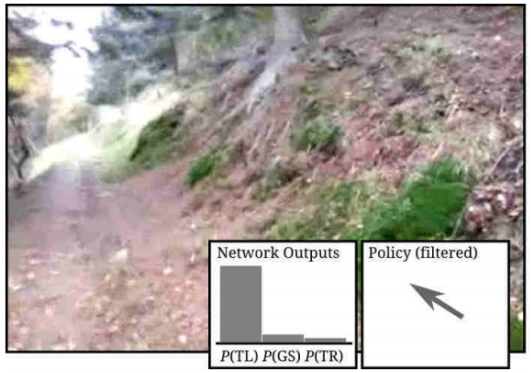
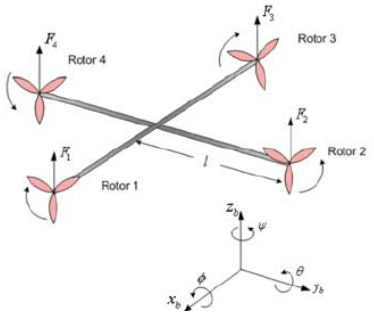
Reinforcement Learning

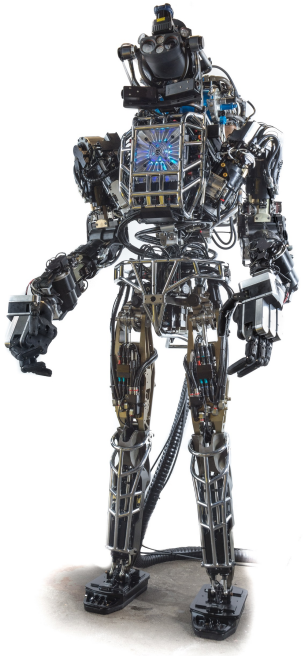
[shallow and deep]

Why go beyond imitation learning?

- Humans need to provide data, which is typically finite
- Humans are not good at providing some kinds of actions

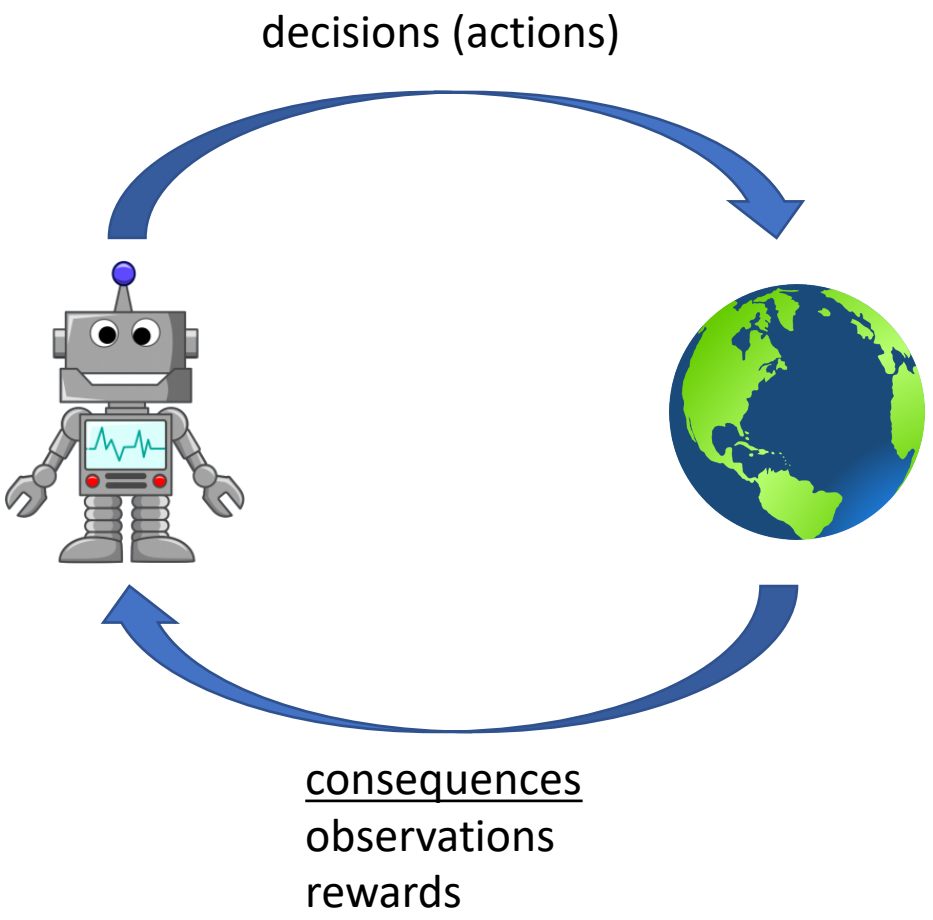






- Humans can learn autonomously; can our machines do the same?
  - Unlimited data from own experience
  - Continuous self-improvement

What is reinforcement learning?



Sequential decision making problem examples

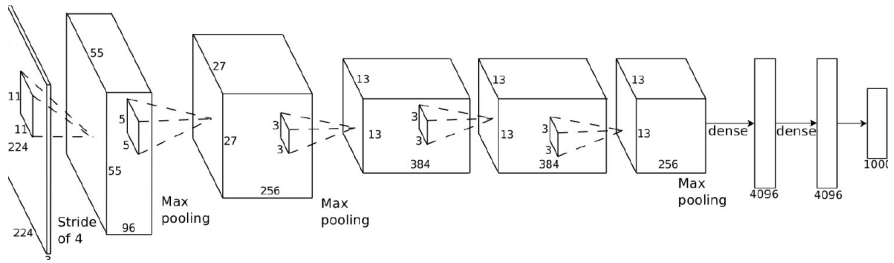
Actions: muscle contractions
Observations: sight, smell
Rewards: food

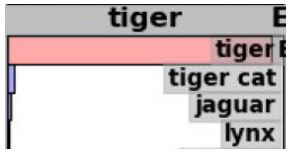
Actions: motor current or torque
Observations: camera images
Rewards: task success measure (e.g., running speed)

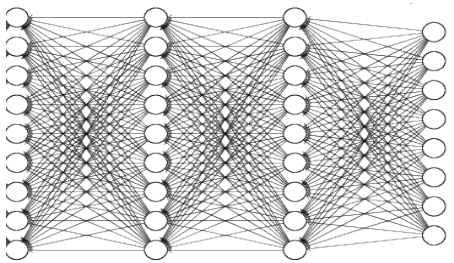
Actions: what to purchase
Observations: inventory levels
Rewards: profit

What does end-to-end/deep learning mean for sequential decision making?

perception



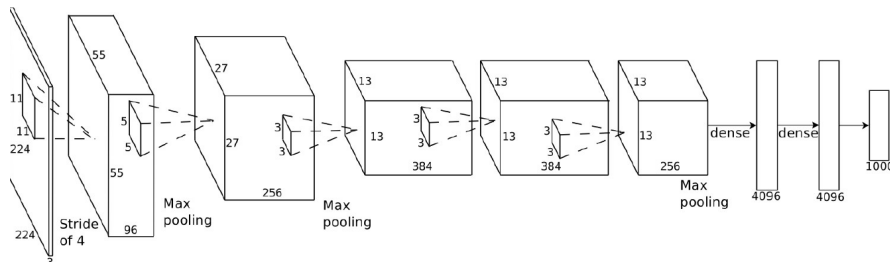


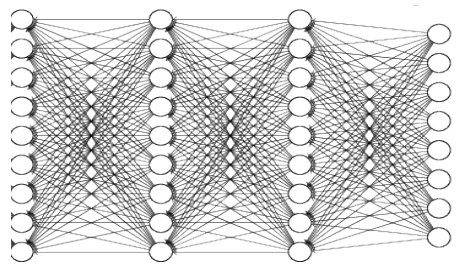


Action
(run away)

action

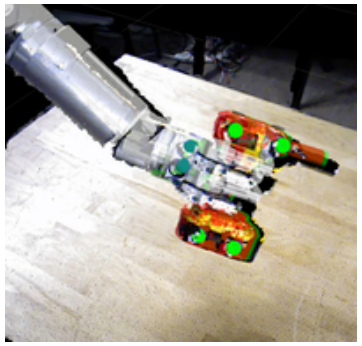
sensorimotor loop

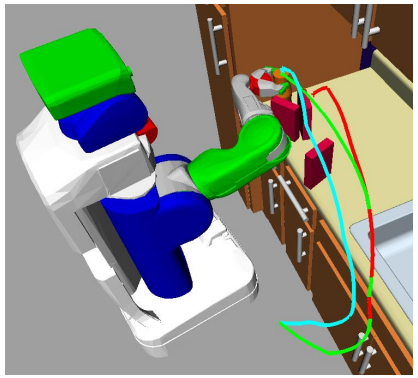




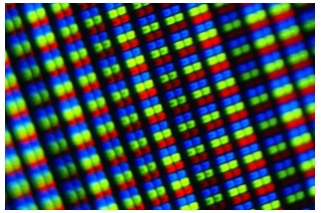
Action
(run away)

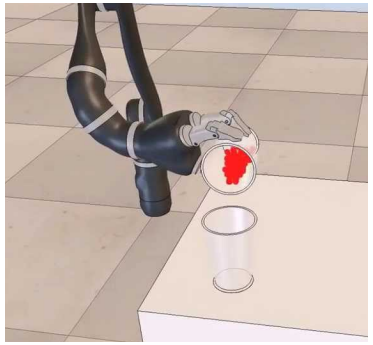
The standard robotic control pipeline

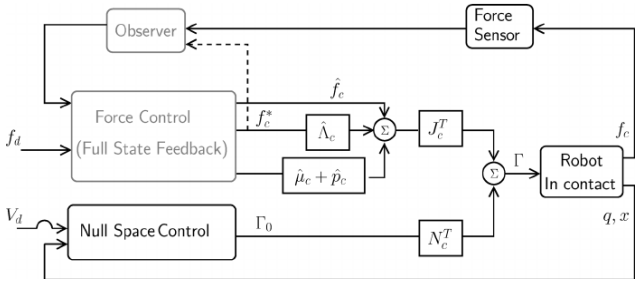




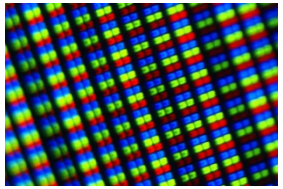
robotic control pipeline







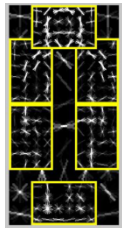
standard computer vision



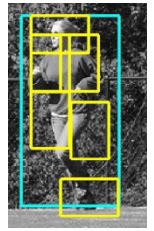
features (e.g. HOG)

mid-level features (e.g. DPM)

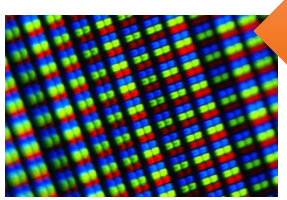
Felzenszwalb ‘08



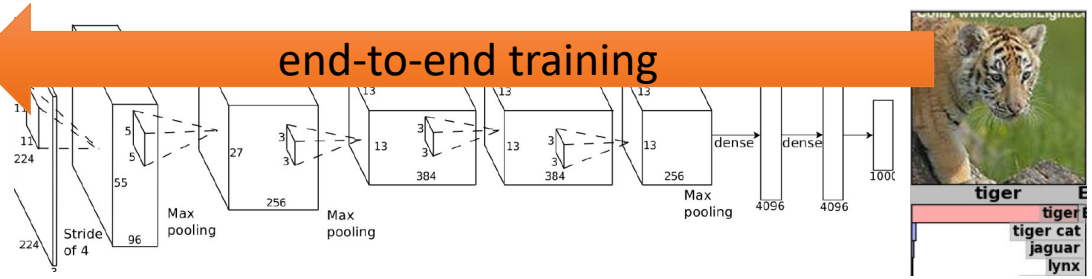
classifier (e.g. SVM)



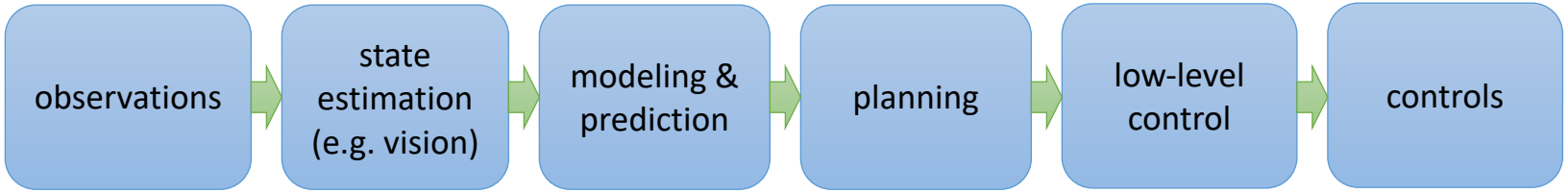
deep learning



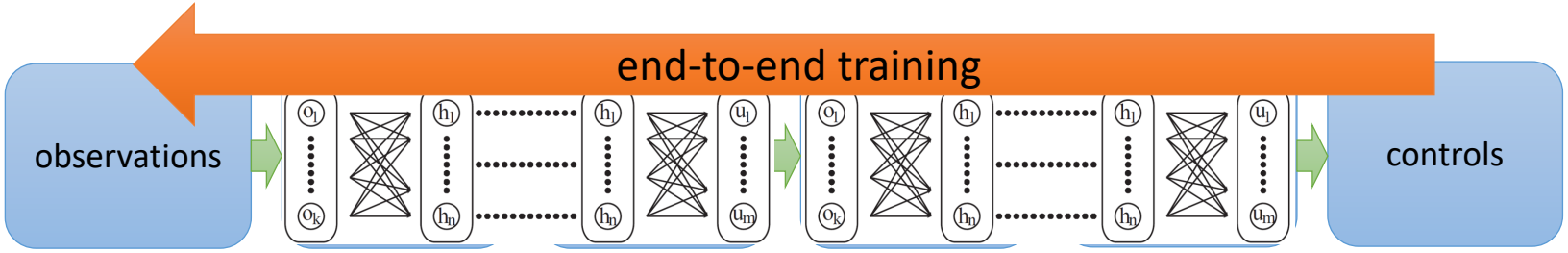
end-to-end training



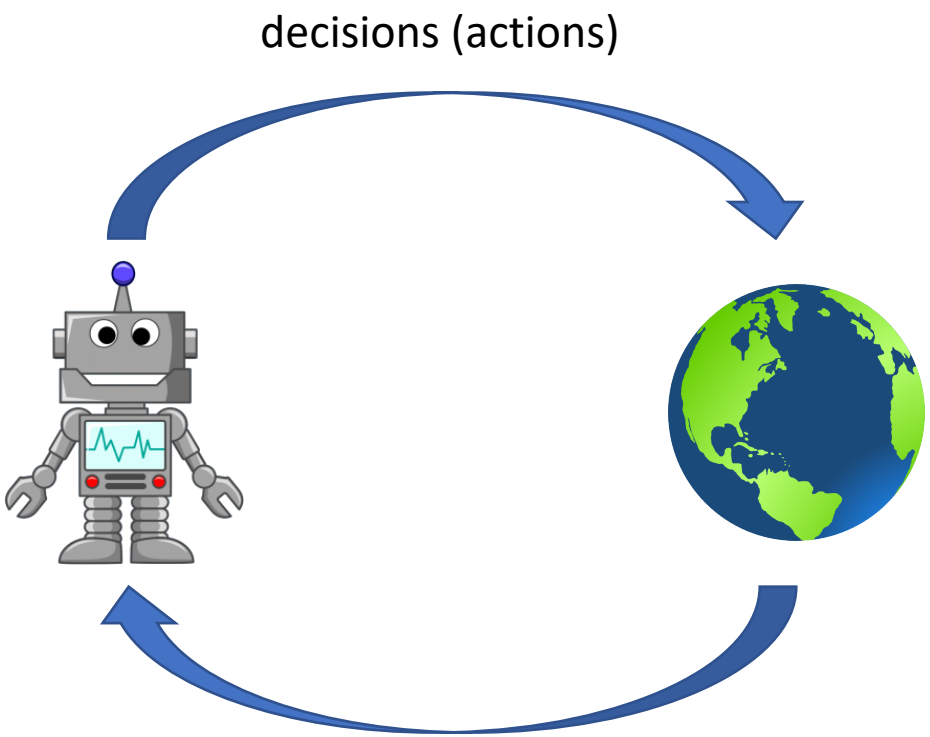
robotic control pipeline



deep robotic learning



The reinforcement learning problem



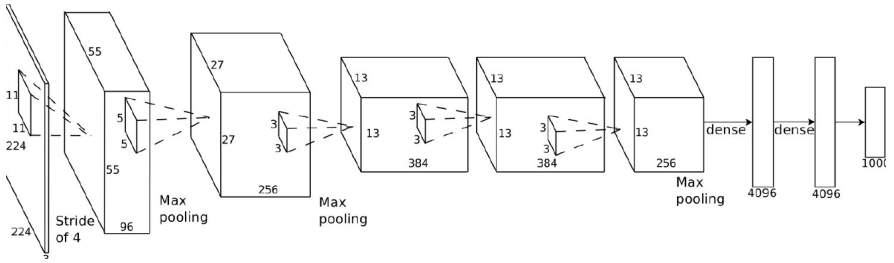
Actions: motor current or torque
Observations: camera images
Rewards: task success measure (e.g., running speed)

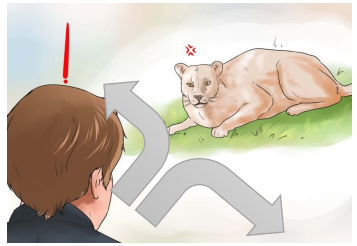
Actions: what to purchase
Observations: inventory levels
Rewards: profit

Actions: words in French
Observations: words in English
Rewards: BLEU score

Captures much of what we consider AI problems!

Recall: Terminology & notation

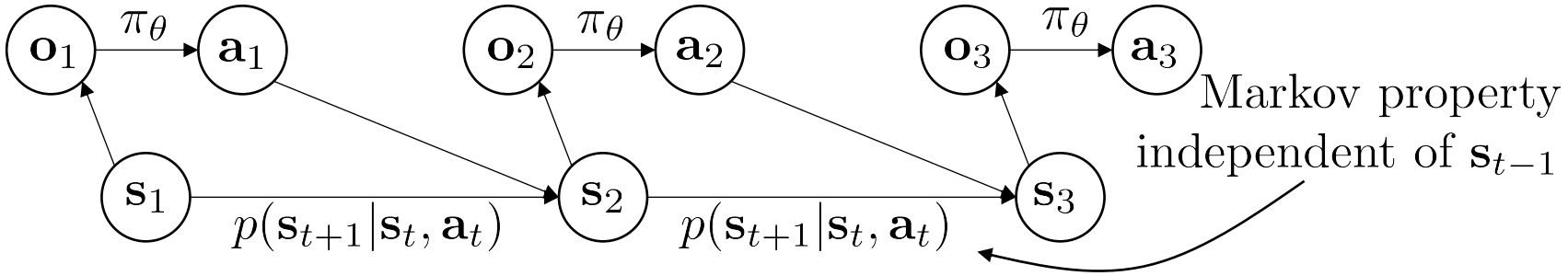




$\mathbf{o}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\mathbf{a}_t$

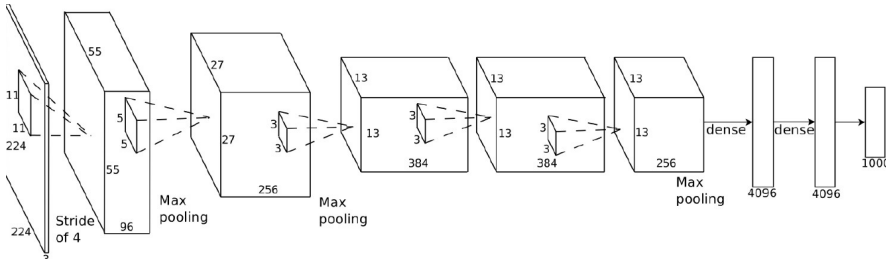
$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



Recall: Imitation Learning

$\mathbf{o}_t$

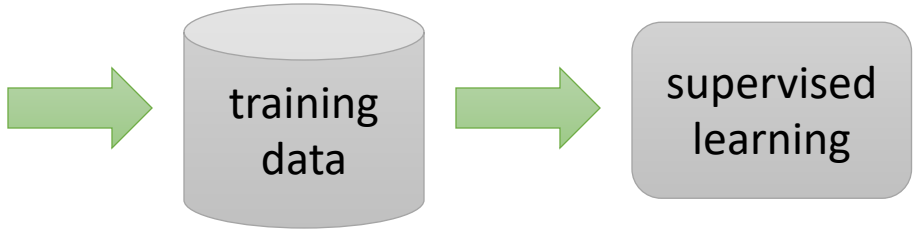


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

$\mathbf{o}_t$

$\mathbf{a}_t$

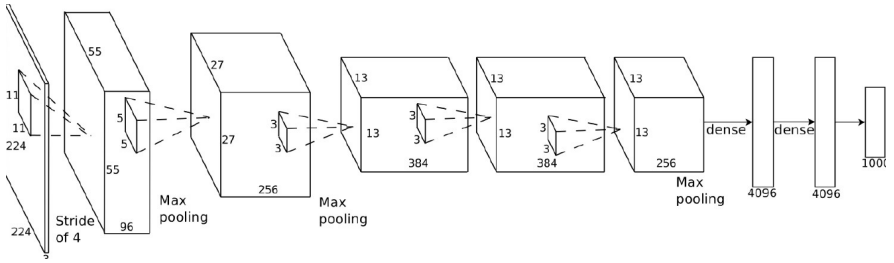


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

Images: Bojarski et al. ‘16, NVIDIA

Learning from Reward Functions

$\mathbf{o}_t$



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

$\mathbf{a}_t$

which action is better or worse?

