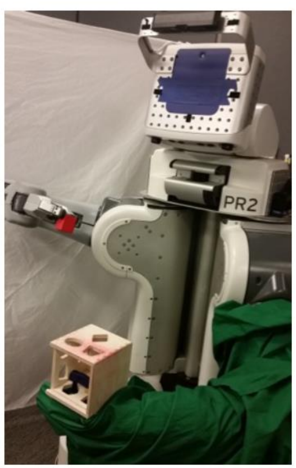
PLATO: Policy Learning with Adaptive Trajectory Optimization

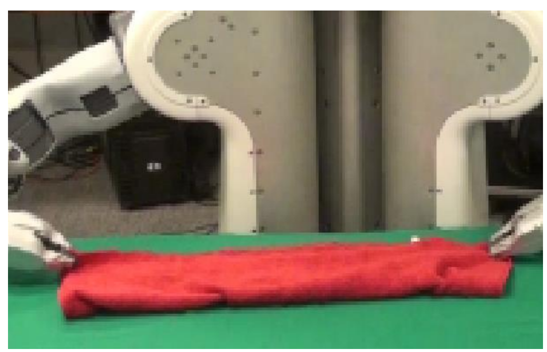


Kahn, Zhang, Levine, Abbeel '16

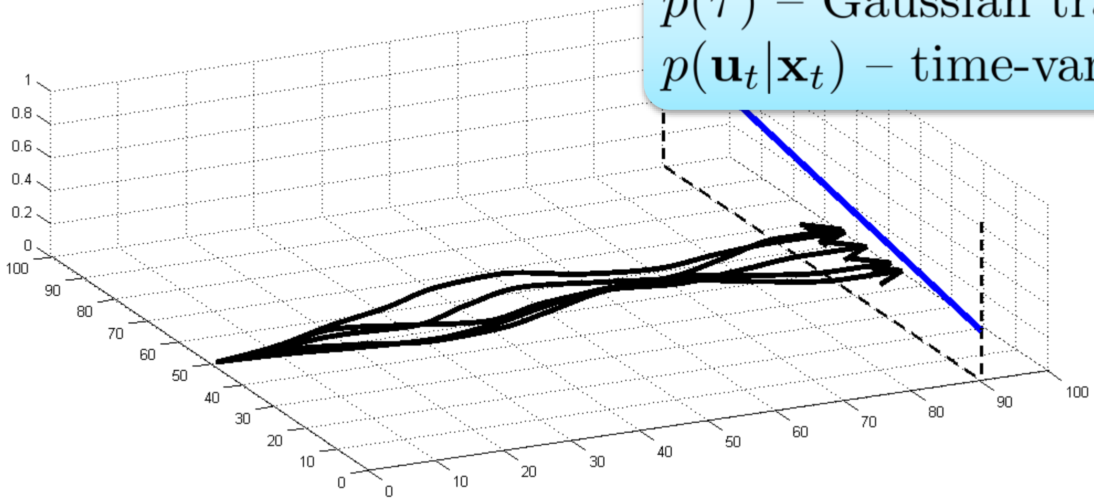
Beyond driving & flying







Trajectory Optimization with Unknown Dynamics



$p(\tau)$ – Gaussian trajectory distribution

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

can **execute** on the robot!

$$\min_{\mathbf{u}_1,\dots,\mathbf{u}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \text{ s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$

$$\min_{\mathbf{u}_1,\dots,\mathbf{u}_T} c(\mathbf{x}_1, \mathbf{u}_1) + c(f(\mathbf{x}_1, \mathbf{u}_1), \mathbf{u}_2) + \cdots + c(f(f(\dots)\dots), \mathbf{u}_T)$$

need $\frac{df}{d\mathbf{x}_t}, \frac{df}{d\mathbf{u}_t}, \frac{dc}{d\mathbf{x}_t}, \frac{dc}{d\mathbf{u}_t}$

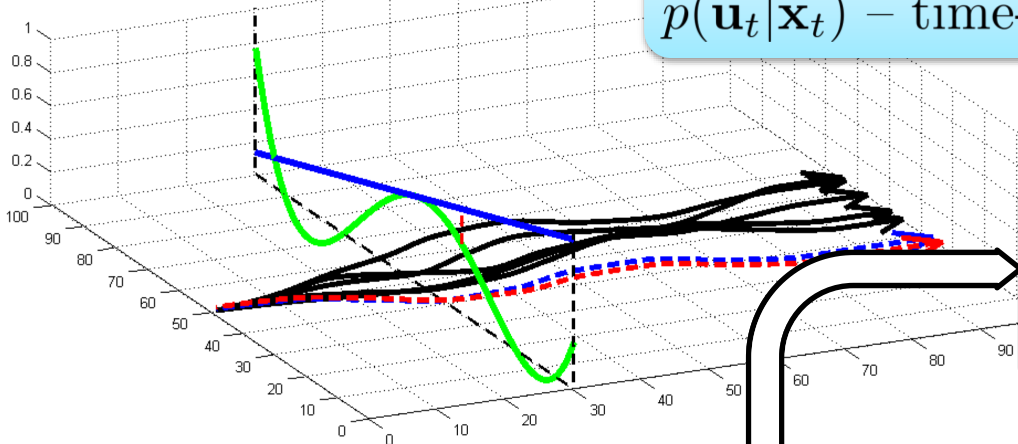
[L. et al. NIPS ‘14]

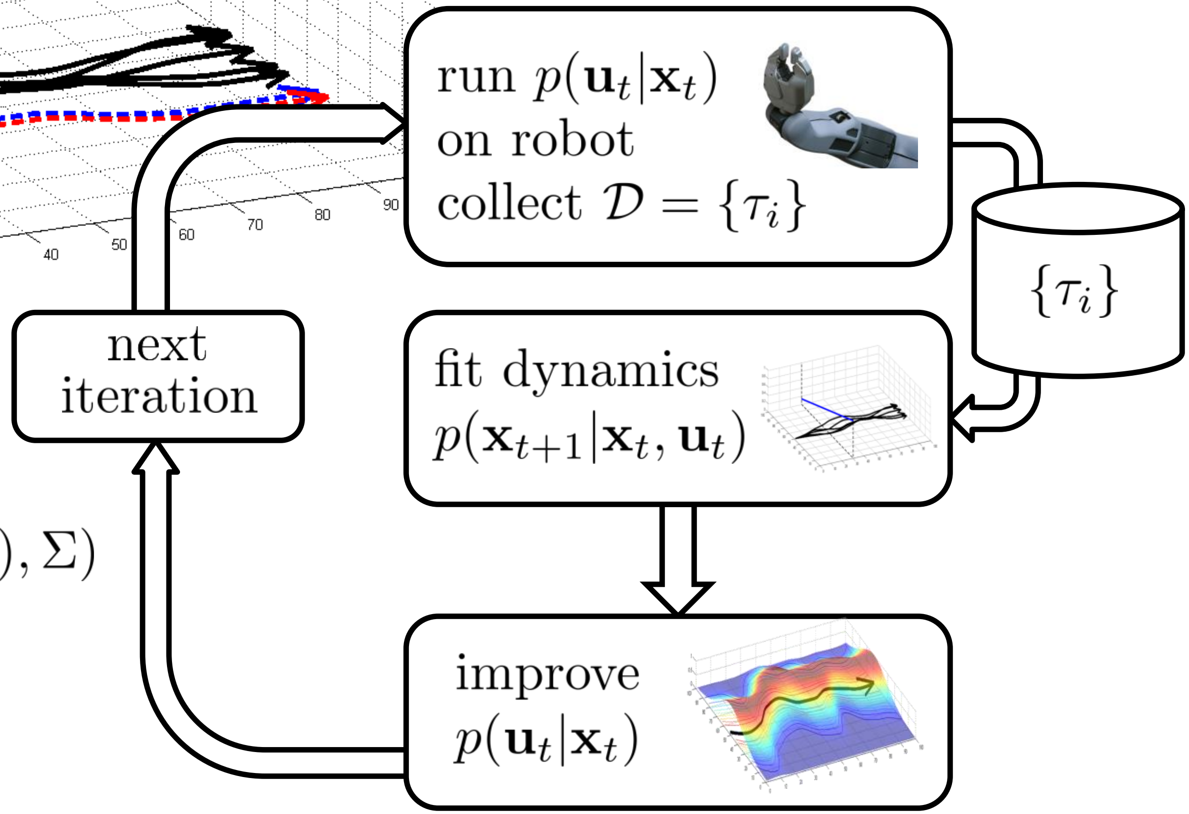
Trajectory Optimization with Unknown Dynamics

$p(\tau)$ – Gaussian trajectory distribution

$p(\mathbf{u}_t|\mathbf{x}_t)$ – time-varying linear-Gaussian <u>controller</u>

can **execute** on the robot!





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

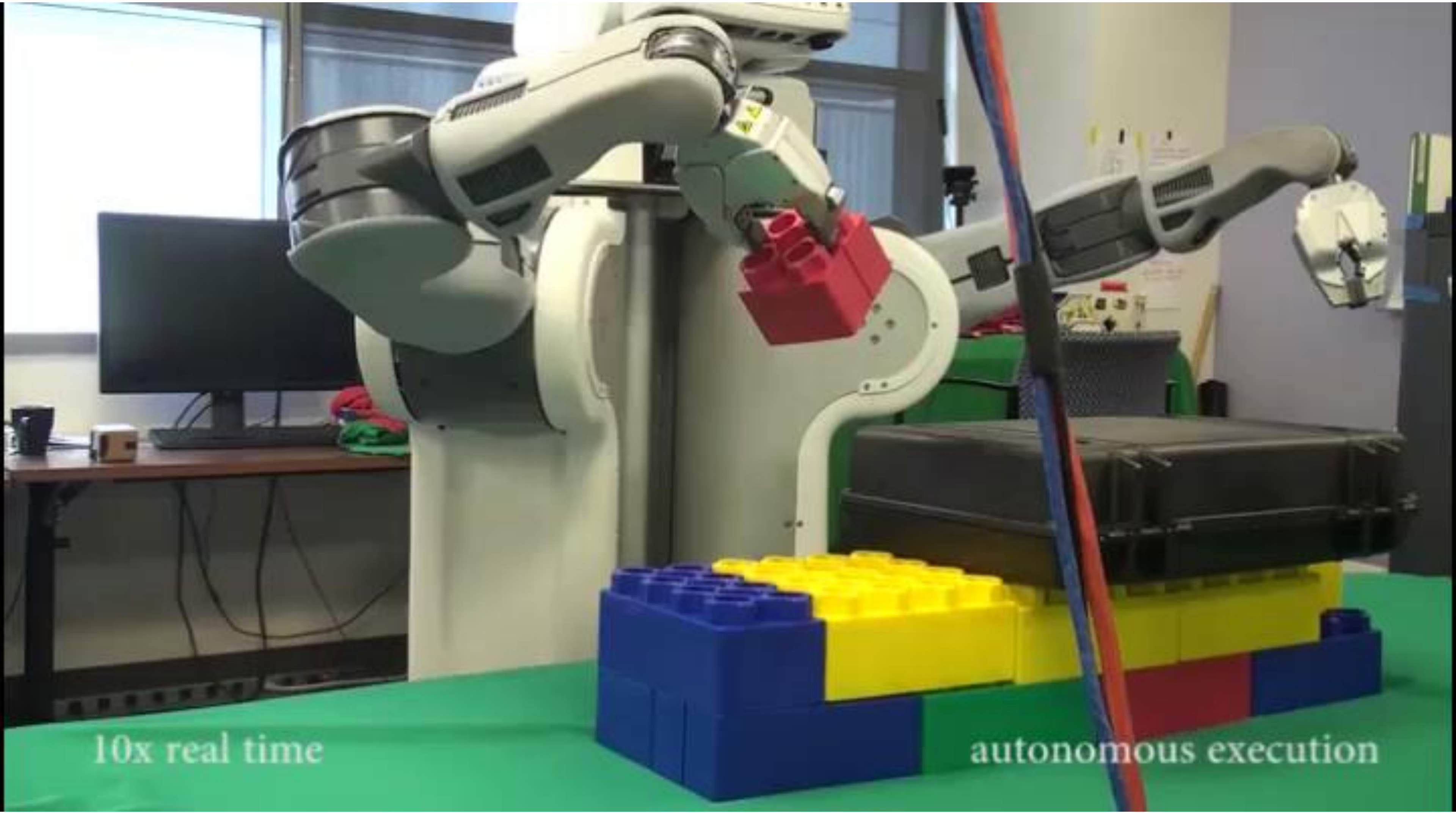
$$f(\mathbf{x}_t, \mathbf{u}_t) \approx \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t$$

$$\mathbf{A}_t = \frac{df}{d\mathbf{x}_t} \qquad \mathbf{B}_t = \frac{df}{d\mathbf{u}_t}$$

$$\min_{p(\tau)} E_p[c(\tau)] \text{ s.t. } D_{KL}(\underbrace{p(\tau)}_{\text{new}} \| \underbrace{\bar{p}(\tau)}_{\text{old}}) \leq \epsilon$$

