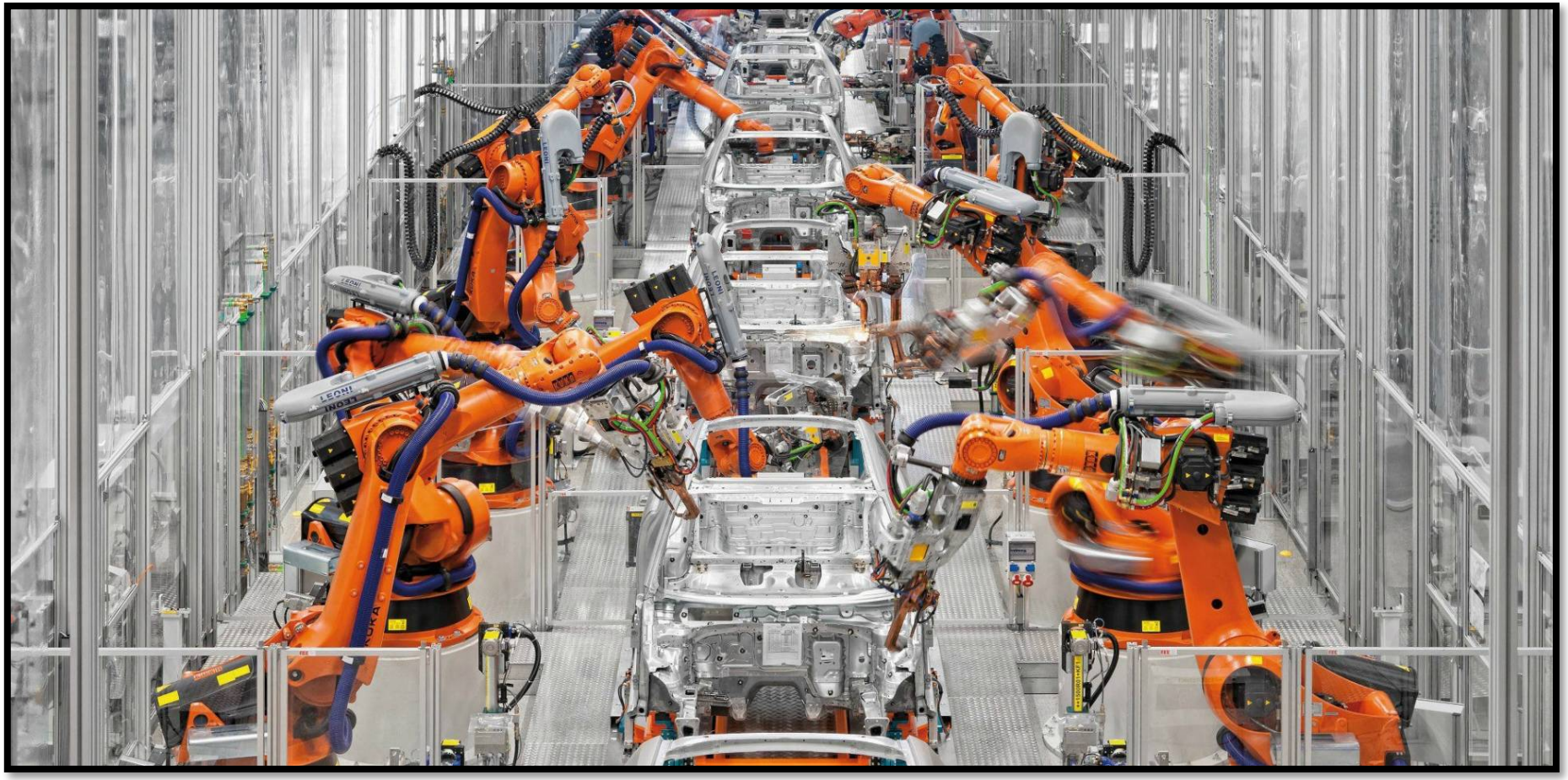
Generalization and Safety in Reinforcement Learning and Control

Aviv Tamar

Berkeley
Artificial Intelligence Research Laboratory

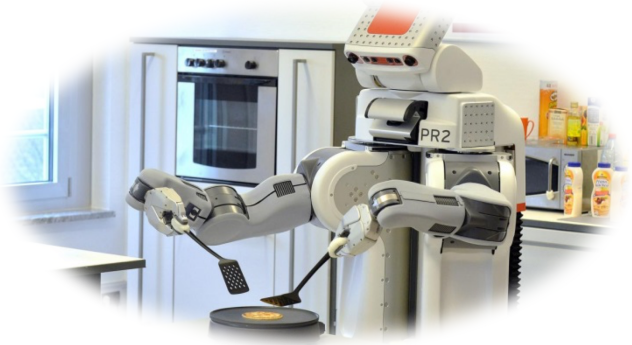
Motivation

- Today: robots in factories



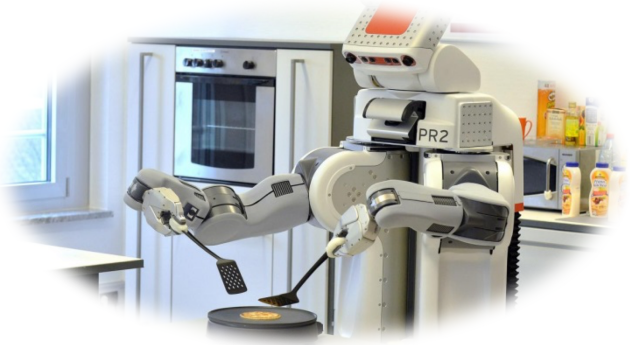
Motivation

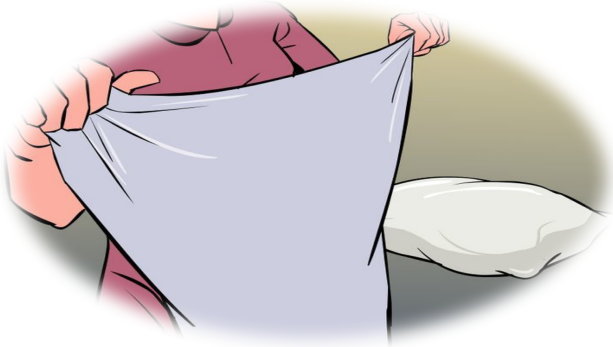
- Future: robots ‘in the wild’
  - Inside homes
  - Alongside humans
  - Rapid manufacturing



Motivation

- Future: robots ‘in the wild’
  - Inside homes
  - Alongside humans
  - Rapid manufacturing

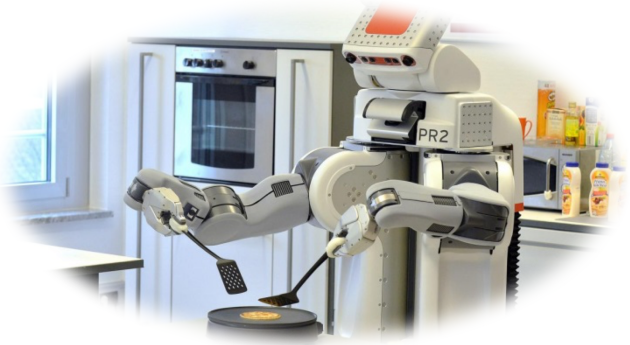


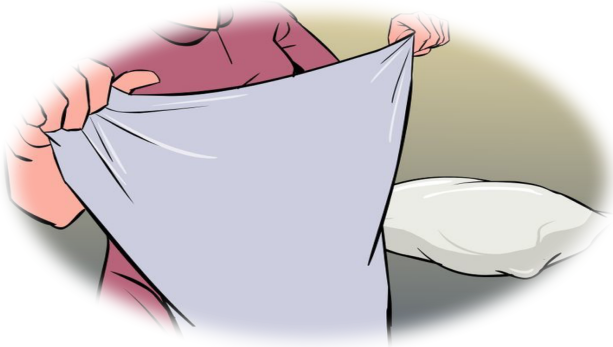


- Challenges:
  - Unstructured environments
  - Nonlinear, non-smooth dynamics
  - True model is never known
  - Safety is essential

Motivation

- Future: robots ‘in the wild’
  - Inside homes
  - Alongside humans
  - Rapid manufacturing

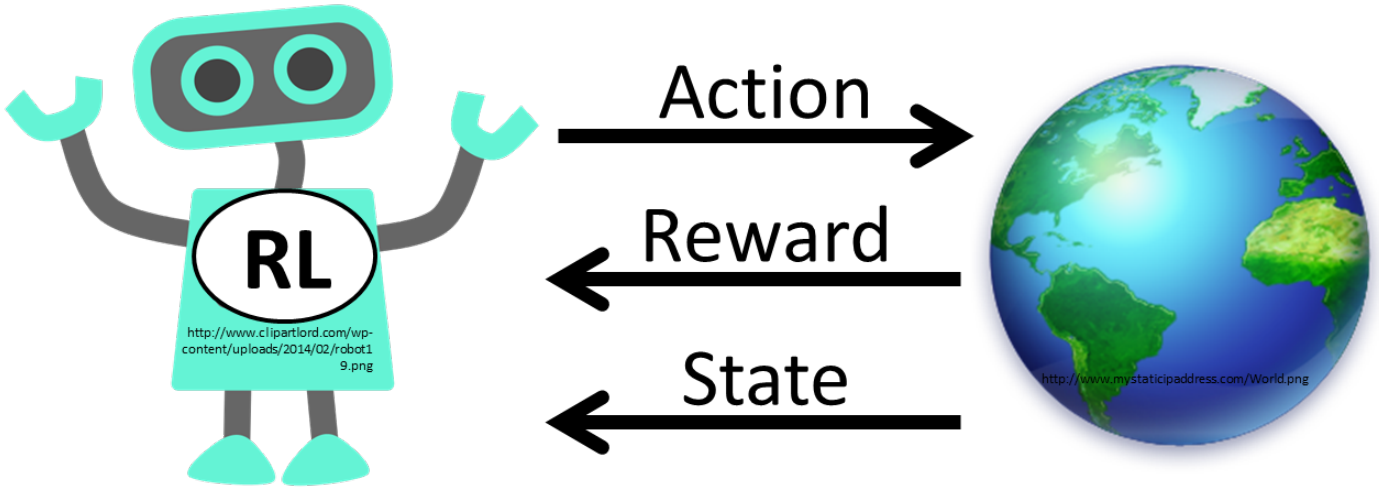




- Challenges:
  - Unstructured environments
  - Nonlinear, non-smooth dynamics
  - True model is never known
  - Safety is essential

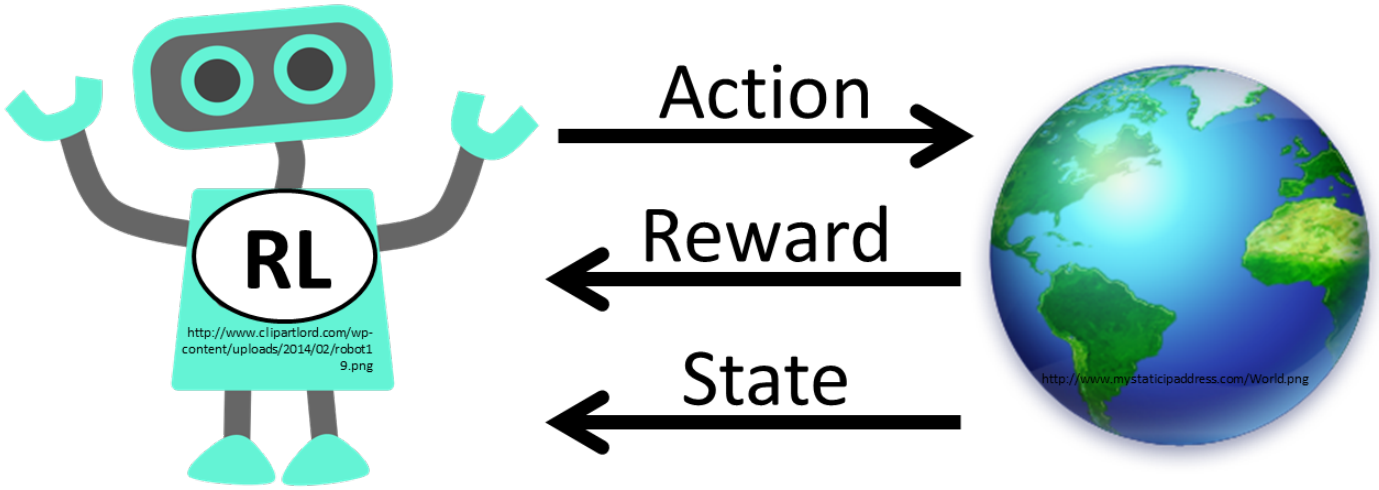
Reinforcement Learning

- Reinforcement learning (RL) model:



Reinforcement Learning

- Reinforcement learning (RL) model:



- Machine learning to learn (state) → (action) that leads to high reward

Reinforcement Learning

- Naturally addresses:
  - ✓Unstructured environments
  - ✓Nonlinear, non-smooth dynamics
  - ✓True model is never known
- Practical challenges:
  - Safety
  - Generalization (to changes in the task)
  - High-dimensional states/actions

This Talk

1. Safety – risk sensitive RL
2. Generalization – value iteration networks
3. Planning-based neural networks for robotics

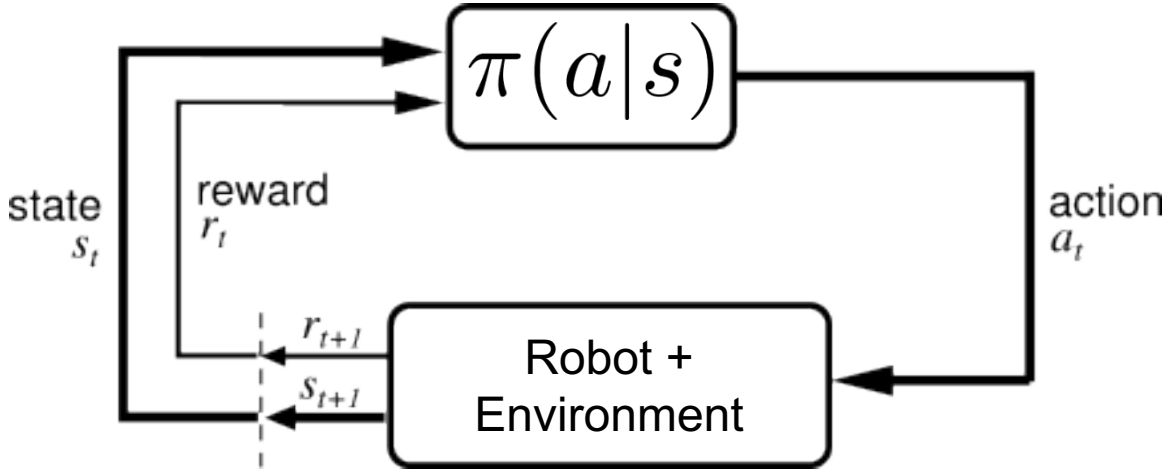
Part 1: Risk-Aware RL

Background

Reinforcement Learning

MDP Formulation

- States s, actions a
- Transitions $P(s'|s,a)$
- Policy $\pi(a|s)$
- Reward $r(s)$
- Discount $\gamma \in [0,1)$



- Goal: $\max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) | \pi\right]$

Background

Planning (when model is known)

- Value function:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \middle| \pi, s_0 = s\right]$$

- Value iteration algorithm:

$$V_{n+1}(s) = \max_a Q_n(s,a) \quad \forall s,$$

$$Q_n(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) V_n(s').$$

Background

Planning (when model is known)

- Value function:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \middle| \pi, s_0 = s\right]$$

- Value iteration algorithm:

$$V_{n+1}(s) = \max_a Q_n(s,a) \quad \forall s,$$

$$Q_n(s,a) = R(s) + \gamma \sum_{s'} P(s'|s,a) V_n(s').$$

- Converges to $V^* = \max_\pi V^\pi$ (γ-geometric rate)
- Optimal policy $\pi^*(a|s) = \arg\max_a Q^*(s,a)$

Background

RL (when model unknown/too large to plan)

1. Approx. value function (e.g., neural-net):

$$V^*(s) \approx V(s;\theta)$$

   - Sampling to estimate θ from interaction

2. Policy gradient:

$$\pi(a|s;\theta)$$

$$\nabla_\theta \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \middle| \pi\right]$$

   - Sampling to estimate gradient from interaction

Risk Averse RL

- Why should we care about risk?

1. The world is stochastic
   - Risk-aware goal: $\max_\pi \rho\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) | \pi\right]$
   - ρ is risk measure, e.g., $\rho(X) = \mathbb{E}[X] - \beta \mathrm{Var}[X]$

Safety against **unluckiness**

Risk Averse RL

- Why should we care about risk?

2. True world model is unknown

   - Robustness objective:

$$\max_{\pi} \min_{\hat{P} \in \text{ possible MDPs}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \,|\, \pi, \hat{P}\right]$$

Safety against **model mismatch**

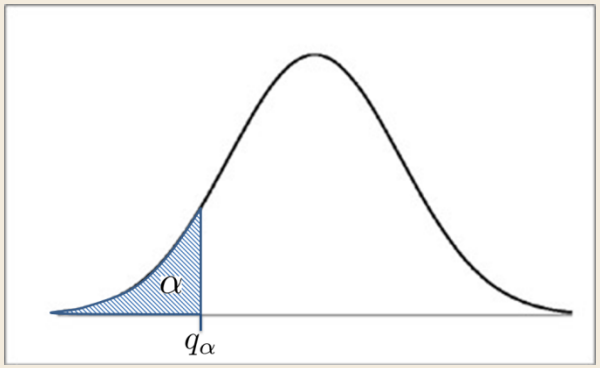
Risk Averse RL

- Why should we care about risk?
  1. Safety against **unluckiness**
  2. Safety against **model mismatch**

Conditional Value at Risk (CVaR)

- Safety against ‘unluckiness’
- Definition:

  - X - random variable
  - $q_\alpha(X)$ - α quantile
  - $\text{CVaR}_\alpha(X) = \mathbb{E}[X | X \leq q_\alpha]$
  - Expected $\alpha\%$ worst cases

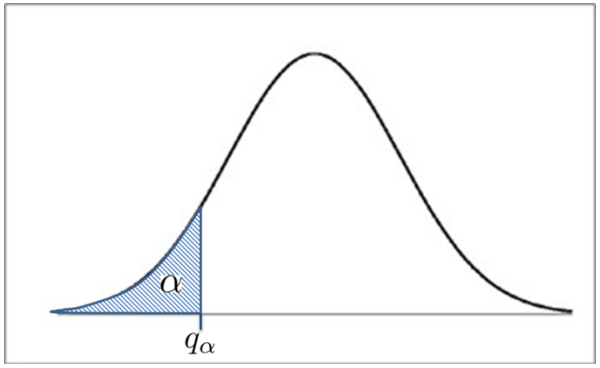


- Risk averse goal:

$$\max_\pi \mathbf{CVaR}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \middle| \pi\right]$$

Conditional Value at Risk (CVaR)

- Popular in finance
- CVaR Risk = Robustness [1]



  - Multiplicative perturbations

$$\hat{P}(x_{t+1}|x_t) = P(x_{t+1}|x_t) \cdot \delta_t(x_{t+1}|x_t)$$

  - Budget

$$\delta_1(x_1|x_0)\delta_2(x_2|x_1)\cdots\delta_H(x_H|x_{H-1}) \leq \eta, \quad \forall x_1, \ldots, x_H.$$

    The worst cannot happen at every time!