$r(\mathbf{s}, \mathbf{a})$: reward function

tells us which states and actions are better

$\mathbf{s}$, $\mathbf{a}$, $r(\mathbf{s}, \mathbf{a})$, and $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ define Markov decision process

high reward

low reward

Definitions: Markov Chain

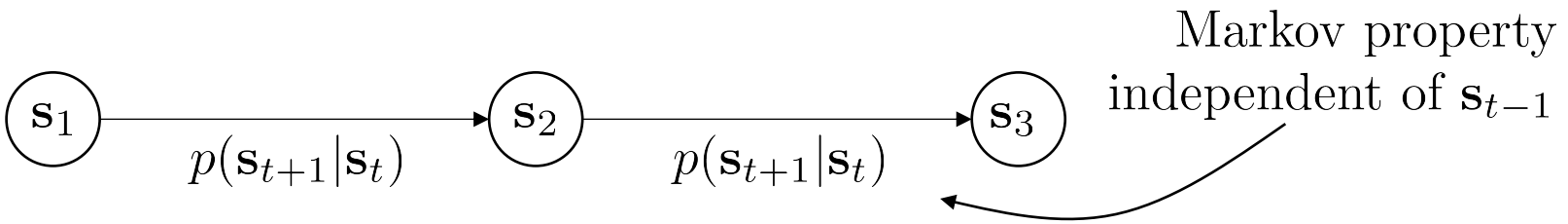
$\mathcal{M} = \{\mathcal{S}, \mathcal{T}\}$

$\mathcal{S}$ – state space — states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{T}$ – transition operator — $p(s_{t+1}|s_t)$

why "operator"? — let $\mu_{t,i} = p(s_t = i)$ — $\vec{\mu}_t$ is a vector of probabilities

let $\mathcal{T}_{i,j} = p(s_{t+1} = i|s_t = j)$ — then $\vec{\mu}_{t+1} = \mathcal{T}\vec{\mu}_t$



Definitions: Markov Decision Process

$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$

$\mathcal{S}$ – state space states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{A}$ – action space actions $a \in \mathcal{A}$ (discrete or continuous)

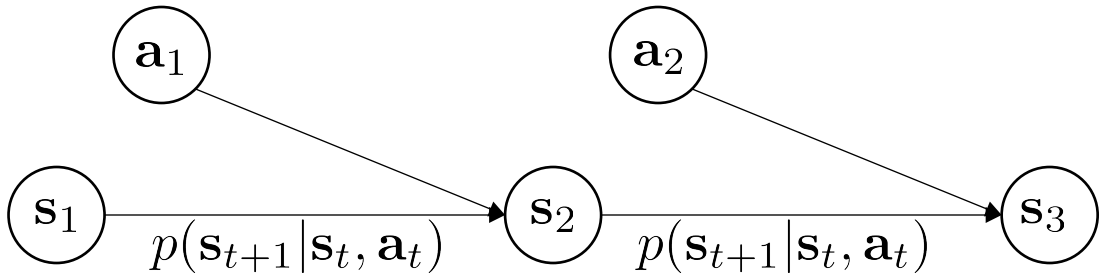
$\mathcal{T}$ – transition operator (now a tensor!)

let $\mu_{t,j} = p(s_t = j)$

let $\xi_{t,k} = p(a_t = k)$

let $\mathcal{T}_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k)$

$$\mu_{t,i} = \sum_{j,k} \mathcal{T}_{i,j,k} \mu_{t,j} \xi_{t,k}$$



Definitions: Markov Decision Process

$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$

$\mathcal{S}$ – state space — states $s \in \mathcal{S}$ (discrete or continuous)

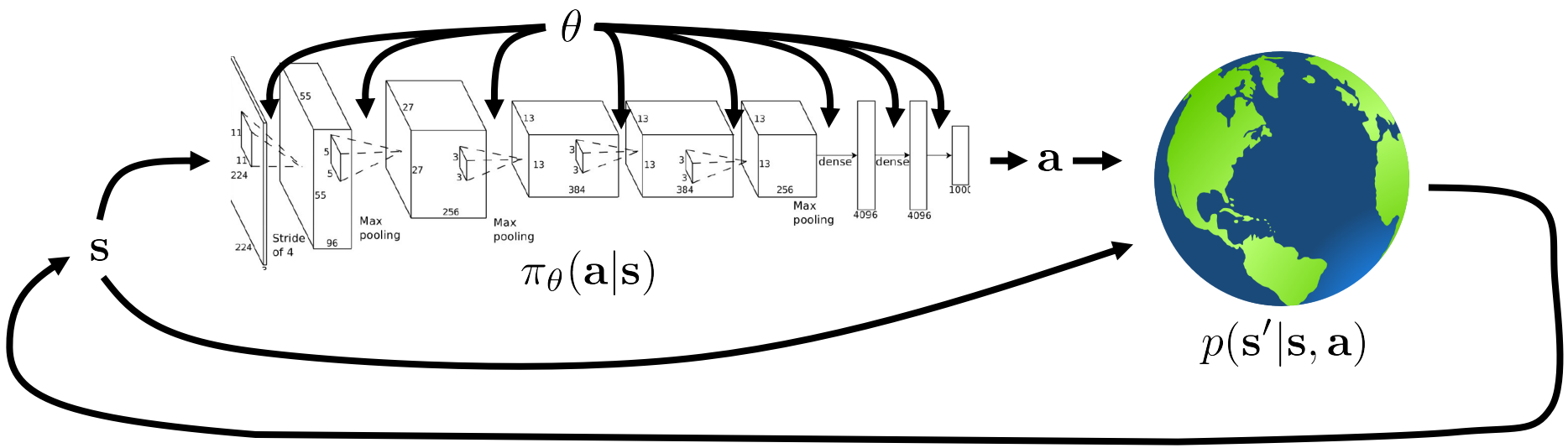
$\mathcal{A}$ – action space — actions $a \in \mathcal{A}$ (discrete or continuous)

$\mathcal{T}$ – transition operator (now a tensor!)

r – reward function — $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

$r(s_t, a_t)$ – reward

The goal of reinforcement learning



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

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Learned Model Predictive Control

[Learning only the MDP transition function]

Prediction

"the idea that we **predict the consequences of our motor commands** has emerged as an important theoretical concept in all aspects of sensorimotor control"

Prediction Precedes Control in Motor Learning

J. Randall Flanagan,[1,*] Philipp Vetter,[2] Roland S. Johansson,[3] and Daniel M. Wolpert[2]

Procedures for details). Figure 1 shows, for a single subject, the hand path (top trace) and the grip (middle)

Predicting the Consequences of Our Own Actions: The Role of Sensorimotor Context Estimation

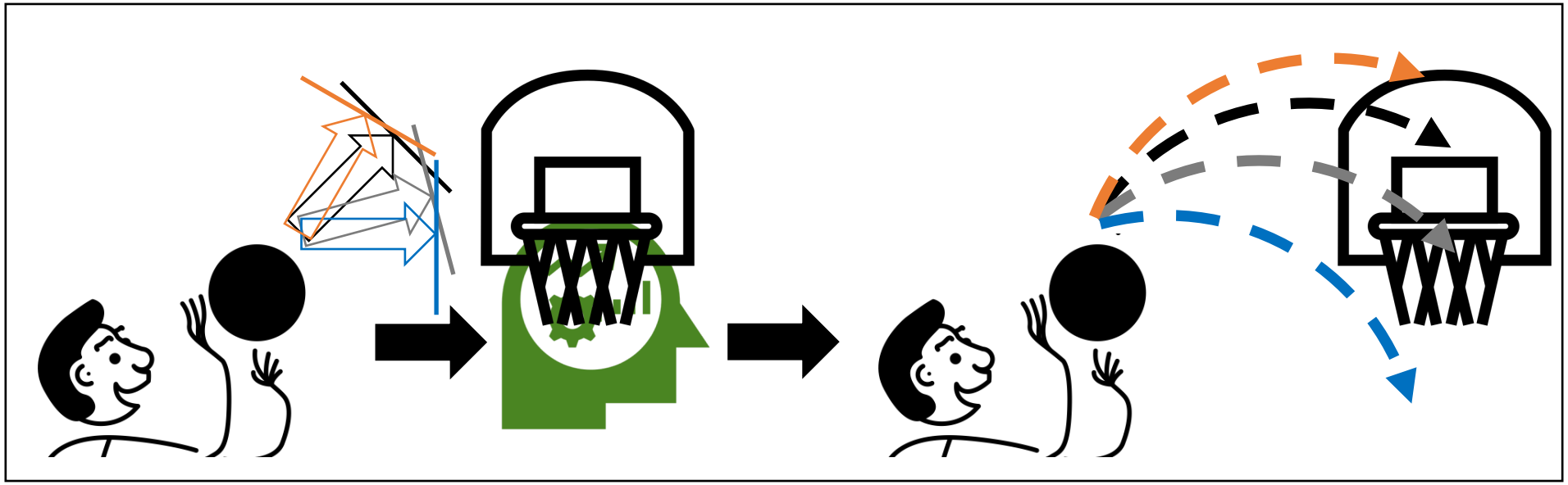
Sarah J. Blakemore, Susan J. Goodbody, and Daniel M. Wolpert

Sobell Department of Neurophysiology, Institute of Neurology, University College London, London WC1N 3BG,

Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects

Rajesh P. N. Rao[1] and Dana H. Ballard[2]

Model Predictive Control [MPC]: the 1-slide version



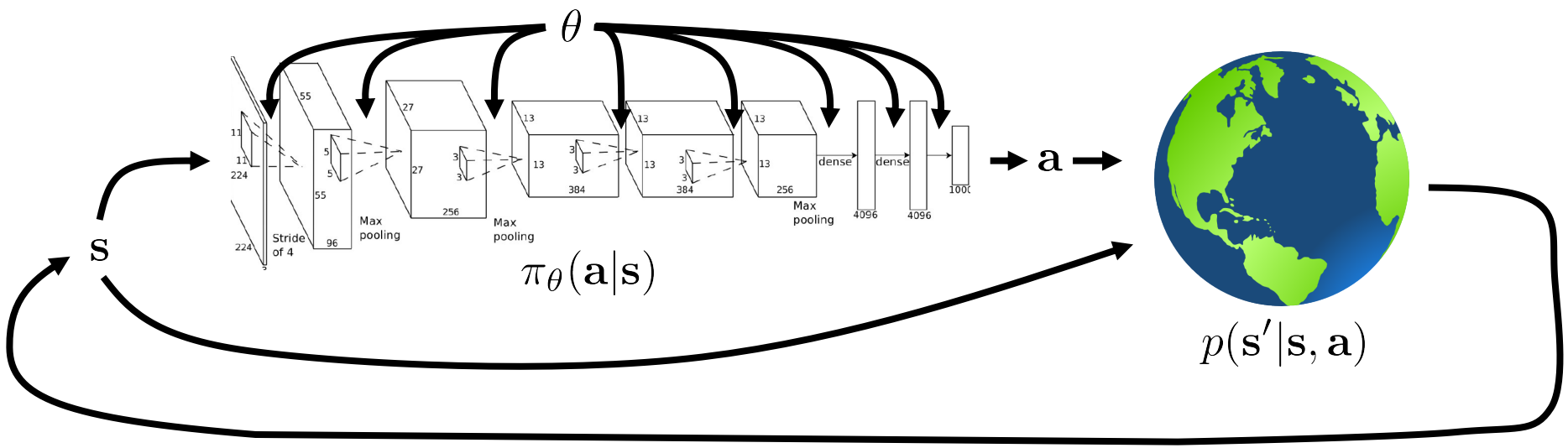
Learn to predict future states (e.g. video frames) given actions.

Test time:

1. Sample actions and forecast their effects.
2. Select action with best forecasted outcome.

Predictive models assist robotic control.

Recap: the reinforcement learning objective



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

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

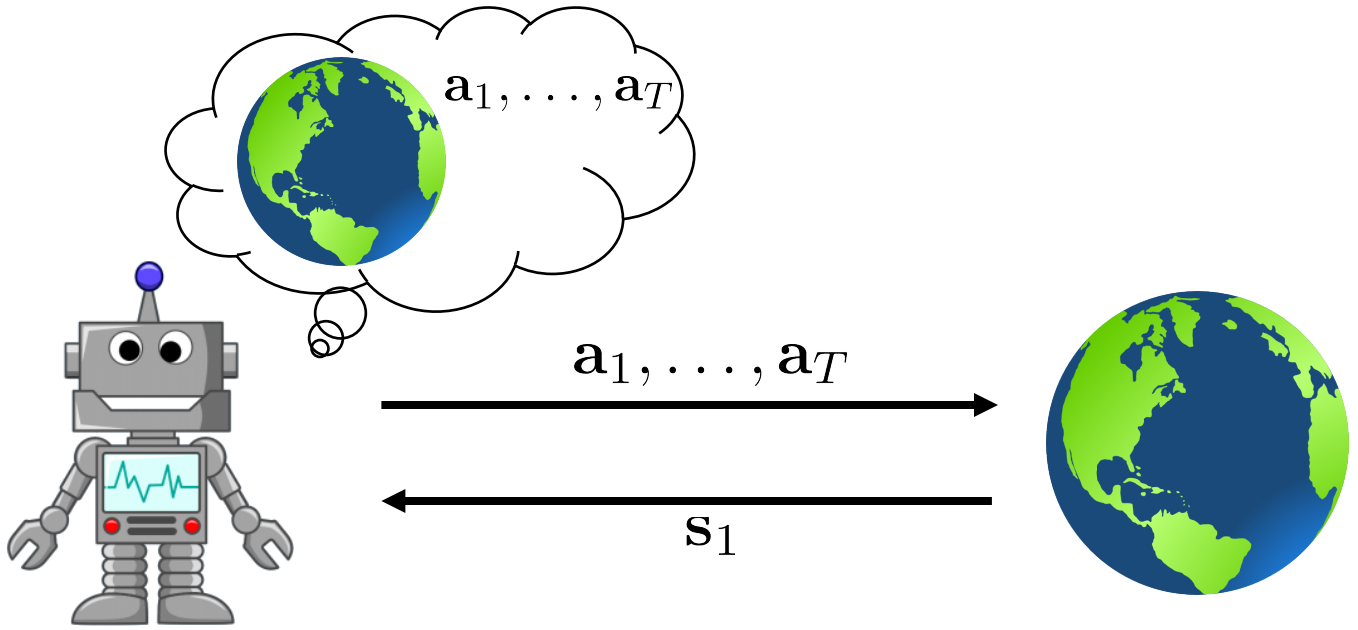
Model-based reinforcement learning

- What if we could learn the transition dynamics P(s_{t+1} | s_t, a_t)? This function is often called the “dynamics model” or simply “model”.

First, assume we known transition function. What could we do then?

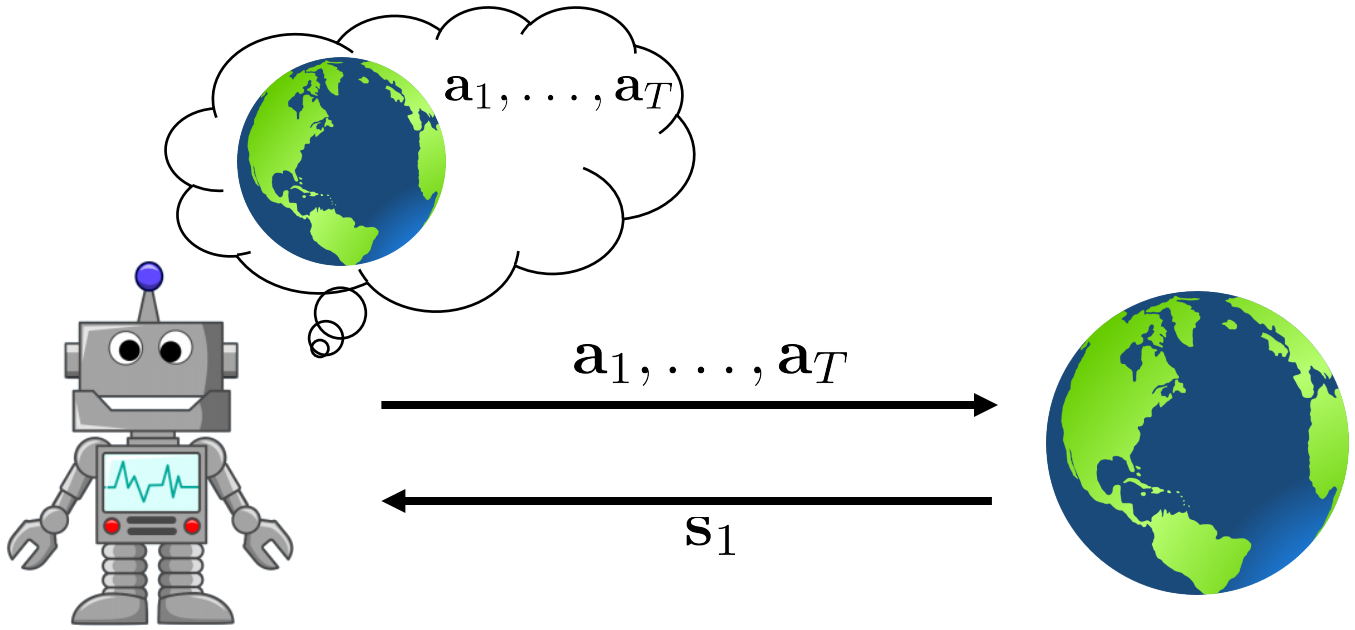
- One possible answer: “Planning through the model”.

Planning: the deterministic case



$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \text{ s.t. } \mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$$

Planning: the stochastic open-loop case



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

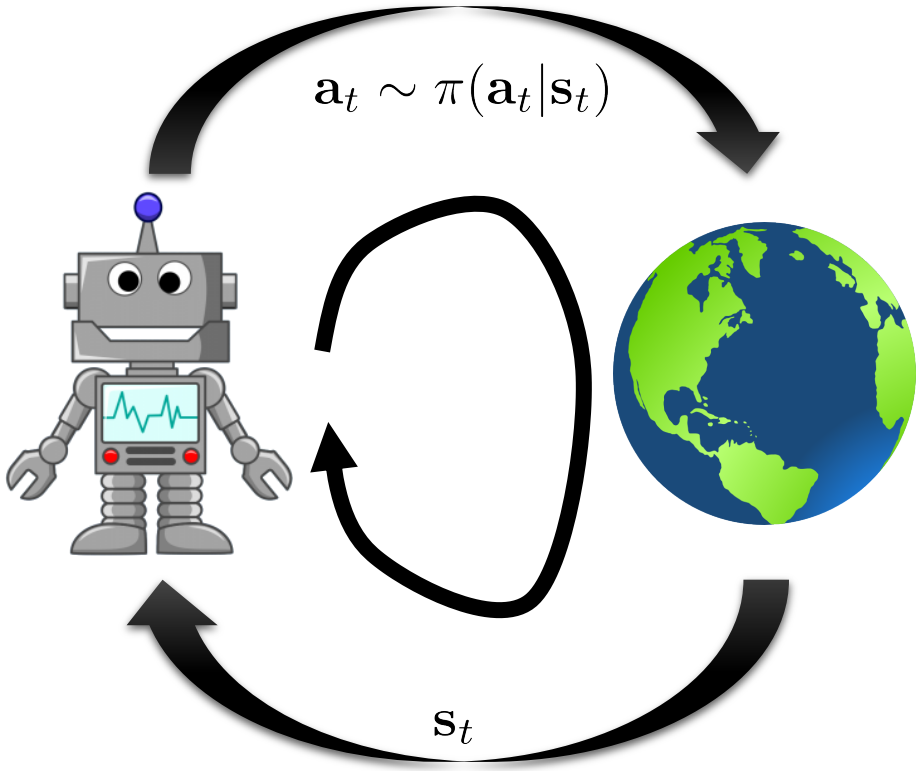
why is this suboptimal?