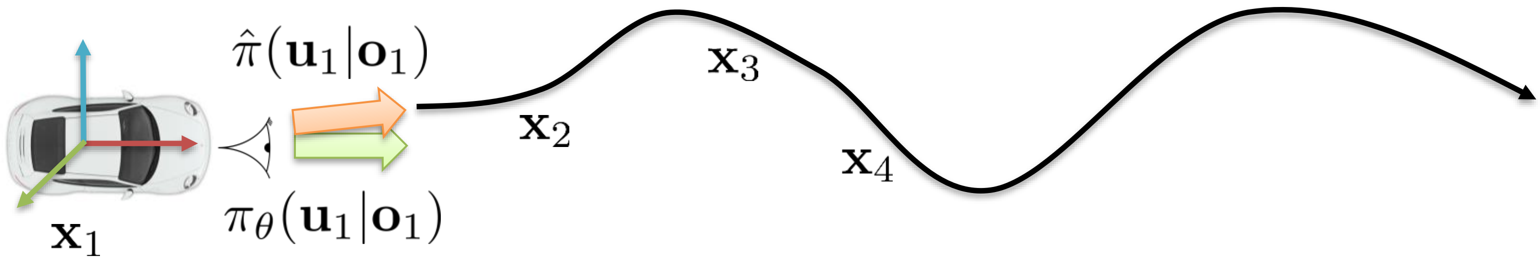
[L. et al. NIPS ‘14]

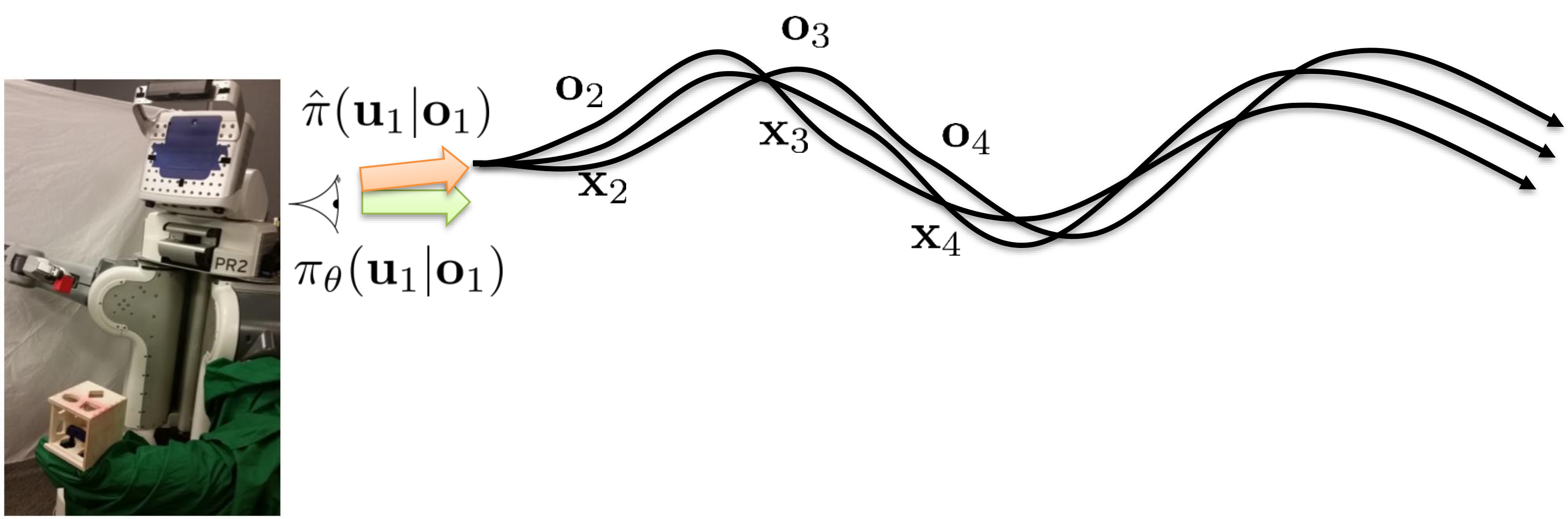
Learning on PR2



[L. et al. ICRA '15]

Combining with Policy Learning

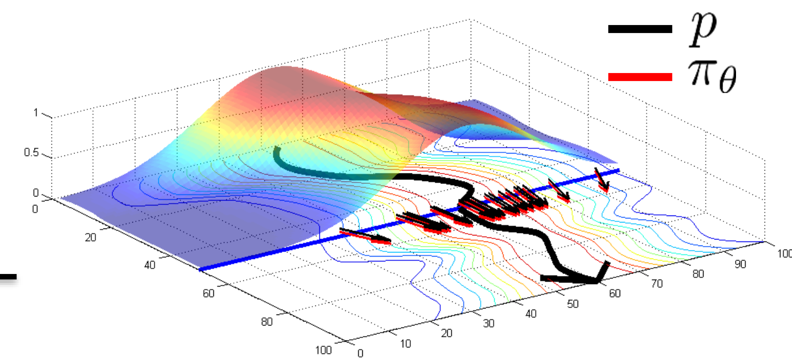




$$\min_{\theta} \overbrace{E_{\pi_\theta}[c(\tau)]}^{\text{expectation under current policy}}$$

$$\min_{\theta, p(\tau)} \quad E_p[c(\tau)]$$

trajectory distribution(s)



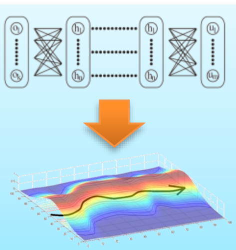
$$s.t. \quad \pi_\theta(\mathbf{u}_t|\mathbf{o}(\mathbf{x}_t)) = p(\mathbf{u}_t|\mathbf{x}_t) \quad \forall t, \mathbf{x}_t, \mathbf{u}_t$$

$$E_{p(\mathbf{x}_t)}[D_{KL}(\pi_\theta(\mathbf{u}_t|\mathbf{o}(\mathbf{x}_t))\|p(\mathbf{u}_t|\mathbf{x}_t))] = 0 \quad \forall t$$

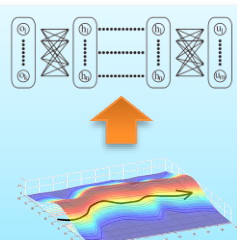
L. et al. ICML '14 (dual descent)

can also use BADMM (L. et al.'15)

$$\mathcal{L}(\theta, p, \lambda) = E_p[c(\tau)] + \sum_{t=1}^{T} \overbrace{\lambda_t}^{\text{Lagrange multiplier}} D_t(\pi_\theta, p)$$



optimize $\mathcal{L}(\theta, p, \lambda)$ w.r.t. $p(\tau)$

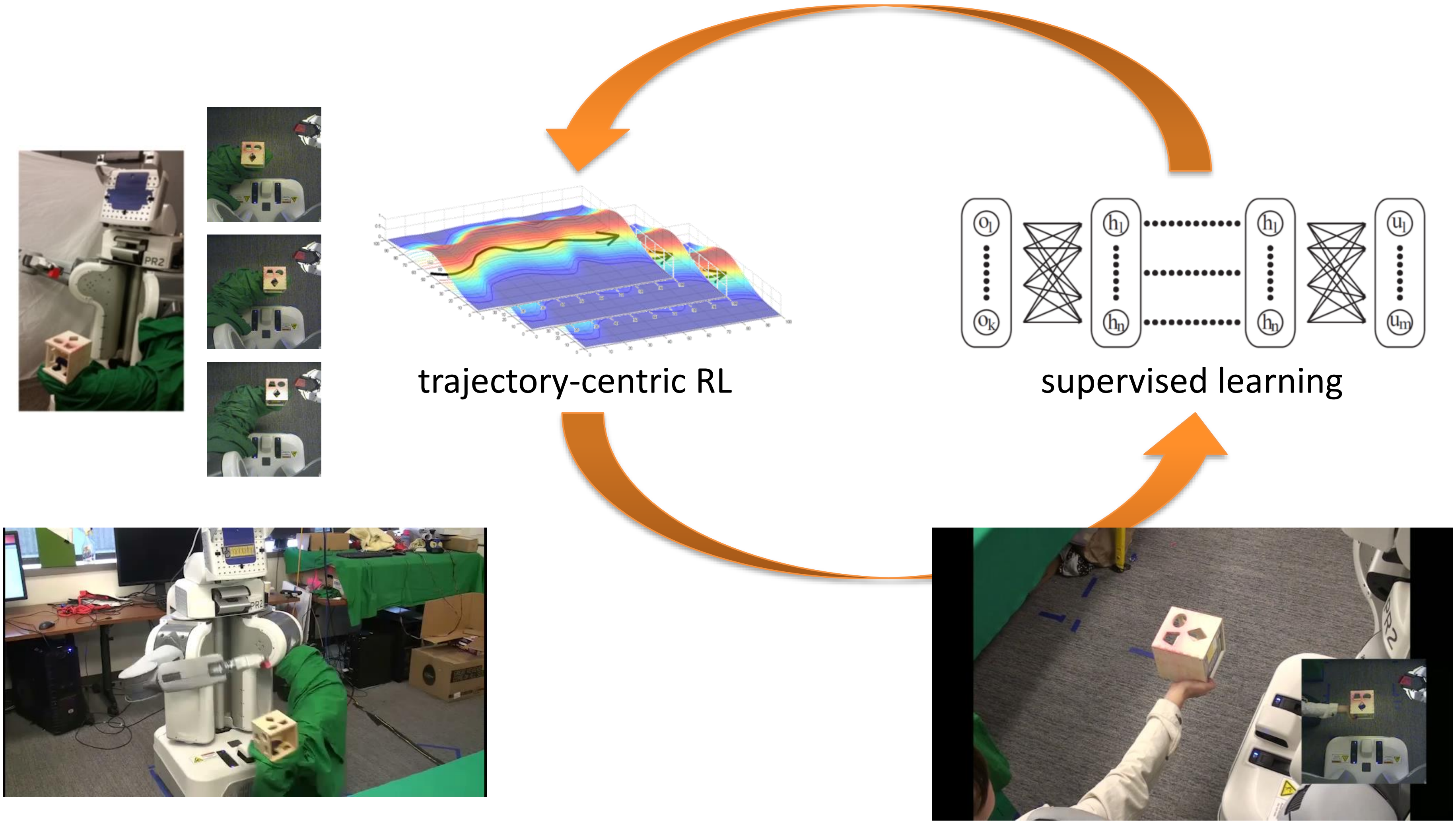


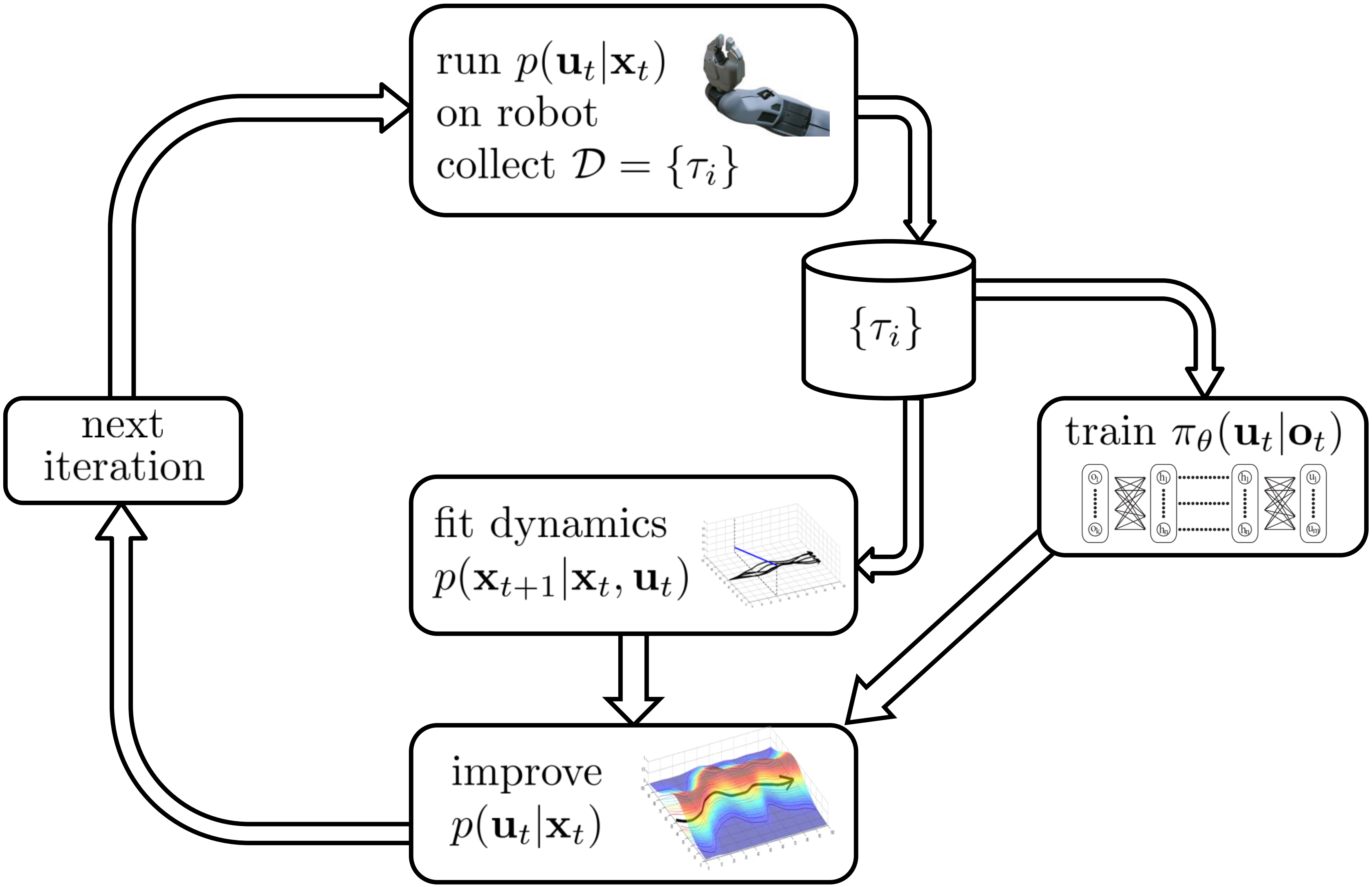
optimize $\mathcal{L}(\theta, p, \lambda)$ w.r.t. θ