  - Set of possible perturbations Δ_η

Theorem

$$\text{CVaR}_{\frac{1}{\eta}}\left(\sum_{t=0}^{H} r(x_t)\right) = \inf_{\hat{P}\in\Delta_\eta} \mathbb{E}\left[\sum_{t=0}^{H} r(x_t) \middle| \hat{P}\right]$$

[1] Chow, T., Mannor, Pavone. NIPS 2015

Conditional Value at Risk (CVaR)

Safety against unluckiness = safety to model errors

- CVaR Can be optimized efficiently:

1. Planning [1]
   - Value iteration + linear prog. + linear interpolation
   - First error bounds + convergence rate
   - γ-geometric rate, poly(S, A) each iteration
2. Large/continuous MDPs, RL [2,3]
   - Policy gradient
   - Convergence (w.p.1) to locally optimal policy
   - Compatible with deep RL

[1] Chow, T., Mannor, Pavone. NIPS 2015
[2] T., Glassner, Mannor. AAAI 2015
[3] T., Chow, Ghavamzadeh, Mannor. NIPS 2015

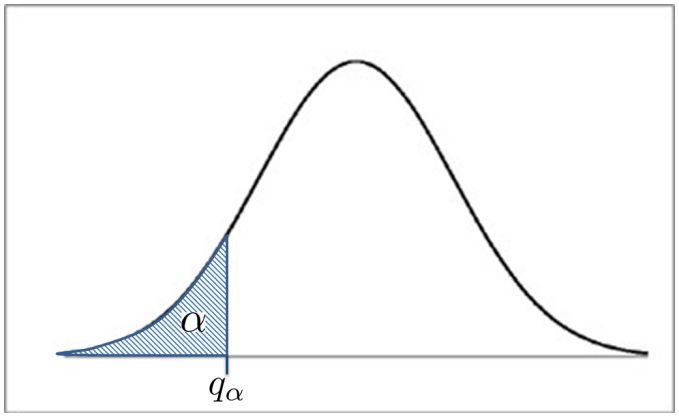
CVaR Policy Gradient

CVaR definition

- X - random variable
- $q_\alpha(X)$ - α quantile
- α-CVaR:

$$\Phi_\alpha(X) = \mathbb{E}[X | X \leq q_\alpha]$$

- Expected α% worst cases
- Prominent in finance
- Sensitive to rare, disastrous events



Estimation

$$\mathbb{E}[X] \approx \frac{1}{N} \sum_{1 \leq i \leq N} x_i$$

$$\Phi_\alpha[X] \approx \frac{1}{\alpha N} \sum_{\alpha N \text{ worst}} x_i$$

T., Glassner, Mannor. AAAI 2015

CVaR Policy Gradient

Gradient estimation

- **Likelihood ratio** method (Glynn 1990; a.k.a. policy gradient)
- Estimate gradient $\nabla \mathbb{E}(X)$

$$
\begin{aligned}
\nabla \mathbb{E}(X) &= \nabla \int_{-\infty}^{\infty} f_X(x) x dx \\
&= \int_{-\infty}^{\infty} \nabla f_X(x) x dx \\
&= \int_{-\infty}^{\infty} \frac{\nabla f_X(x)}{f_X(x)} f_X(x) x dx \\
&= \mathbb{E}\left(\frac{\nabla f_X(X)}{f_X(X)} X \right) \\
&\approx \frac{1}{N} \sum_{1 \le i \le N} \frac{\nabla f_X(x_i)}{f_X(x_i)} x_i
\end{aligned}
$$

CVaR Policy Gradient

Gradient estimation - CVaR

- Estimate gradient $\nabla \Phi_\alpha(X)$
- Maybe:

$$\nabla \Phi_\alpha(X) \approx \frac{1}{\alpha N} \sum_{\alpha N \text{ worst}} \frac{\nabla f_X(x_i)}{f_X(x_i)} x_i$$

CVaR Policy Gradient

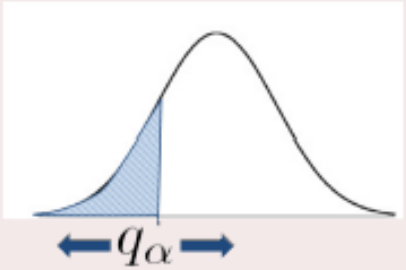
Gradient estimation - CVaR

- Estimate gradient $\nabla \Phi_\alpha(X)$
- Maybe:

$$\nabla \Phi_\alpha(X) \approx \frac{1}{\alpha N} \sum_{\alpha N \text{ worst}} \frac{\nabla f_X(x_i)}{f_X(x_i)} x_i$$

- **No!**
- Why?

$$\nabla \Phi_\alpha(X) = \nabla \int_{-\infty}^{q} \alpha^{-1} f_X(x) x dx$$



$$= \int_{-\infty}^{q} \alpha^{-1} \nabla f_X(x) x dx + \alpha^{-1} \nabla q f_X(q) q$$

CVaR Policy Gradient

Proposition

We have

$$\nabla \Phi_\alpha(R(X)) = \mathbb{E}\left[\frac{\nabla f_X(X)}{f_X(X)} (R(X) - q) \,\middle|\, R(X) \leq q \right]$$

Estimation algorithm

$$\nabla \Phi_\alpha(R(X)) \approx \frac{1}{\alpha N} \sum_{\alpha N \text{ worst}} \frac{\nabla f_X(x_i)}{f_X(x_i)} (R(x_i) - \hat{q})$$

where $\hat{q}$ is empirical quantile.

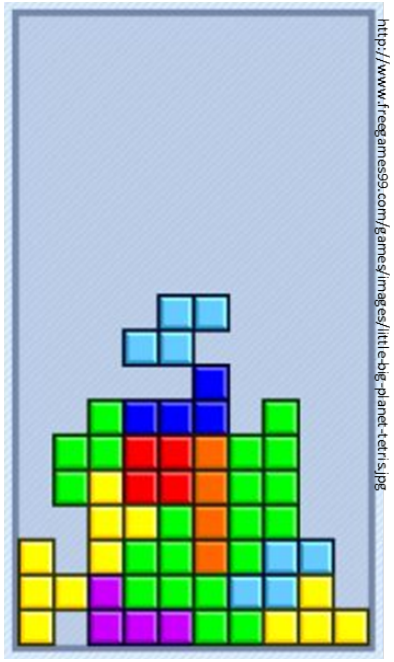
Guarantees

- Gradient estimate bias is $\mathcal{O}(N^{-1/2})$
- Convergence w.p. 1 of SGD to local CVaR optimum

CVaR Policy Gradient

Tetris

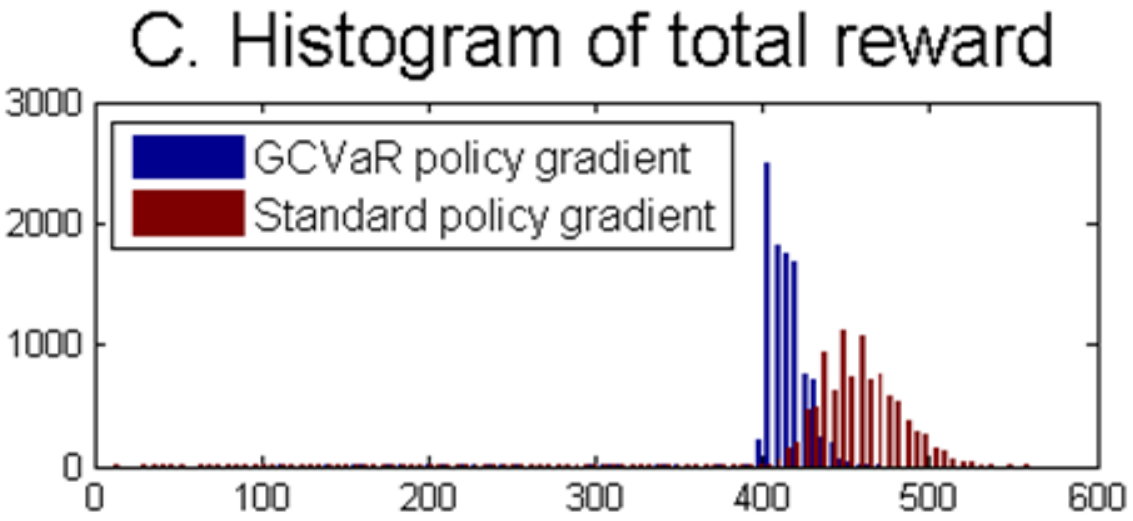
- Softmax policy, standard features (Thiery and Scherrer, 2009)
- Bonus for clearing multiple rows
- Compare standard policy gradient with CVaRSGD



CVaR Policy Gradient

Tetris

- Softmax policy, standard features (Thiery and Scherrer, 2009)
- Bonus for clearing multiple rows
- Compare standard policy gradient with CVaRSGD

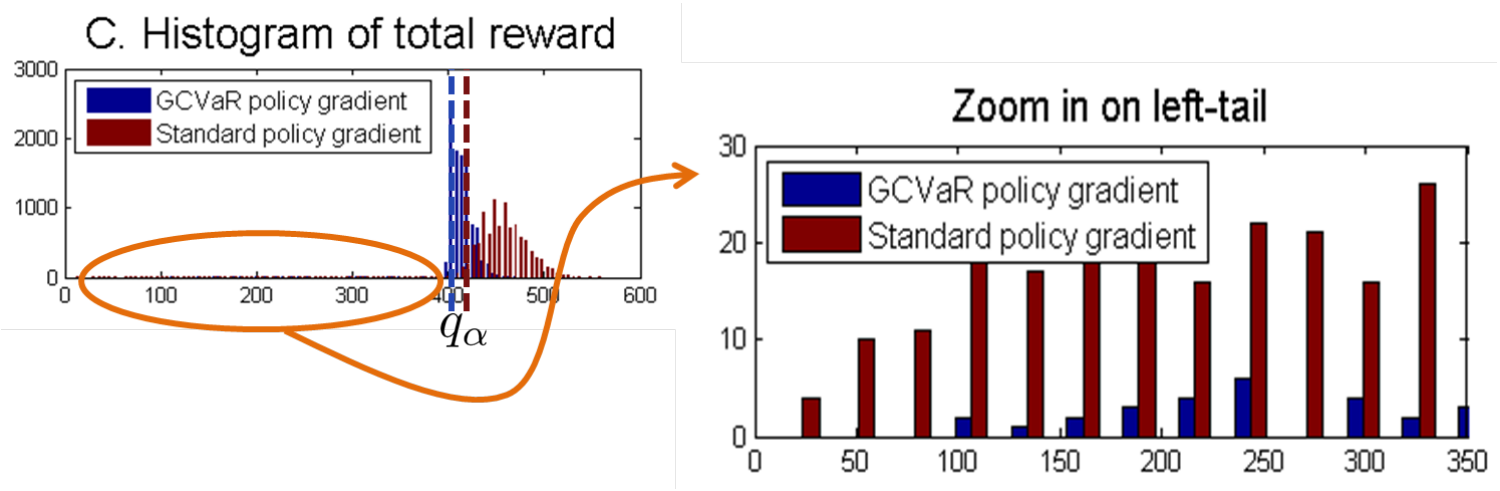


Avg. reward: **451** vs. 414

CVaR Policy Gradient

Tetris

- Softmax policy, standard features (Thiery and Scherrer, 2009)
- Bonus for clearing multiple rows
- Compare standard policy gradient with CVaRSGD



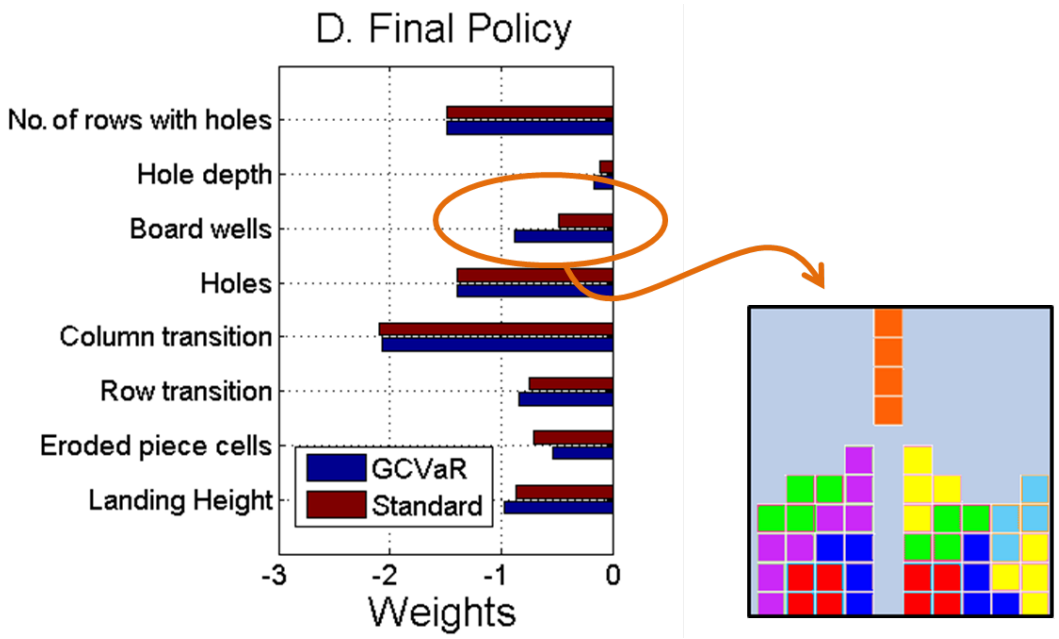
Avg. reward: **451** vs. 414

Reward CVaR: 323 vs. **394**

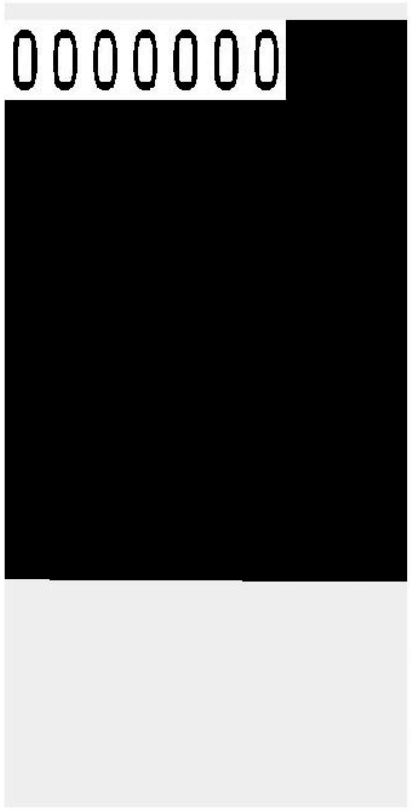
CVaR Policy Gradient

Tetris

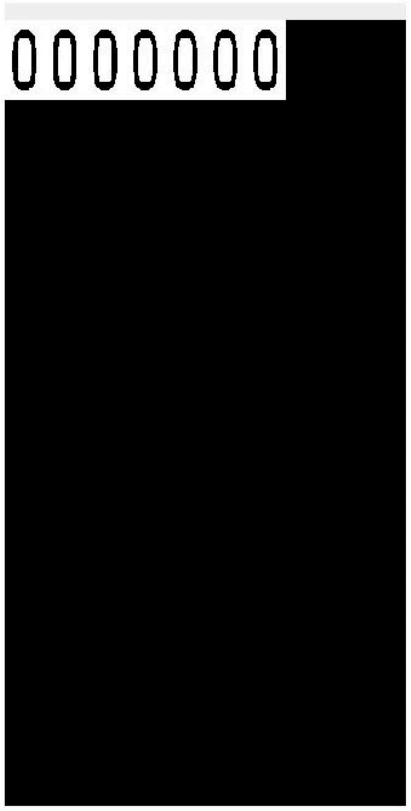
- Softmax policy, standard features (Thiery and Scherrer, 2009)
- Bonus for clearing multiple rows
- Compare standard policy gradient with CVaRSGD



CVaR Policy Gradient



Risk-Averse



Standard

Risk Averse RL - Summary

- Unified approach to safety
  - Unluckiness
  - Model mismatch
- Efficient algorithms
- Applications
  - Finance (e.g., options[1])
  - Robotics (simulation to real-world[2], safety)

[1] T., Mannor, Xu., ICML 2014
[2] Rajeswaran et al., 2016

Part 2:

Value Iteration Networks

T., Wu, Thomas, Levine, Abbeel. NIPS 2016

Motivation

- Goal: autonomous robots

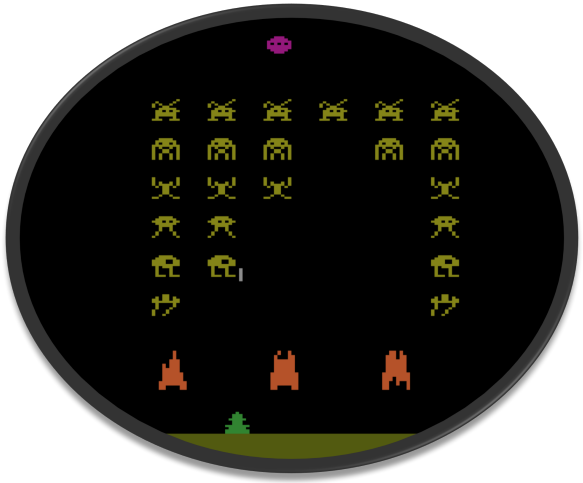
Robot, bring me the milk bottle!

- Solution: reinforcement learning?

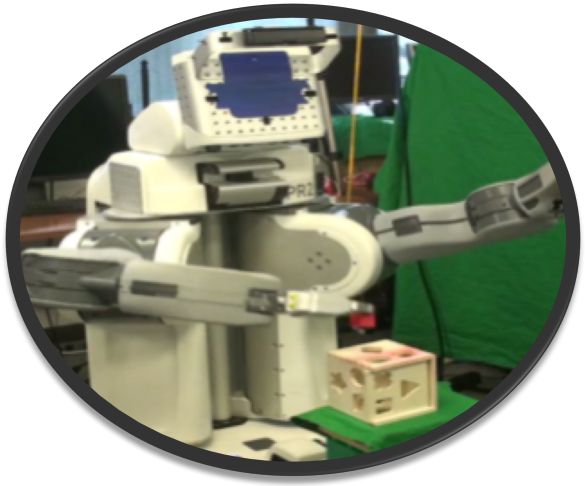
Image credit: http://www.wellandgood.com/wp-content/uploads/2015/02/Shira-fridge.jpg

Motivation

- Deep RL success: visual input → action



Mnih V., et al. *Nature* 2015

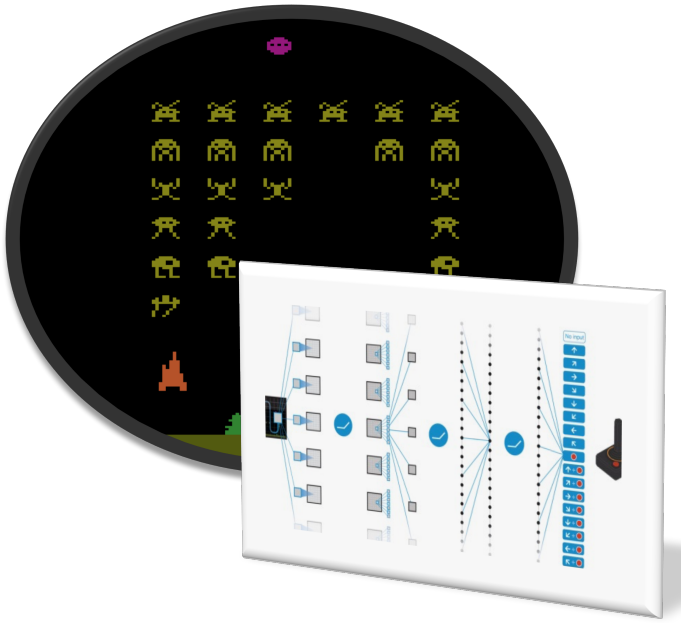


Levine S., et al. *JMLR* 2016

And many more…

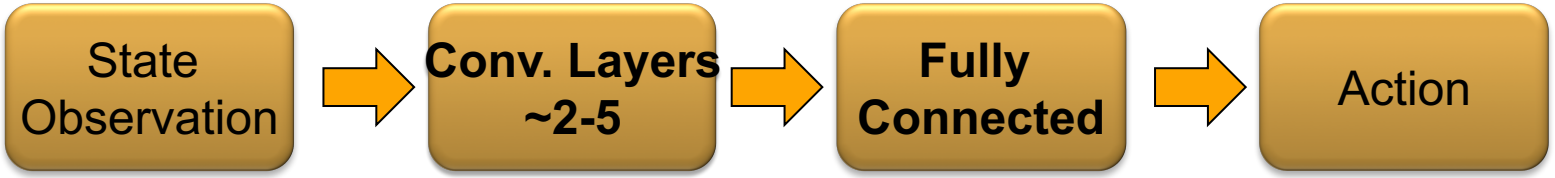
Motivation

- Deep RL success: visual input → action



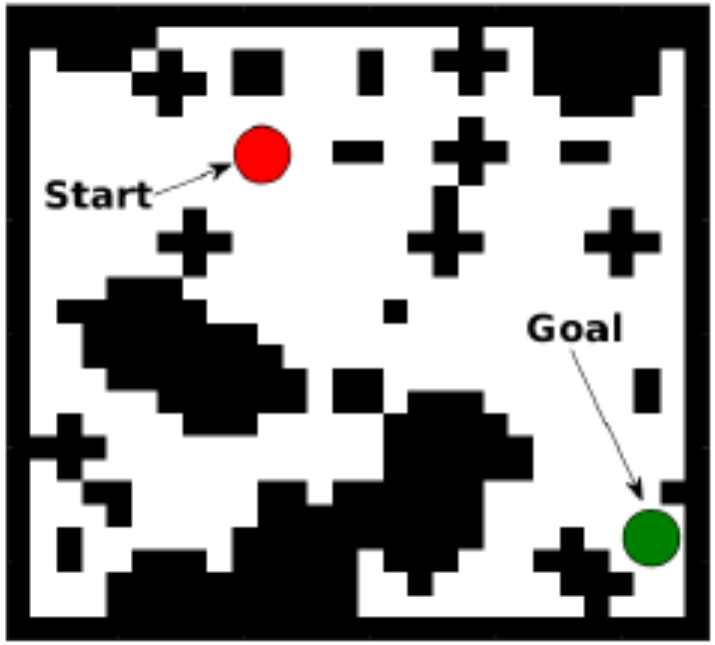
And many more…

- **Reactive** policies:



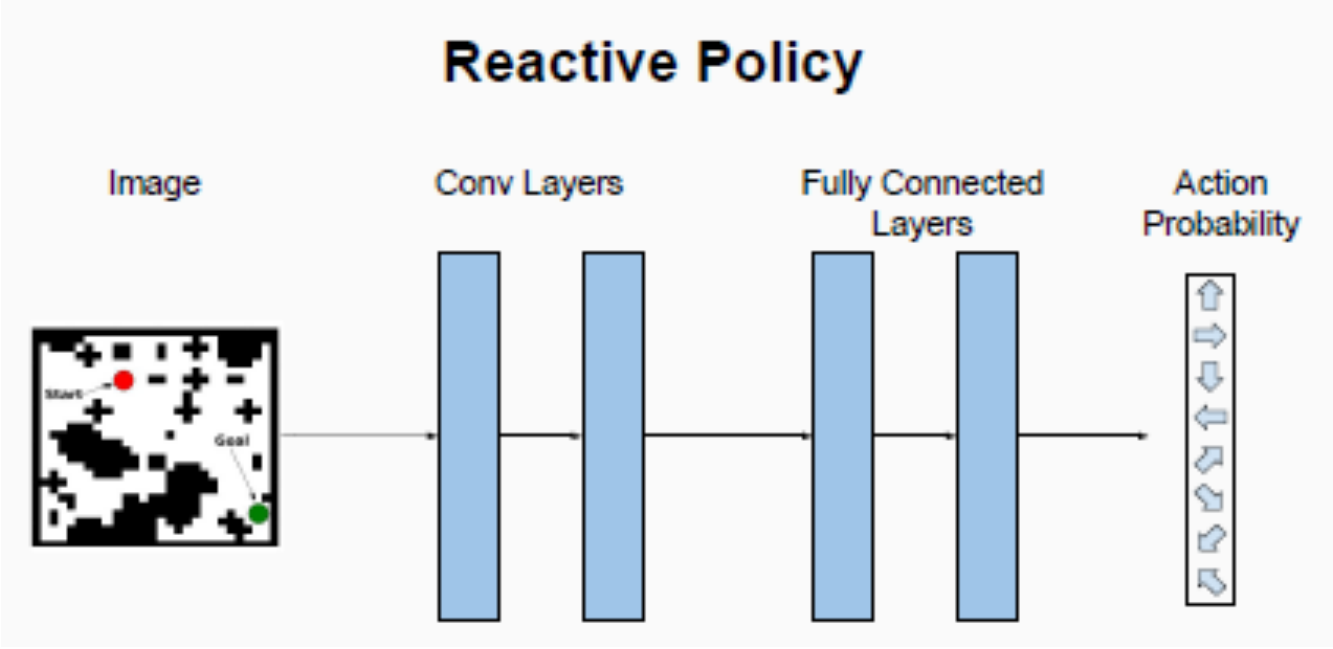
Generalization in RL

- Reactive policies can solve complex tasks
- But do they **understand**?
- A simple test: **generalization** on grid worlds



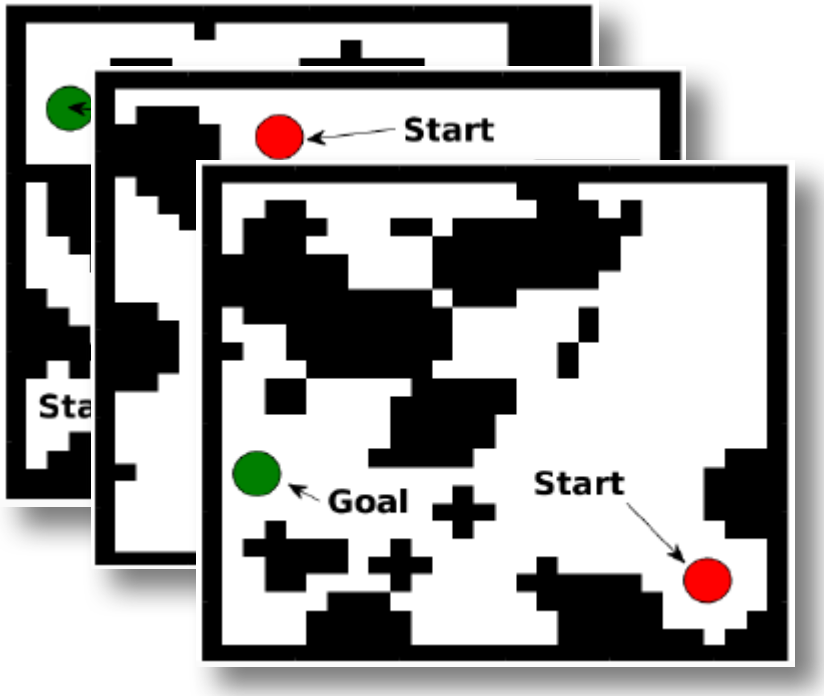
Generalization in RL

- Reactive policies can solve complex tasks
- But do they **understand**?
- A simple test: **generalization** on grid worlds

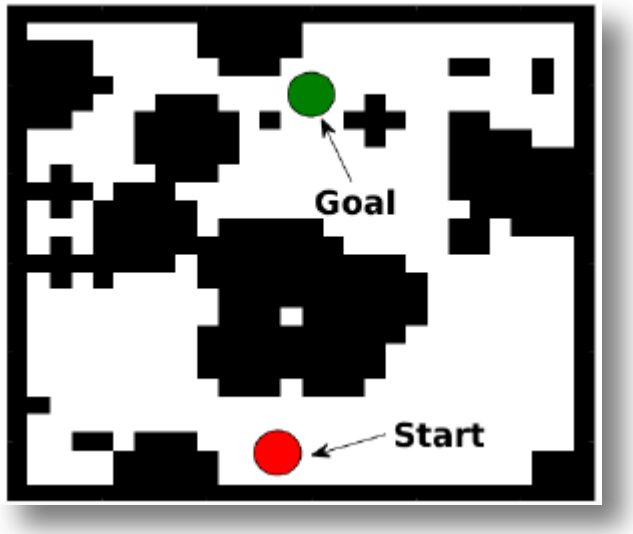


Generalization in RL

Train

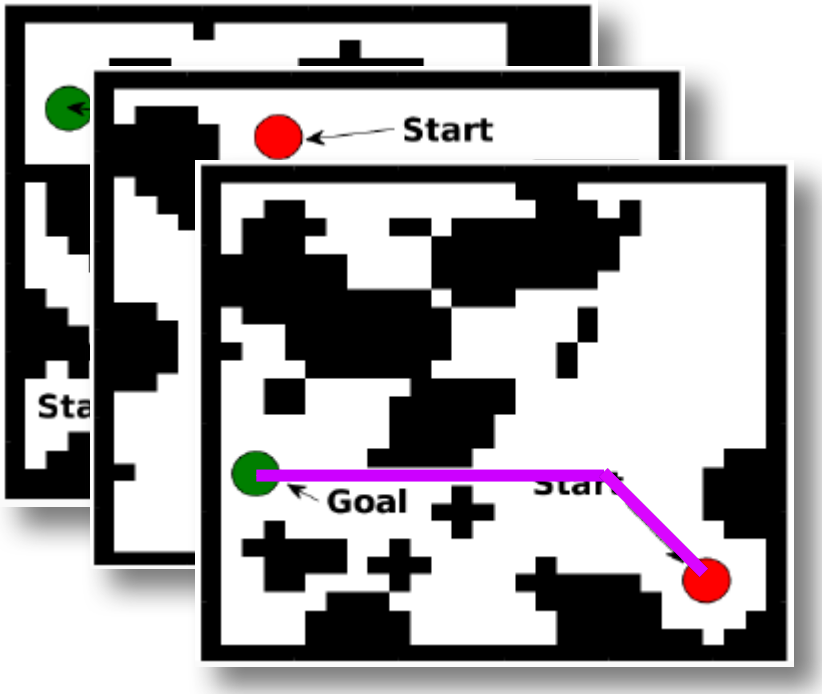


Test

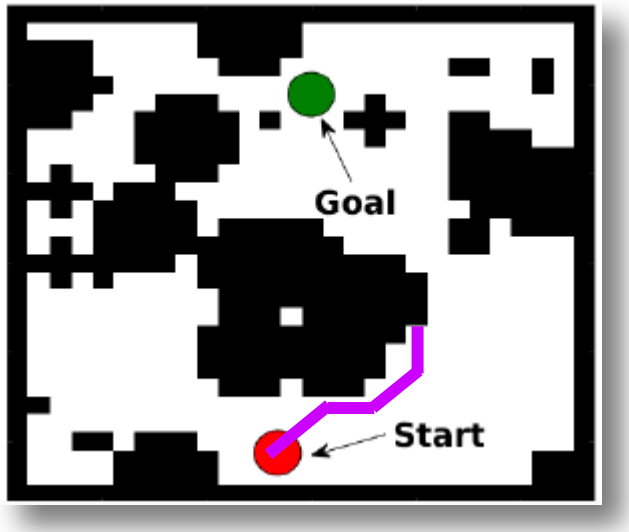


Generalization in RL

Train



Test



Observation: reactive policies do not generalize well

Generalization in RL

Why do reactive policies fail to generalize?

- A sequential task requires a planning computation
- RL gets around that – learns a mapping

- Q/advantage/expert: planning on training domains

Generalization in RL

Why do reactive policies fail to generalize?

- A sequential task requires a planning computation
- RL gets around that – learns a mapping

- Q/advantage/expert: planning on training domains
- New task – need a new plan

Generalization in RL

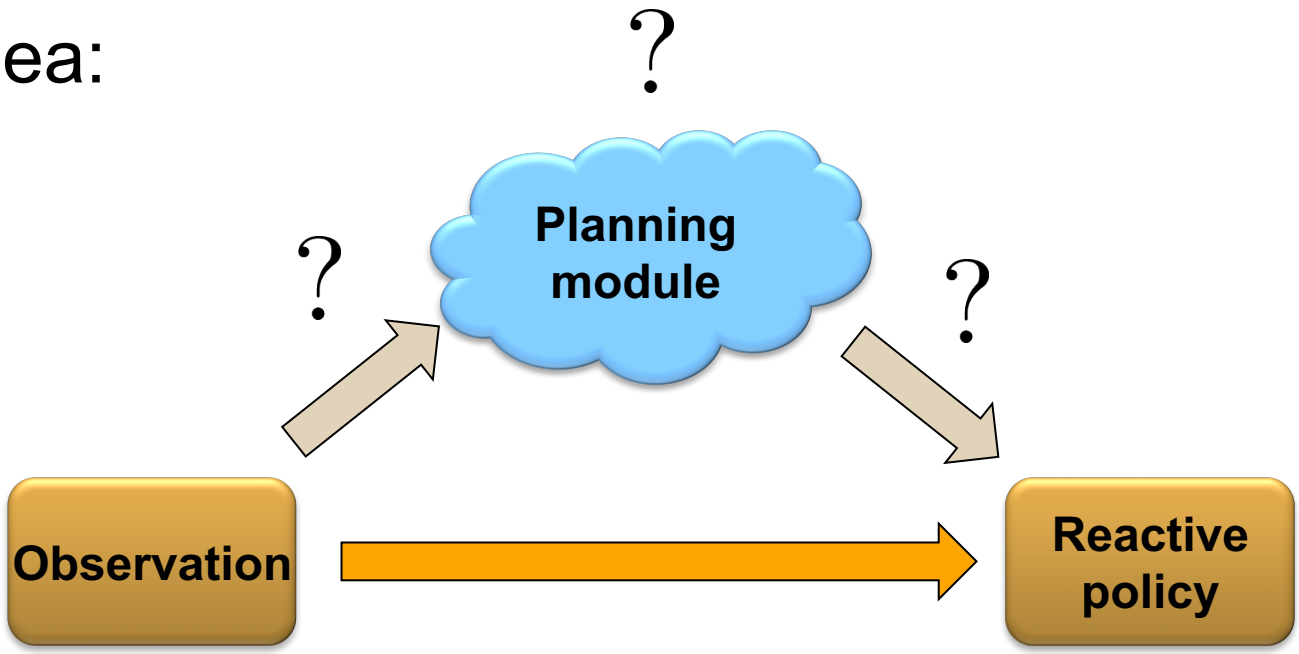
Value iteration networks:

- Learn to plan
- Generalize to unseen tasks

Model

Network that Can Plan

Main idea:

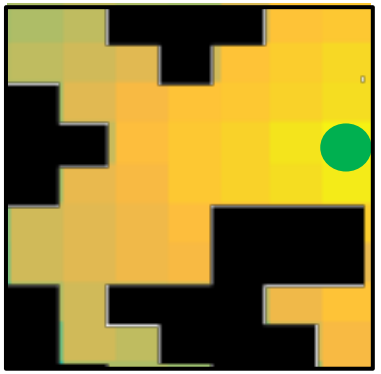


Challenges:

1. What to plan?
2. How to plan?
3. How to use the plan?
4. How to learn?

Value Iteration Planning

- Value function: state → long-term value
  - Shortest-distance to-go
  - Max. sum of future rewards from state

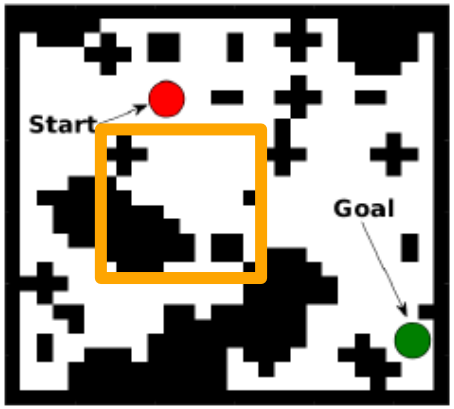


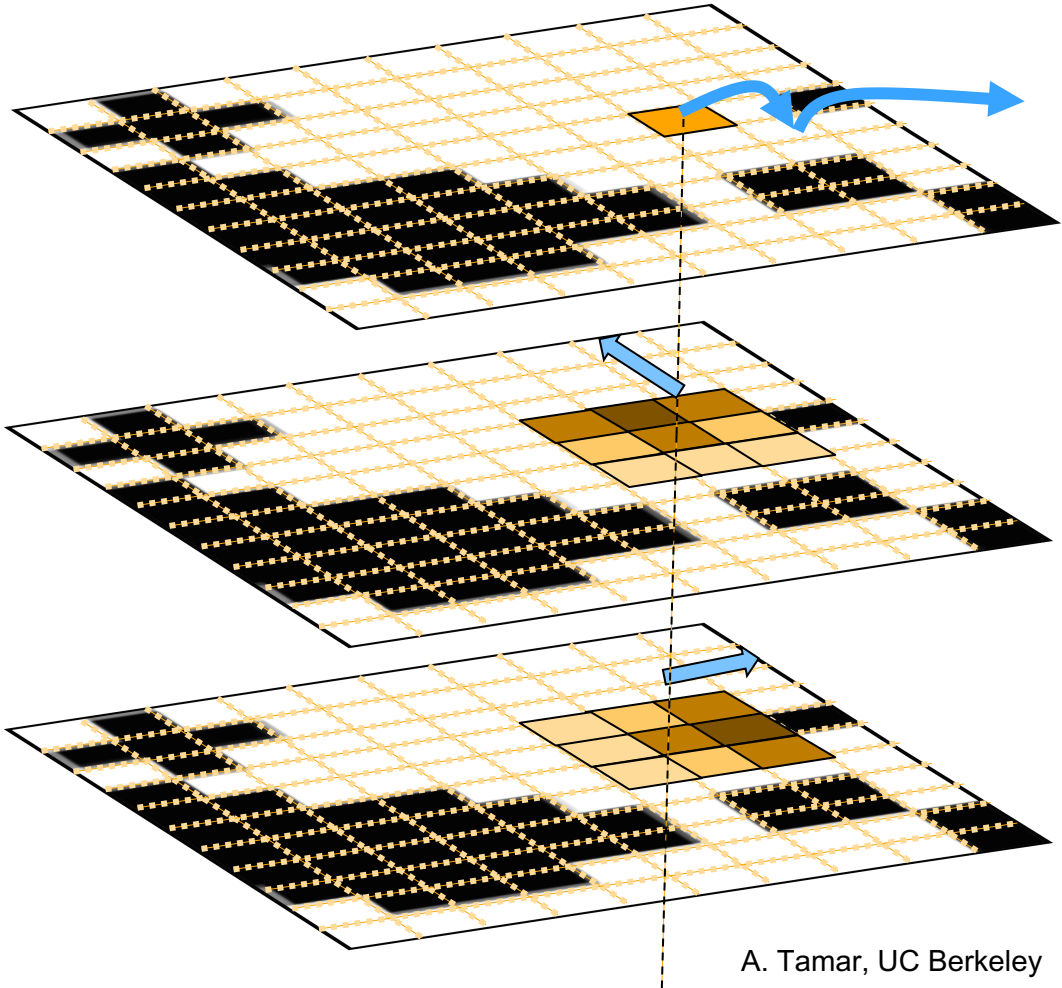
$$V(s) = \max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) | \pi, s_0 = s\right]$$

- Sufficient for **reactive policy**
  - Action that maximizes value = optimal
- How to calculate?

Value Iteration Planning

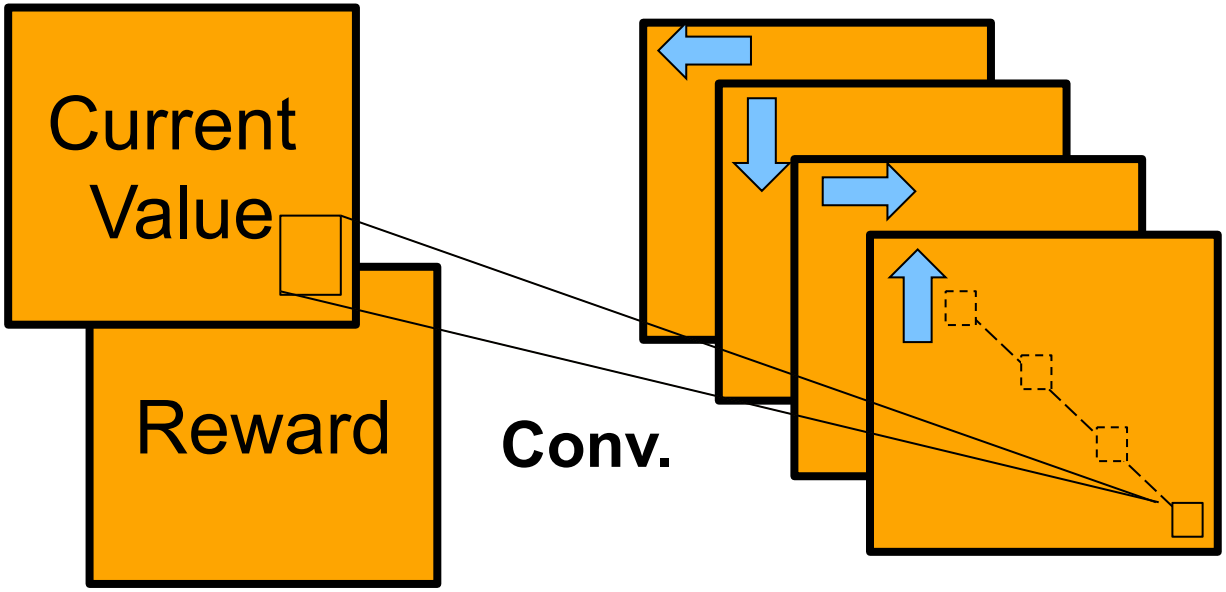
$$\text{Value} := \max_{\text{action}} \mathbb{E}\{\text{reward} + \text{next value}\}$$





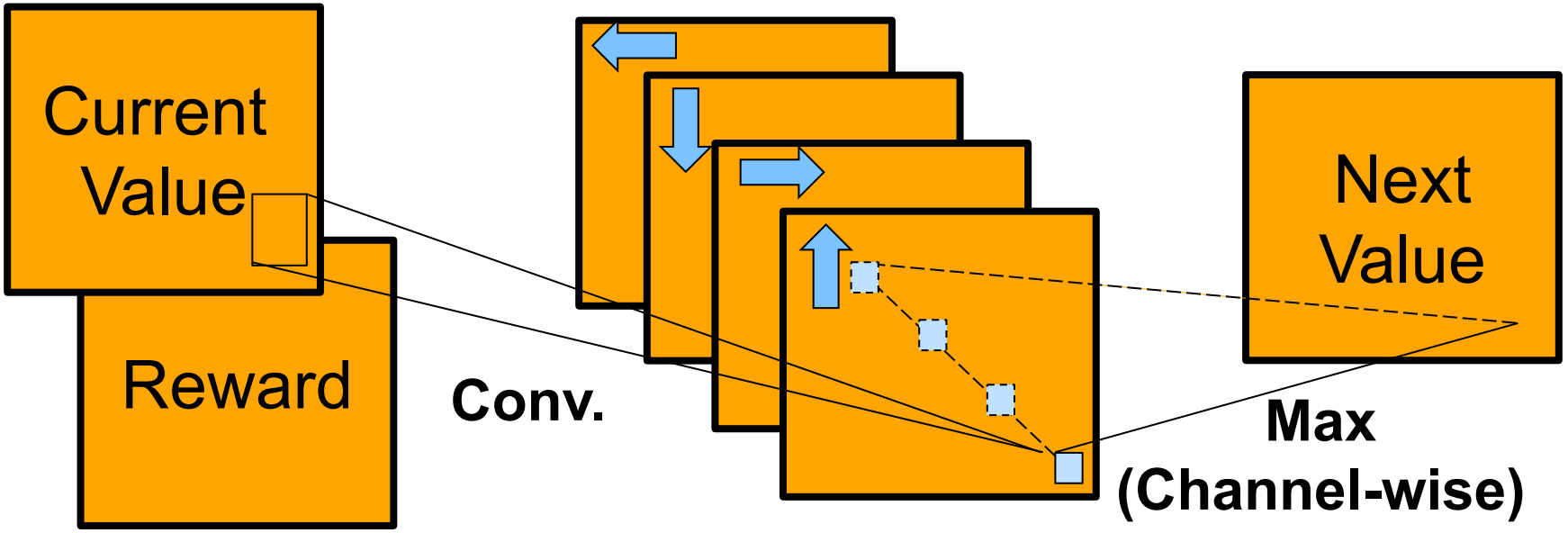
Value Iteration Planning

$$\text{Value} := \max_{\text{action}} \mathbb{E}\{\text{reward} + \text{next value}\}$$