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} E \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{a}_1, \ldots, \mathbf{a}_T \right]$$

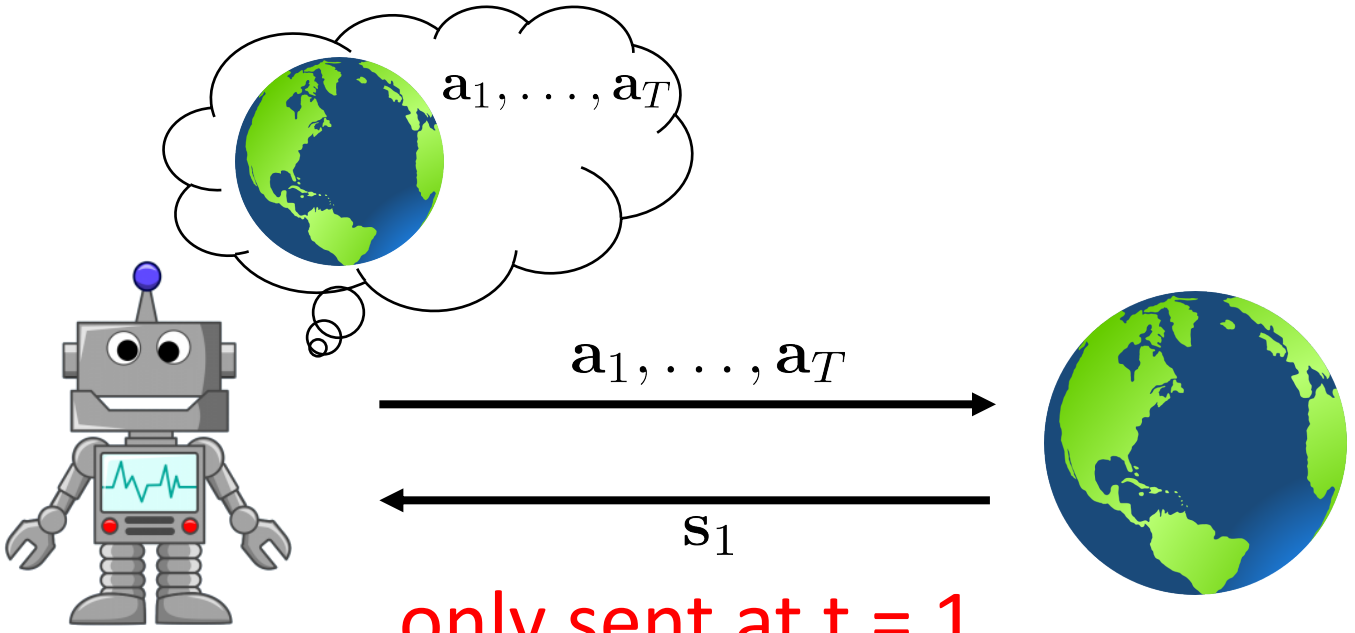
Aside: terminology

what is this “loop”?

closed-loop

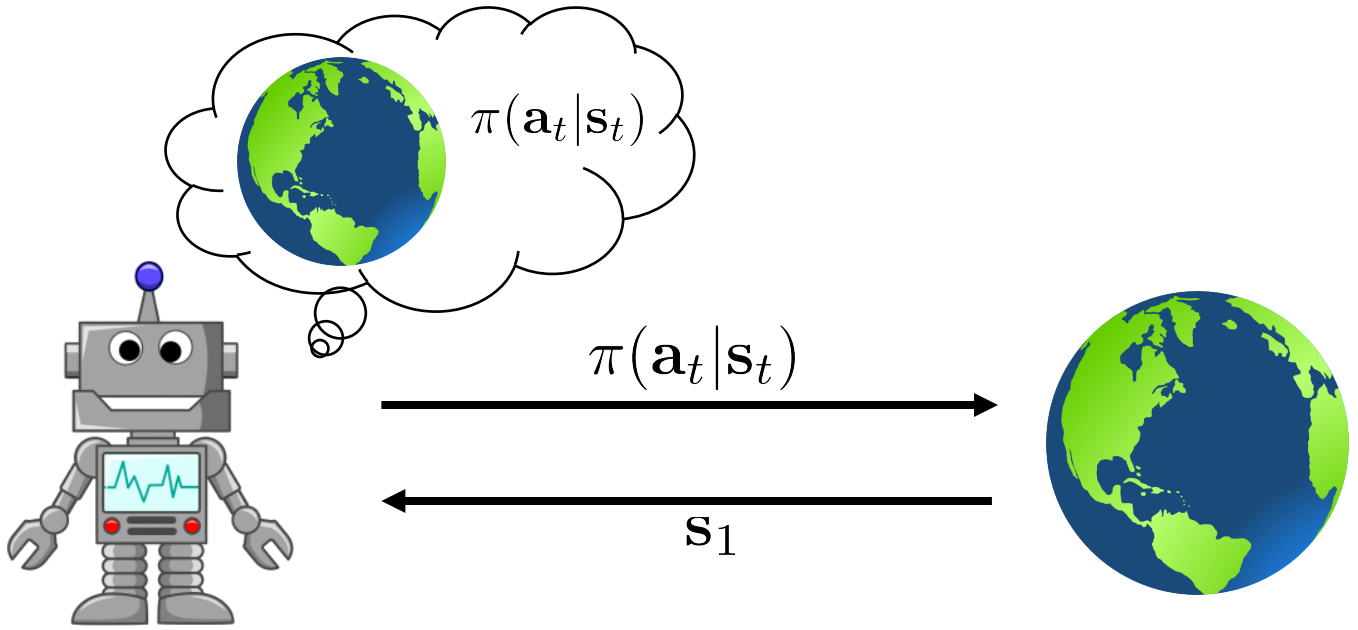


open-loop



only sent at t = 1,
then it’s one-way!

Planning: the stochastic **closed**-loop case



form of π?

neural net

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

$$\pi = \arg\max_{\pi} E_{\tau \sim p(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Sampling-based maximization

$$\pi = \arg\max_{\pi} E_{\tau \sim p(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

abstract away optimal control/planning:

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg\max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \underbrace{J(\mathbf{a}_1, \ldots, \mathbf{a}_T)}_{\text{don't care what this is}}$$

$$\mathbf{A} = \arg\max_{\mathbf{A}} J(\mathbf{A})$$

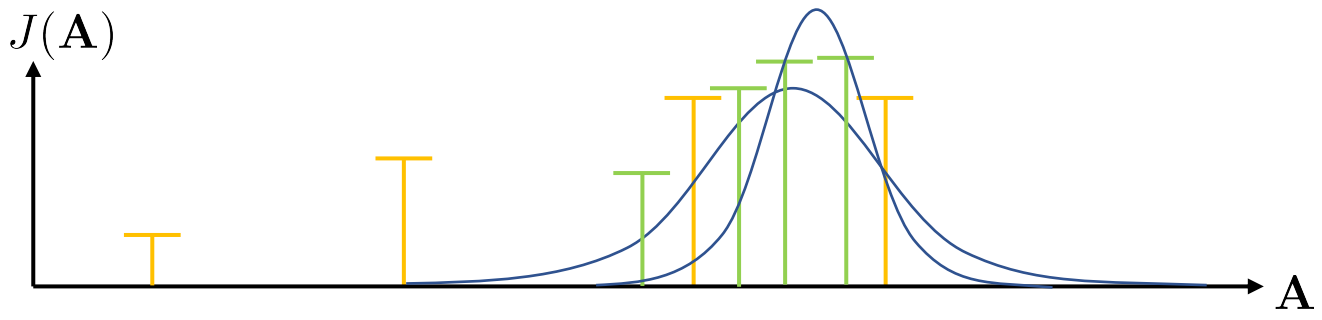
simplest method: guess & check

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

Cross-entropy method (CEM)

1. pick $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from some distribution (e.g., uniform)
2. choose $\mathbf{A}_i$ based on $\arg\max_i J(\mathbf{A}_i)$

can we do better?



typically use Gaussian distribution

see also: CMA-ES (sort of like CEM with momentum)

cross-entropy method with continuous-valued inputs:

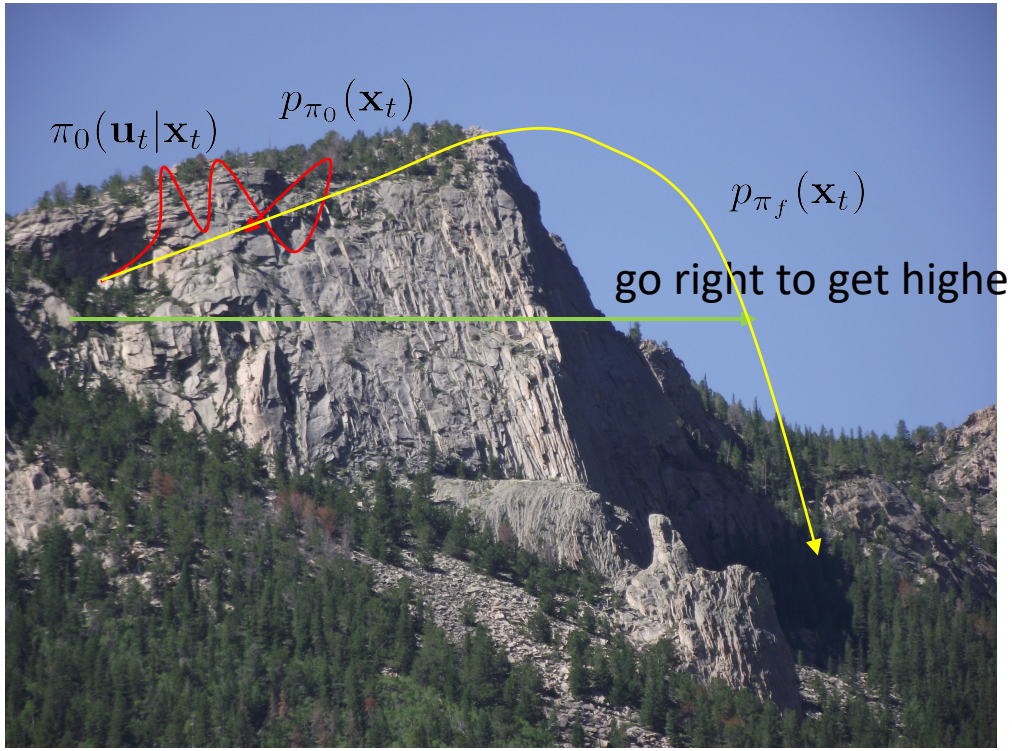
1. sample $\mathbf{A}_1, \ldots, \mathbf{A}_N$ from $p(\mathbf{A})$
2. evaluate $J(\mathbf{A}_1), \ldots, J(\mathbf{A}_N)$
3. pick the *elites* $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$ with the highest value, where $M < N$
4. refit $p(\mathbf{A})$ to the elites $\mathbf{A}_{i_1}, \ldots, \mathbf{A}_{i_M}$

How could we use this idea with a *learned* model?

model-based reinforcement learning version 0.5:

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions

Distribution mismatch issue



1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions

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

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

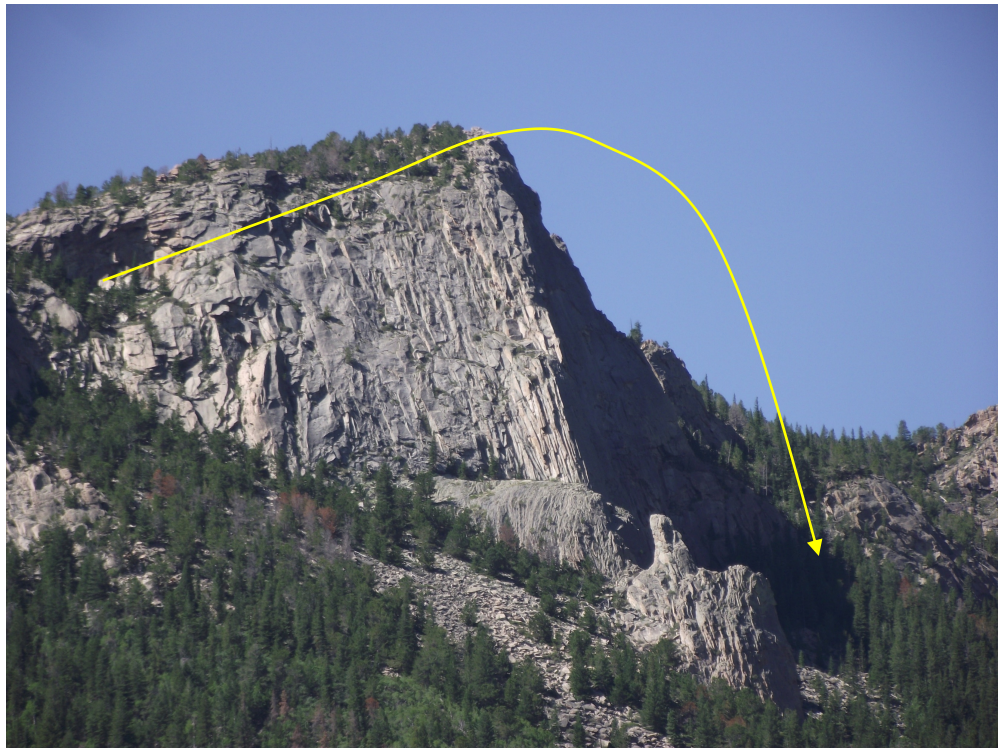
Remember how we mitigate distribution shift?

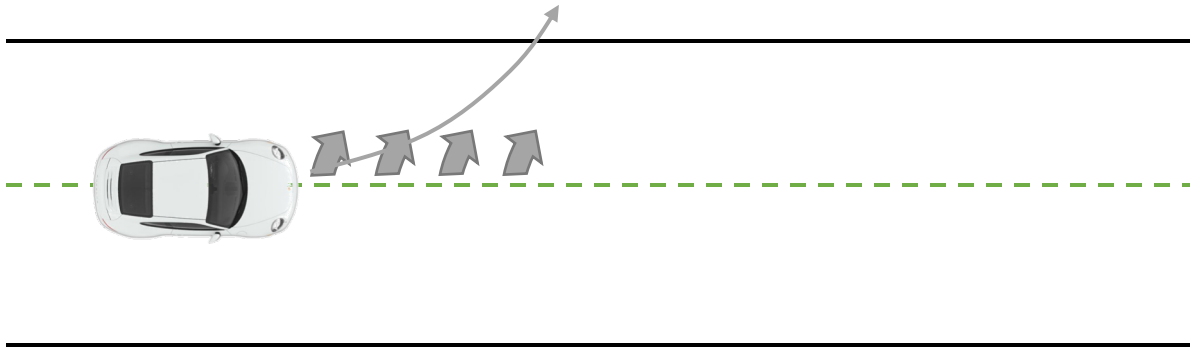
Hint: Imitation Learning… DAGGER

model-based reinforcement learning version 1.0:

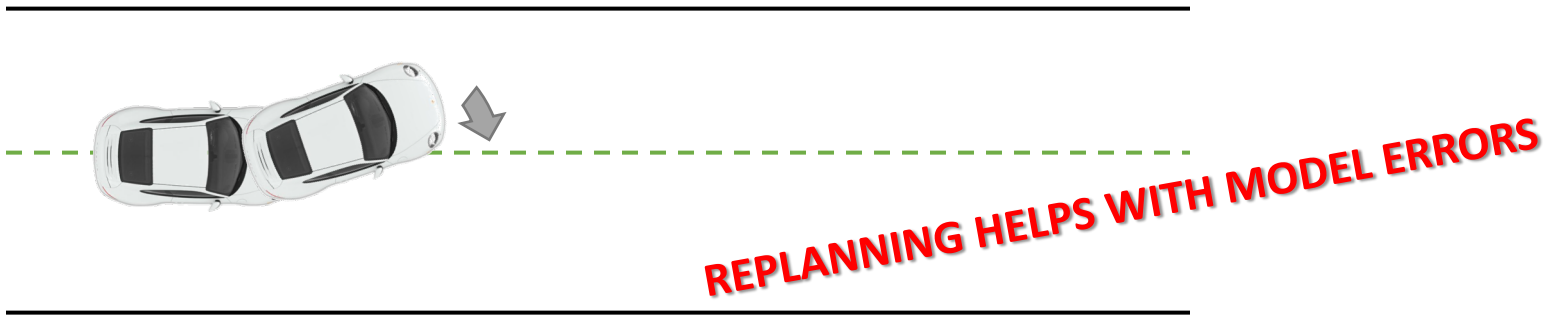
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s}, \mathbf{a})$ to choose actions
4. execute those actions and add the resulting data $\{(\mathbf{x}, \mathbf{u}, \mathbf{x}')_j\}$ to $\mathcal{D}$

What if we make a mistake?





Replanning for robustness to model errors



model-based reinforcement learning version 1.5:

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

every N steps

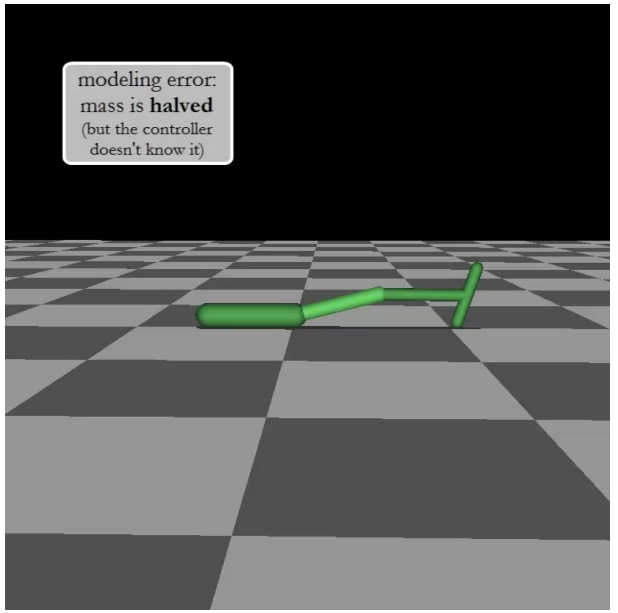
How to replan?

model-based reinforcement learning version 1.5:

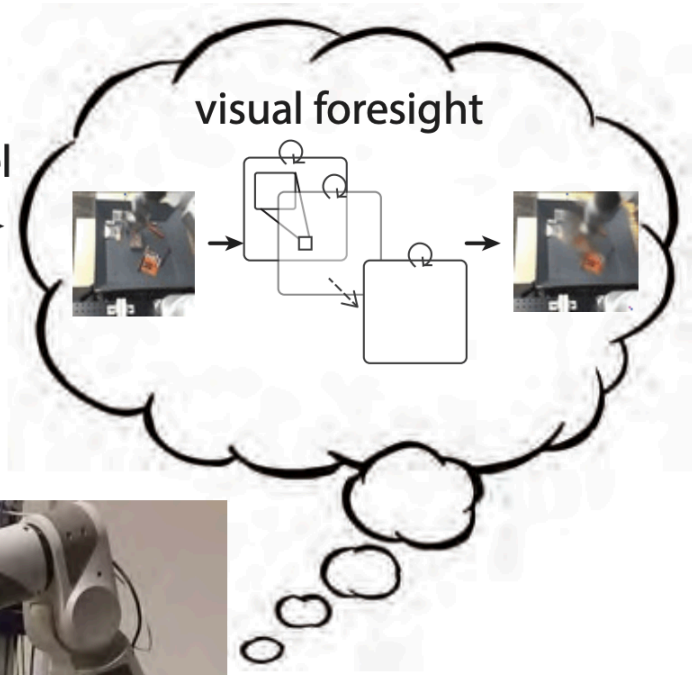
1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn dynamics model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \|f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i\|^2$
3. plan through $f(\mathbf{s},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

every N steps

- The more you replan, the less perfect each individual plan needs to be
- Can use shorter horizons
- Even random sampling can often work well here!