update λ with subgradient descent:

$$\lambda_t \leftarrow \lambda_t + \eta D_t(\pi_\theta, p)$$

Guided Policy Search

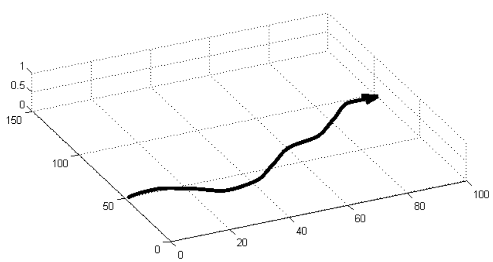




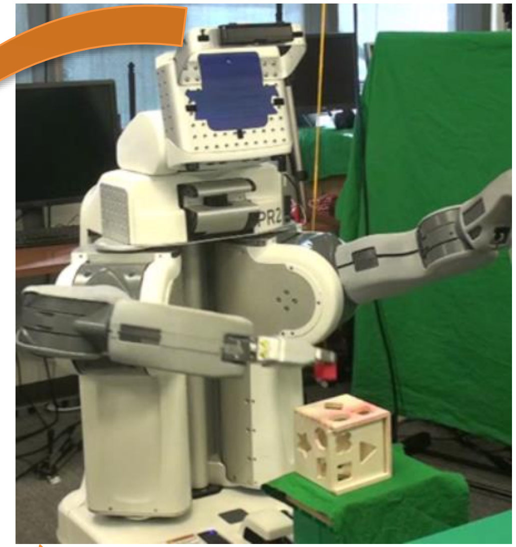
[see L. et al. NIPS '14 for details]

training time

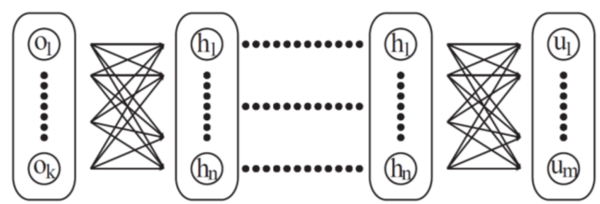
$$\mathbf{x}_t \rightarrow \mathbf{u}_t$$



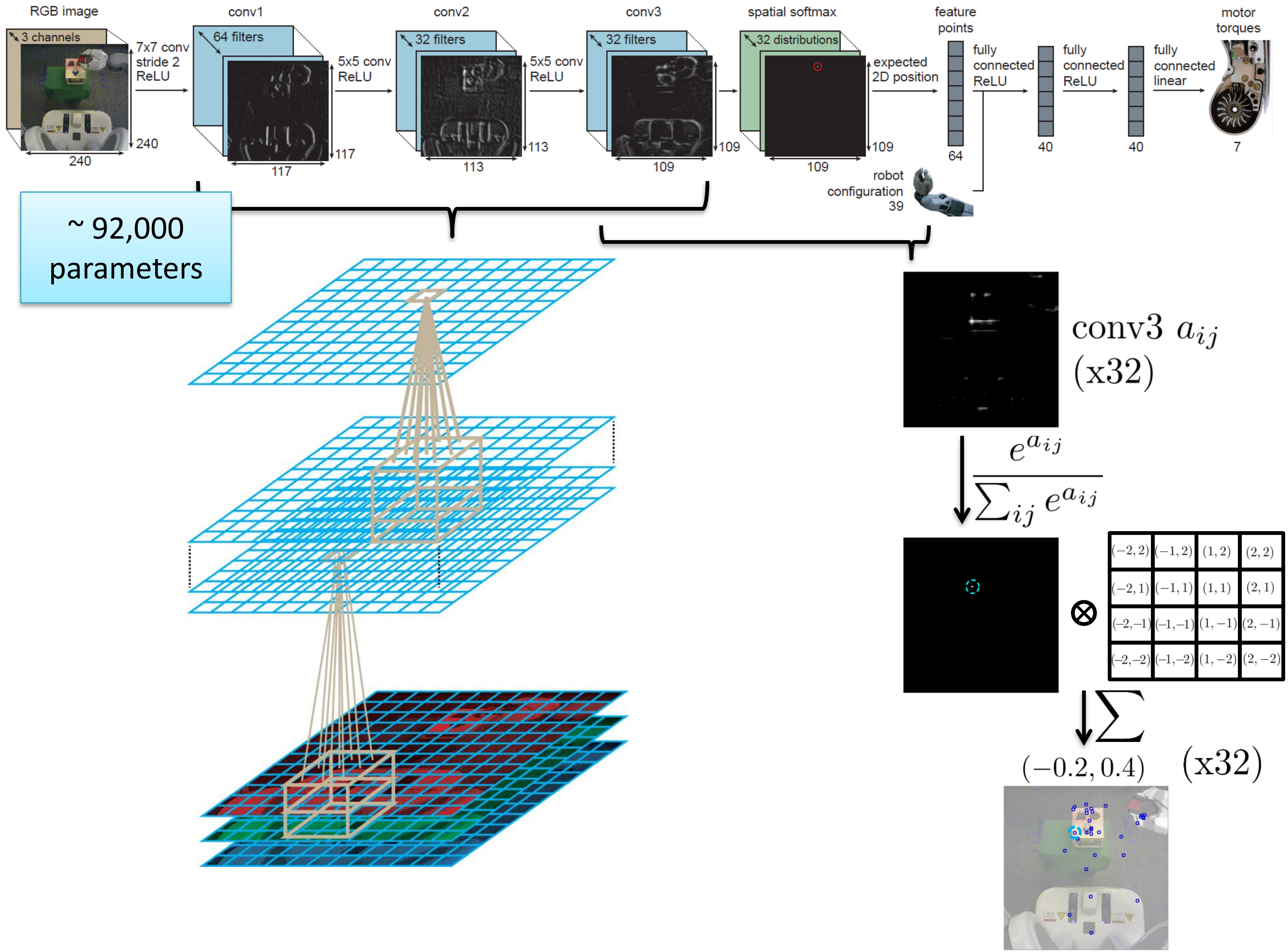
test time



$$\mathbf{o}_t \rightarrow \mathbf{u}_t$$



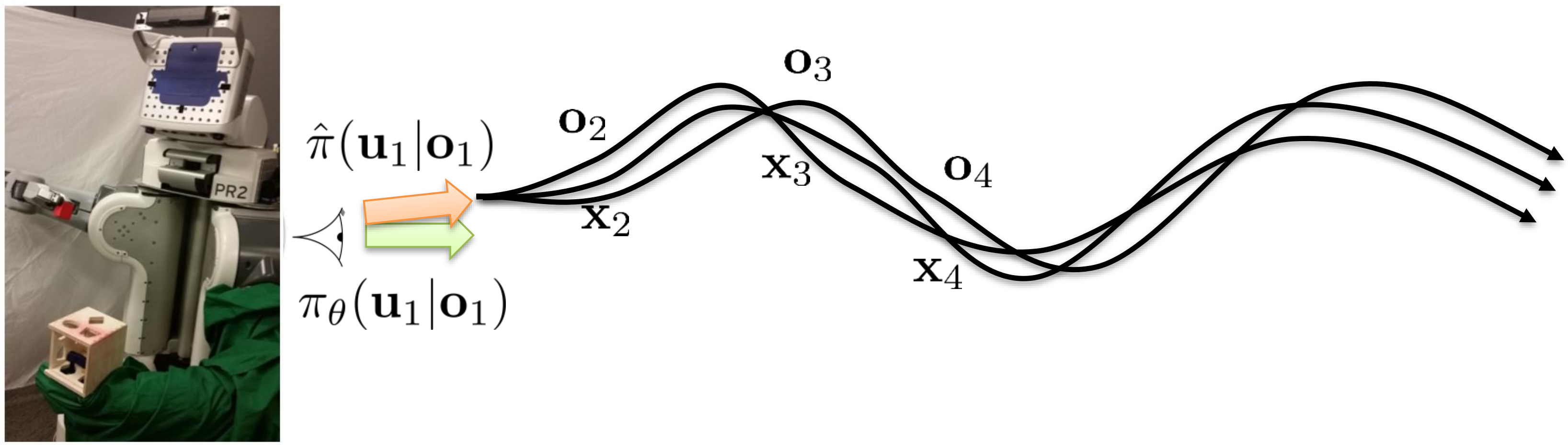
L.*, Finn*, Darrell, Abbeel '16



RGB image
3 channels
7x7 conv
stride 2
ReLU
240
240
conv1
64 filters
117
117
5x5 conv
ReLU
conv2
32 filters
113
113
5x5 conv
ReLU
conv3
32 filters
109
109
spatial softmax
32 distributions
109
109
expected
2D position
feature
points
64
fully
connected
ReLU
40
fully
connected
ReLU
40
fully
connected
linear
motor
torques
7
robot
configuration
39
~ 92,000
parameters
conv3 a_{ij}
(x32)
$\frac{e^{a_{ij}}}{\sum_{ij} e^{a_{ij}}}$
$\otimes$
$(-2,2)$ $(-1,2)$ $(1,2)$ $(2,2)$
$(-2,1)$ $(-1,1)$ $(1,1)$ $(2,1)$
$(-2,-1)$ $(-1,-1)$ $(1,-1)$ $(2,-1)$
$(-2,-2)$ $(-1,-2)$ $(1,-2)$ $(2,-2)$
$\sum$
$(-0.2, 0.4)$
(x32)

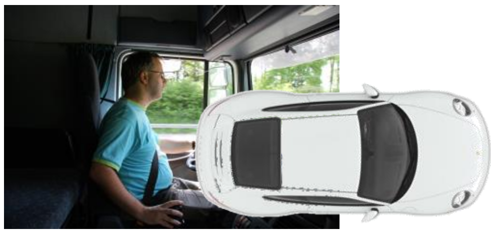
Experimental Tasks

Imitating optimal control: questions

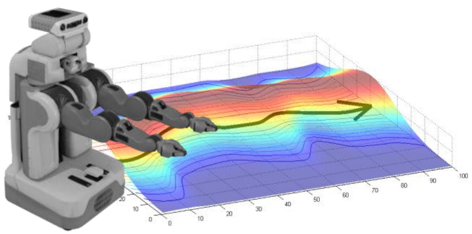


- Any difference from standard imitation learning?
  - Can change behavior of the "teacher" programmatically
- Can the policy help optimal control (rather than just the other way around?)

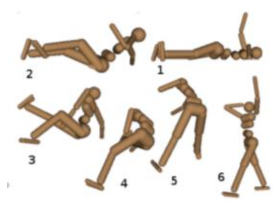
Contents



Imitation learning



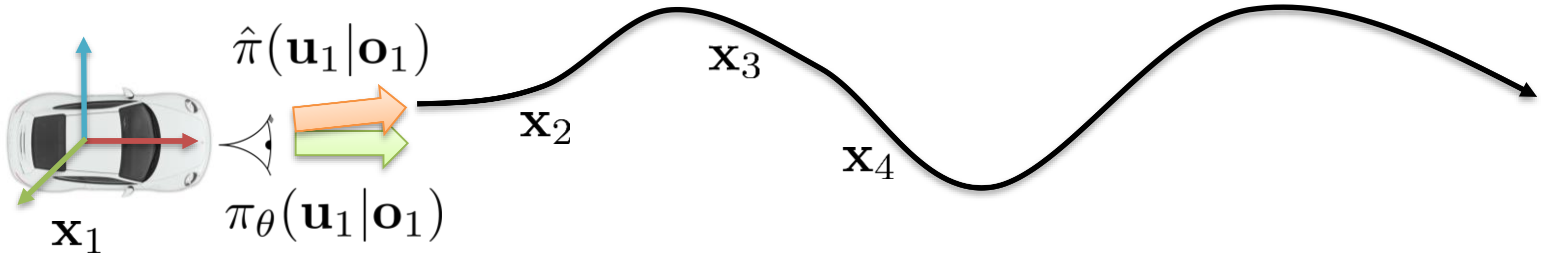
Imitation without a human

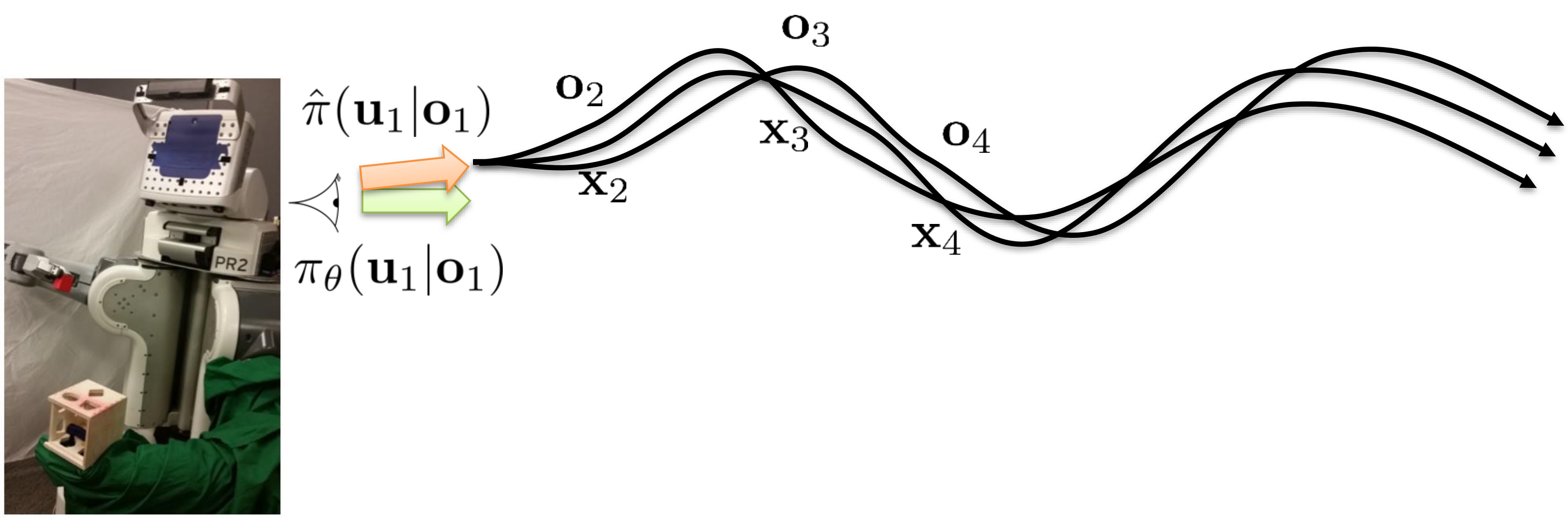


Reinforcement learning

Research frontiers

Can we avoid dynamics completely?