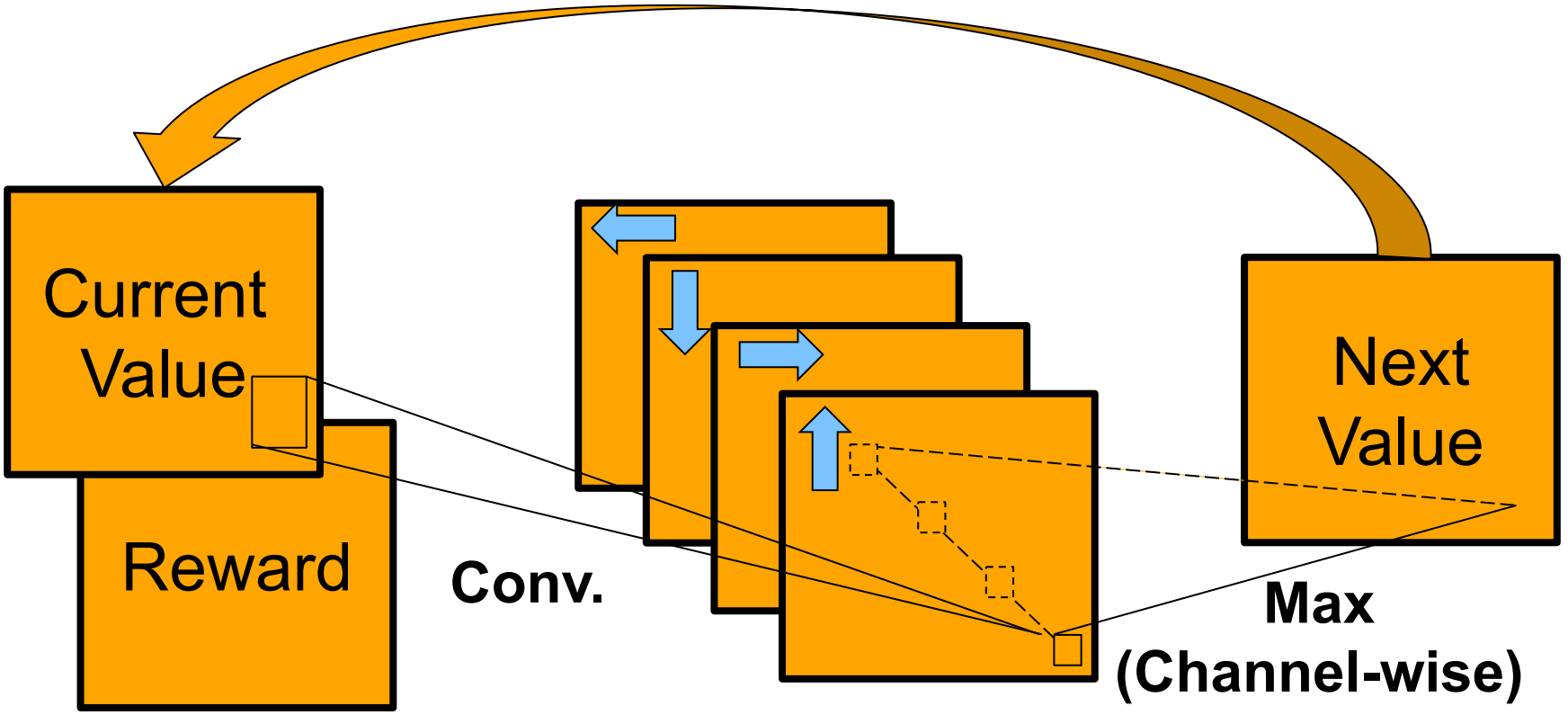
Value Iteration Planning

$$\text{Value} := \max_{\text{action}} \mathbb{E}\{\text{reward} + \text{next value}\}$$



Value Iteration Planning

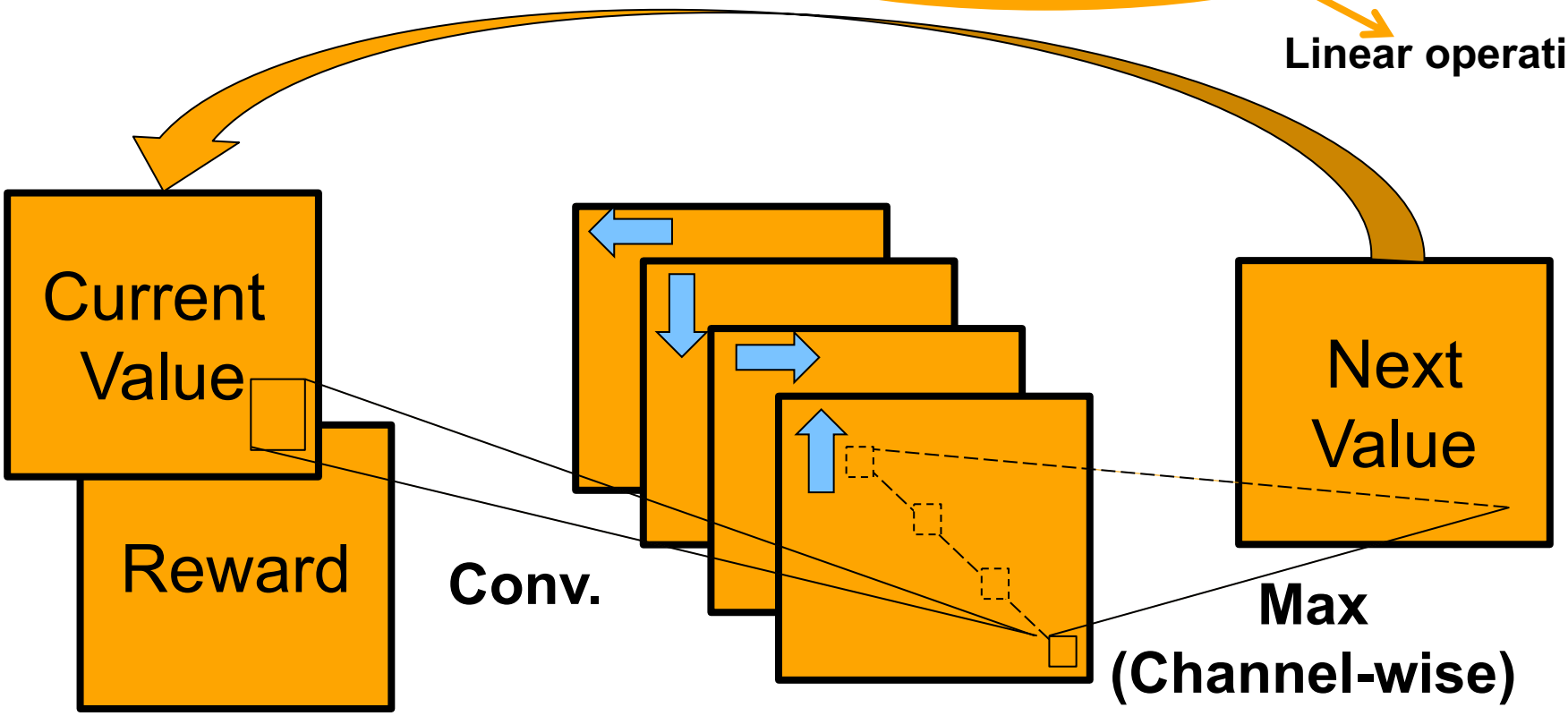
$$\text{Value} := \max_{\text{action}} \mathbb{E}\{\text{reward} + \text{next value}\}$$



Value Iteration Planning

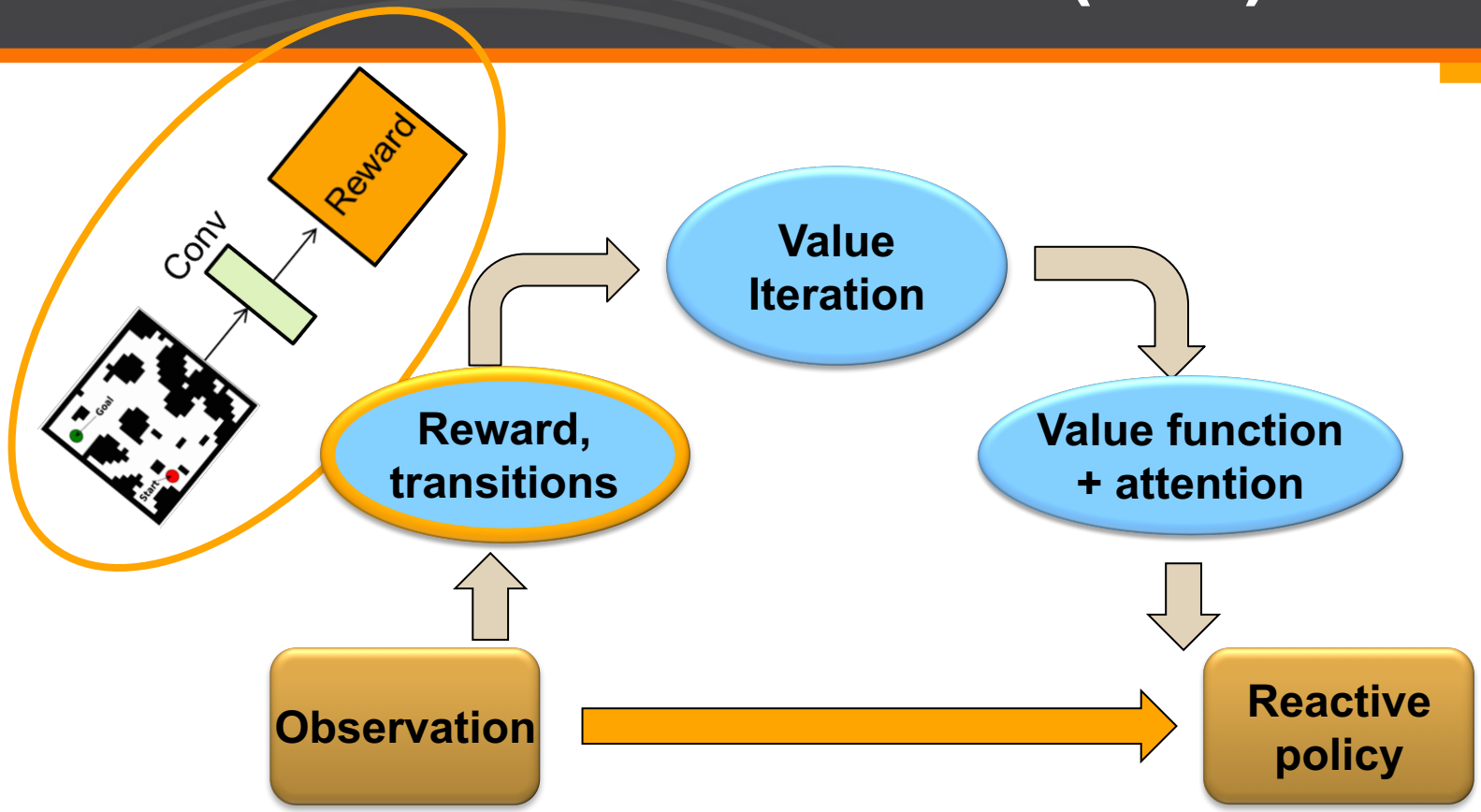
$$\text{Value} := \max_{\text{action}} \mathbb{E}\{\text{reward} + \text{next value}\}$$

Linear operation



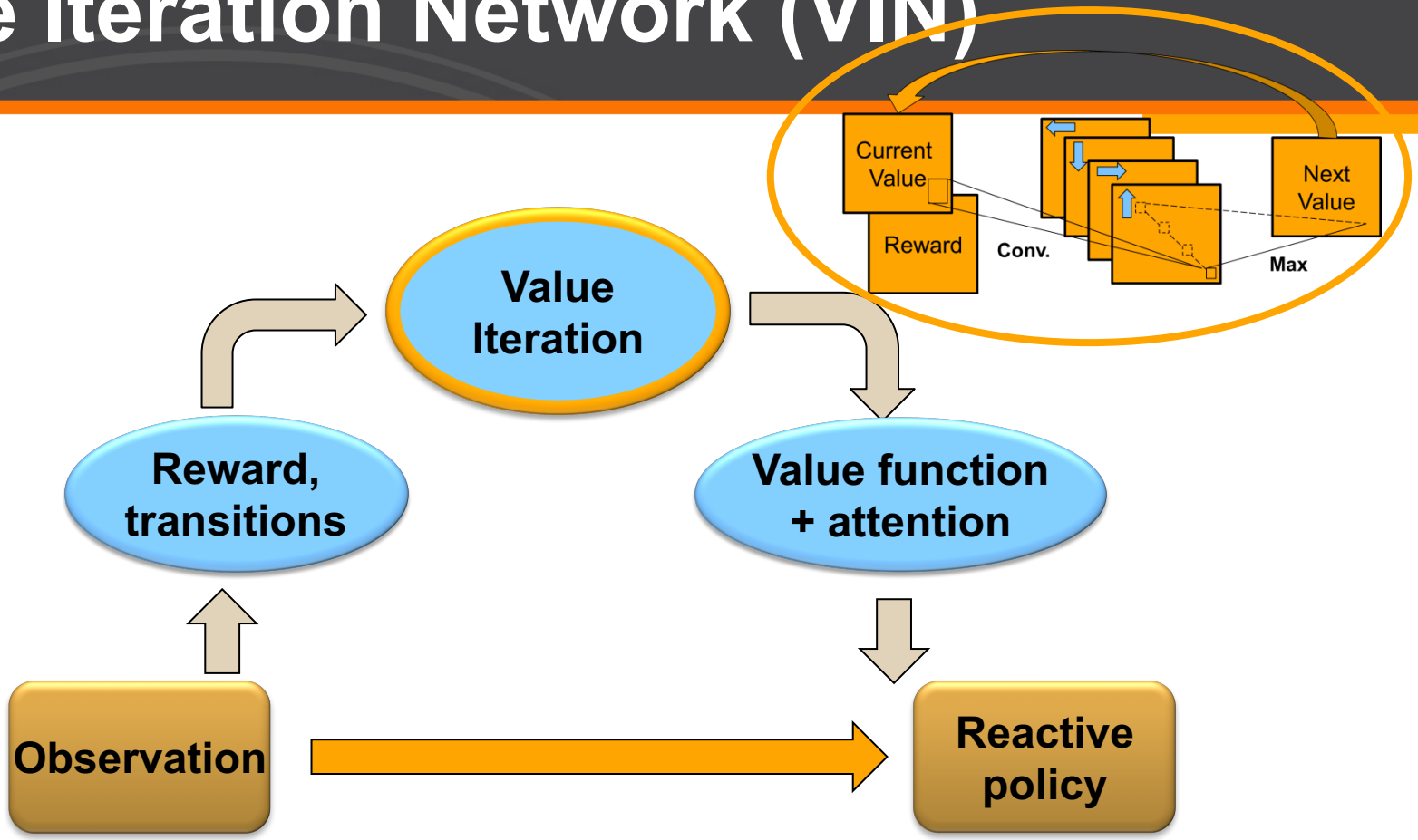
Value Iteration = Convnet !

Value Iteration Network (VIN)



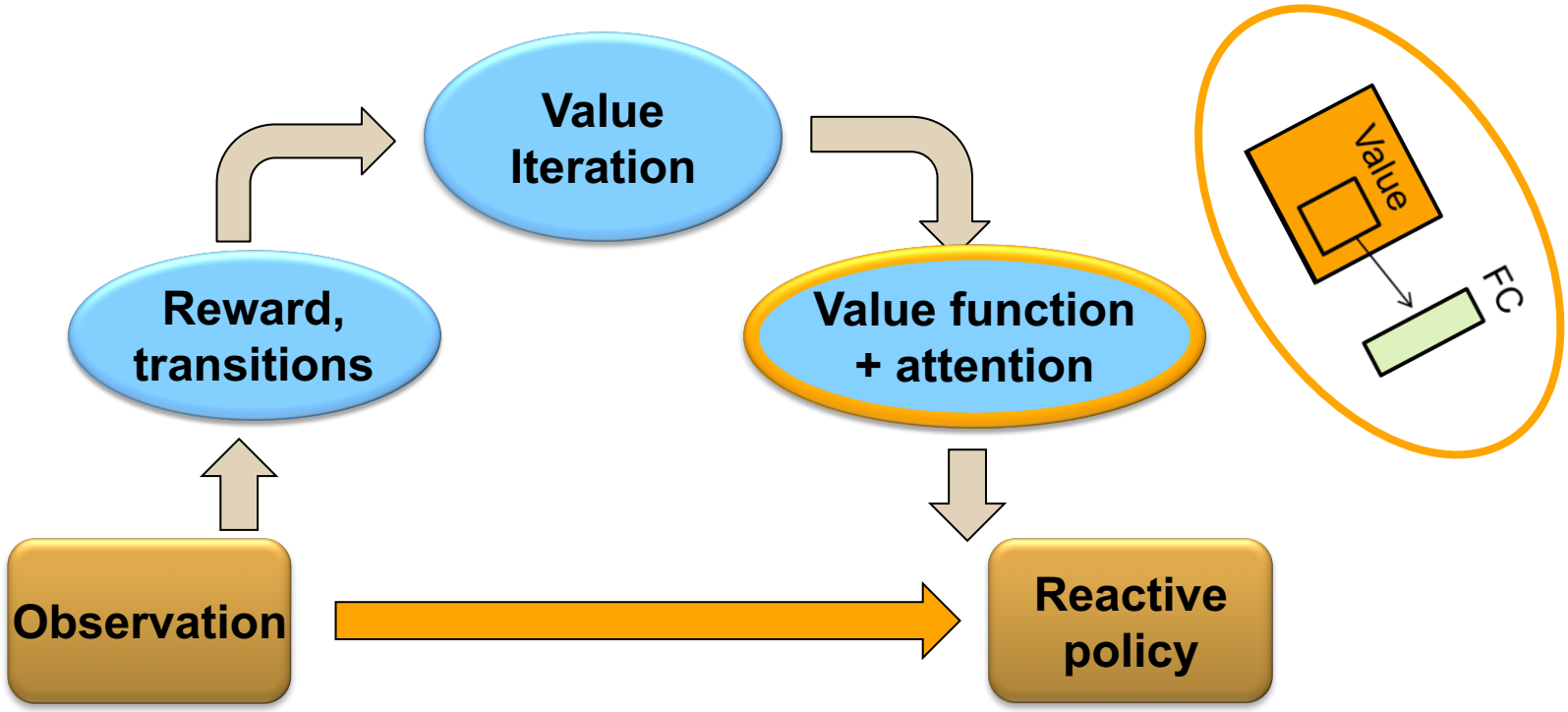
1. What to plan?