Example: Visual MPC for object pushing



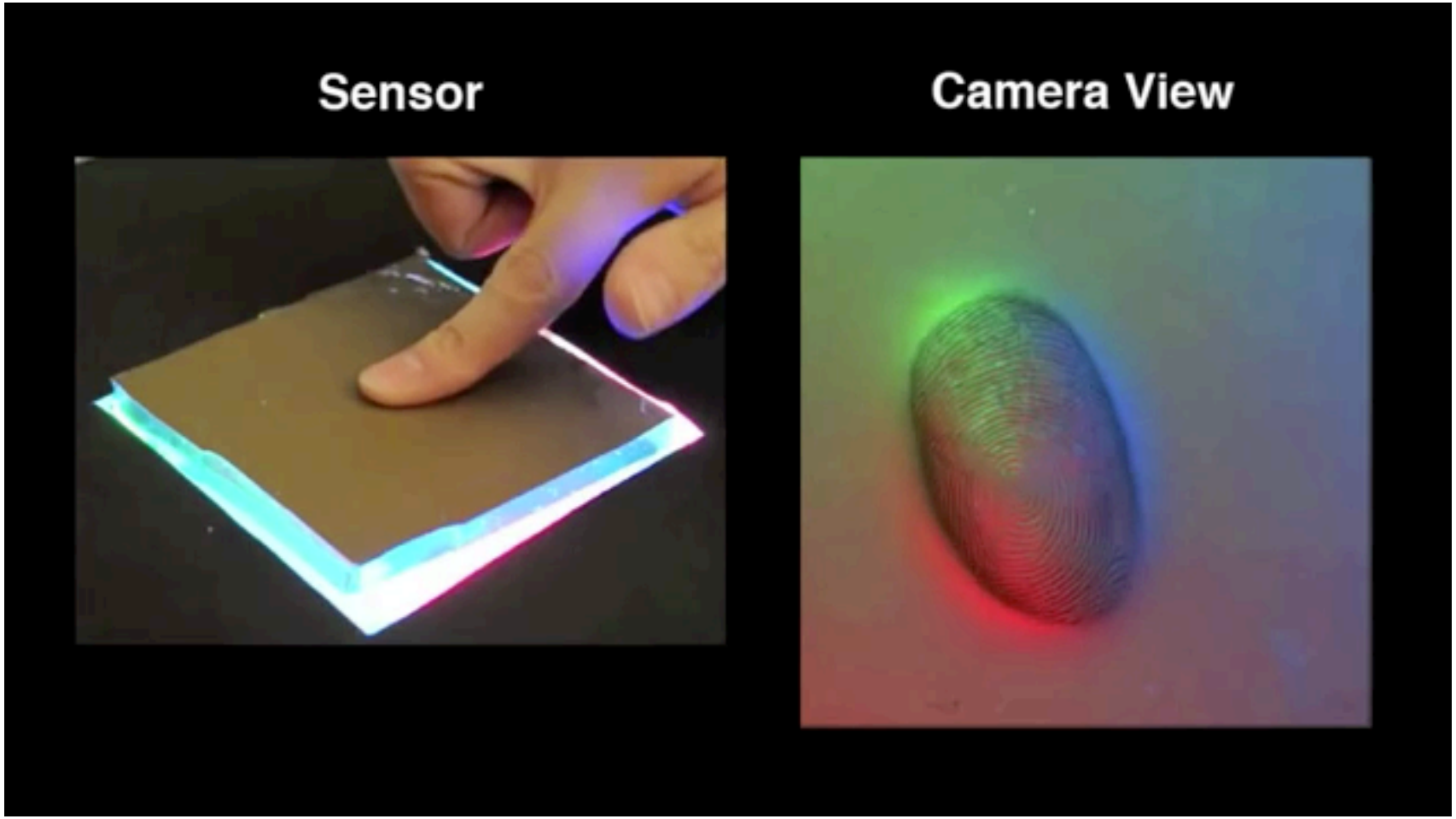
indicated goal

Ebert 2017

“Tactile Video” Prediction

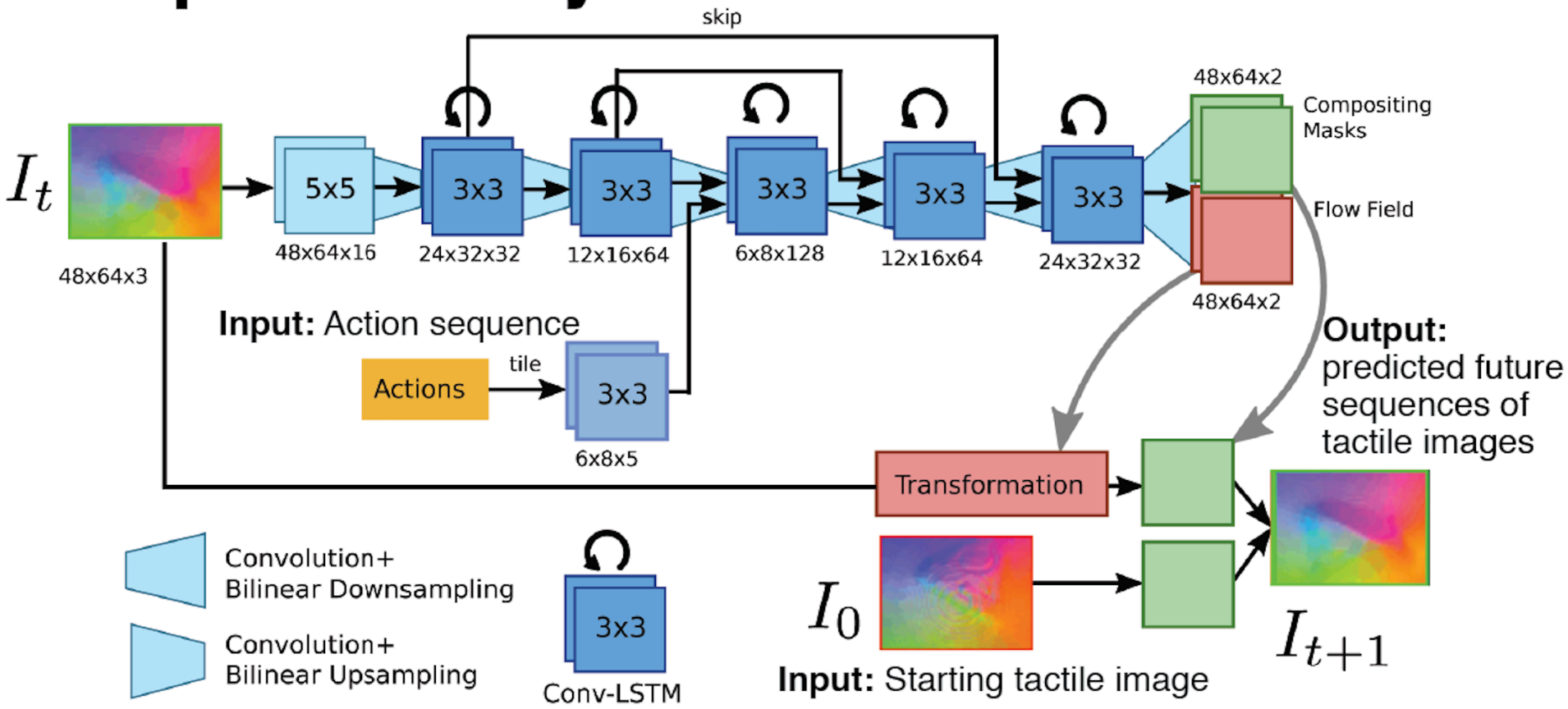
[Example use cases for MPC]

Gelsight: touch from vision



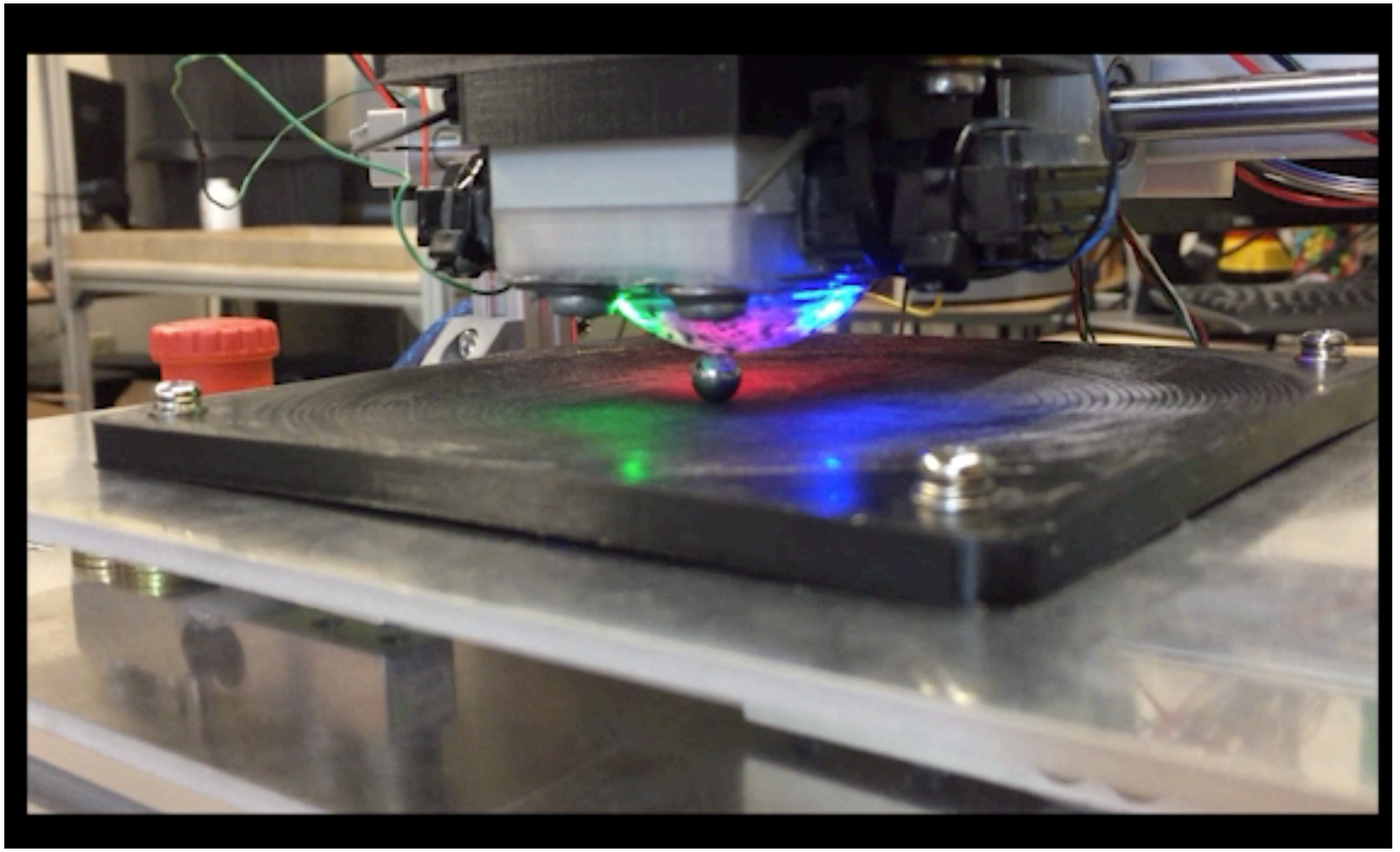
Johnson, Adelson, Raj. Siggraph 2009

Deep Video Prediction



[Oh et al, NIPS 2015; Mathieu et al, ICLR 2015; Vondrick et al, NIPS 2016; Xue et al, NIPS 2016; Larsen et al, ICML 2016; Walker et al, ECCV 2016; Finn et al, 2016; Finn et al, ICRA 2017; Ebert et al, CORL 2017; Denton et al, ICML 2018; Jayaraman et al, ICLR 2019]

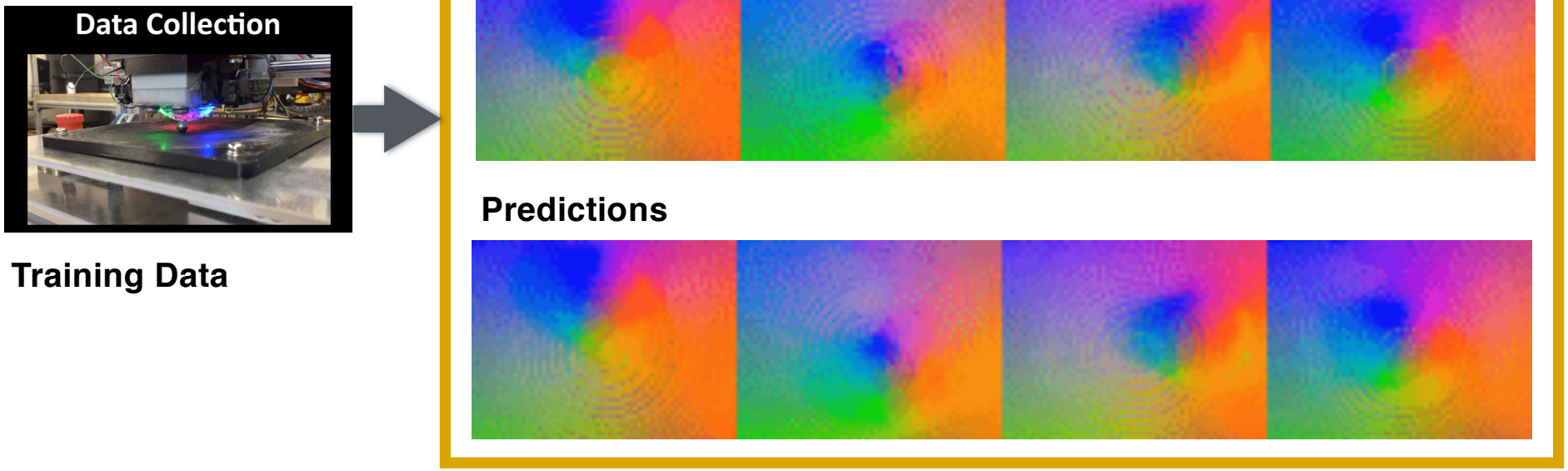
Self-supervision through exploration



Stephen Tian*, Frederik Ebert*, Dinesh Jayaraman, Roberto Calandra, Mayur Mudigonda, Sergey Levine and Chelsea Finn, ICRA 2019

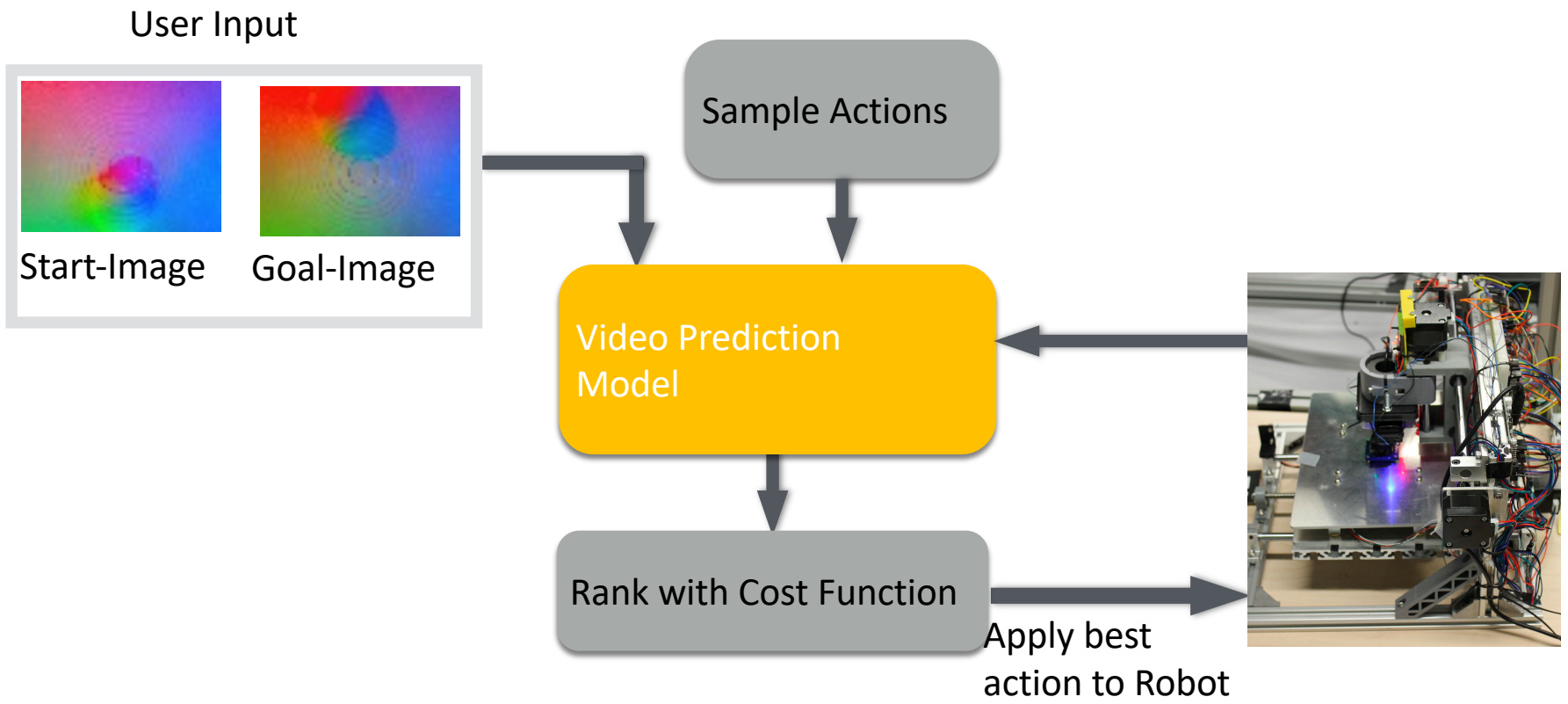
Self-supervised prediction

Action-Conditioned Video-Prediction Model



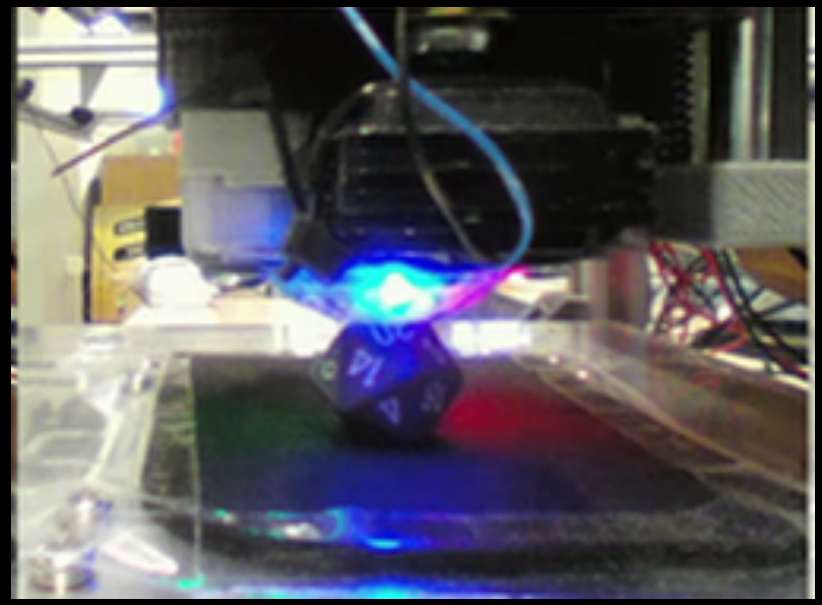
Stephen Tian*, Frederik Ebert*, Dinesh Jayaraman, Roberto Calandra, Mayur Mudigonda, Sergey Levine and Chelsea Finn, ICRA 2019