A simple reinforcement learning algorithm

$$\min_{\boldsymbol{\theta}} \underbrace{\sum_{t=1}^{T} E_{(\mathbf{x}_t,\mathbf{u}_t)\sim p_\theta(\mathbf{x}_t,\mathbf{u}_t)}[c(\mathbf{x}_t,\mathbf{u}_t)]}_{J(\theta)}$$

$$\min_{\boldsymbol{\theta}} E_{\tau\sim p_\theta(\tau)}[c(\tau)] = E_{\tau\sim p_\theta(\tau)}\left[\sum_{t=1}^{T} c(\mathbf{x}_t,\mathbf{u}_t)\right]$$

$$p_\theta(\tau) = p_\theta(\mathbf{x}_1,\mathbf{u}_1,\ldots,\mathbf{x}_T,\mathbf{u}_T) = p(\mathbf{x}_1)\prod_{t=1}^{T} p(\mathbf{x}_{t+1}|\mathbf{x},\mathbf{u})\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$$

$$\nabla_\theta J(\theta)$$

$$\nabla_\theta p_\theta(\tau)$$

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)[\nabla_\theta \log p_\theta(\tau)]c(\tau)d\tau$$

$$\left.\right) c(\tau)\Big]$$

$$\nabla_\theta \log p_\theta(\tau) = \nabla_\theta\left[\cancel{\log p(\mathbf{x}_1)} + \sum_{t=1}^{T} \cancel{\log p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t)} + \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\right]$$

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=1}^{T}\left[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\right]$$

REINFORCE likelihood ratio policy gradient

$$\nabla_\theta J(\theta) = E\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\right)\left(\sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t)\right)\right]$$

example: $\pi_\theta(\mathbf{u}_t, \mathbf{x}_t) = \mathcal{N}(f_{\text{neural network}}(\mathbf{x}_t); \Sigma)$

$$\log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t) = -\frac{1}{2}\|f(\mathbf{x}_t) - \mathbf{u}_t\|_\Sigma^2 + \text{const}$$

$$\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t) = -\frac{1}{2}\Sigma^{-1}(f(\mathbf{x}_t) - \mathbf{u}_t)\frac{df}{d\theta}$$

how? just backpropagate $-\frac{1}{2}\Sigma^{-1}(f(\mathbf{x}_t) - \mathbf{u}_t)$

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{u}_t^i|\mathbf{x}_t^i)\right)\left(\sum_t c(\mathbf{x}_t^i, \mathbf{u}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Williams '92

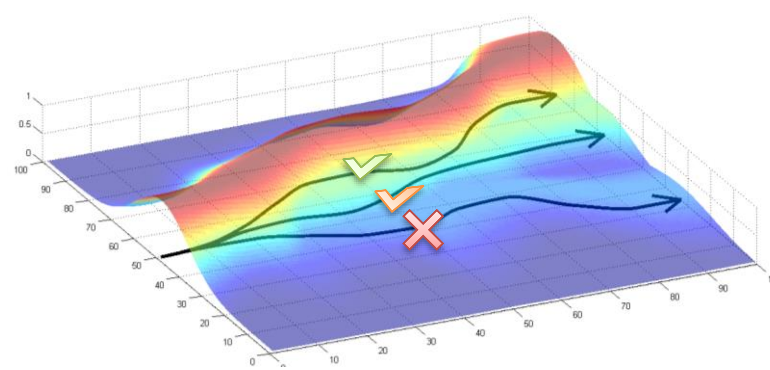
What the heck did we just do?

$$\nabla_\theta J(\theta) = E\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\right)\left(\sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) - b\right)\right]$$

$$b = E\left[\sum_t c(\mathbf{x}_t, \mathbf{u}_t)\right]$$

one more piece…

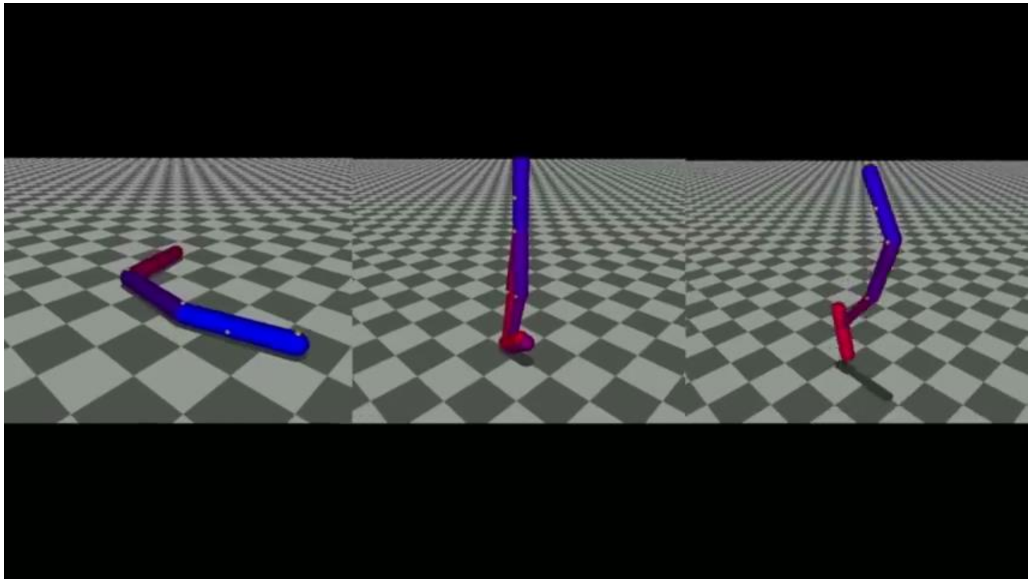
$$E[\nabla \log p(y) b]$$



Policy gradient challenges

$$\nabla_\theta J(\theta) = E\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\right)\left(\sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) - b\right)\right]$$

- High variance in gradient estimate
  - Smarter baselines
- Poor conditioning
  - Use higher order methods (see: natural gradient)
- Very hard to choose step size
  - Trust region policy optimization (TRPO)



Schulman, L., Moritz, Jordan, Abbeel ‘15

Value functions

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E\left[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t) \underbrace{E\left[\sum_{t'=t}^{T} c(\mathbf{x}_{t'}, \mathbf{u}_{t'})\right]}_{\text{total } \textit{remaining} \text{ cost of executing } \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}\right]$$

$$Q^\pi(\mathbf{x}_t, \mathbf{u}_t)$$

$Q^\pi(\mathbf{x}_t, \mathbf{u}_t)$ – total cost of running π after taking action $\mathbf{u}_t$ in state $\mathbf{x}_t$

$$Q^\pi(\mathbf{x}_t, \mathbf{u}_t) = c(\mathbf{x}_t, \mathbf{u}_t) + E_{\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)}[V^\pi(\mathbf{x}_{t+1})]$$

$V^\pi(\mathbf{x}_t)$ – total cost of running π from state $\mathbf{x}_t$

$$Q^\pi(\mathbf{x}_t, \mathbf{u}_t) = c(\mathbf{x}_t, \mathbf{u}_t) + E_{\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)}[E_{\mathbf{u}_{t+1} \sim \pi_\theta(\mathbf{u}_{t+1}|\mathbf{x}_{t+1})}[Q^\pi(\mathbf{x}_{t+1}, \mathbf{u}_{t+1})]]$$

Value functions

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E\left[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\left[\sum_{t'=t}^{T} c(\mathbf{x}_{t'}, \mathbf{u}_{t'})\right]\right]$$

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E\left[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\hat{Q}^\pi_\phi(\mathbf{x}_t, \mathbf{u}_t)\right]$$

$$\hat{Q}^\pi_\phi(\mathbf{x}_t, \mathbf{u}_t) \approx E\left[\sum_{t'=t}^{T} c(\mathbf{x}_{t'}, \mathbf{u}_{t'})\right]$$

example: $\hat{Q}^\pi_\phi(\mathbf{x}_t, \mathbf{u}_t)$ is a neural network, trained via regression

$$\phi \leftarrow \arg\min_\phi \sum_{i=1}^{N}\sum_{t=1}^{T}\left\|\hat{Q}^\pi_\phi(\mathbf{x}_t^i, \mathbf{u}_t^i) - \left[\sum_{t'=t}^{T} c(\mathbf{x}_{t'}^i, \mathbf{u}_{t'}^i)\right]\right\|^2$$

$$\phi \leftarrow \arg\min_\phi \sum_{i=1}^{N}\sum_{t=1}^{T}\left\|\hat{Q}^\pi_\phi(\mathbf{x}_t^i, \mathbf{u}_t^i) - \left[c(\mathbf{x}_t^i, \mathbf{u}_t^i) + \hat{Q}^\pi_{\phi_{\mathrm{old}}}(\mathbf{x}_{t+1}^i, \mathbf{u}_{t+1}^i)\right]\right\|^2$$

Policy gradient with value function approximation:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ (run it on the robot)
2. Use samples $\{\tau^i\}$ to fit $\hat{Q}^\pi_\phi$
3. $\nabla_\theta J(\theta) \approx \sum_i \sum_t \nabla_\theta \log \pi_\theta(\mathbf{u}_t^i|\mathbf{x}_t^i)\hat{Q}^\pi_\phi(\mathbf{x}_t^i, \mathbf{u}_t^i)$
4. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Value functions challenges

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E\left[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\hat{Q}^\pi_\phi(\mathbf{x}_t, \mathbf{u}_t)\right]$$

high bias?

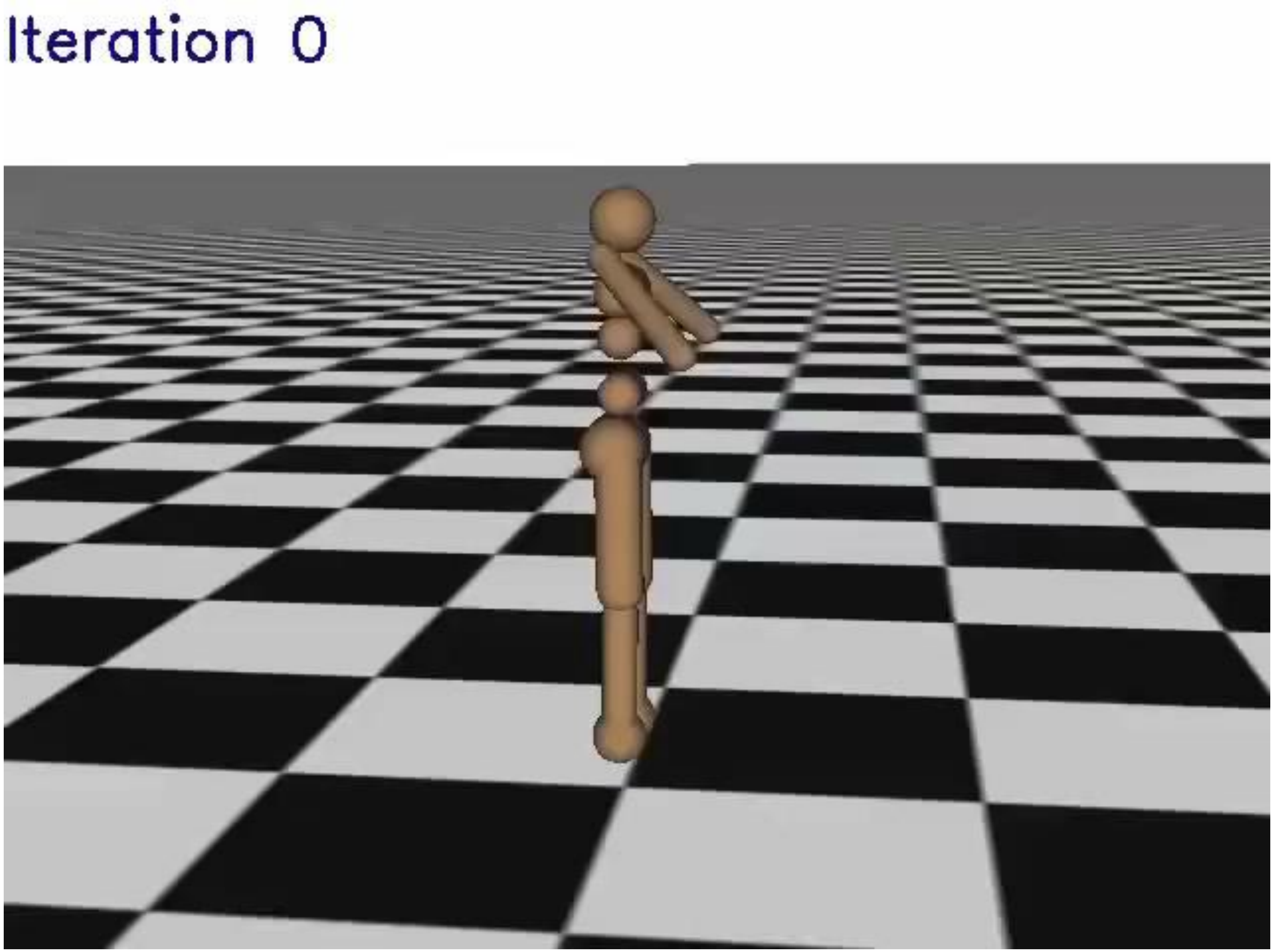
$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E\left[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t) E\left[\sum_{t'=t}^{T} c(\mathbf{x}_{t'}, \mathbf{u}_{t'})\right]\right]$$

high variance?