Value Iteration Network (VIN)



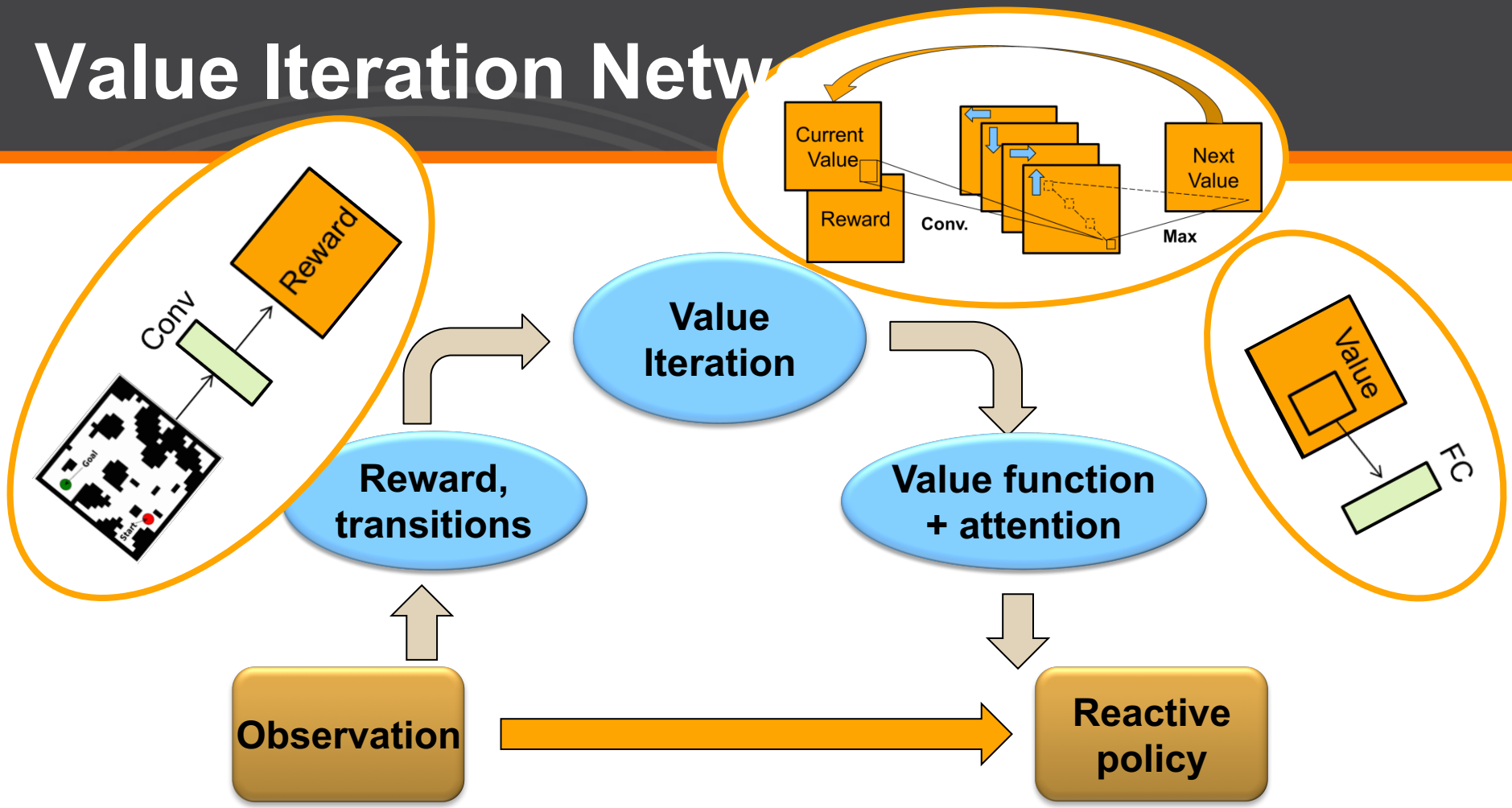
1. What to plan?
2. How to plan?

Value Iteration Network (VIN)



1. What to plan?
2. How to plan?
3. How to use the plan?

Value Iteration Netw



Backprop through the whole network!

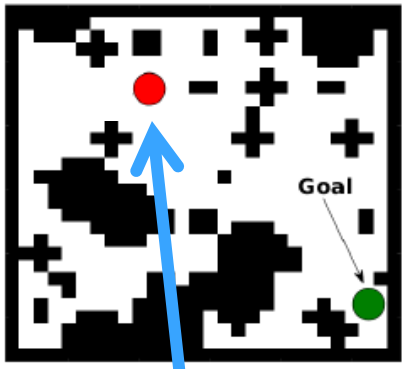
Learn planning computation that's useful for predicting actions

Results

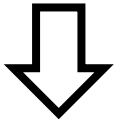
Grid-World Domain

- Supervised learning

Data:



,



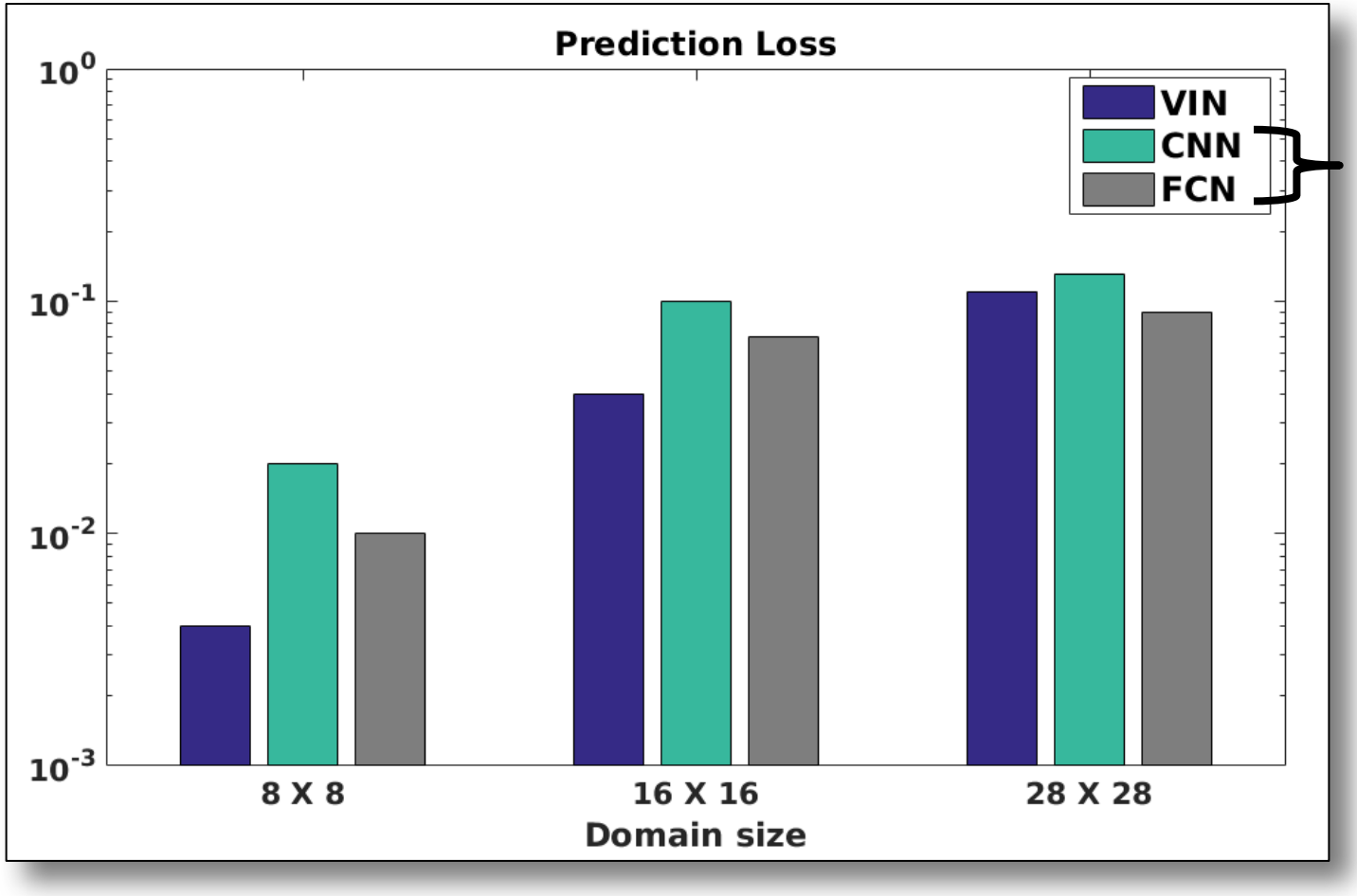
Shortest-path direction

5000 random maps X 7 trajectories

- Compare VINs with:
  - Convnet inspired by DQN architecture (Mnih et al. *Nature* 2015)
  - Fully convolutional net (FCN) (Long et al. *CVPR* 2015)

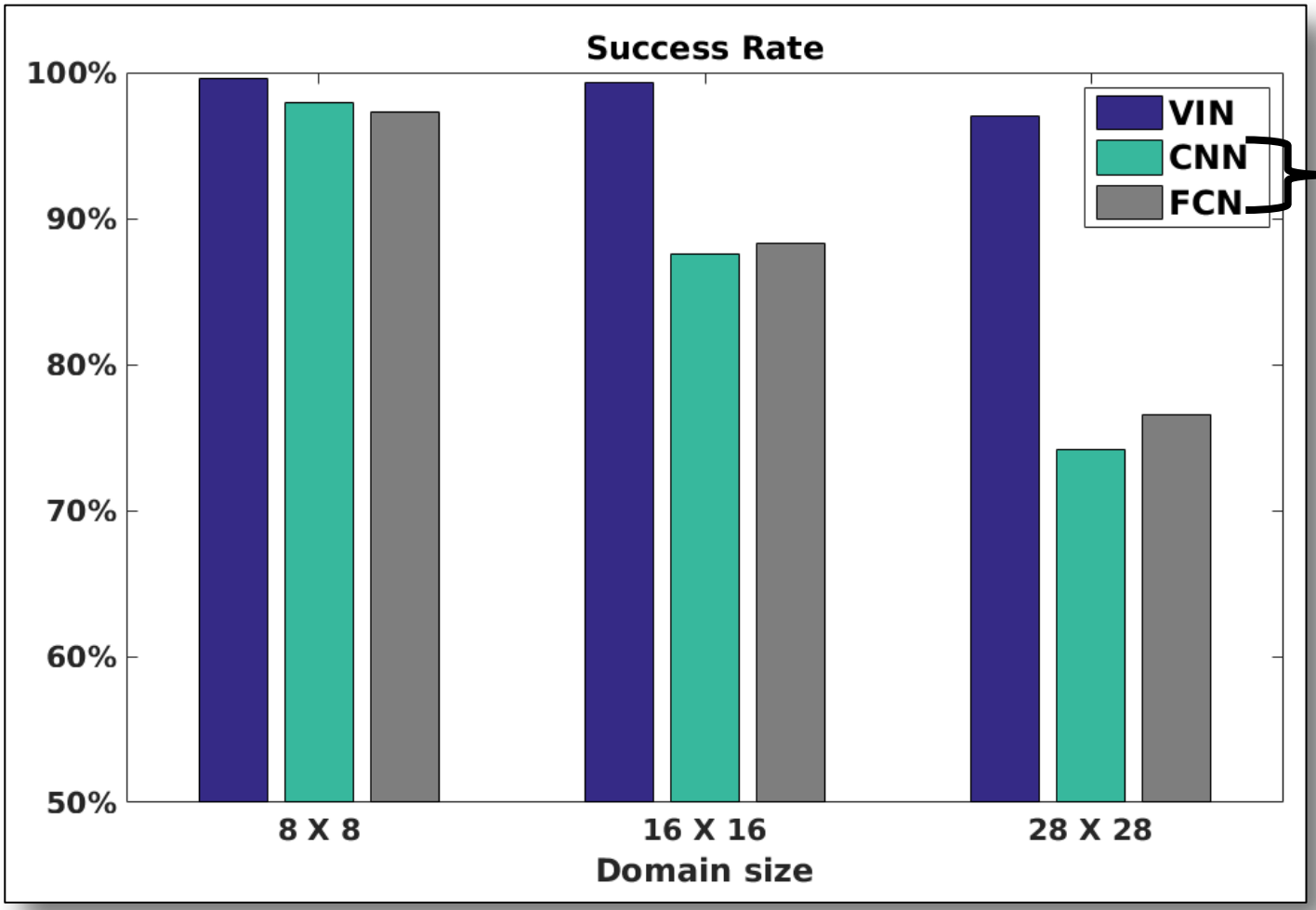
Grid-World Domain

Prediction loss on test set

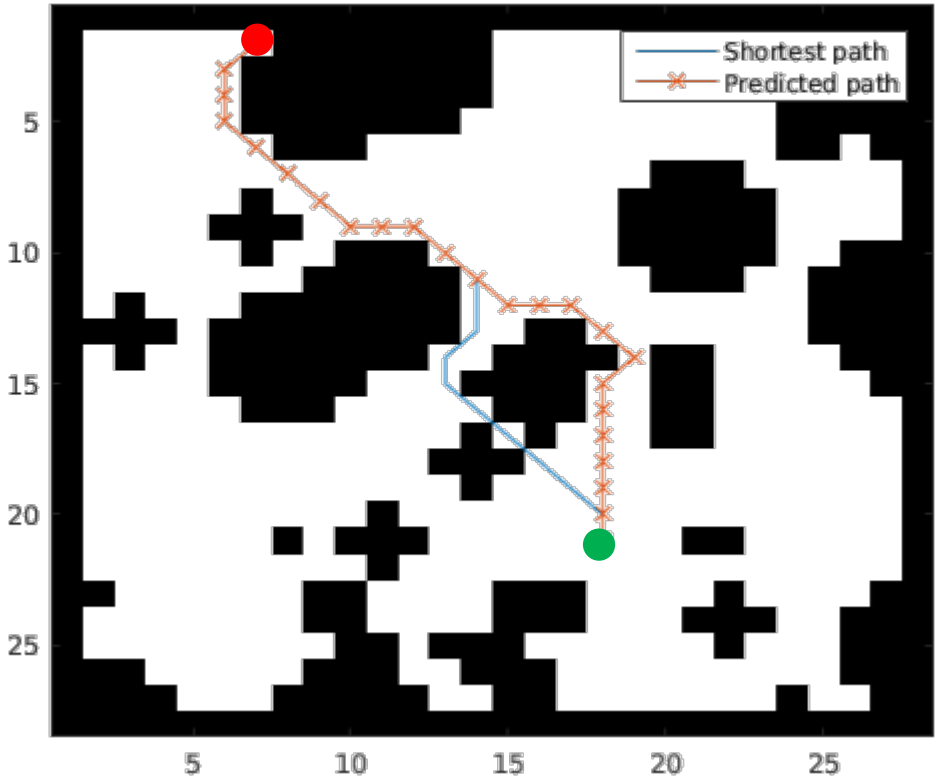


Grid-World Domain

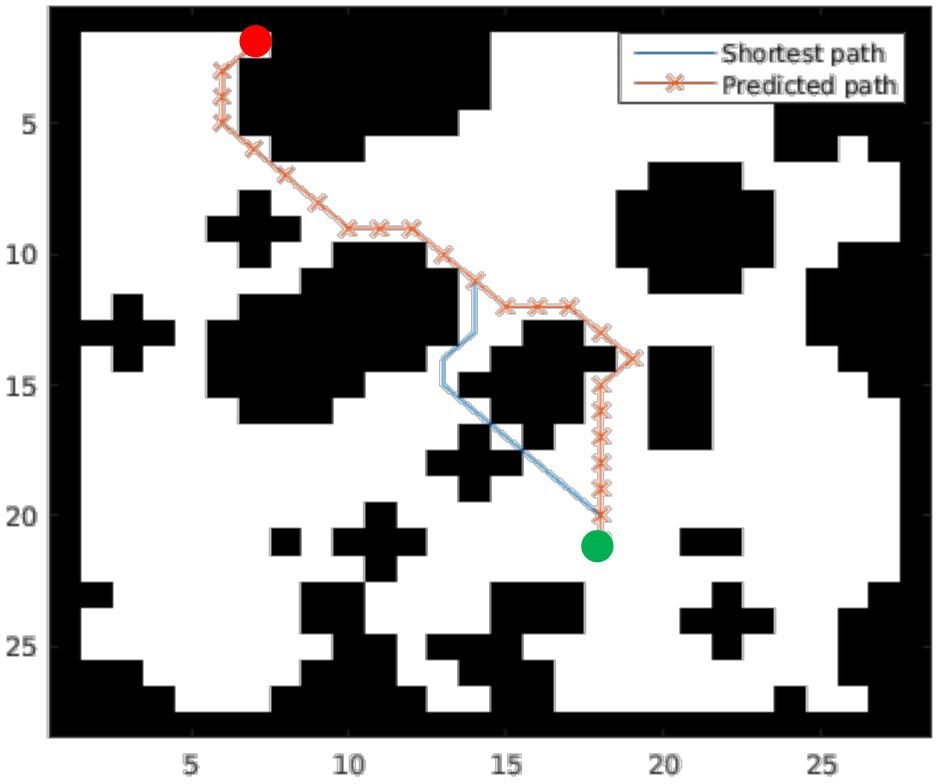
Success of reaching goal

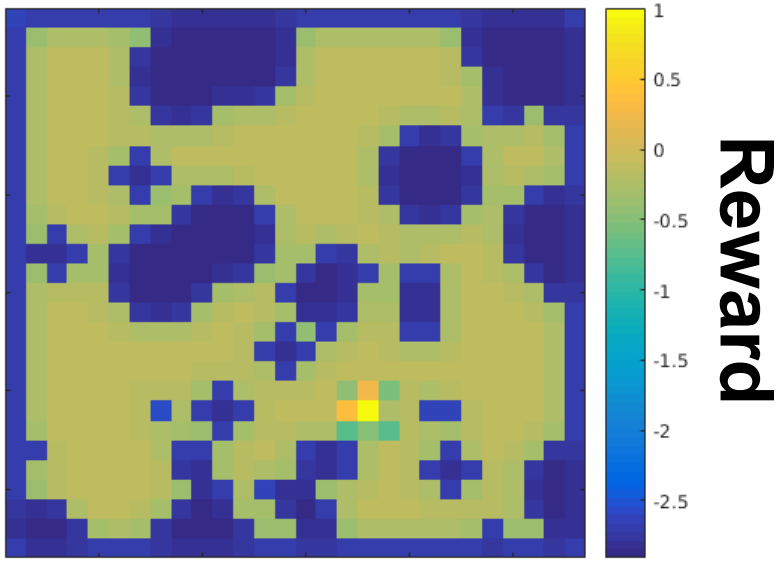


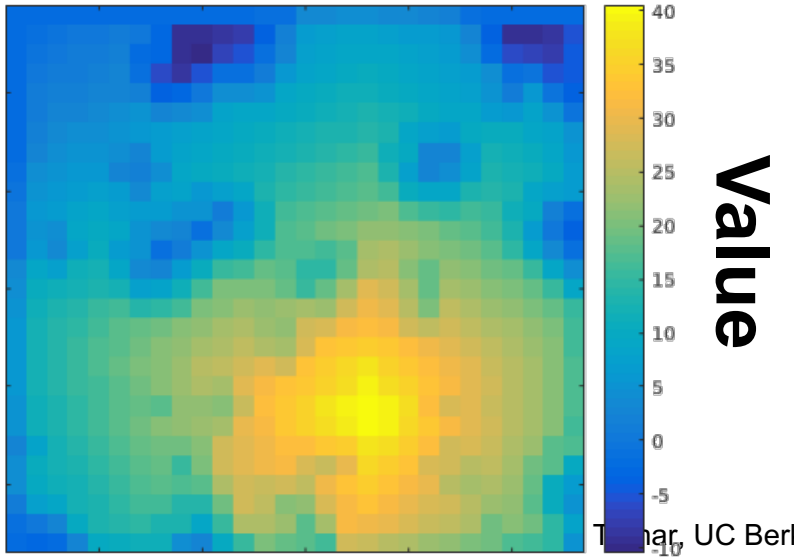
Grid-World Domain



Grid-World Domain



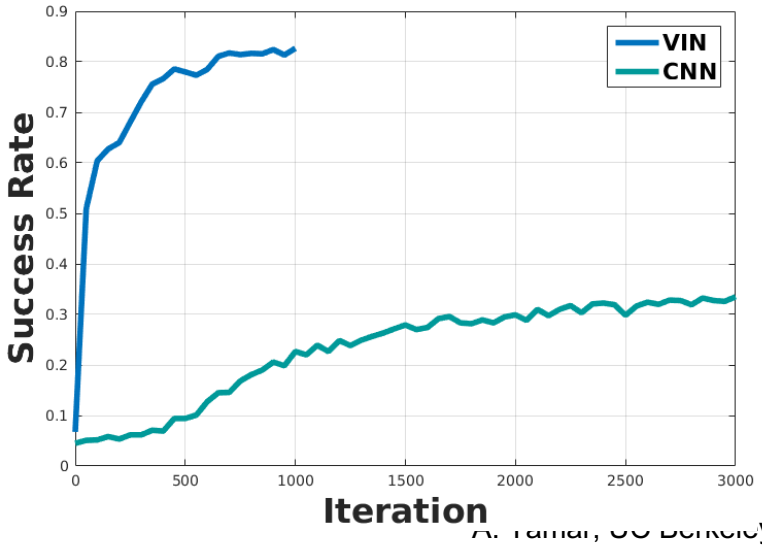




Grid-World Domain

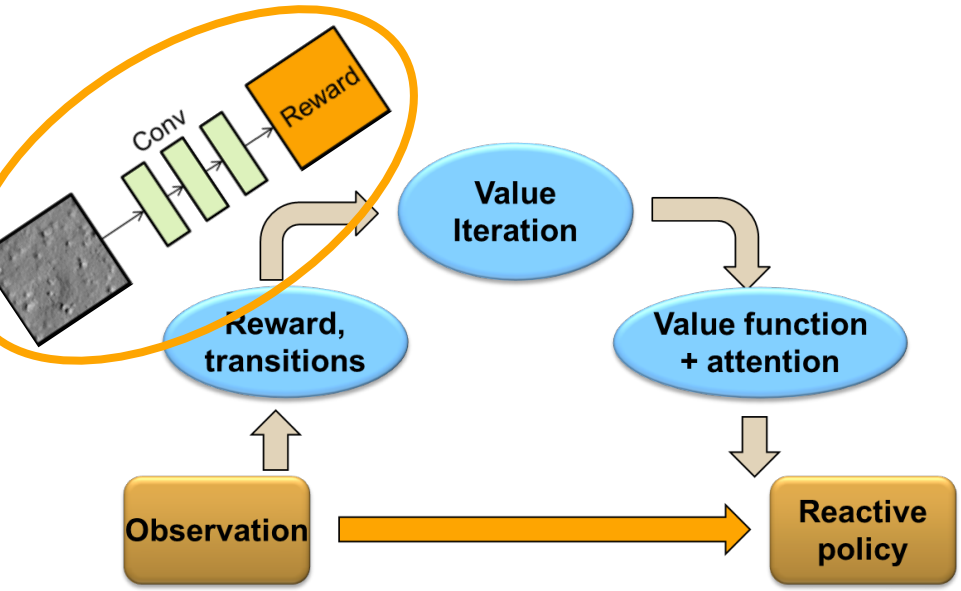
- Is it only ‘depth’?
  - VIN with untied weights
  - Degrades performance, esp. with less data