Visual MPC: Test Time



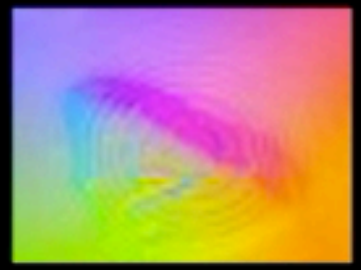
Stephen Tian*, Frederik Ebert*, Dinesh Jayaraman, Roberto Calandra, Mayur Mudigonda, Sergey Levine and Chelsea Finn, ICRA 2019

20-sided Die Repositioning Task

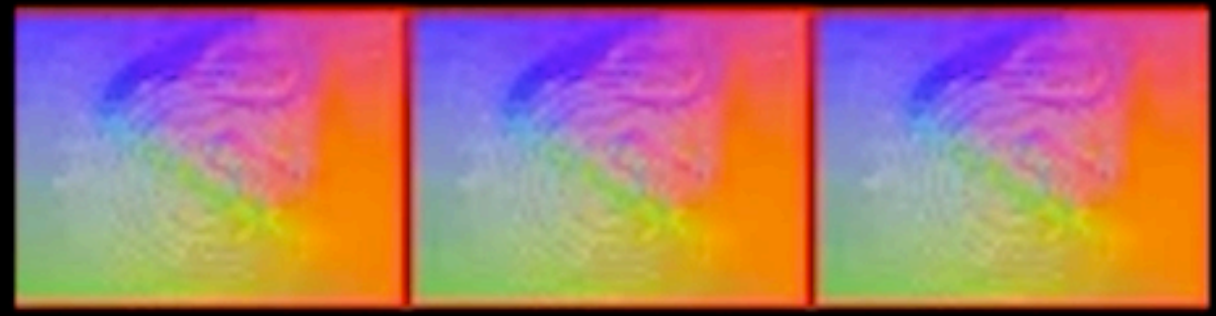


20-sided die: Example 1

Goal Image



Best 3 predicted trajectories



Quantitative results

Quantitative Final Step Results

	Median L2 dist [mm]		Success Rate
	Ball Rolling	Analog Stick	Die
Tactile MPC	**2.10**	**5.31**	**86.6% (26/30)**
Centroid Baseline	2.97	8.86	46.6% (14/30)

Time-Agnostic Prediction

[How to let your dynamics model cheat a little]

Compounding prediction error over time

Real video

Predicted

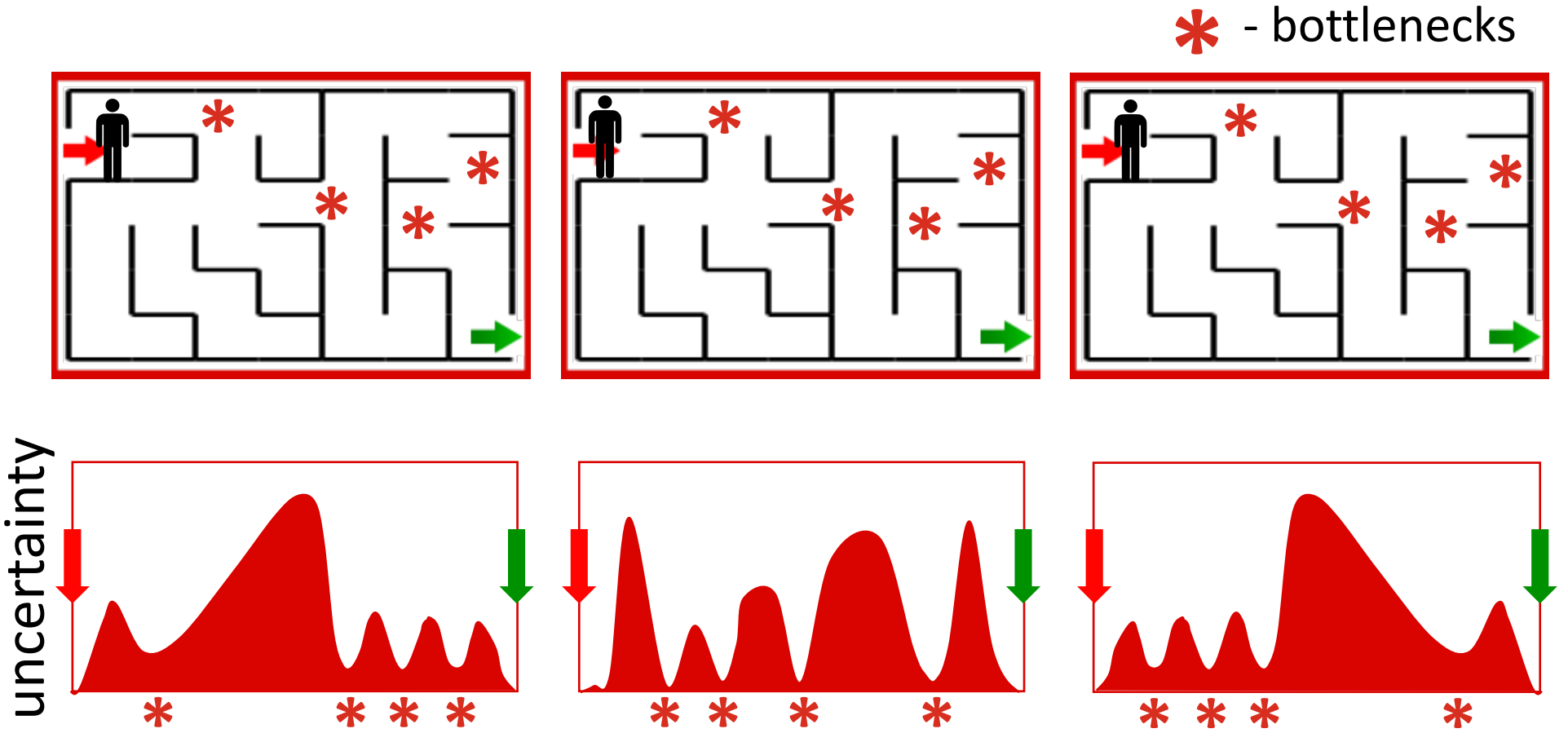
[Finn et al, NIPS 2016]

“It’s hard to make predictions, especially about the future.”

- Danish proverb

Jayaraman, Ebert, Efros, Levine, ICLR 2019

Low-uncertainty “bottlenecks”



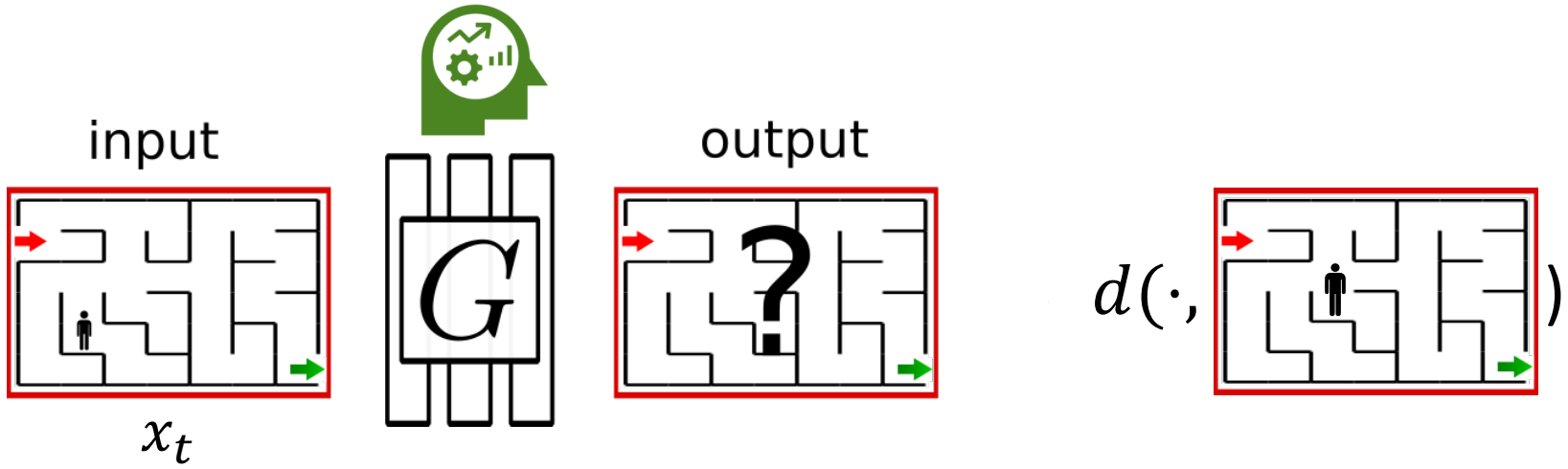
Bottleneck states: happen predictably but cannot say when.

Jayaraman, Ebert, Efros, Levine, ICLR 2019

Standard fixed-offset prediction

"Predict what will happen after a 5-second offset"

[Oh et al, NIPS 2015; Mathieu et al, ICLR 2015; Vondrick et al, NIPS 2016; Xue et al, NIPS 2016; Larsen et al, ICML 2016; Walker et al, ECCV 2016; Finn et al, ICRA 2017; Ebert et al, CORL 2017; Denton et al, ICML 2018; …]



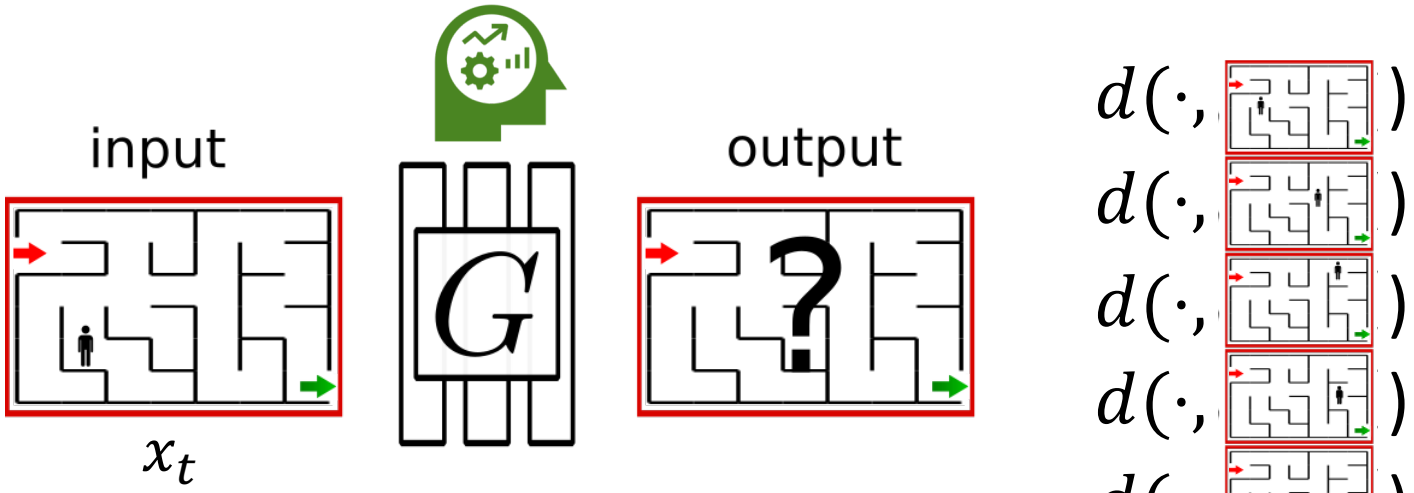
$$\theta_G^* = \min_{\theta_G} d(G(x_t), x_{t+\Delta})$$

Jayaraman, Ebert, Efros, Levine, ICLR 2019

Idea: time-agnostic prediction (TAP)

Incentive to latch on to bottlenecks.

"Predict something that will happen ***at some point***"

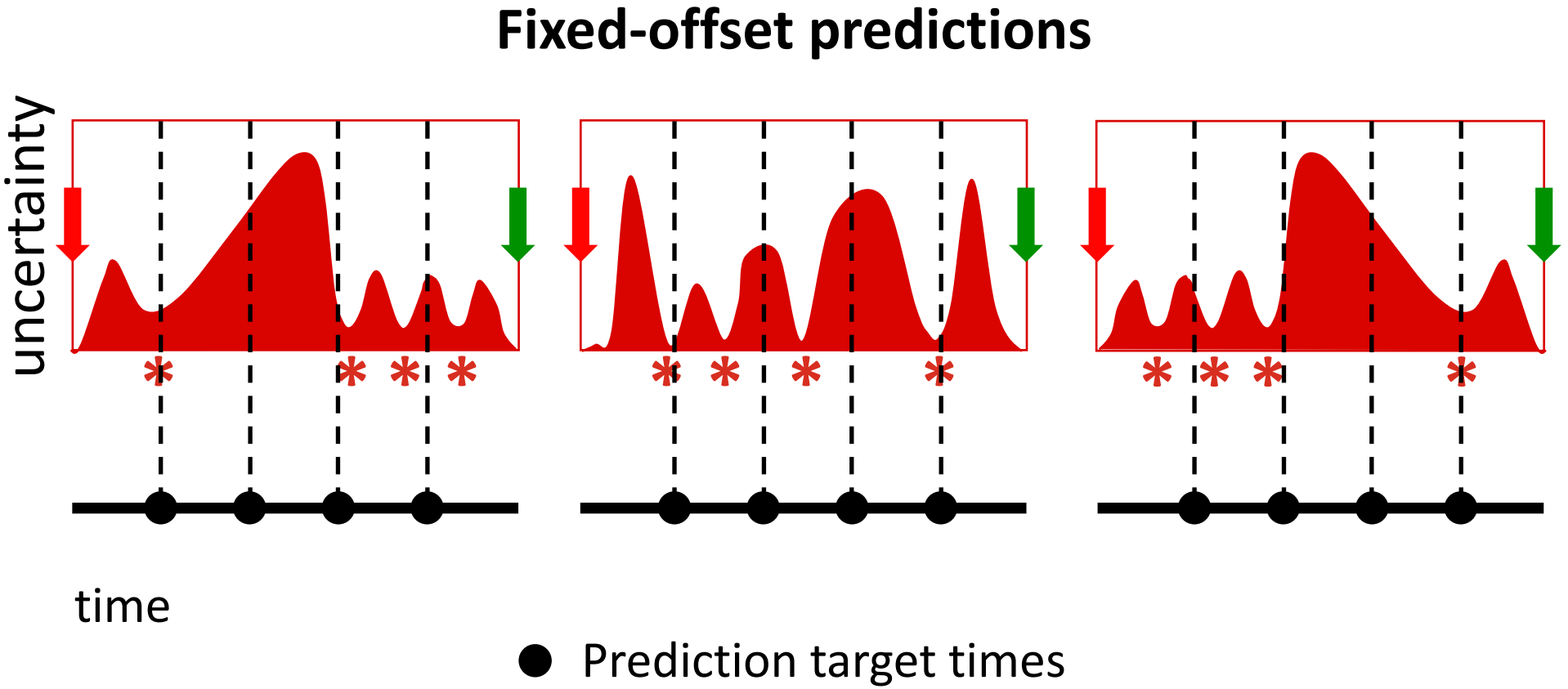


$$\theta_G^* = \min_{\theta_G} \min_{\Delta} d(G(x_t), x_{t+\Delta})$$

Agnostic to time offset Δ

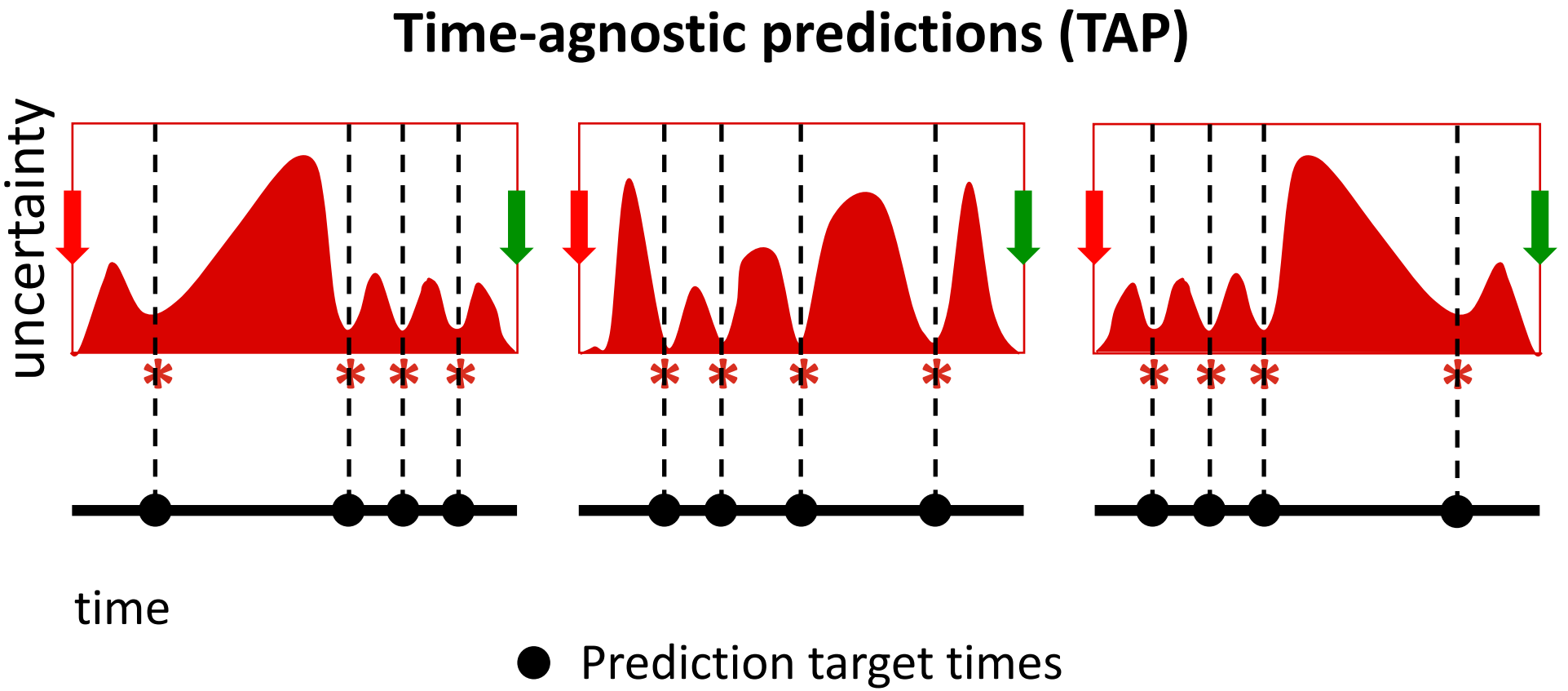
Jayaraman, Ebert, Efros, Levine, ICLR 2019

Seeking bottlenecks



Jayaraman, Ebert, Efros, Levine, ICLR 2019

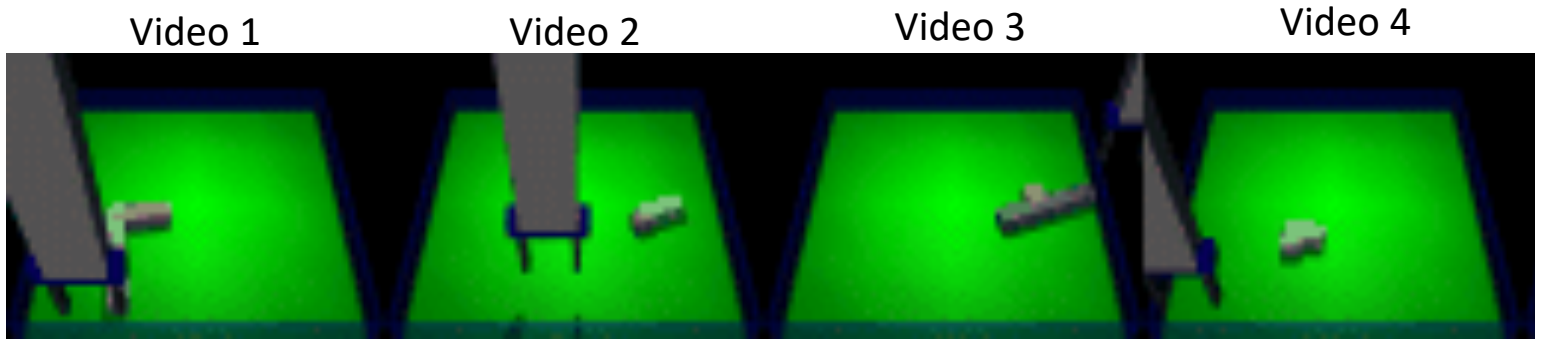
Seeking bottlenecks



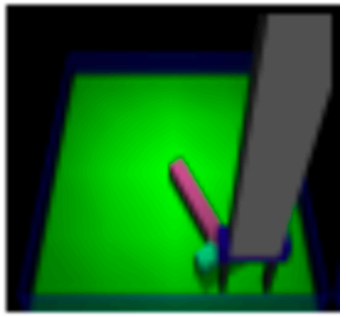
Time-agnostic predictions should naturally target bottlenecks