- The usual problems with policy gradient
  - Poor conditioning (use natural gradient)
  - Hard to choose step size (use TRPO)
- Bias/variance tradeoff
  - Combine Monte Carlo and function approximation: Generalized Advantage Estimation (GAE)
- Instability/overfitting
  - Limit how much the value function estimate changes per iteration

Schulman, Moritz, L. Jordan, Abbeel ‘16

Generalized advantage estimation



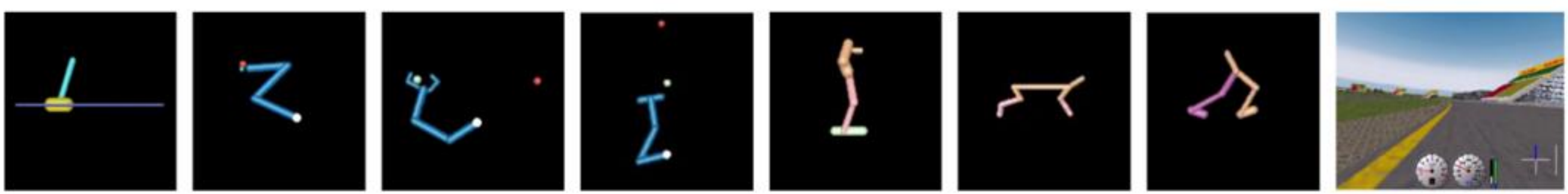
Schulman, Moritz, L., Jordan, Abbeel ‘16

Online actor-critic methods

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ (run it on the robot)
2. Use samples $\{\tau^i\}$ to fit $\hat{Q}^\pi_\phi$
3. $\nabla_\theta J(\theta) \approx \sum_i \sum_t \nabla_\theta \log \pi_\theta(\mathbf{u}_t^i|\mathbf{x}_t^i)\hat{Q}^\pi_\phi(\mathbf{x}_t^i, \mathbf{u}_t^i)$

1. make one decision (time step) with $\mathbf{u} \sim \pi_\theta(\mathbf{u}|\mathbf{x})$, add to buffer $\mathcal{D}$
2. sample batch $\{\mathbf{x}^i, \mathbf{u}^i\}$ from $\mathcal{D}$, fit $\hat{Q}^\pi_\phi$
3. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{u}^i|\mathbf{x}^i)\hat{Q}^\pi_\phi(\mathbf{x}^i, \mathbf{u}^i)$
4. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Deep deterministic policy gradient (DDPG)



Continuous Control with Deep Reinforcement Learning (Lillicrap et al. '15)

Just the Q function?

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E\left[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\hat{Q}^\pi_\phi(\mathbf{x}_t,\mathbf{u}_t)\right]$$

$$\phi \leftarrow \arg\min_\phi \sum_{i=1}^{N}\sum_{t=1}^{T}\left\|\hat{Q}^\pi_\phi(\mathbf{x}_t^i,\mathbf{u}_t^i) - \left[c(\mathbf{x}_t^i,\mathbf{u}_t^i) + \gamma \hat{Q}^\pi_\phi(\mathbf{x}_{t+1}^i,\mathbf{u}_{t+1}^i)\right]\right\|^2$$

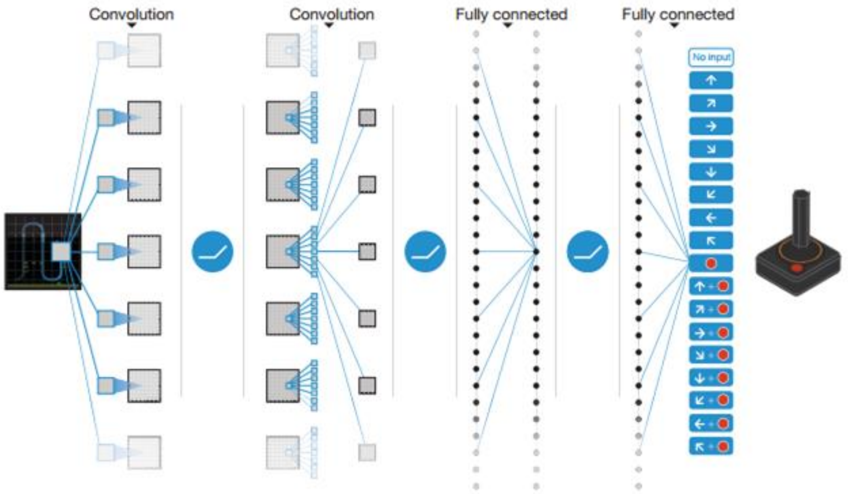
$$\phi \leftarrow \arg\min_\phi \sum_{i=1}^{N}\sum_{t=1}^{T}\left\|\hat{Q}_\phi(\mathbf{x}_t^i,\mathbf{u}_t^i) - \left[c(\mathbf{x}_t^i,\mathbf{u}_t^i) + \gamma \min_{\mathbf{u}_{t+1}} \hat{Q}_{\phi_{\text{old}}}(\mathbf{x}_{t+1}^i,\mathbf{u}_{t+1})\right]\right\|^2$$

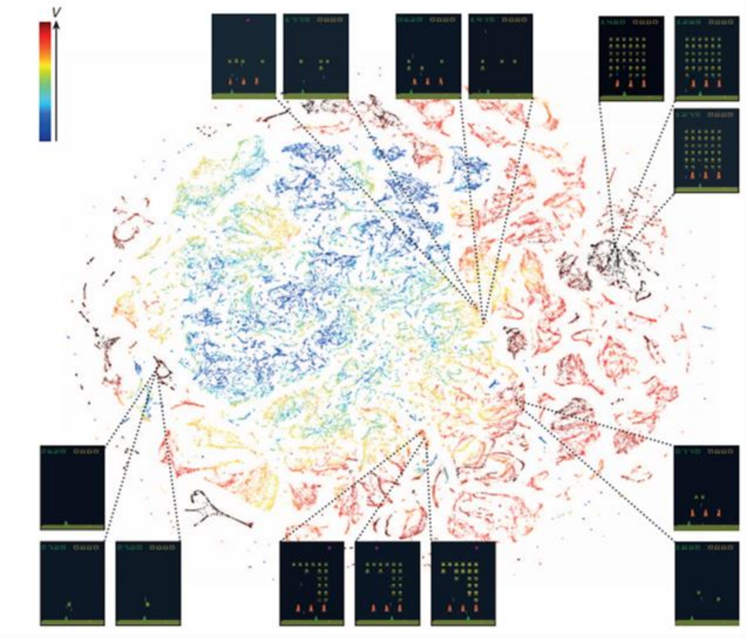
$\pi(\mathbf{u}_t|\mathbf{x}_t) \propto \exp(-\hat{Q}_\phi(\mathbf{x}_t,\mathbf{u}_t))$ $\qquad$ $\pi(\mathbf{u}_t|\mathbf{x}_t) \propto \epsilon + \delta(\mathbf{u}_t = \arg\min_{\mathbf{u}_t} \hat{Q}_\phi(\mathbf{x}_t,\mathbf{u}_t))$

Q-learning for deep RL

1. make one decision (time step) with $\mathbf{u} \sim \pi_\theta(\mathbf{u}|\mathbf{x})$, add to buffer $\mathcal{D}$
2. sample minibatch $\{\mathbf{x}^i,\mathbf{u}^i\}$ from $\mathcal{D}$, fit $\hat{Q}_\phi$
3. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{u}^i|\mathbf{x}^i)\hat{Q}^\pi_\phi(\mathbf{x}^i,\mathbf{u}^i)$
3. choose $\pi(\mathbf{u}|\mathbf{x})$ based on $\hat{Q}_\phi$, e.g. $\pi(\mathbf{u}|\mathbf{x}) \propto \exp(-\hat{Q}_\phi(\mathbf{x},\mathbf{u}))$
4. $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$

Discrete Q-learning





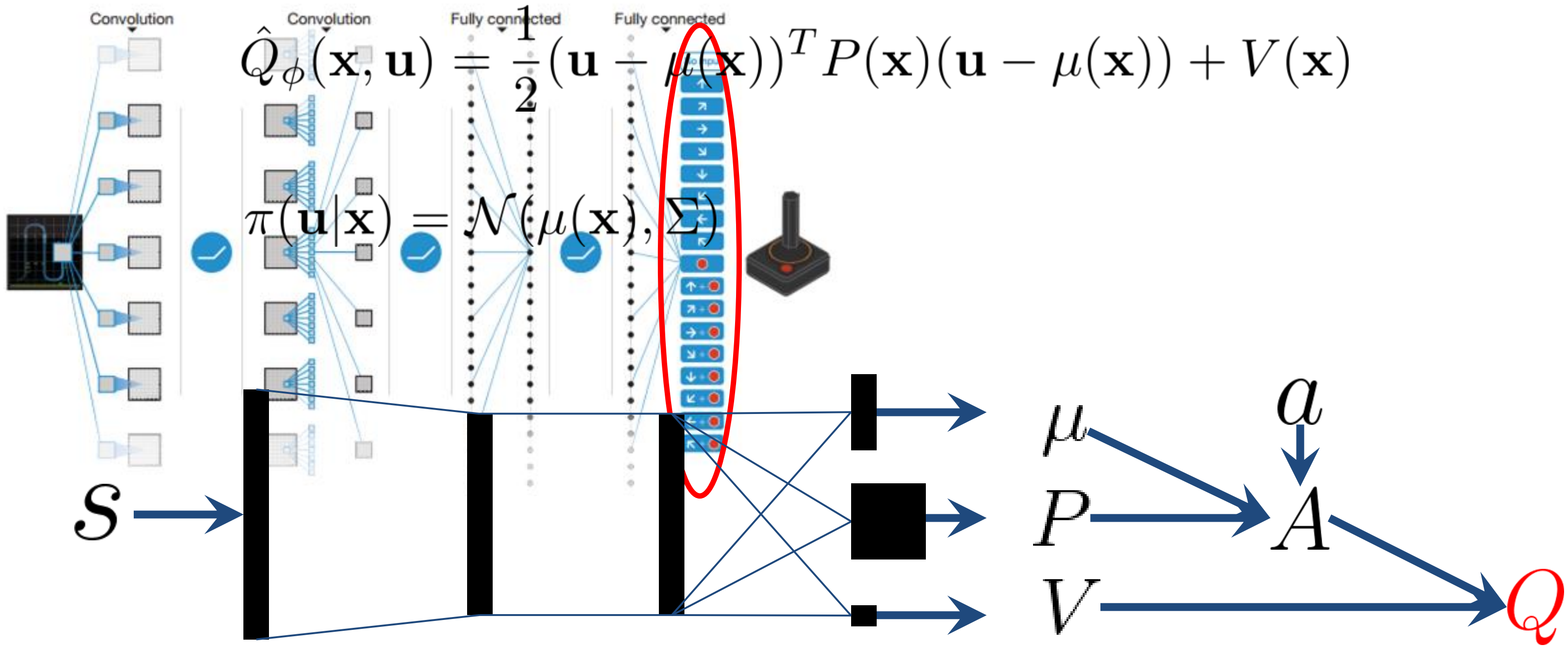
Human-Level Control Through Deep Reinforcement Learning (Mnih et al. ‘15)