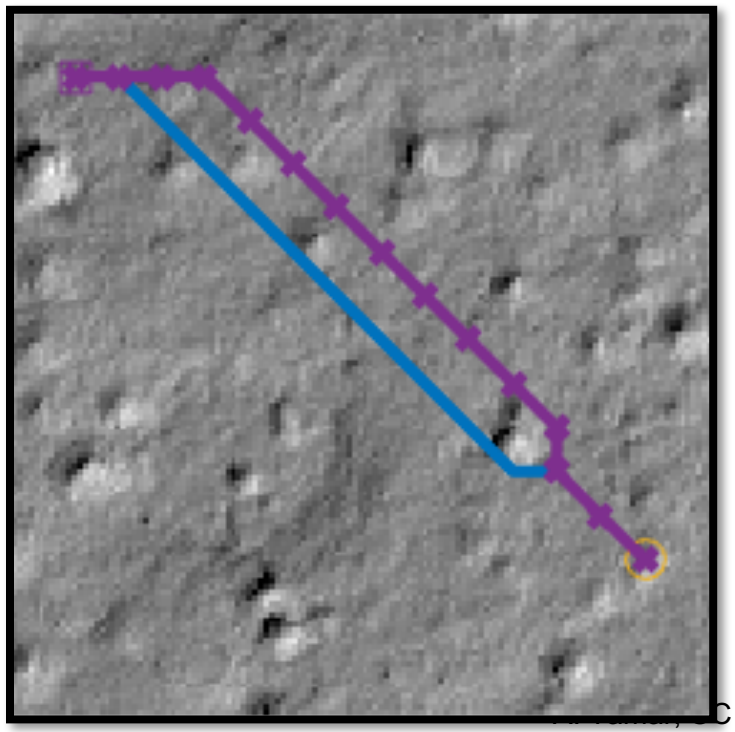
- Train using RL
  - TRPO (Schulman et al. *ICML* 2015)



Perception & Control

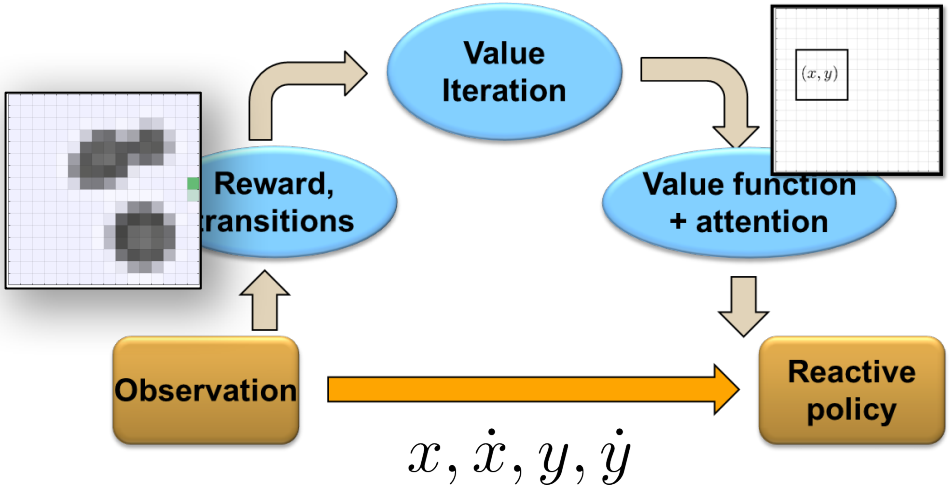
- A VIN is just a neural network
- Compose with perception & control modules
- Natural image input: Mars navigation

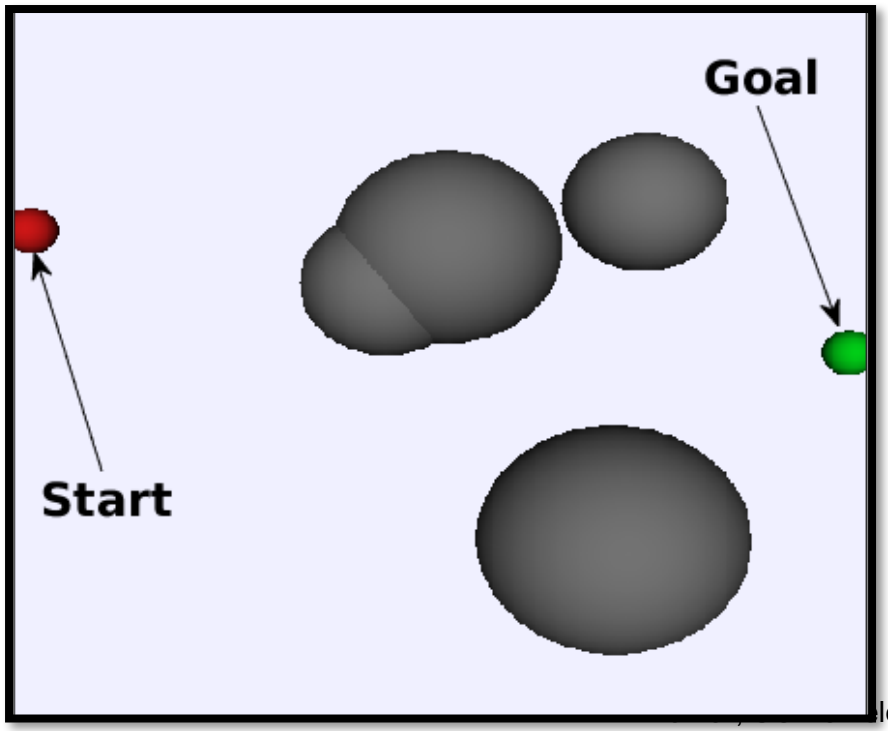




Perception & Control

- Continuous control
  - Discrete planning in continuous domain?
  - Let the network figure it out!





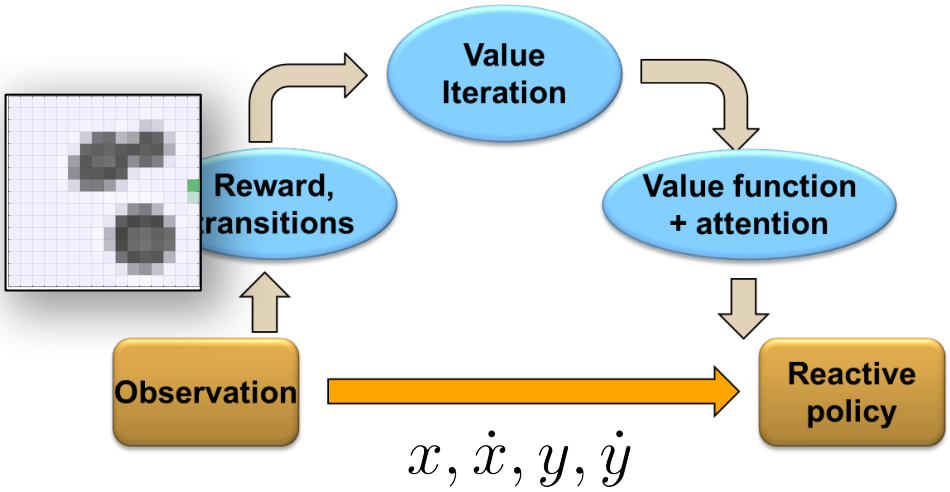
Perception & Control

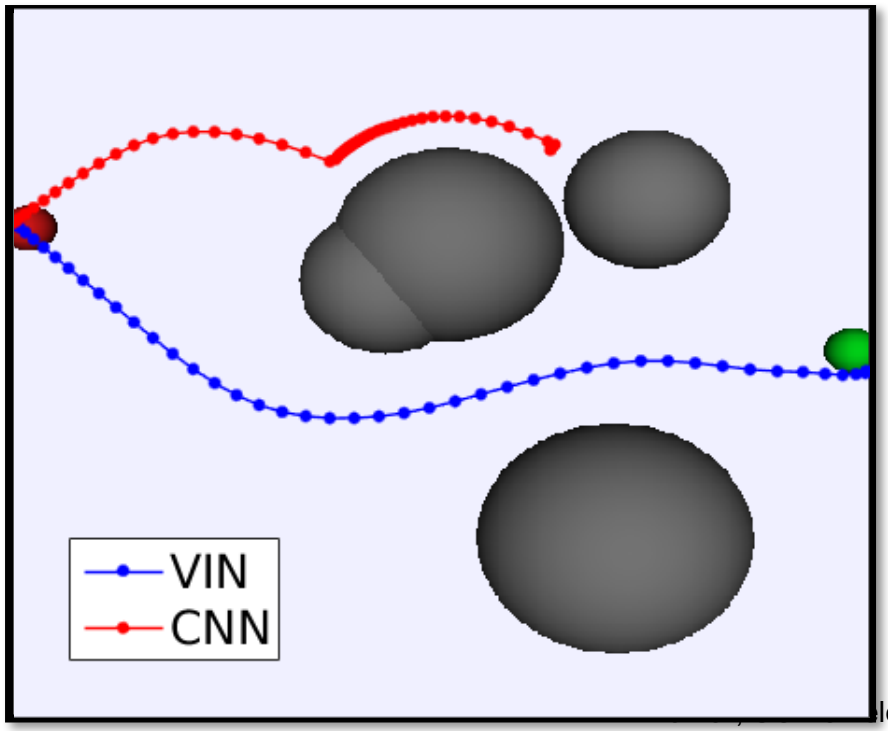
- Train: Guided policy search (unknown dynamics)

(Levine & Abbeel, *NIPS 2014*)

- Compare: conv-net

(Mnih et al. Nature 2015, Lillicrap et al. ICLR 2016)





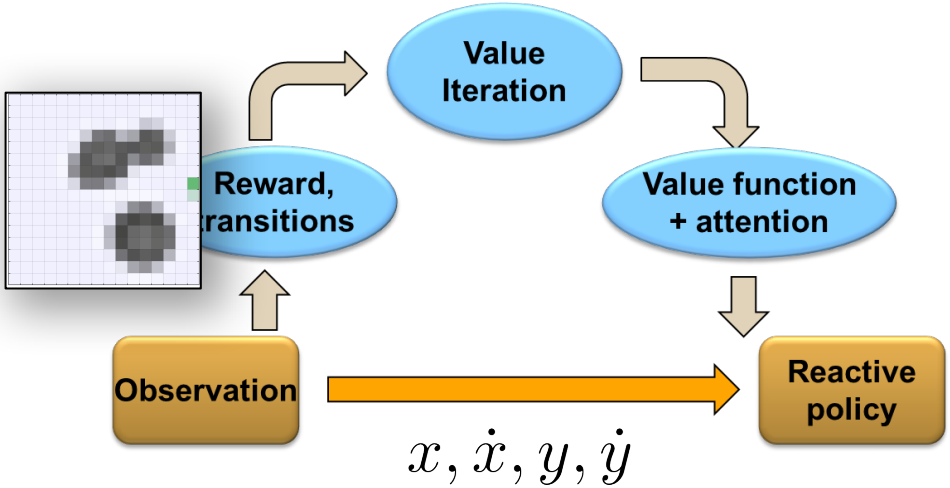
Perception & Control

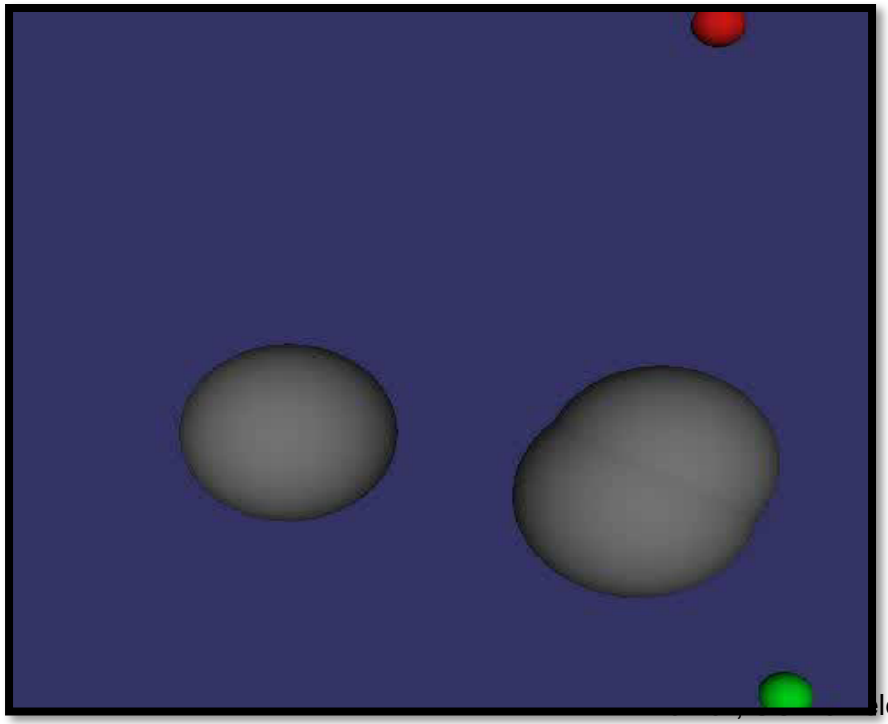
- Train: Guided policy search (unknown dynamics)

(Levine & Abbeel, *NIPS 2014*)

- Compare: conv-net

(Mnih et al. Nature 2015, Lillicrap et al. ICLR 2016)



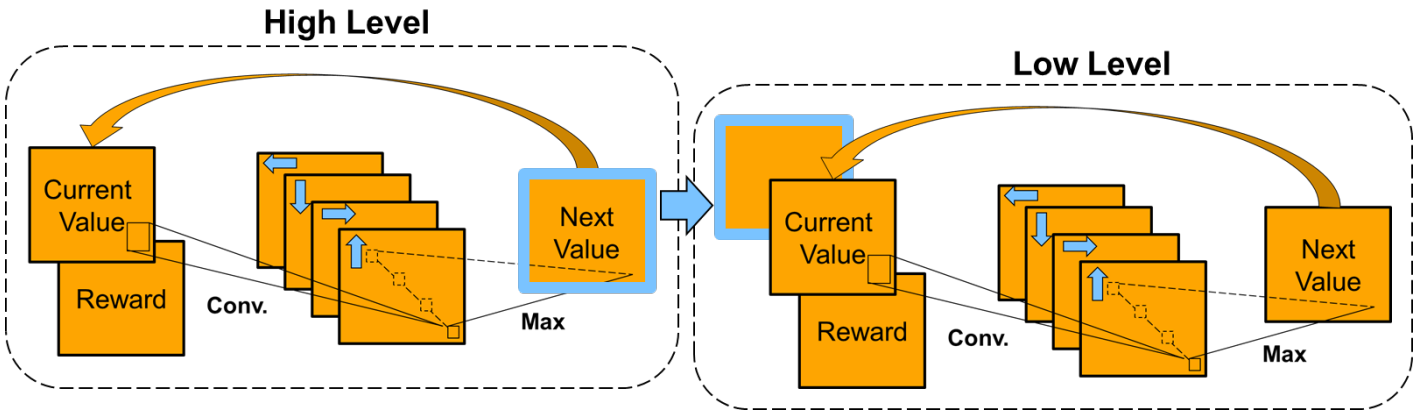


More Examples

- Navigate website links to find query

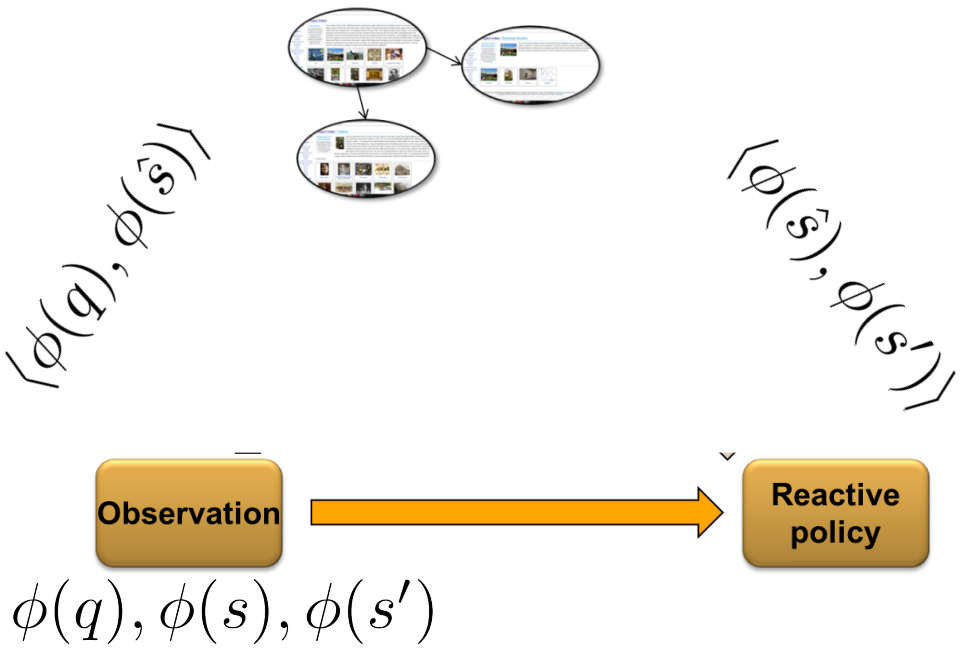
(Nogueira & Cho, NIPS 2016)

- Hierarchical composition

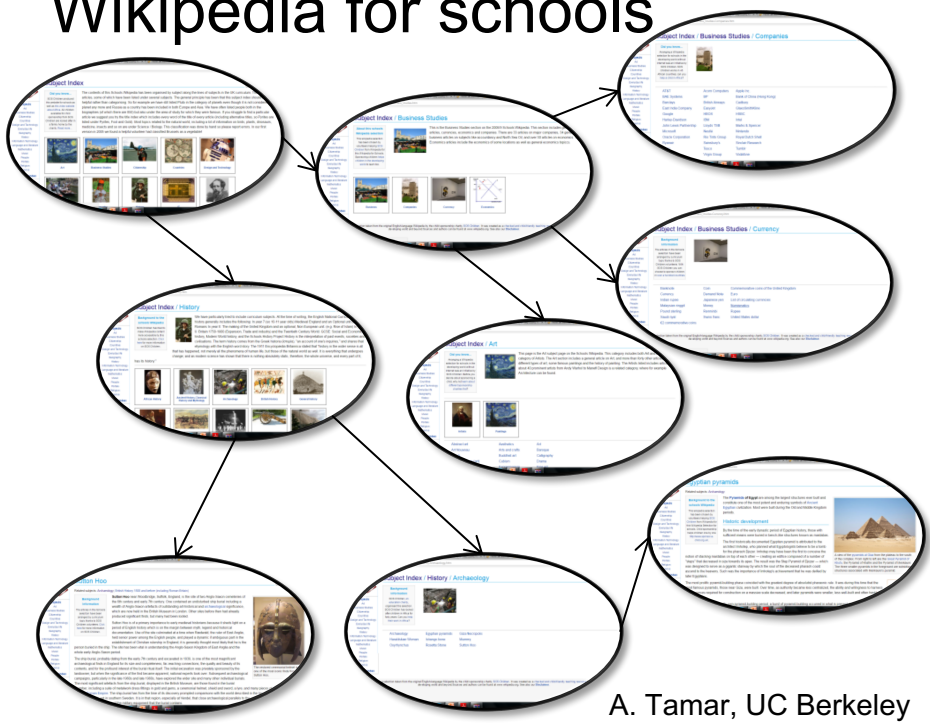


Plan on Approximate Graph

- Navigate website links to find query (Nogueira & Cho, NIPS 2016)
- Plan on approx. graph (1st+2nd level categories)
- Learn feature similarity mappings