Jayaraman, Ebert, Efros, Levine, ICLR 2019

Pick-and-place task results

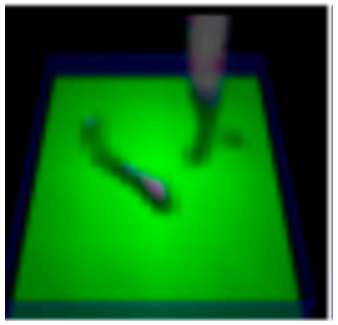


start (input)

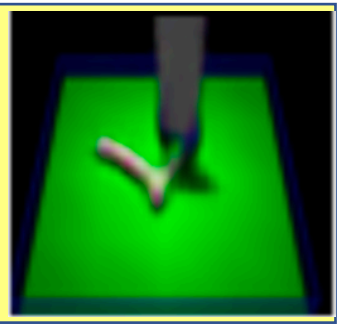


predictions

fixed-offset



TAP (ours)

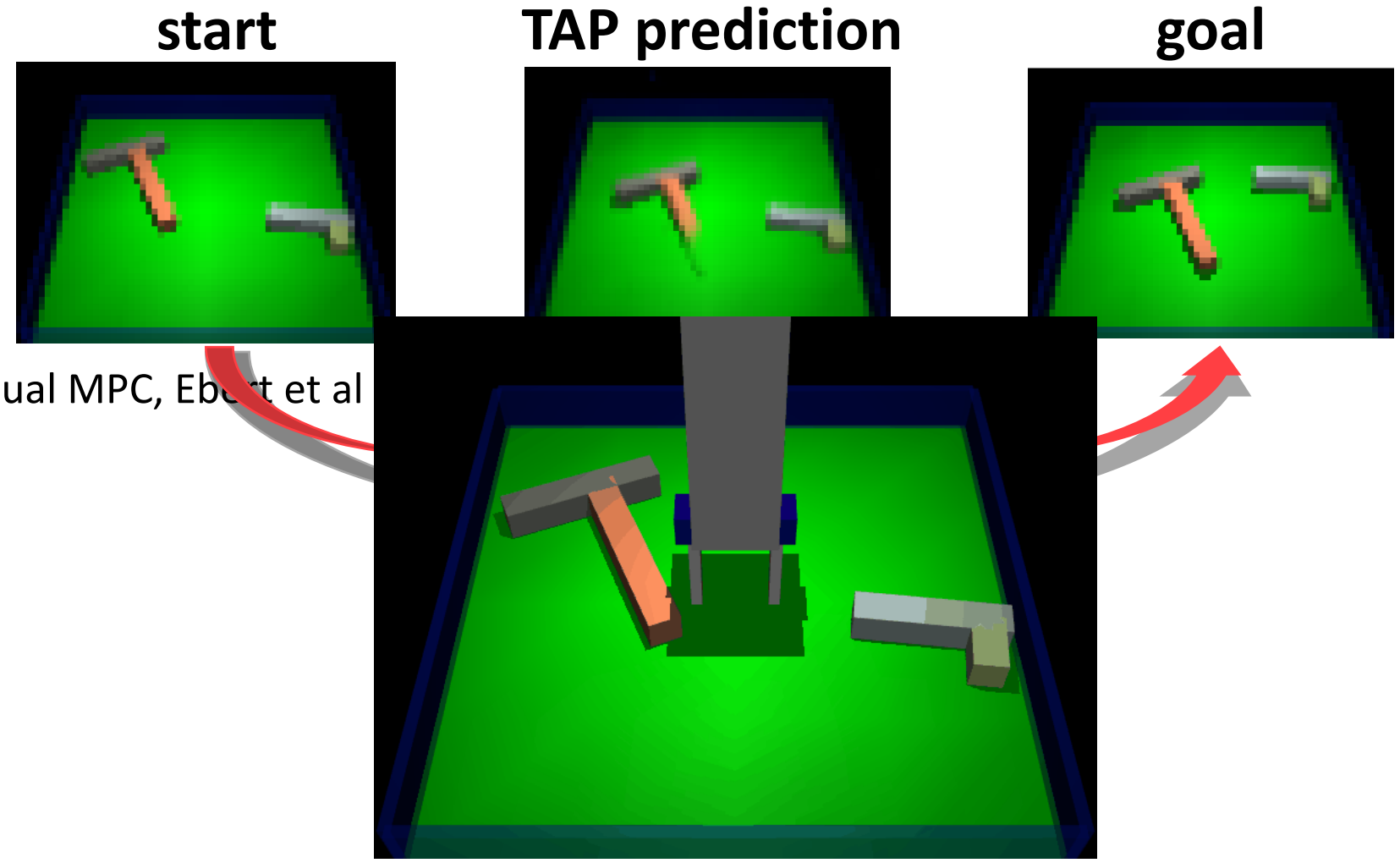


goal (input)

Jayaraman, Ebert, Efros, Levine, ICLR 2019

Evaluating subgoals

Planning with TAP: bottleneck predictions = subgoals?

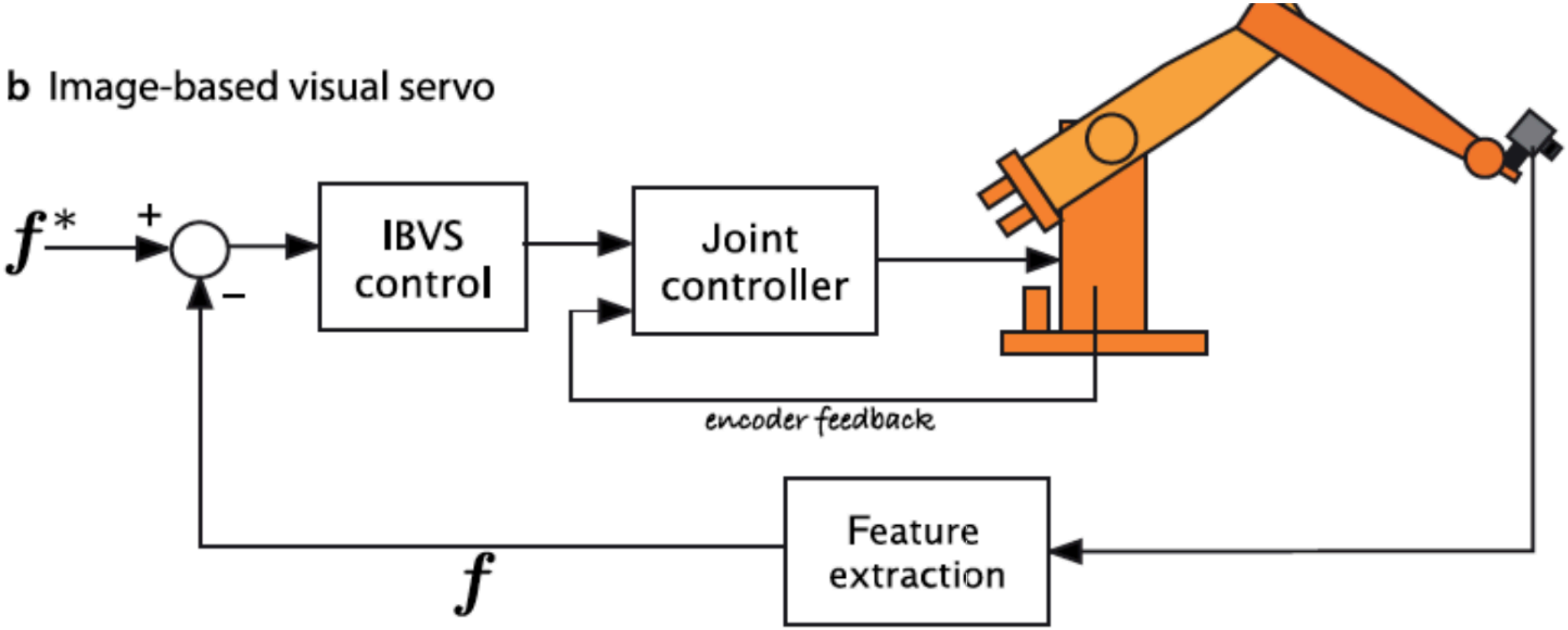


Jayaraman, Ebert, Efros, Levine, ICLR 2019

Visual Servoing

[Meanwhile in the control theory community]

Basic Idea of Image-Based Visual Servoing



The Image Jacobian

- How does features move when we move the robot?

$$\dot{\mathbf{f}} = \mathbf{J}\,\dot{\mathbf{q}}$$

- Image Jacobian

$$J(\mathbf{q}) = \left[\ \frac{\delta \mathbf{f}}{\delta \mathbf{q}}\ \right] = \begin{bmatrix} \frac{\delta f_1(\mathbf{q})}{\delta q_1} & \cdots & \frac{\delta f_1(\mathbf{q})}{\delta q_m} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_k(\mathbf{q})}{\delta q_1} & \cdots & \frac{\delta f_k(\mathbf{q})}{\delta q_m} \end{bmatrix}$$

Connection to model-based planning / RL

- The Jacobian is a linear predictive model.
- Works in a small neighborhood.
- Usually learned on the fly.
- Only permits greedy planning => can get stuck in local optima.
- Naturally handles any robot morphology, but needs to be shown what feature to extract => fiducial markers / trained detectors.

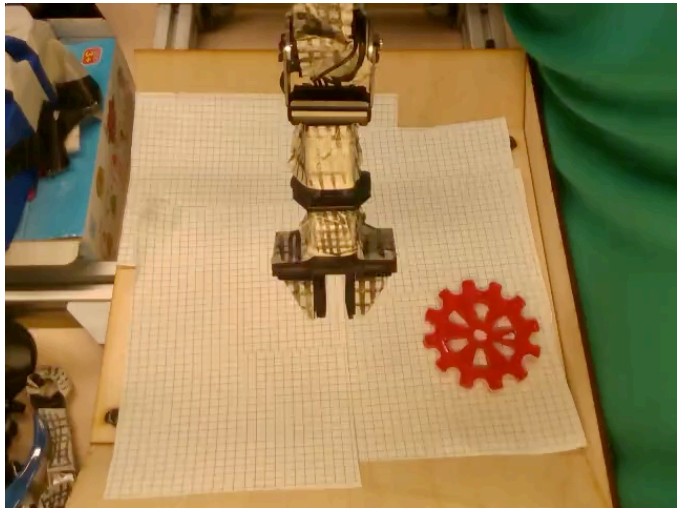
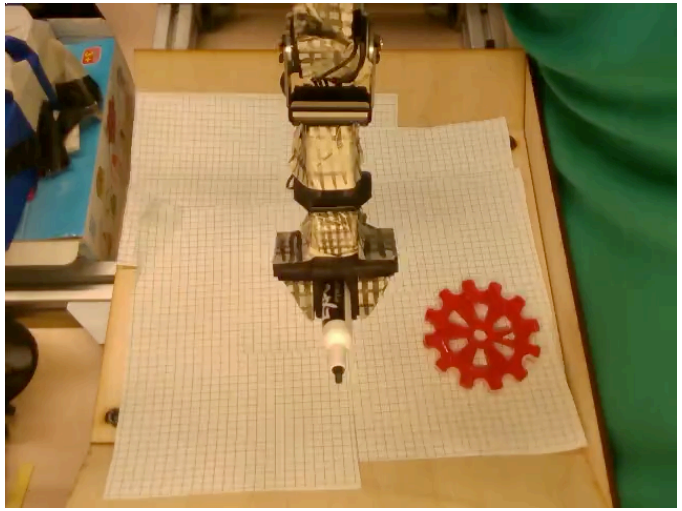
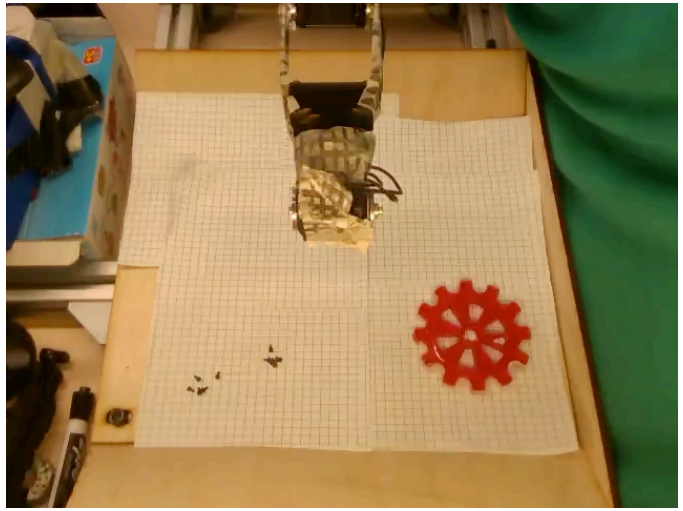
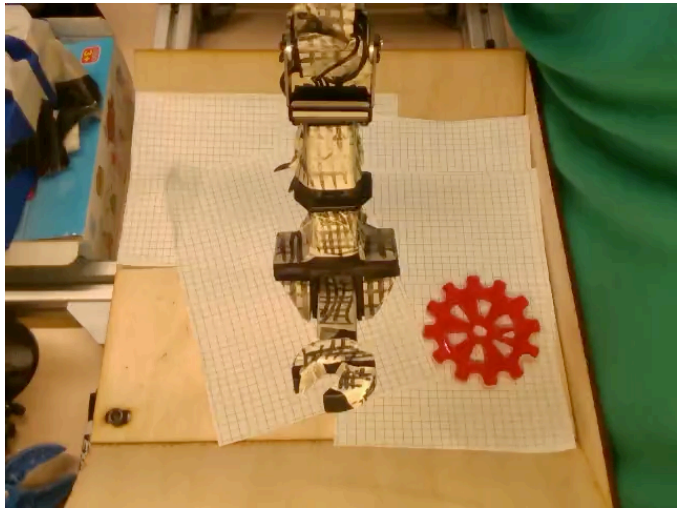
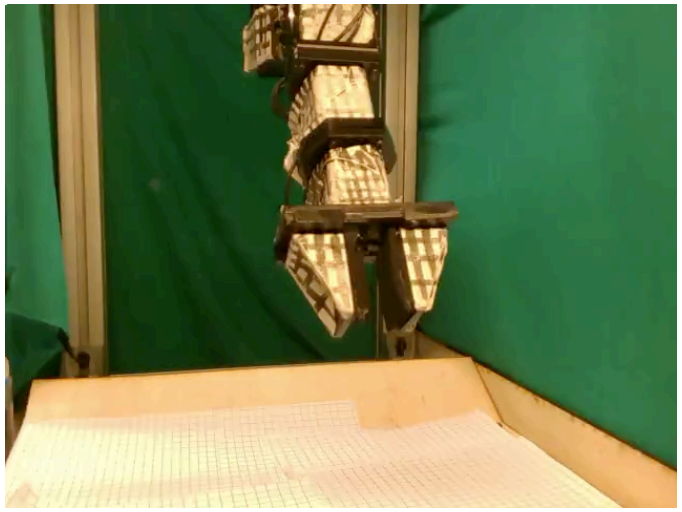
MARVC: visual “self-recognition” + servoing

- Mirror self-recognition in animals

- Could robots do this? Recognize which part of the world they see is their body?

[Soon on arXiv]

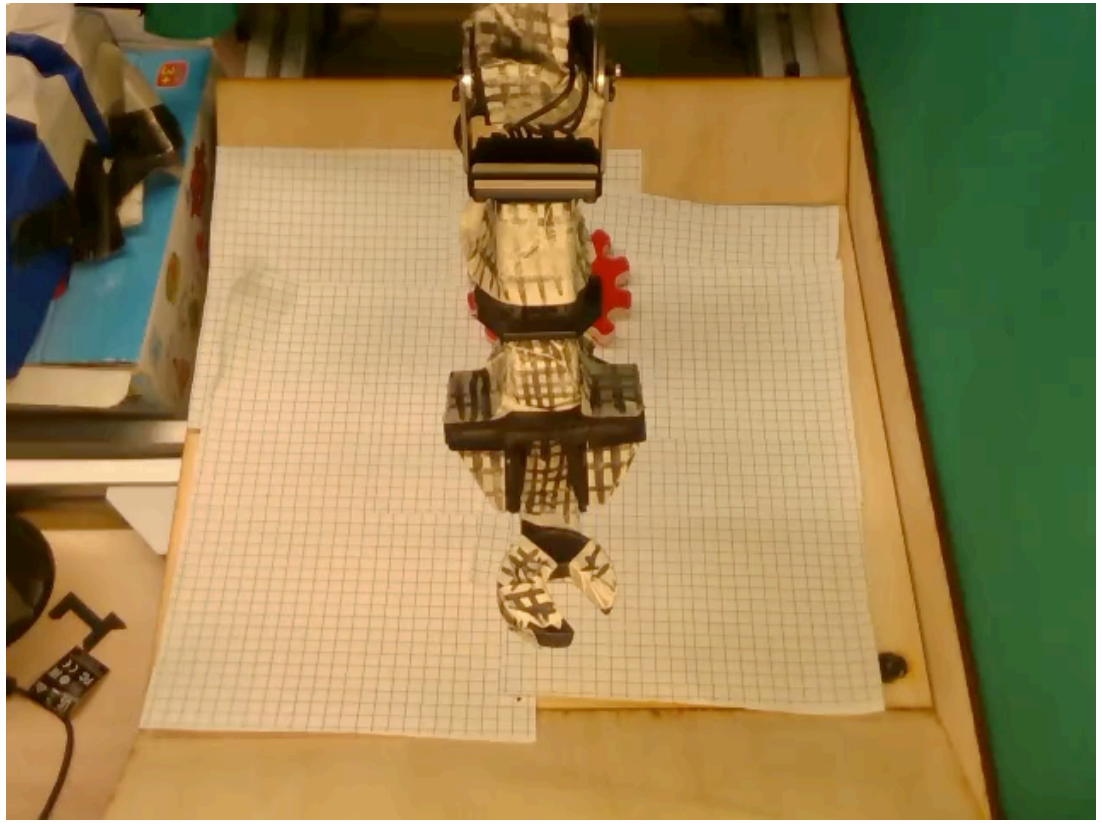
MAVRC self-recognition examples

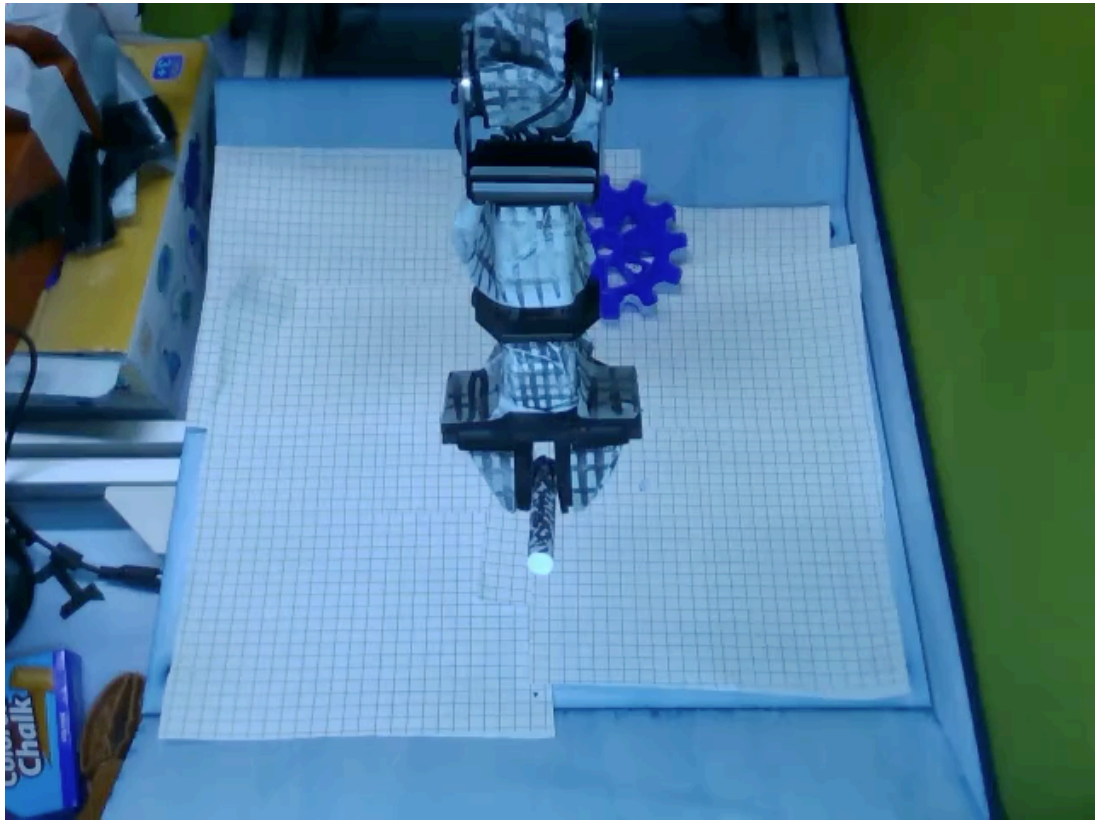






[Soon on arXiv]