Continuous Q-learning

$$\phi \leftarrow \arg\min_{\phi} \sum_{i=1}^{N} \sum_{t=1}^{T} \left\| \hat{Q}_{\phi}(\mathbf{x}_t^i, \mathbf{u}_t^i) - \left[c(\mathbf{x}_t^i, \mathbf{u}_t^i) + \gamma \underline{\min_{\mathbf{u}_{t+1}} \hat{Q}_{\phi_{\text{old}}}(\mathbf{x}_{t+1}^i, \mathbf{u}_{t+1})} \right] \right\|^2$$

$$\hat{Q}_{\phi}(\mathbf{x}, \mathbf{u}) = \frac{1}{2}(\mathbf{u} - \mu(\mathbf{x}))^T P(\mathbf{x})(\mathbf{u} - \mu(\mathbf{x})) + V(\mathbf{x})$$

$$\pi(\mathbf{u}|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \Sigma)$$

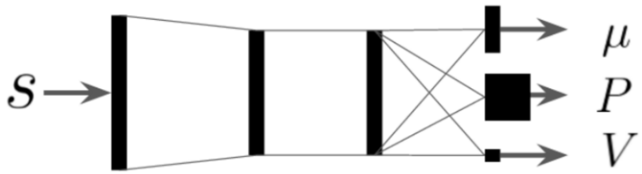


Normalized Advantage Functions

Domains	Random	DDPG	episode	NAF	episode
Cartpole	-2.1	-0.601	420	-0.604	**190**
Reacher	-2.3	-0.509	1370	**-0.331**	**1260**
Peg	-11	-0.950	690	**-0.438**	**130**
Gripper	-29	1.03	2420	**1.81**	**1920**
GripperM	-90	-20.2	1350	**-12.4**	**730**
Canada2d	-12	-4.64	1040	**-4.21**	900
Cheetah	-0.3	**8.23**	**1590**	7.91	2390
Swimmer6	-325	-174	220	**-172**	**190**
Ant	-4.8	-2.54	2450	-2.58	**1350**
Walker2d	0.3	**2.96**	**850**	1.85	1530

*Random, DDPG, NAF policies: final rewards and episodes to converge

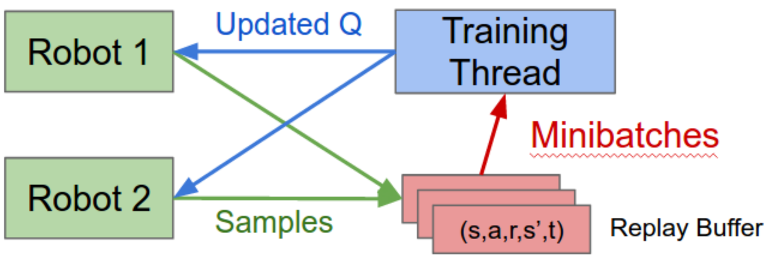
Policy Learning with Multiple Robots: Deep RL with NAF

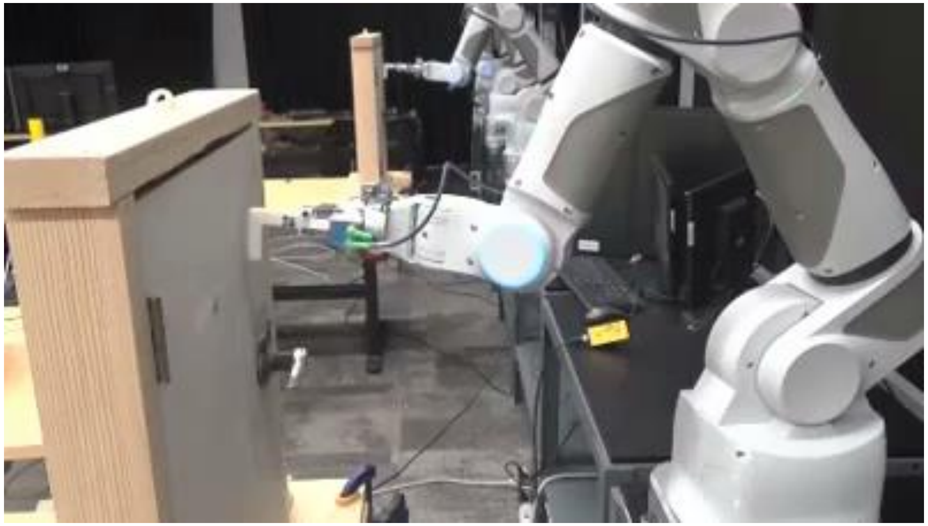


NAF Architecture.

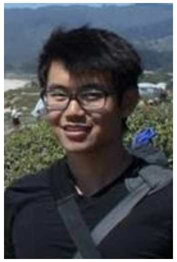
$$Q(\boldsymbol{x},\boldsymbol{u}|\theta^Q) = A(\boldsymbol{x},\boldsymbol{u}|\theta^A) + V(\boldsymbol{x}|\theta^V)$$

$$A(\boldsymbol{x},\boldsymbol{u}|\theta^A) = -\frac{1}{2}(\boldsymbol{u}-\boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))^T \boldsymbol{P}(\boldsymbol{x}|\theta^P)(\boldsymbol{u}-\boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))$$





Shane Gu

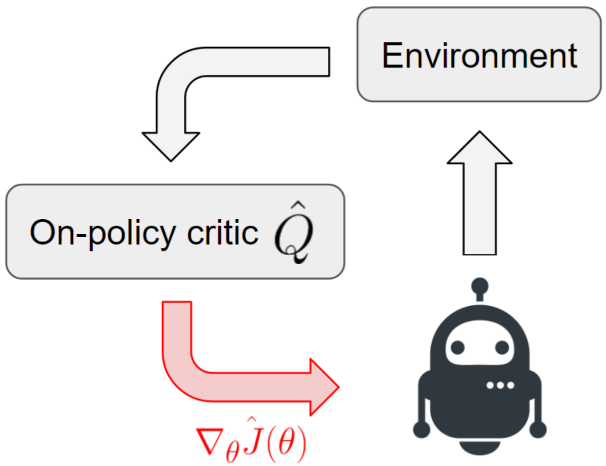


Ethan Holly

Tim Lillicrap

Gu*, Holly*, Lillicrap, L., '16

Deep RL with Policy Gradients

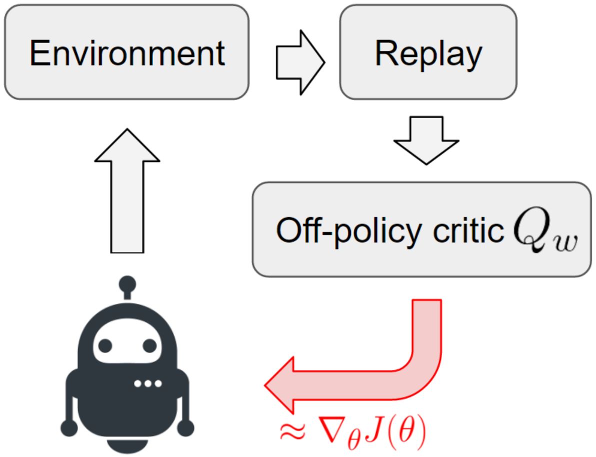


- Unbiased but high-variance gradient
- Stable
- Requires many samples
- Example: TRPO [Schulman et al. ‘15]

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\hat{Q}(\mathbf{x}_t, \mathbf{u}_t)]$$

$$\hat{Q}(\mathbf{x}_t, \mathbf{u}_t) \approx \sum_{t'=t}^{\infty} \gamma^{t-t'} r(\mathbf{x}_t, \mathbf{u}_t)$$

Deep RL with Off-Policy Q-Function Critic

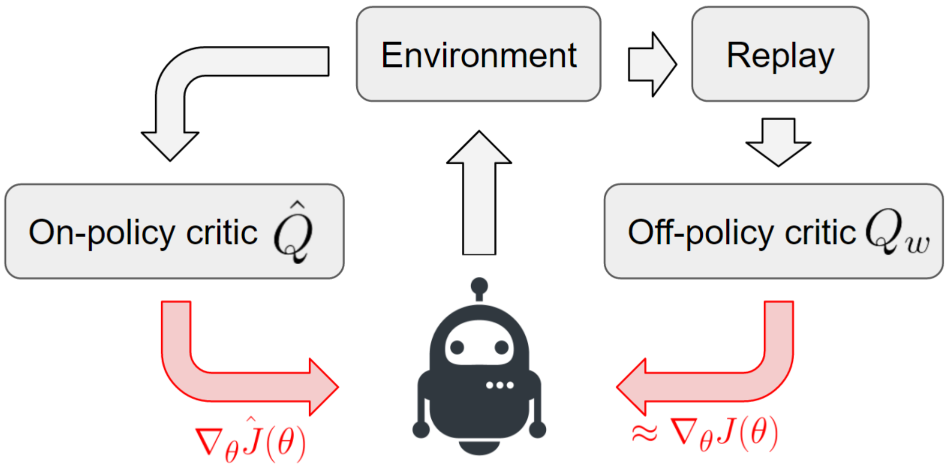


- Low-variance but biased gradient
- Much more efficient (because off-policy)
- Much less stable (because biased)
- Example: DDPG [Lillicrap et al. ‘16]

$$Q_w \leftarrow \min_w E[(Q_w(\mathbf{x}_t, \mathbf{u}_t) - (r(\mathbf{x}_t, \mathbf{u}_t) + \gamma Q_w(\mathbf{x}_{t+1}, \pi_\theta(\mathbf{x}_t)))^2]$$

$$\nabla_\theta J(\theta) = E[\nabla_{\mathbf{u}_t} Q_w(\mathbf{x}_t, \pi_\theta(\mathbf{x}_t)) \nabla_\theta \pi_\theta(\mathbf{x}_t)]$$

Improving Efficiency & Stability with Q-Prop



- Unbiased gradient, stable
- Efficient (uses off-policy samples)
- Critic comes from off-policy data
- Gradient comes from on-policy data
- Automatic variance-based adjustment

Policy gradient:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)\hat{Q}(\mathbf{x}_t, \mathbf{u}_t)]$$

Q-function critic:

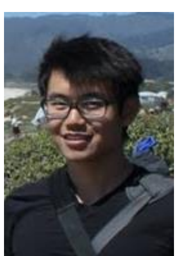
$$\nabla_\theta J(\theta) = E[\nabla_{\mathbf{u}_t} Q_w(\mathbf{x}_t, \mu_\theta(\mathbf{x}_t))\nabla_\theta \mu_\theta(\mathbf{x}_t)]$$

Q-Prop:

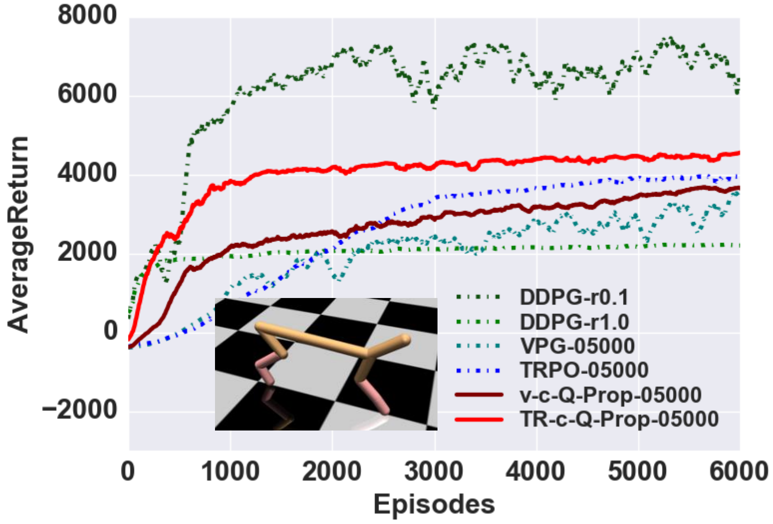
$$\nabla_\theta J(\theta) = E_{\pi_\theta(\mathbf{x}_t)}[\nabla_{\mathbf{u}_t} Q(\mathbf{x}_t, \mu_\theta(\mathbf{x}_t))\nabla_\theta \mu_\theta(\mathbf{x}_t)] +$$
$$E_{\pi_\theta(\mathbf{x}_t, \mathbf{u}_t)}[\nabla_\theta \log \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)(\hat{Q}(\mathbf{x}_t, \mathbf{u}_t) - \bar{Q}(\mathbf{x}_t, \mathbf{u}_t))]$$

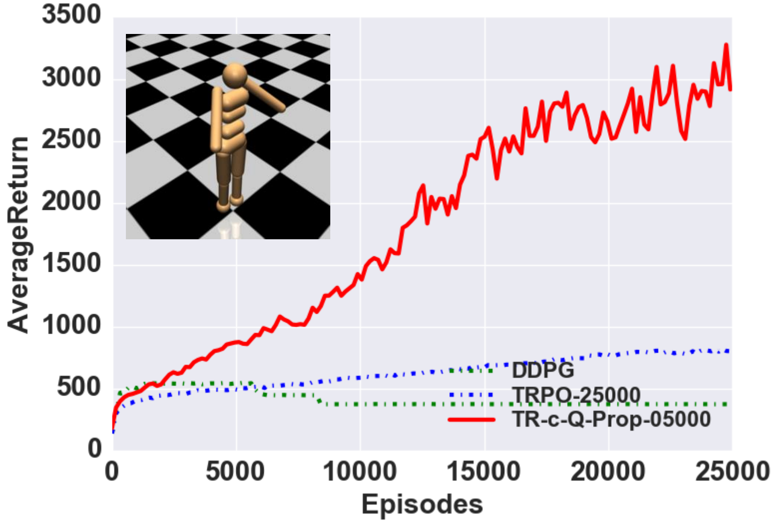
$$\bar{Q}(\mathbf{x}_t, \mathbf{u}_t) = \nabla_{\mathbf{u}_t} Q(\mathbf{x}_t, \mu_\theta(\mathbf{x}_t))(\mathbf{u}_t - \mu_\theta(\mathbf{x}_t))$$