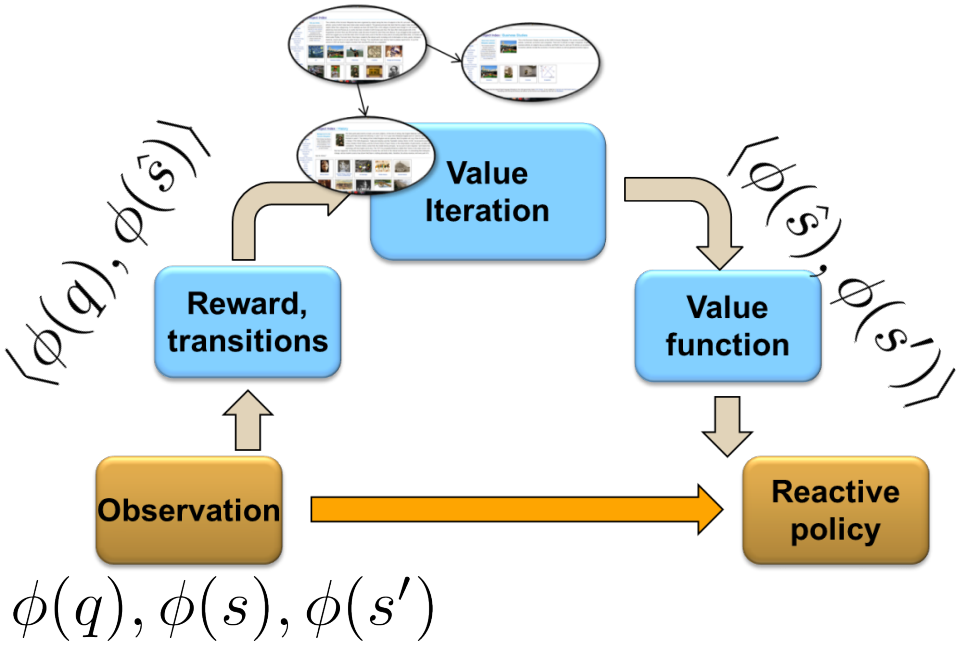


Wikipedia for schools

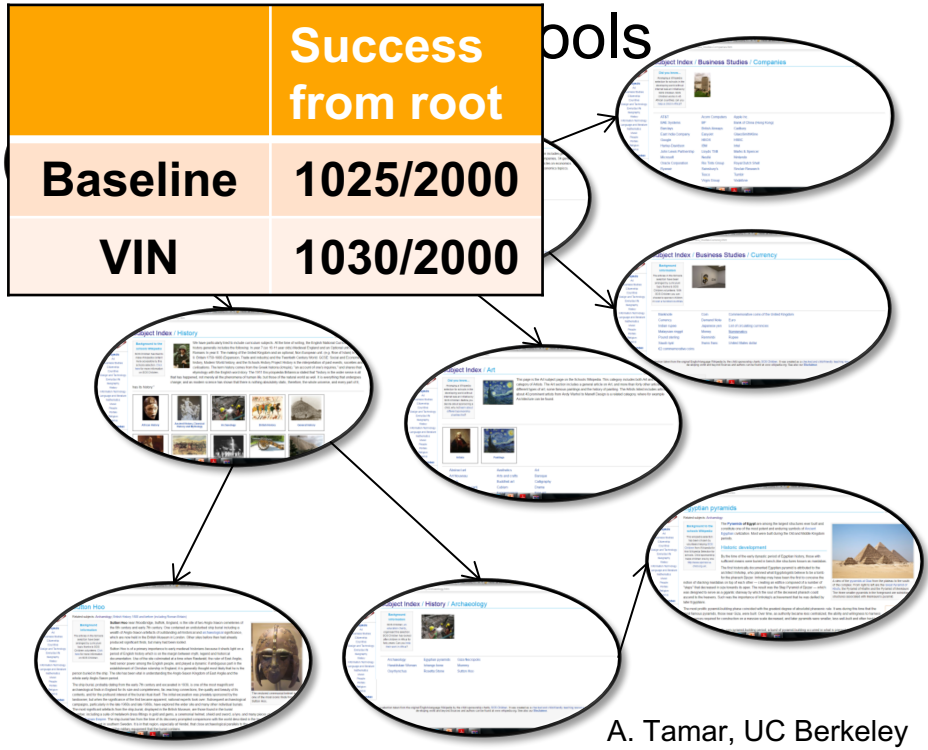


Plan on Approximate Graph

- Navigate website links to find query (Nogueira & Cho, NIPS 2016)
- Plan on approx. graph (1st+2nd level categories)
- Learn feature similarity mappings

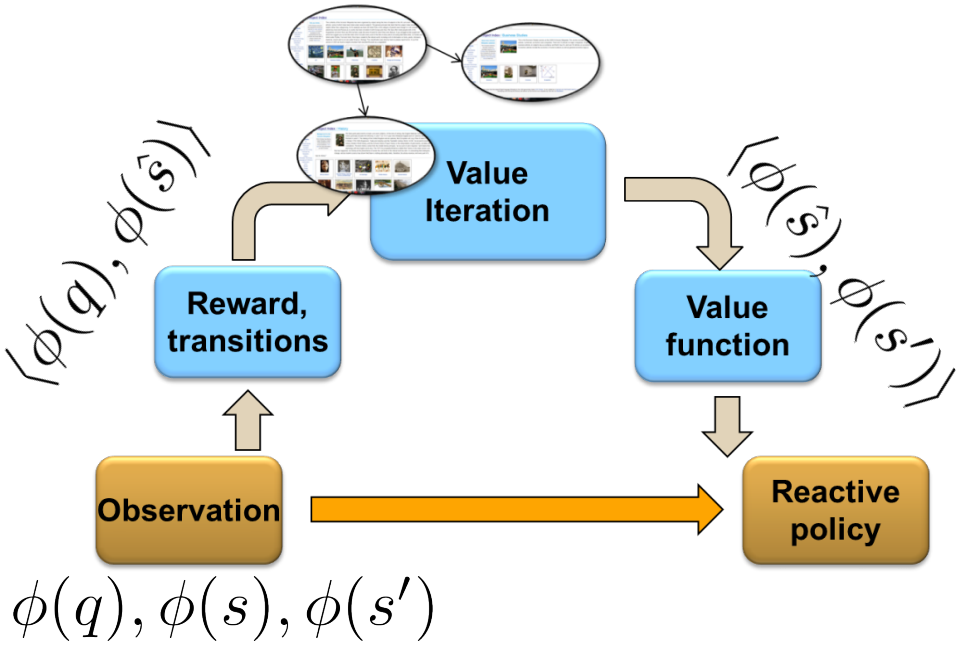


	Success from root
Baseline	1025/2000
VIN	1030/2000

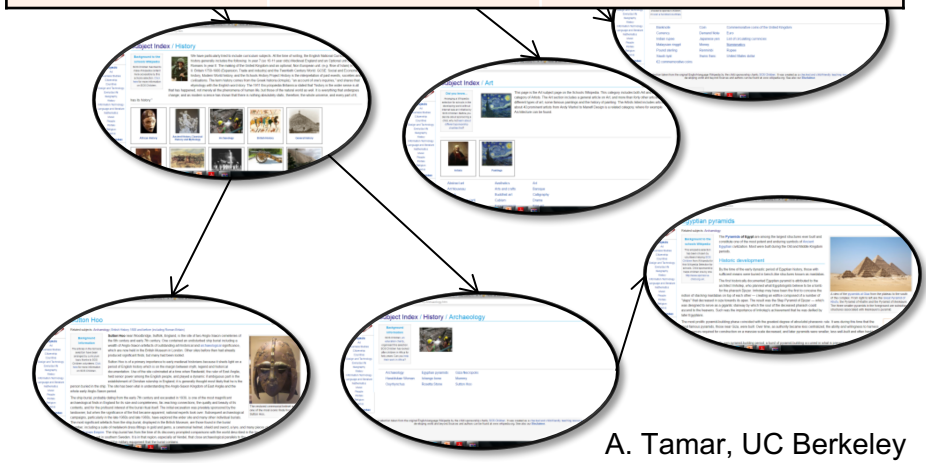


Plan on Approximate Graph

- Navigate website links to find query (Nogueira & Cho, NIPS 2016)
- Plan on approx. graph (1st+2nd level categories)
- Learn feature similarity mappings

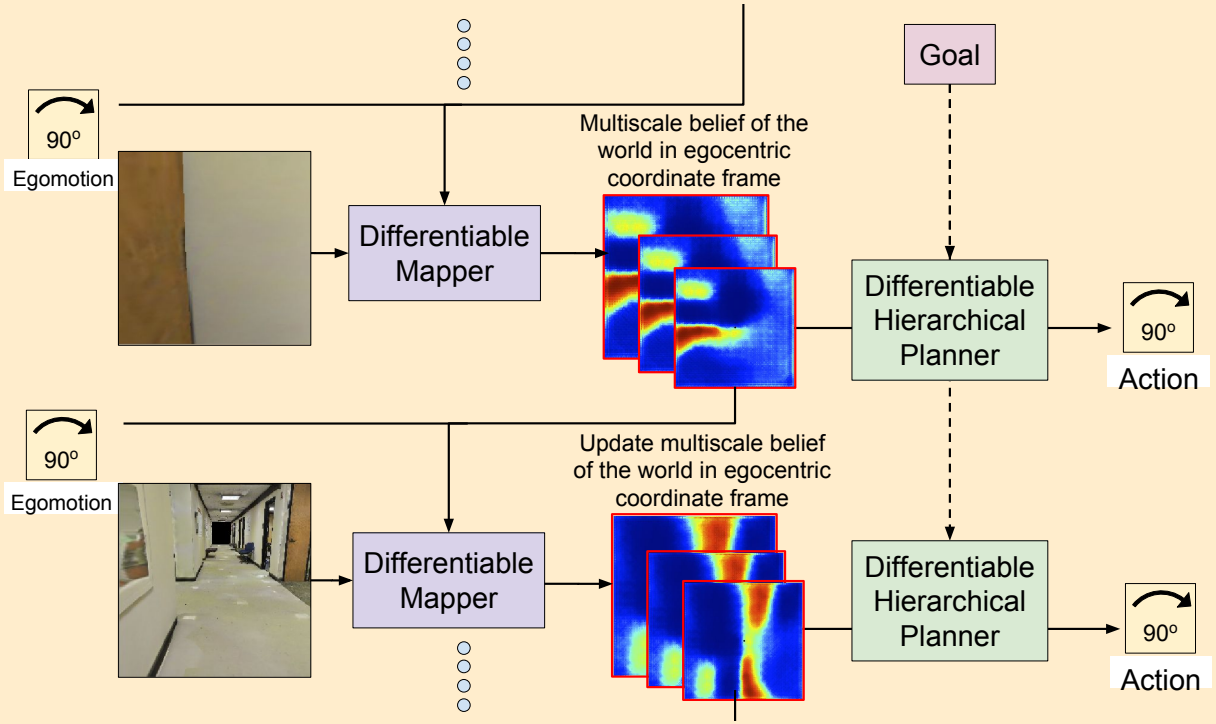


	Success from root	Success from random
Baseline	1025/2000	304/4000
VIN	1030/2000	346/4000



Extensions

- Follow-up work (independent)
  - 1st person navigation



Cognitive Mapping and Planning for Visual Navigation,
S. Gupta, J. Davidson, S. Levine, R. Sukthankar, J. Malik, arXiv 2017

Extensions

Cognitive Mapping and Planning for Visual Navigation,
S. Gupta, J. Davidson, S. Levine, R. Sukthankar, J. Malik, arXiv 2017

Summary – Value Iteration Networks

- Learn to plan → generalization
- VIN framework
  - Motivated by planning theory
  - Differentiable planner (VI = convnet)
  - Easy to compose with perception/control
- Challenges
  - Planning in high-dim spaces

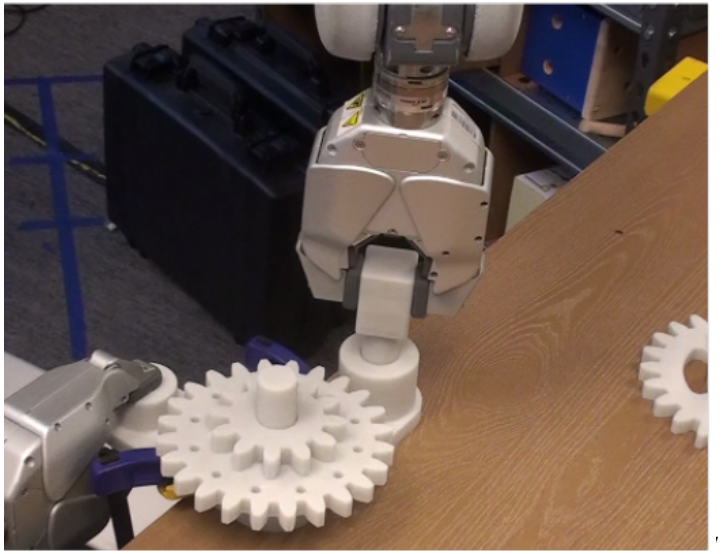
Part 3:

Hindsight MPC

T., Thomas, Zhang, Levine, Abbeel. ICRA 2017

Motivation

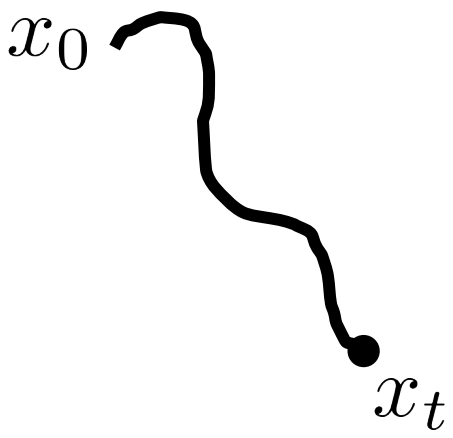
- Robotic manipulation
- Accurate dynamics model not available
- MPC: well-established control method
- Recently: adaptive MPC for manipulation [1,2]



[1] Fu, Levine, Abbeel. IROS 2016
[2] Lenz, Knepper, Saxena. RSS 2015

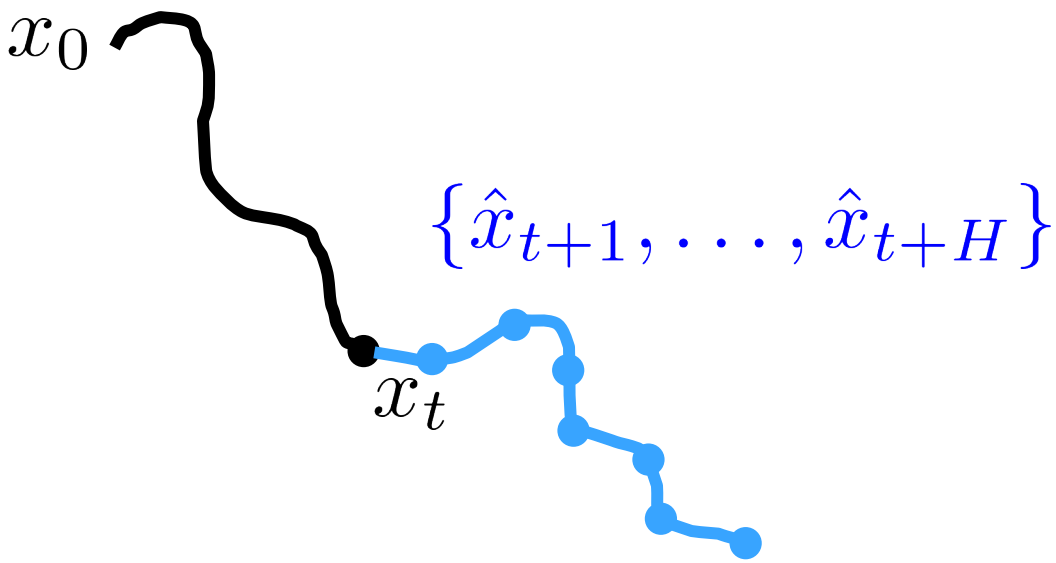
Background - Adaptive MPC

At time t:



Background - Adaptive MPC

Predict H-step trajectory:

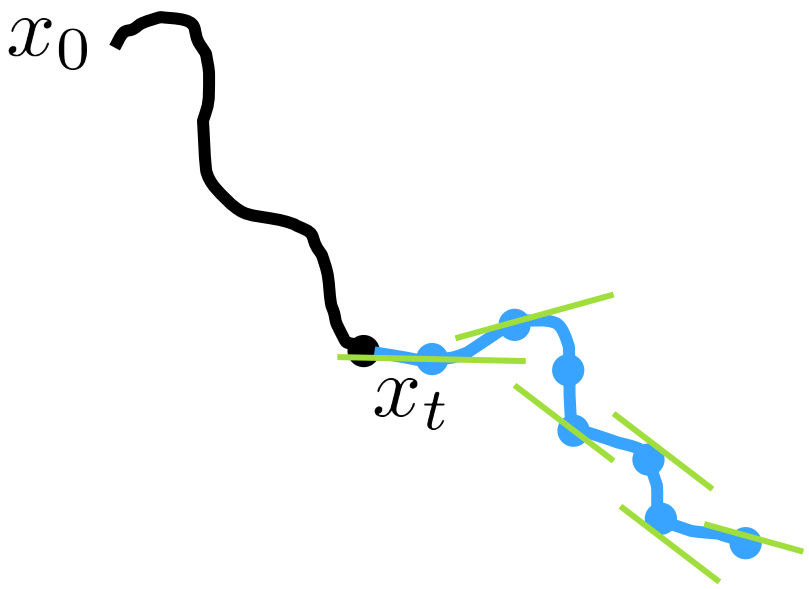


Using current dynamics estimate & policy:

$$\hat{x}_{s+1} = \hat{f}^t(\hat{x}_s, \pi^t(\hat{x}_s))$$

Background - Adaptive MPC

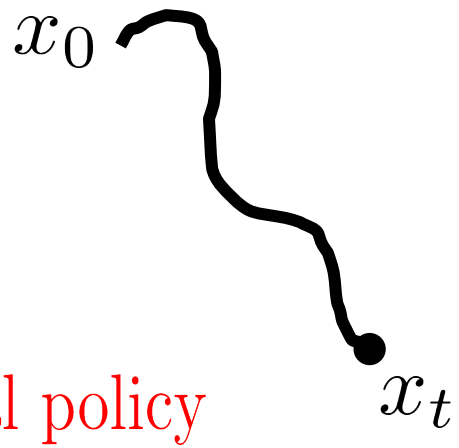
Linearize dynamics:



$$\hat{f}_s(x, u) \approx A_s x + B_s u$$

Background - Adaptive MPC

Solve linear quadratic regulator (LQR):



H-horizon optimal policy

$$\arg\min_{u_t,\ldots,u_{t+H}} \sum_{s=t}^{t+H} \ell(x_s, u_s)$$

$$\text{s.t.} \quad x_{s+1} = A_s x_s + B_s u_s$$

Background - Adaptive MPC

Solve linear quadratic regulator (LQR):

$$\arg\min_{u_t,\ldots,u_{t+H}} \sum_{s=t}^{t+H} \ell(x_s, u_s)$$

linear

quadratic

$$\text{s.t.} \quad x_{s+1} = A_s x_s + B_s u_s$$

$$\ell(x_s, u_s) = x_s^T Q x_s + u_s^T R u_s$$

Dynamic programming solution:

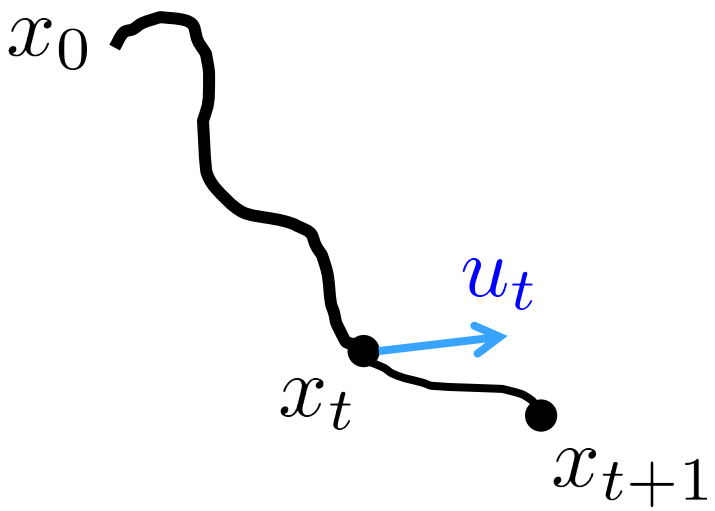
$$u_s = -F_s x_s$$

$$F_s = (R + B^T P_s B)^{-1}(B^T P_{s+1} A)$$

$$P_{s-1} = A^T P_s A - (A^T P_s B)(R + B^T P_s B)^{-1}(B^T P_{s+1} A) + Q$$

Background - Adaptive MPC

Take action u_t



Update dynamics estimate

Repeat...