MAVRC: visual self-recognition + servoing

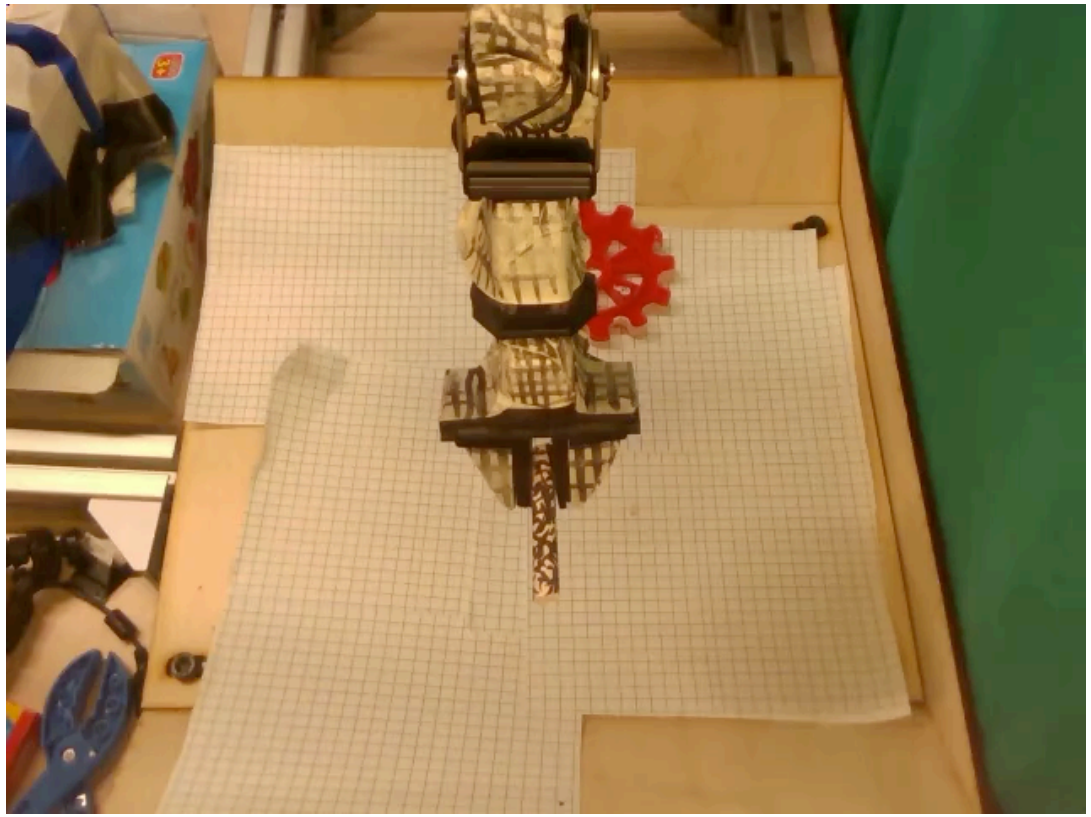
Morphology-Agnostic Visual Robotic Control [MAVRC]

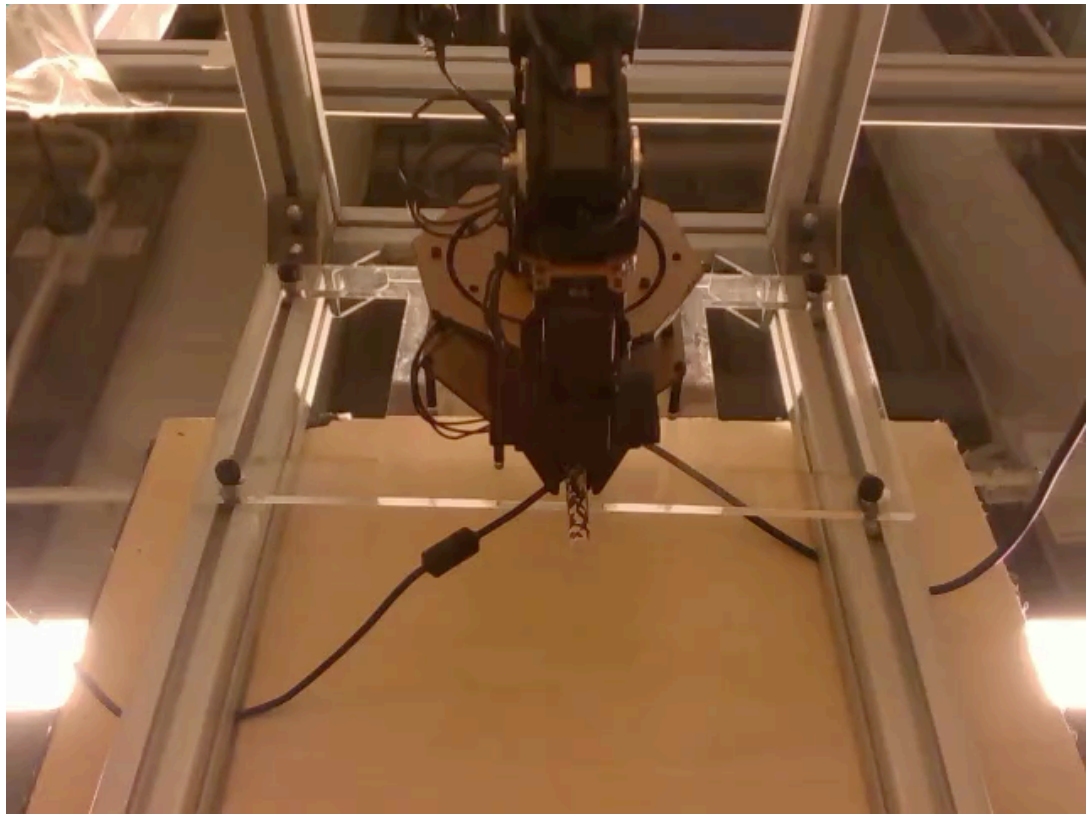




[Soon on arXiv]

MAVRC robot-to-robot imitation





[Soon on arXiv]

How does it work?

- MAVRC detects the body as the set of particles in the world that are “responsive” to control inputs.

$$R_i \triangleq I(\Delta S_i; A)$$

$$B_\delta \triangleq \{P_i : R_i > \delta\}$$

- Once this is done, standard uncalibrated image-based visual servoing takes over, trying to move the “end-effector” of the detected body to the goal.

[Soon on arXiv]

Summary of model-based RL approaches

Pros:

- Mature field of study, drawing heavily on control theory, planning etc.
- Has been demonstrated on more real-world cases, esp in robotics, than model-free approaches.
- Sample efficiency
- The same dynamics model can solve many tasks.

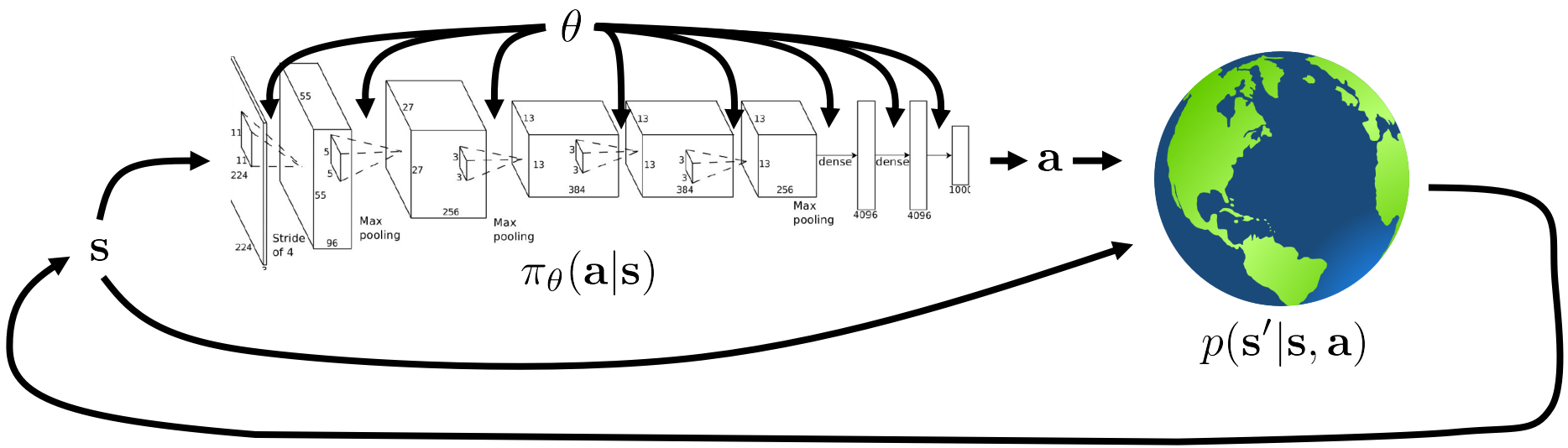
Cons:

- Modeling full high-dimensional states is sometimes too hard.
- **Core problem: the model does not receive gradients from the task.**

Deep Q-Learning

[a modern classic]

Recap: the goal of reinforcement learning



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

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Definition: Q-function

$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t\right]$: total reward from taking $\mathbf{a}_t$ in $\mathbf{s}_t$

Definition: value function

$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t\right]$: total reward from $\mathbf{s}_t$

$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$

$E_{\mathbf{s}_1 \sim p(\mathbf{s}_1)}[V^\pi(\mathbf{s}_1)]$ is the RL objective!

Bellman Equations

- Recursive dynamic programming-style equations that state properties of V(s) and Q(s,a)

$$Q^{\pi}(s,a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right]$$
$$= \mathbb{E}_{s'}\left[r + \gamma Q^{\pi}(s', a') \mid s, a\right]$$

- Optimal Q* satisfies:

$$Q^{*}(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^{*}(s', a') \mid s, a\right]$$

Q-learning: the 1-slide basic version

- By definition: $Q^*(s,a) = E[r(s,a) + \max'_a Q^*(s',a')]$
- Q-learning = “fake it till you make it.” (like EM).
  - Start with random Q(s,a).
  - From current state s, perform “Q-optimal” action a = argmax Q(s,.), observe reward r.
  - Then update Q(s, a) by its Bellman error, to:

$$Q(s,a) + \eta[Q(s,a) - r(s,a) - \max_{a'} Q(s,a')]$$

  - Repeat ``Q-optimal” action selection and Q update, till convergence.

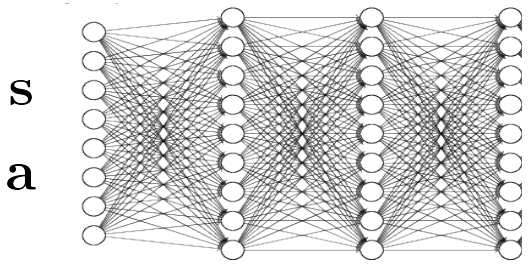
Yes, incredibly, this converges. Sadly, no, not in the neural net case.

Online Q-learning with function approximator

"off policy", so many choices here!

online Q iteration algorithm:

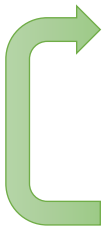
1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\mathbf{y}_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i)$



$Q_\phi(\mathbf{s}, \mathbf{a})$
parameters ϕ

Problem: test policy is greedy

online Q iteration algorithm:



1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\mathbf{y}_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i)$

final policy:

$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$$

why is this a bad idea for step 1?

Solution: epsilon-greedy

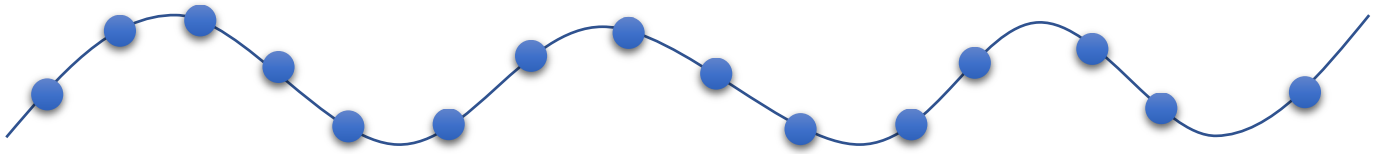
$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 - \epsilon \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ \epsilon/(|\mathcal{A}| - 1) \text{ otherwise} \end{cases}$$

Problem: correlated samples in online Q-learning

online Q iteration algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$

- sequential states are strongly correlated
- target value is always changing



One solution: replay buffers

online Q iteration algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$

special case with K = 1, and one gradient step

full fitted Q-iteration algorithm:

~~1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy~~

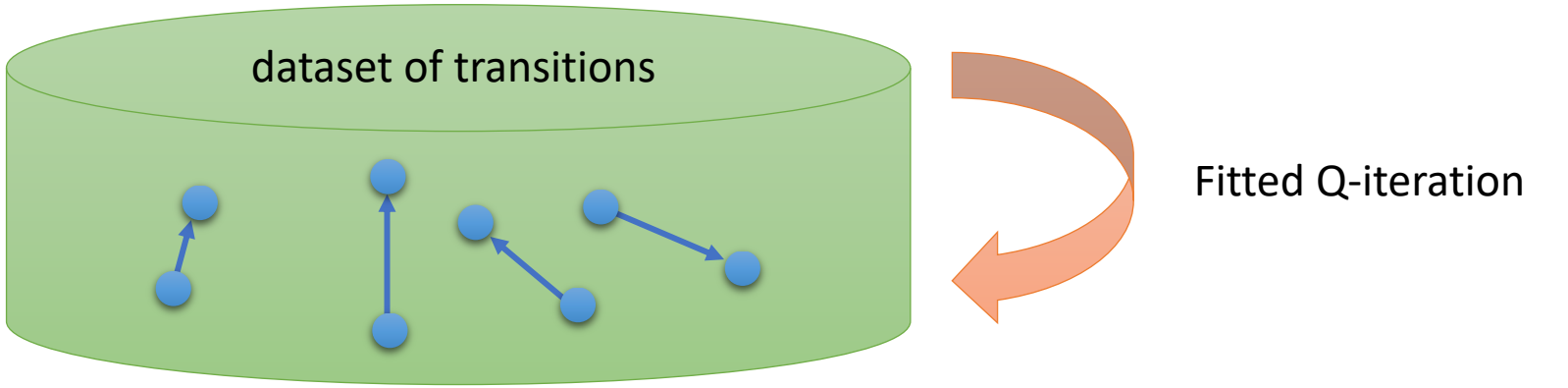
$K\times$

2. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'_i} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

any policy will work! (with broad support)

just load data from a buffer here

still use one gradient step



Putting it together

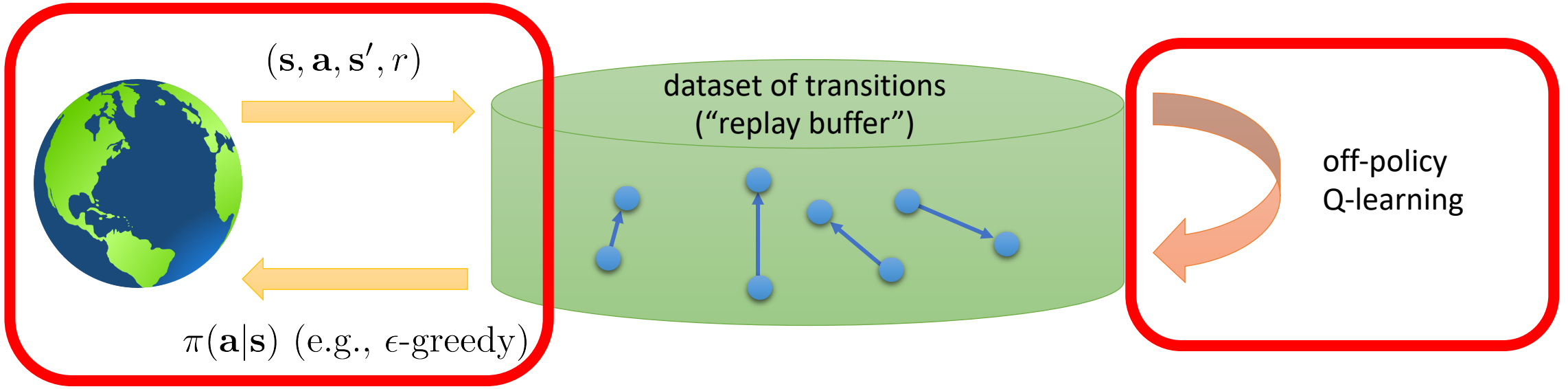
full Q-learning with replay buffer:

1. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{B}$

$K\times$

2. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
3. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$

K = 1 is common, though larger K more efficient



Problem: moving target for Q regression

online Q iteration algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\mathbf{y}_i = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
3. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i)$

Q-learning is *not* gradient descent!

$$\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$$

no gradient through target value

This is still a problem!

Solution: target networks

Q-learning with replay buffer and target network:

1. save target network parameters: $\phi' \leftarrow \phi$

$N\times$

2. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{B}$

$K\times$

3. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$

4. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}'_i, \mathbf{a}'_i)])$

targets don't change in inner loop!

supervised regression

“Classic” deep Q-learning algorithm (DQN)

Q-learning with replay buffer and target network:

1. save target network parameters: $\phi' \leftarrow \phi$
2. collect dataset $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$ using some policy, add it to $\mathcal{B}$
3. sample a batch $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$ from $\mathcal{B}$
4. $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}'_i, \mathbf{a}'_i)])$

steps 3–4 repeated K times; steps 2–4 repeated N times

“classic” deep Q-learning algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{B}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{B}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$

steps 2–4 repeated $K = 1$ times

5. update ϕ': copy ϕ every N steps

You’ll implement this in HW3!

Mnih et al. ‘13

Example applications of DQN variants

Action for Perception

[What if I am interested in improving computer vision performance]

What does action do for perception?

"… perception is facilitated by interaction with the environment:

- Interaction with the environment creates a rich sensory signal that would otherwise not be present.
- Knowledge of the regularity in the combined space of sensory data and action parameters facilitates the prediction and interpretation of the sensory signal."