Shane Gu



Comparisons

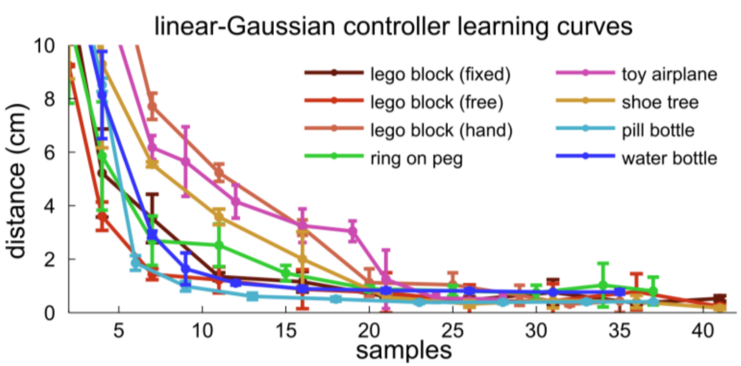




- Works with smaller batches than TRPO
- More efficient than TRPO
- More stable than DDPG with respect to hyperparameters
  - Likely responsible for the better performance on harder task

Sample complexity

- Deep reinforcement learning is very data-hungry
  - DQN: about 100 hours to learn Breakout
  - GAE: about 50 hours to learn to walk
  - DDPG/NAF: 4-5 hours to learn basic manipulation, walking
- Model-based methods are more efficient
  - Time-varying linear models: 3 minutes for **real world** manipulation
  - GPS with vision: 30-40 minutes for **real world visuomotor** policies

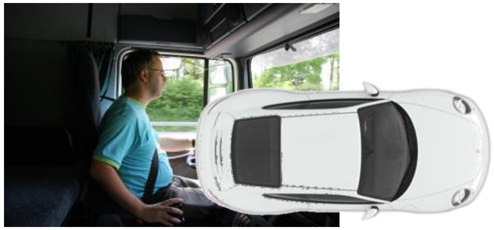


	number of trials		
task	trajectory pretraining	end-to-end training	total
coat hanger	120	36	156
shape cube	90	81	171
toy hammer	150	90	240
bottle cap	180	108	288

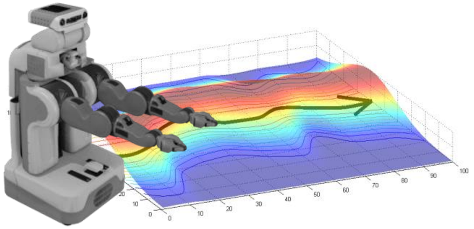
Reinforcement learning tradeoffs

- Reinforcement learning (for the purpose of this slide) = **model-free** RL
- Fewer assumptions
  - Don't need to model dynamics
  - Don't (in general) need state definition, only observations
  - Fully general stochastic environments
- Much slower (model-based acceleration?)
- Hard to stabilize
  - Few convergence results with neural networks

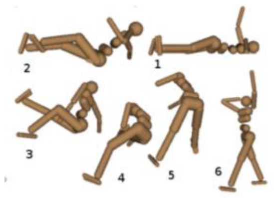
Contents



Imitation learning



Imitation without a human



Reinforcement learning

Research frontiers

ingredients for success in learning:

supervised learning:	learning sensorimotor skills:
✓ computation	✓ computation
✓ algorithms	~ algorithms
✓ data	? data

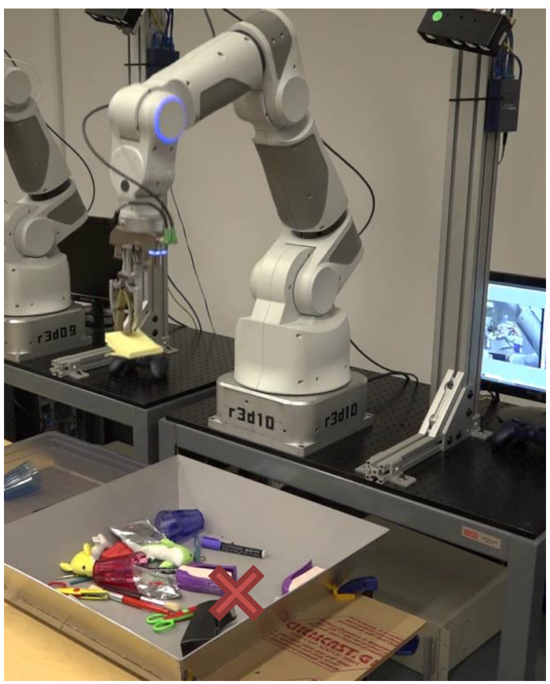
L., Pastor, Krizhevsky, Quillen '16

Grasping with Learned Hand-Eye Coordination

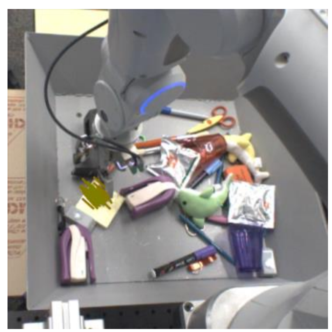
- 800,000 grasp attempts for training (3,000 robot-hours)
- monocular camera (no depth)
- 2-5 Hz update
- no prior knowledge