Jeanette Bohg et al, *Interactive Perception*, 2016

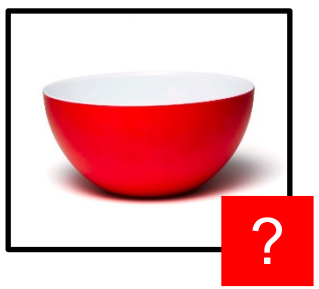
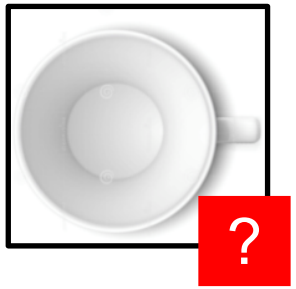
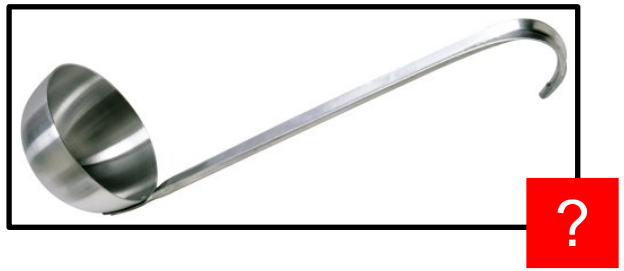
Status quo: passive snapshot recognition

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

IMAGENET

COCO
Common Objects in Context

Object recognition

Scene understanding

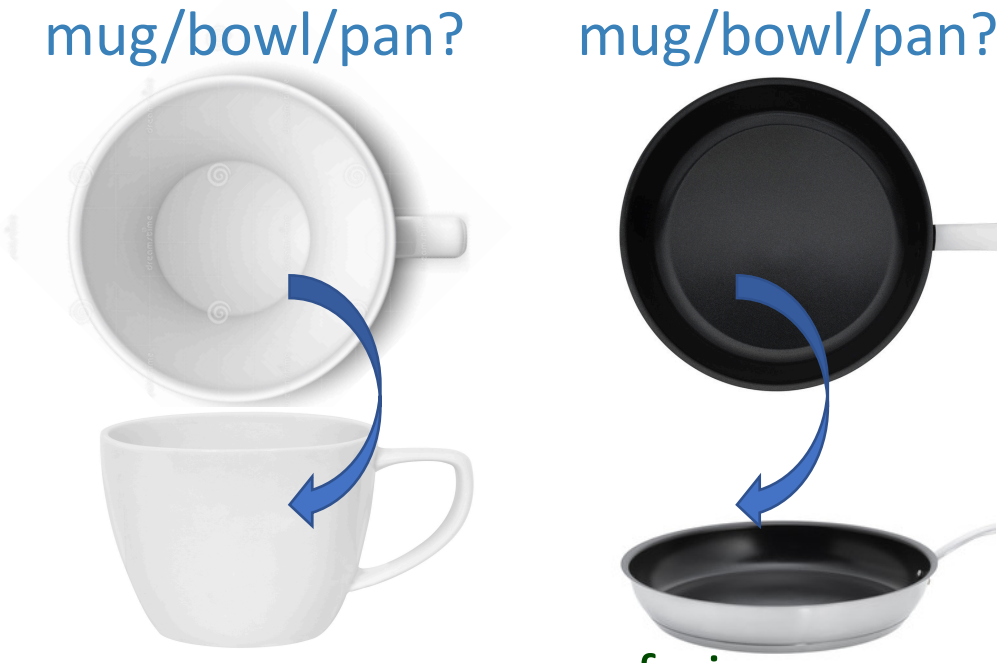

Jayaraman and Grauman, ECCV 2016, TPAMI 2018, CVPR 2018

Status quo: passive snapshot recognition

Jayaraman and Grauman, ECCV 2016, TPAMI 2018, CVPR 2018

Video credit Yu-Chuan Su

Example 1: Active Recognition

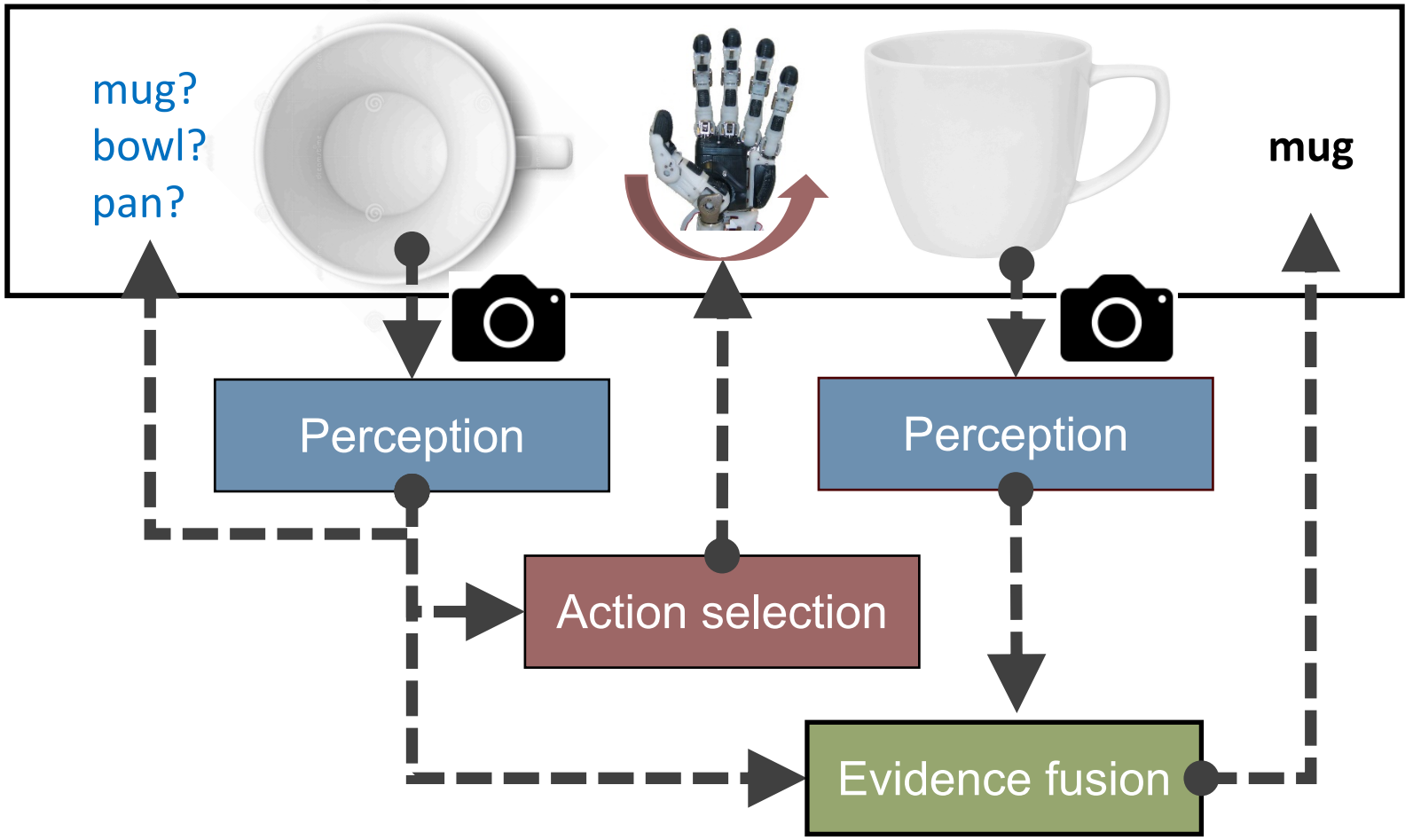


Active recognition: select which views to see.

Bajcsy 1988, Aloimonos 1988, Ballard 1991, Wilkes 1992, Dickinson 1997, Schiele 1998, Denzler 2002, Soatto 2009, Ramanathan 2011, Malmir 2015, Ammirato 2017

Jayaraman and Grauman, ECCV 2016, TPAMI 2018, CVPR 2018

Active recognition system



Bajcsy 1988, Aloimonos 1988, Ballard 1991, Wilkes 1992, Dickinson 1997, Schiele 1998, Denzler 2002, Soatto 2009, Ramanathan 2011, Malmir 2015, Ammirato 2017

Jayaraman and Grauman, ECCV 2016, TPAMI 2018, CVPR 2018

Results

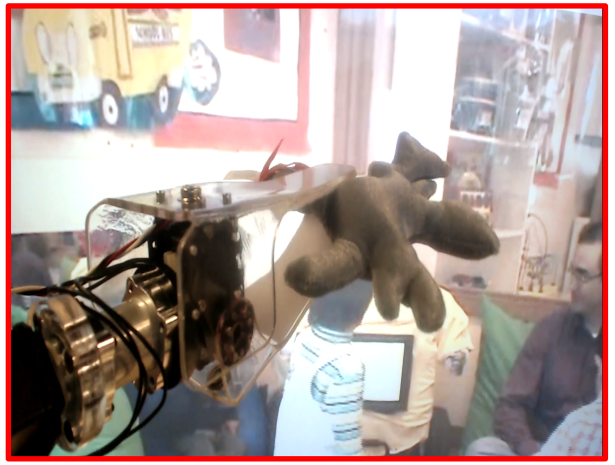
Jayaraman and Grauman, ECCV 2016, TPAMI 2018

Results

Predicted label:

T=1

T=2



T=3

Jayaraman and Grauman, ECCV 2016, TPAMI 2018

Example 2: Better features from action

After moving:

Jayaraman and Grauman, ICCV15 & IJCV Special Issue of Best Papers from ICCV15

egomotion ↔ vision for recognition

- Egomotion-conditioned view prediction requires:

  - Depth, 3D geometry
  - Semantics
  - Context …

} Also key to recognition!

- No manual labels!

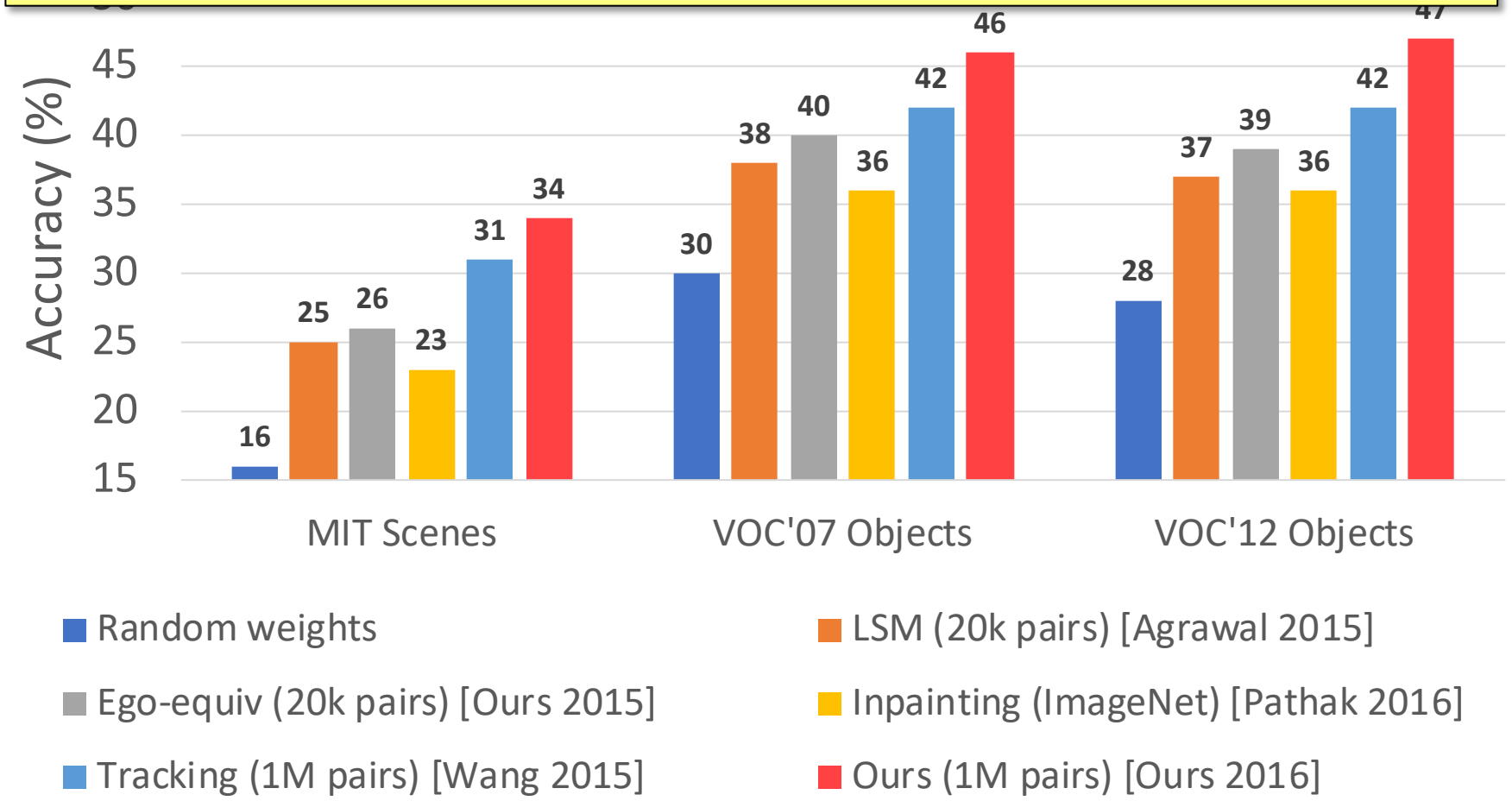
Our approach:

Train features in which egomotion-conditioned future prediction is easy!

Jayaraman and Grauman, ICCV15 & IJCV Special Issue of Best Papers from ICCV15

State of the art

Knowledge of actions contains important information for perception.



Gao, Jayaraman, Grauman, ACCV16

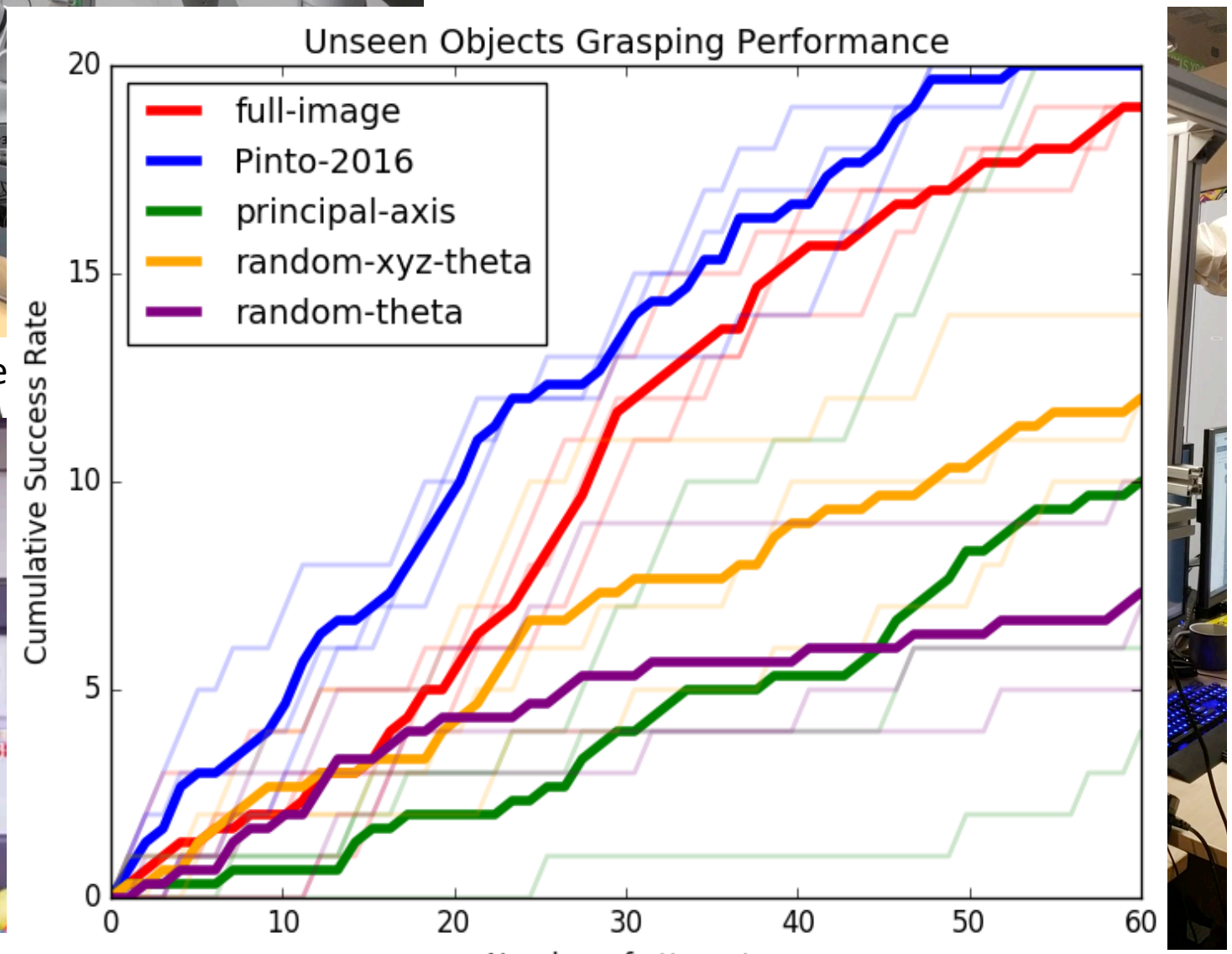
nchmark

Pinto-2016

Full-Image

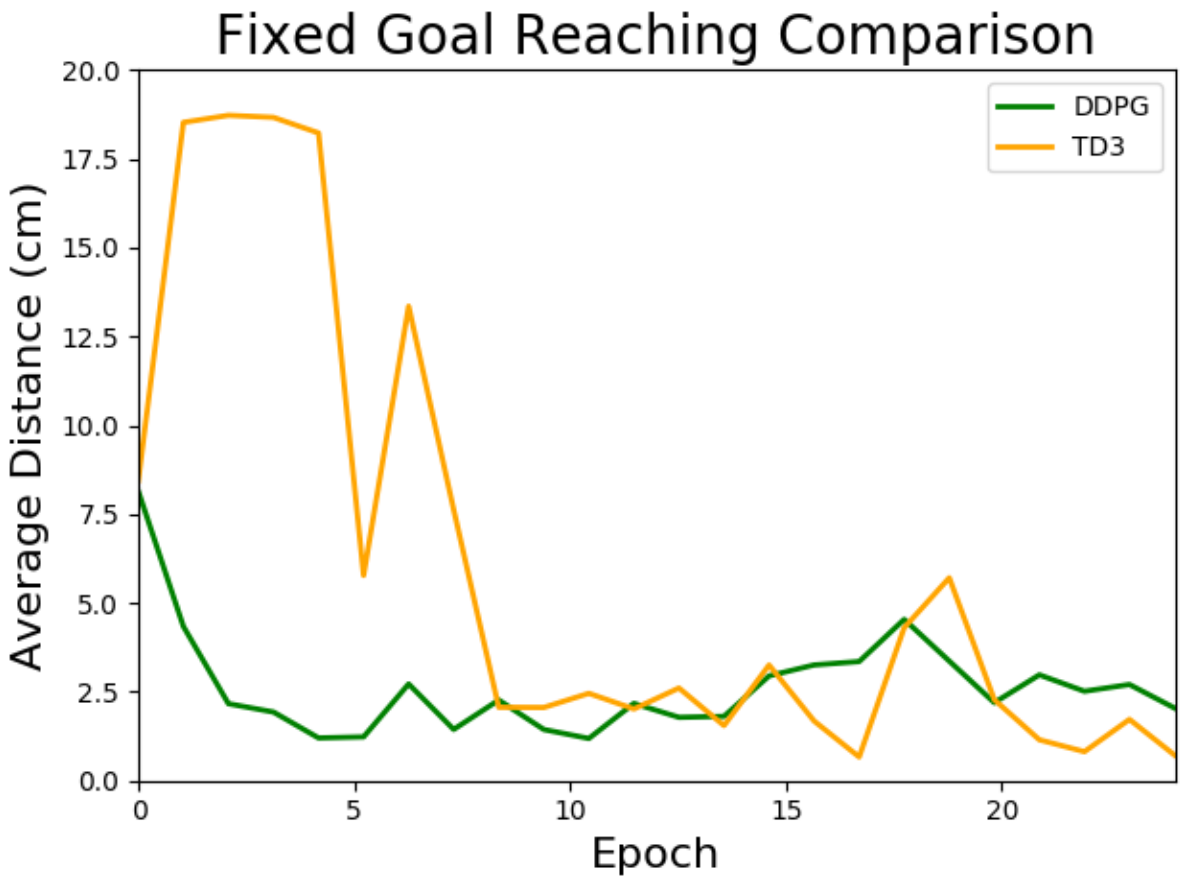
Levine e

Mahler e



Robot RL on REPLAB

Task: Joint space velocity control for 3D reaching (6-dim action space)
Reward: negative euclidean distance from end-effector to target
Baselines: TD3, DDPG (more soon!)



Why REPLAB

- Out-of-the-box baselines
- Benchmarking progress
- Lower entry barrier into robotics
- Code sharing!

Build a REPLAB cell yourself! (2k USD, < 5 hours)

Future work: challenge tasks/benchmarks, datasets, and more control approaches!

Project page: https://goo.gl/5F9dP4
Github: bhyang/replab
Docker: bhyang12/replab

For more information on topics covered here:

- See Sergey Levine's UC Berkeley course on Deep Reinforcement Learning, from which several of the slides in this talk have been borrowed/adapted with permission.