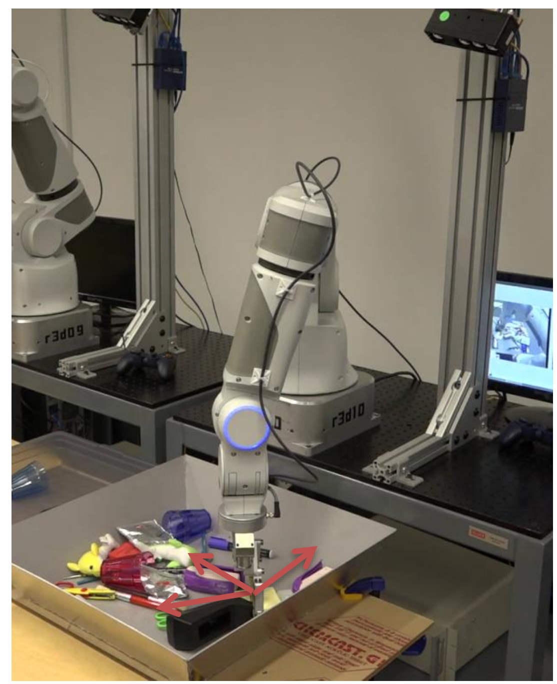
L., Pastor, Krizhevsky, Quillen '16

Using Grasp Success Prediction

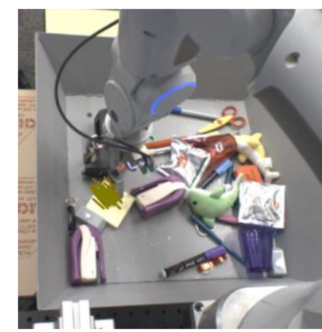


training





testing

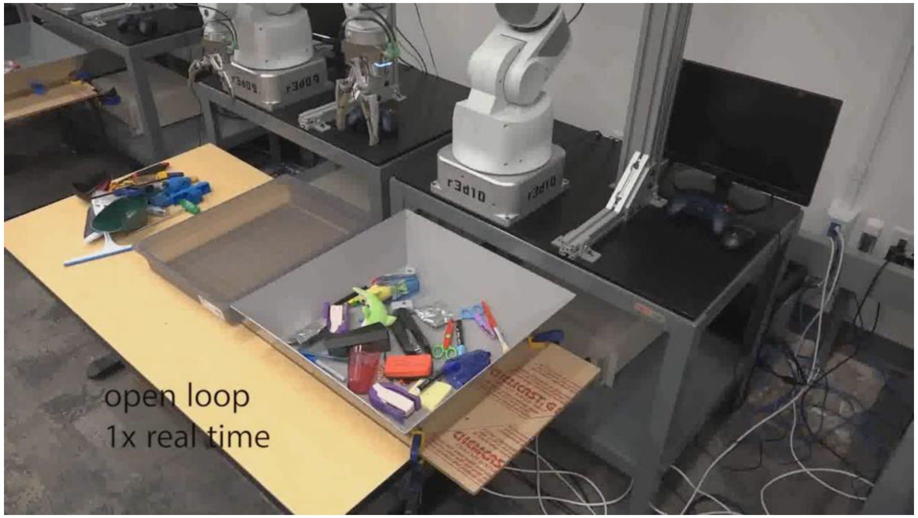


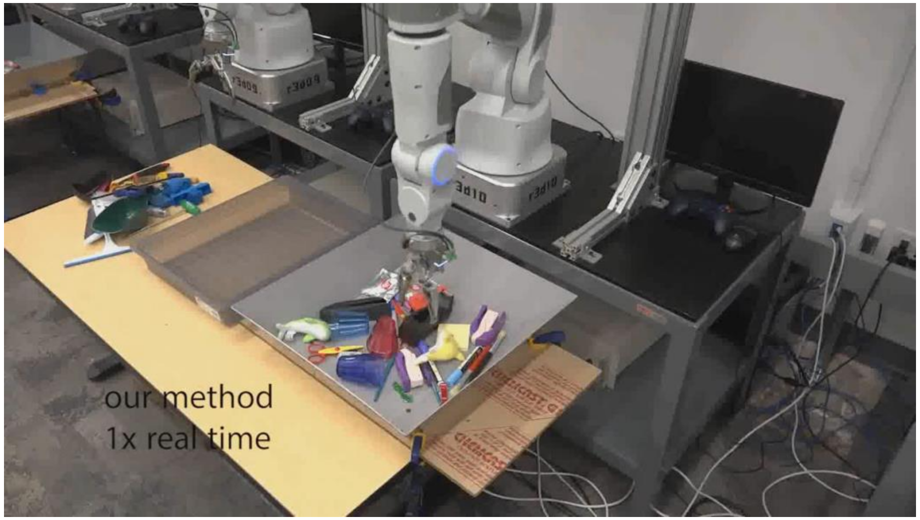
L., Pastor, Krizhevsky, Quillen ‘16

Open-Loop vs. Closed-Loop Grasping

open-loop grasping

closed-loop grasping





failure rate: 33.7%

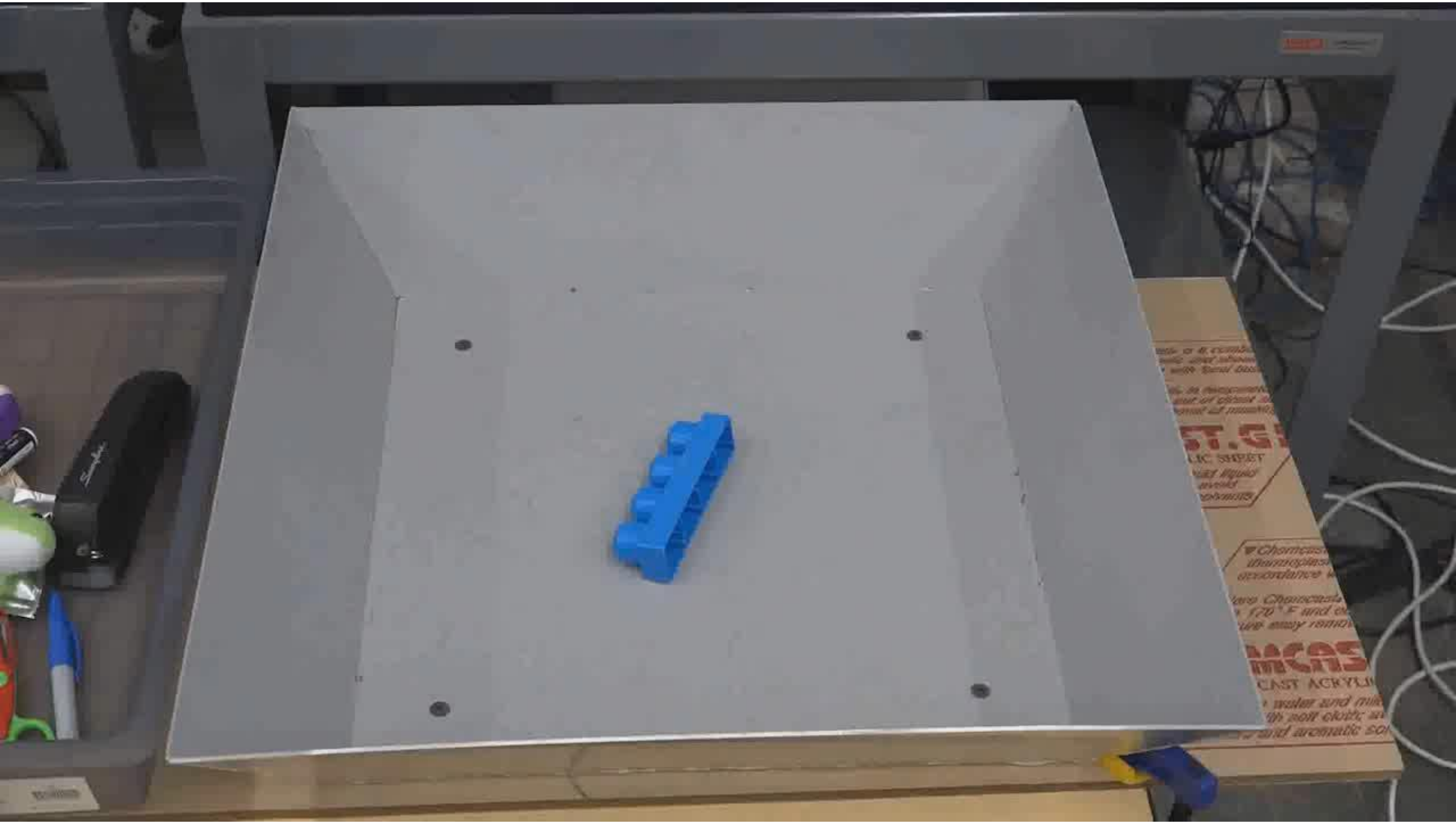
depth + segmentation
failure rate: 35%

failure rate: 17.5%

Pinto & Gupta, 2015

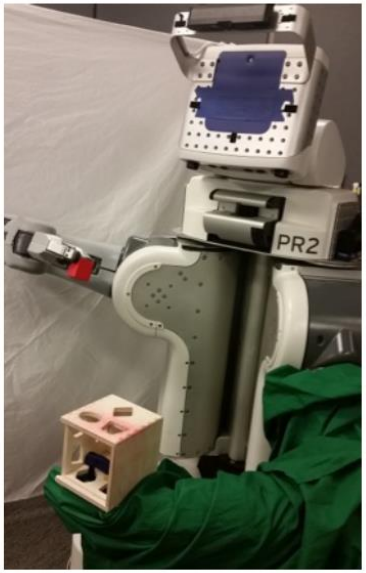
L., Pastor, Krizhevsky, Quillen '16

Grasping Experiments

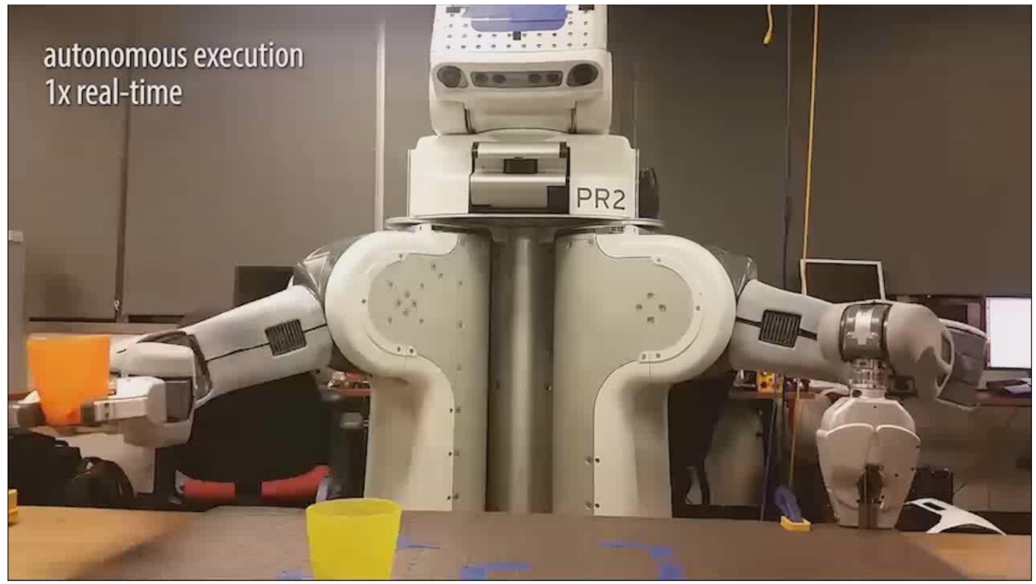


L., Pastor, Krizhevsky, Quillen ‘16

Learning what Success Means

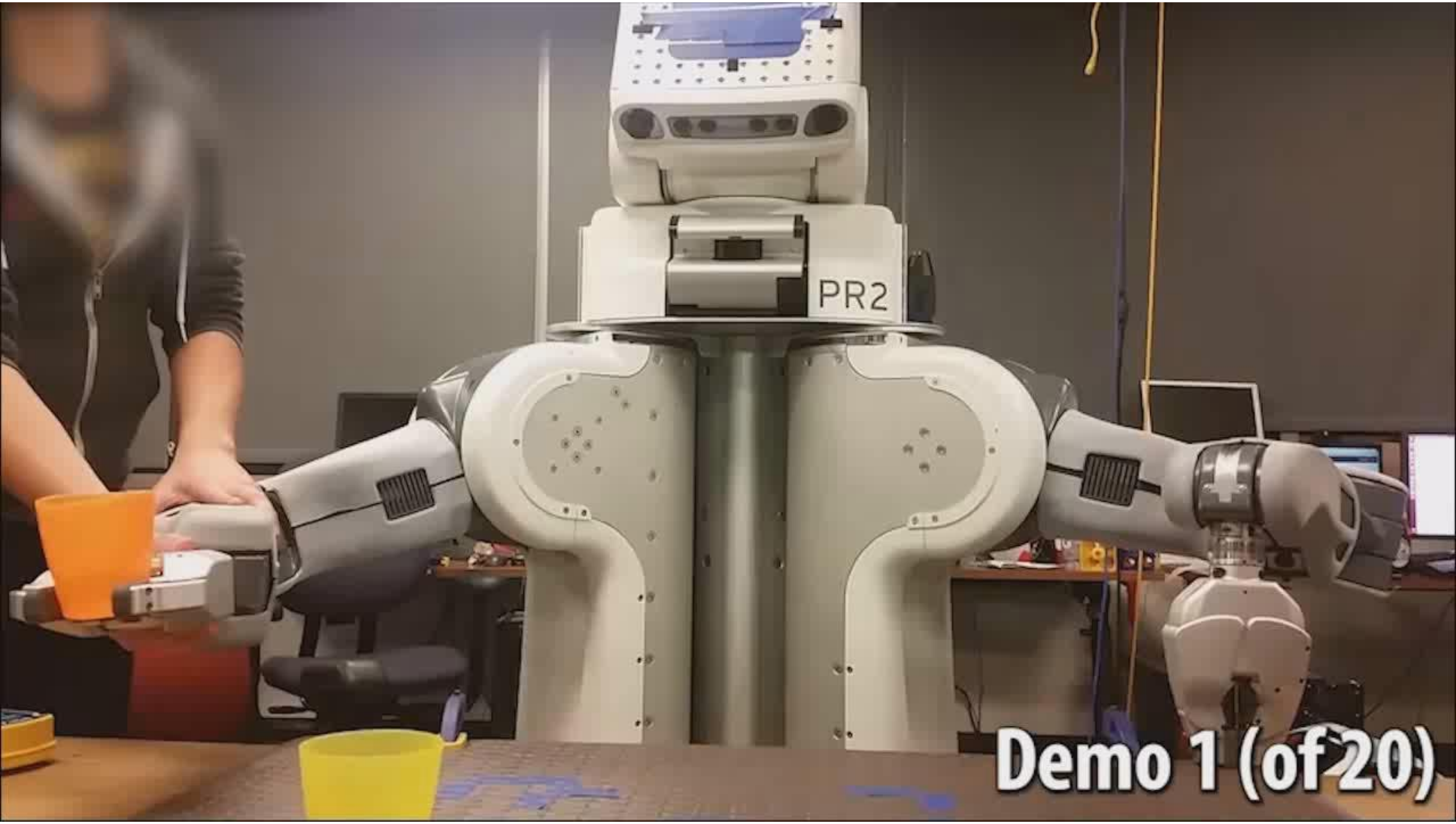


$$c(\mathbf{x}, \mathbf{u}) =$$
$$w_1 f_{\text{target}}(\mathbf{x}) +$$
$$w_2 f_{\text{torque}}(\mathbf{u})$$



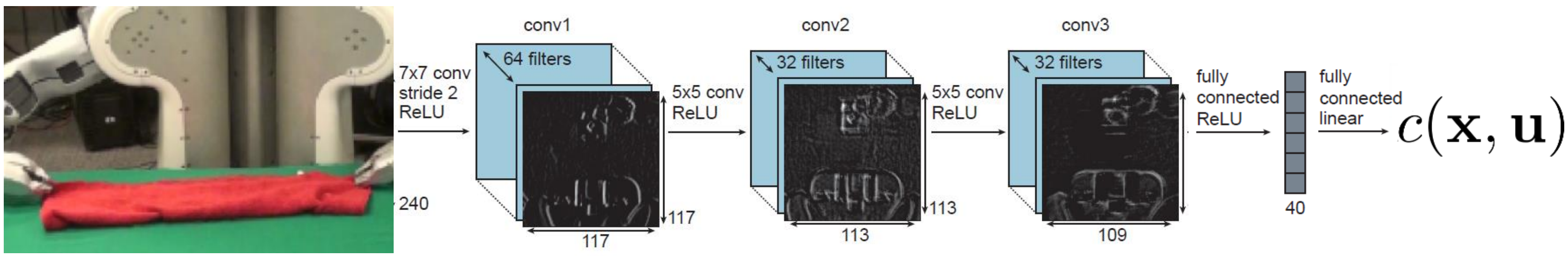
can we *learn* the cost with visual features?

Learning what Success Means

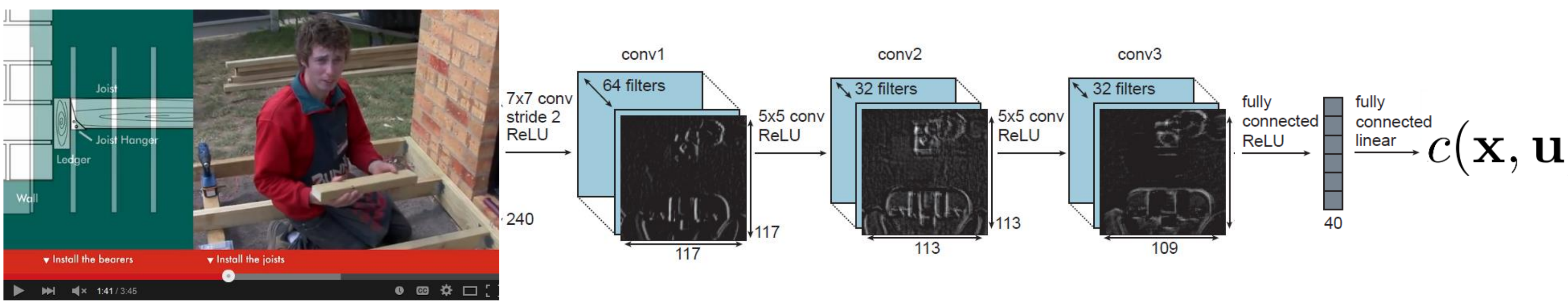


with C. Finn, P. Abbeel

Learning what Success Means

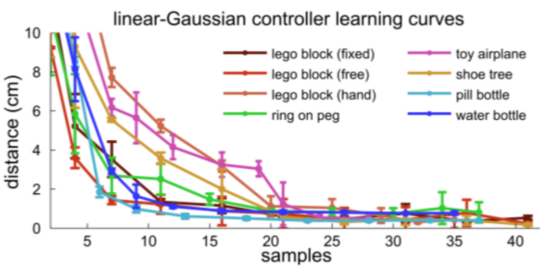


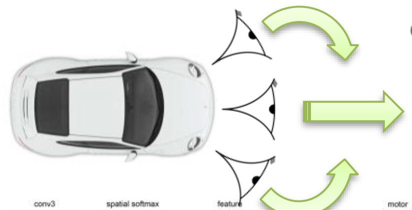
$$c(\mathbf{x}, \mathbf{u}) = ???$$

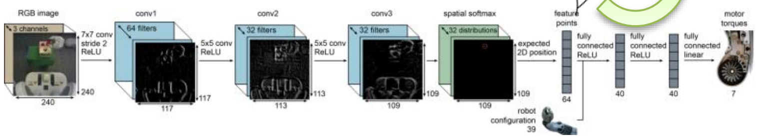


Challenges & Frontiers

- Algorithms
  - Sample complexity
  - Safety
  - Scalability
- Supervision
  - Automatically evaluate success
  - Learn cost functions



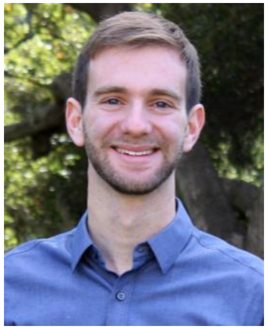




$$c(\mathbf{x}, \mathbf{u}) = ???$$

Acknowledgements

Greg Kahn



Tianhao Zhang

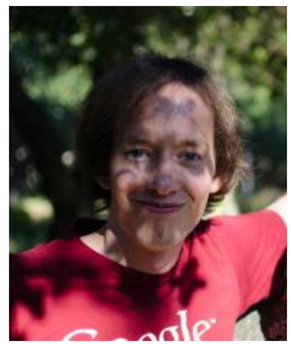
Chelsea Finn

Trevor Darrell

Pieter Abbeel

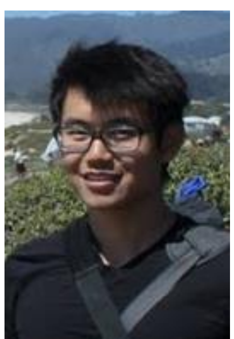
John Schulman

Philipp Moritz



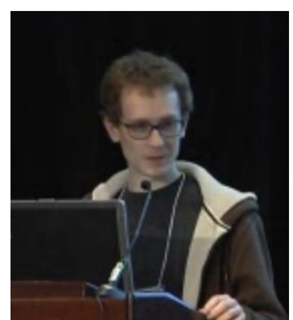
Michael Jordan

Shane Gu



Peter Pastor

Alex Krizhevsky



Deirdre Quillen