Adaptive MPC

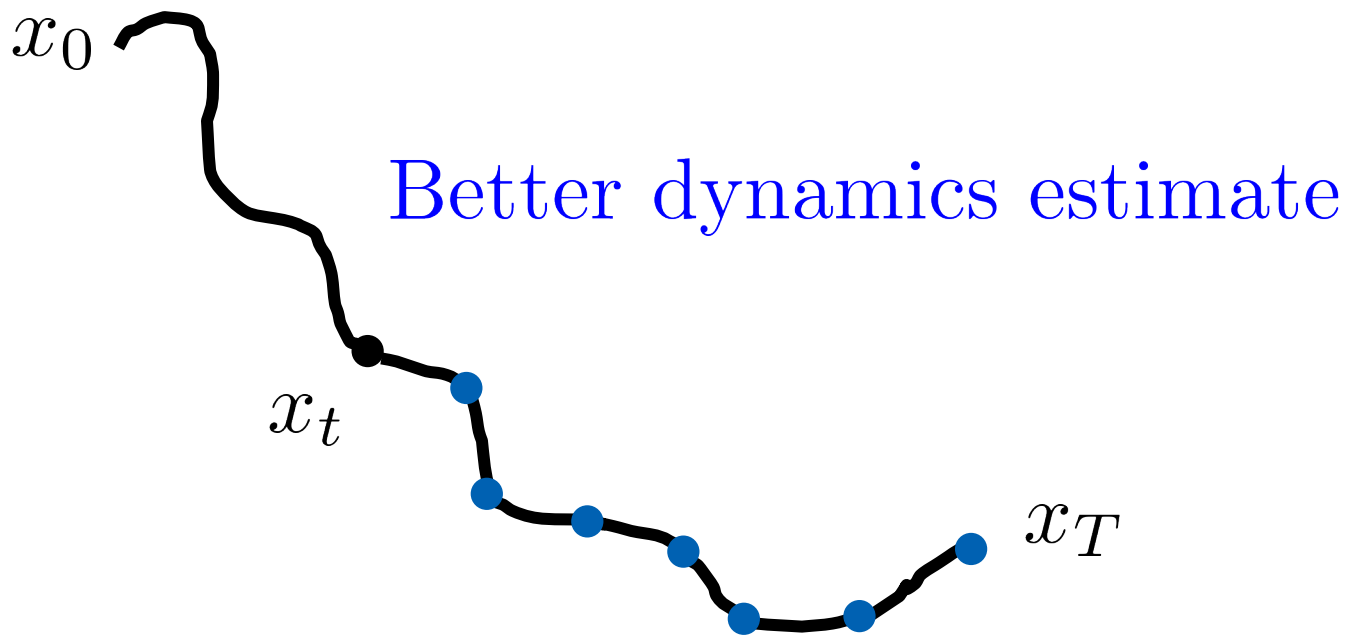
- Sub-optimality:
  - Prediction (dynamics) error
  - Finite horizon
- Typically **H << T**:
  - Mitigate model errors
  - Computational feasibility

Question: how to **improve** in a repeated task?

Idea: the Hindsight Plan

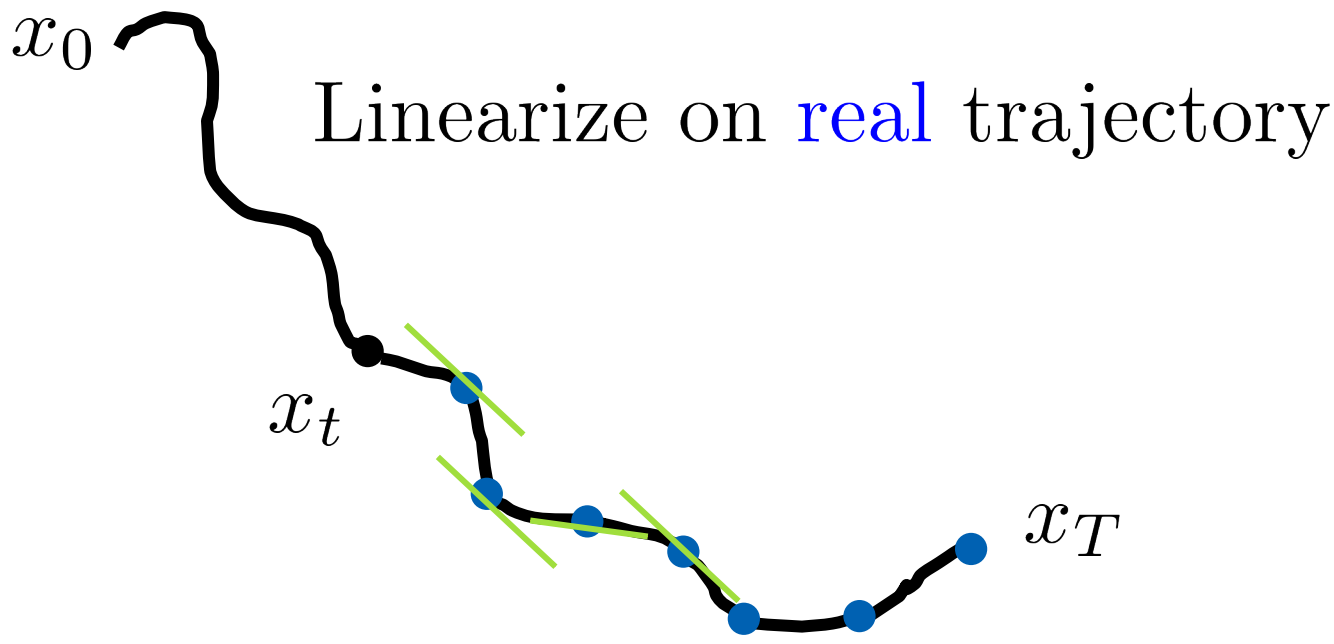
Hindsight MPC

Revisit time t (offline):



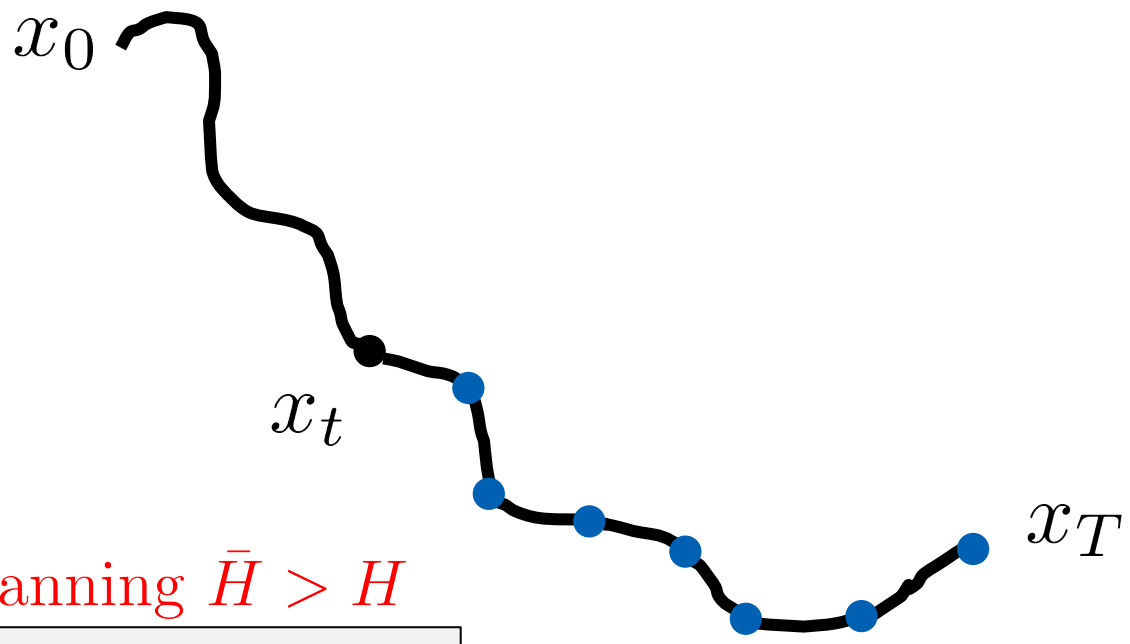
Hindsight MPC

Revisit time t (offline):



Hindsight MPC

Revisit time t (offline):

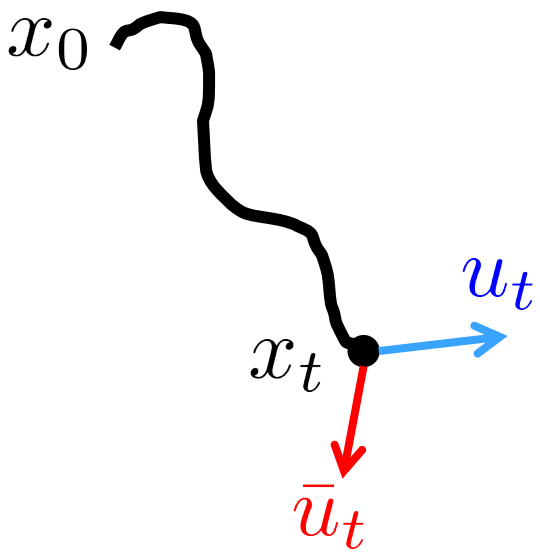


Longer horizon planning $\bar{H} > H$

$$\arg\min_{u_t,\ldots,u_{t+\bar{H}}} \sum_{s=t}^{t+\bar{H}} \ell(x_s, u_s)$$

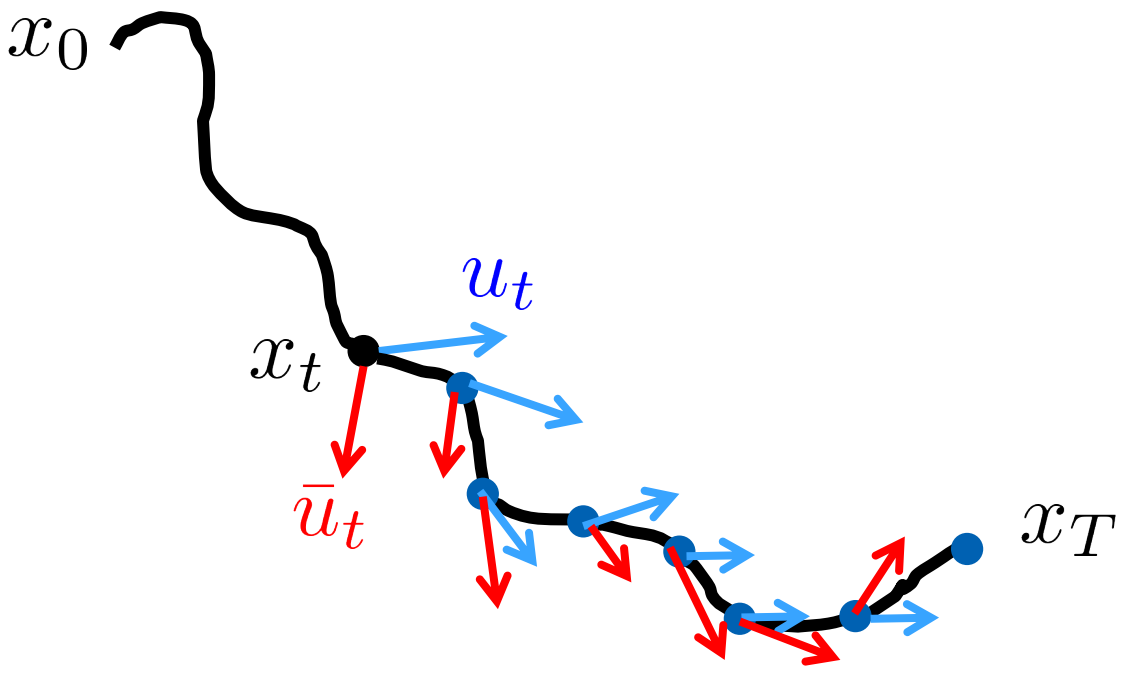
Adaptive MPC

Hindsight action $\bar{u}_t$



Adaptive MPC

Hindsight action $\bar{u}_t$



Hindsight MPC

- MPC Sub-optimality:
  - ✓ Prediction (dynamics) error
  - ✓ Finite horizon
- Hindsight actions are better!
  - How to improve (online) MPC?

Change cost such that online MPC would mimic the hindsight actions!

Hindsight MPC

- Original LQR cost

goal

$$\sum_{s=t}^{t+H}(\mathbf{x}_s - \mathbf{x}^*)^\top Q_s(\mathbf{x}_s - \mathbf{x}^*) + \mathbf{u}_s^\top R_s \mathbf{u}_s$$

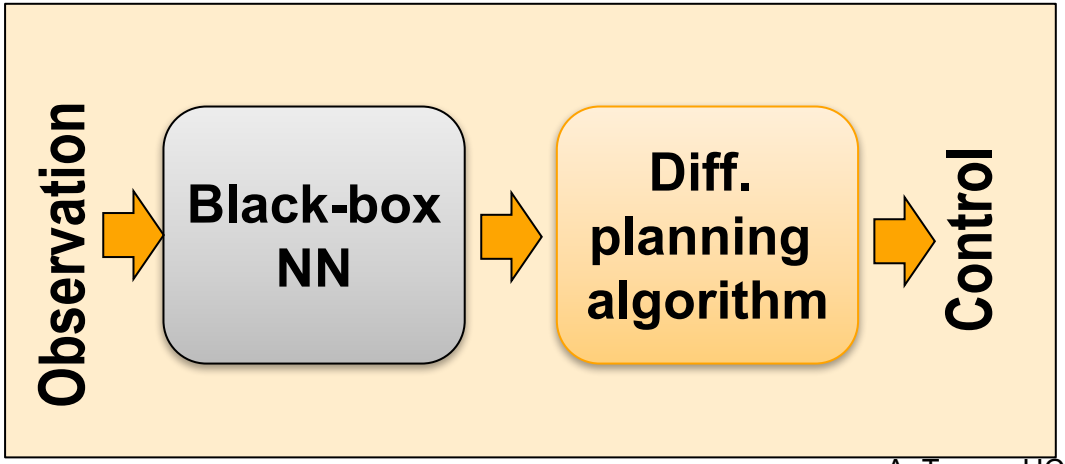
- Shaped LQR cost

shaped goal

$$\sum_{s=t}^{t+H}(\mathbf{x}_s - \mathbf{x}^*(\mathbf{x}_t,\theta))^\top Q_s(\mathbf{x}_s - \mathbf{x}^*(\mathbf{x}_t,\theta)) + \mathbf{u}_s^\top R_s \mathbf{u}_s$$

$$\mathbf{x}^*(\mathbf{x}_t,\theta) = \mathbf{x}^* + \mathrm{NN}(\mathbf{x}_t,\theta)$$

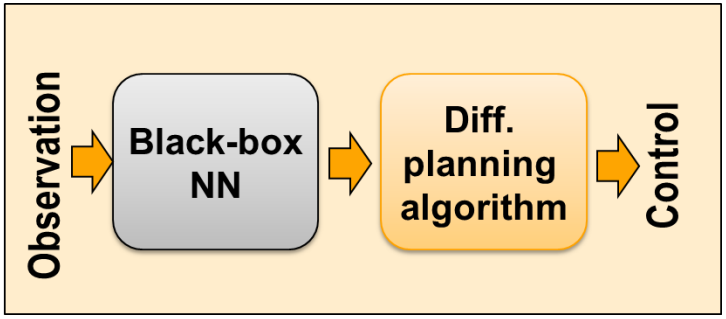
Intuition: learn way points!



Hindsight MPC

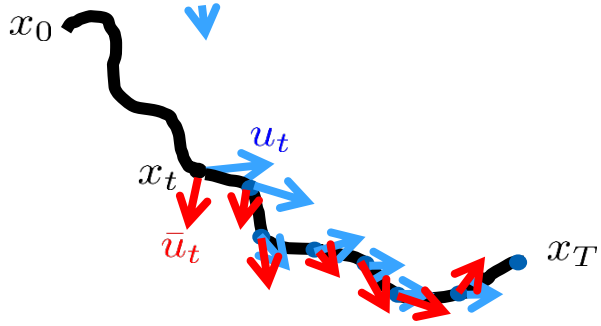
- Learning problem:

$$\text{Minimize} \quad \mathcal{L}(\theta) = \sum_{t=0}^{T} \|\mathbf{u}_t(\theta) - \bar{\mathbf{u}}_t\|^2$$



$\mathbf{u}_t(\theta)$

Hindsight actions



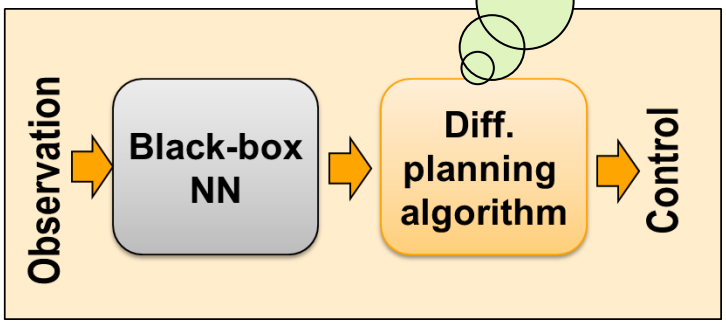
Make the online MPC mimic the hindsight plan!

Hindsight MPC

- Learn

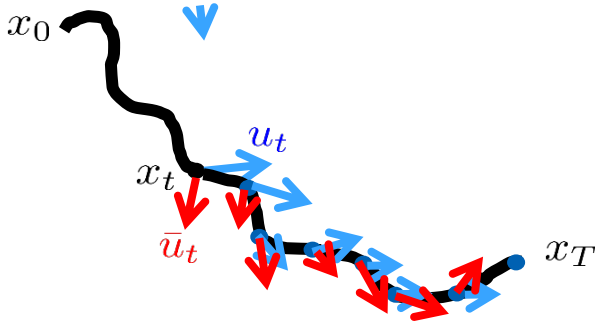
$\frac{\partial \mathbf{u}_t}{\partial \theta}$: **diff. through dynamic programming LQR calculation**

$\mathbf{u}_t(\theta)$



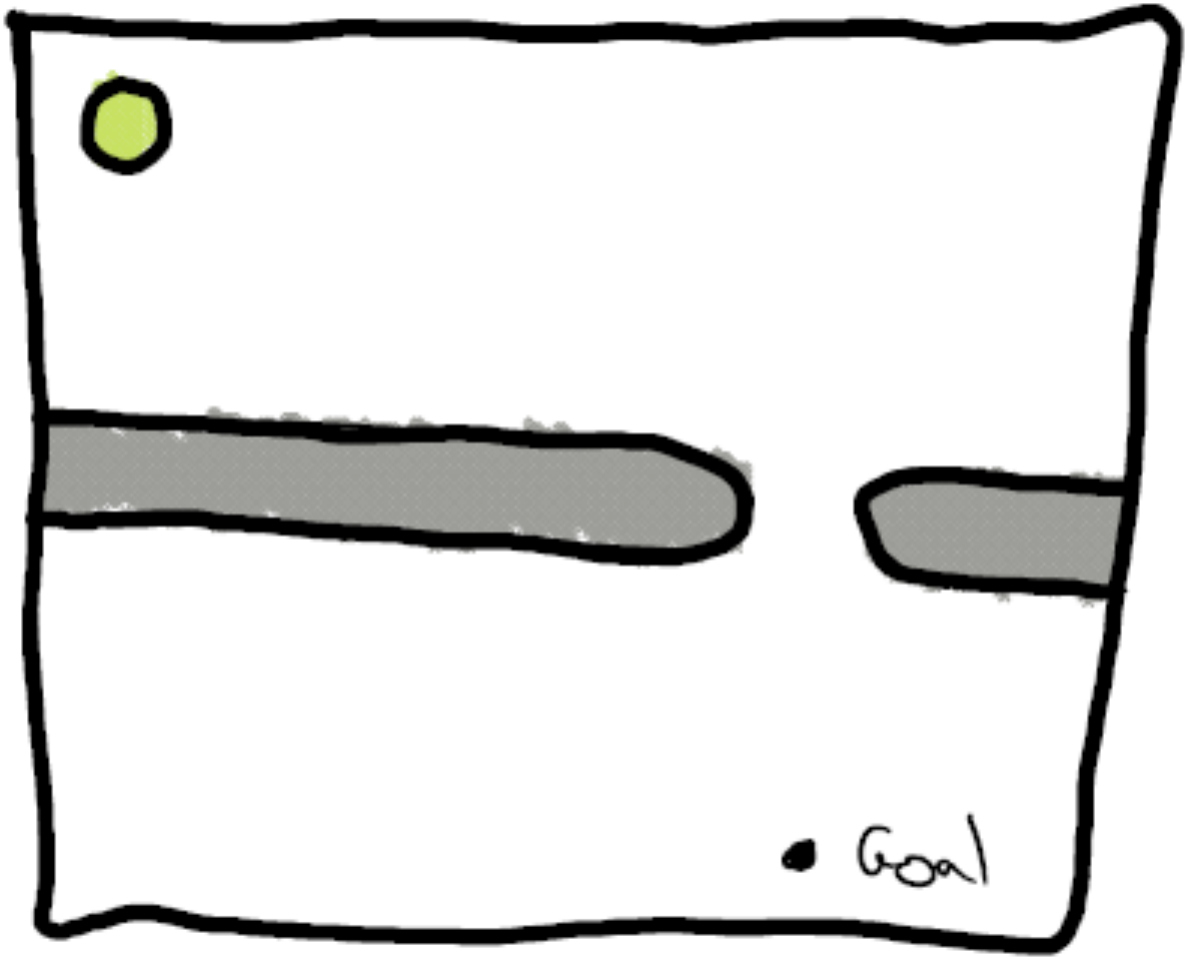
$\| \cdot \|^2$

Hindsight actions

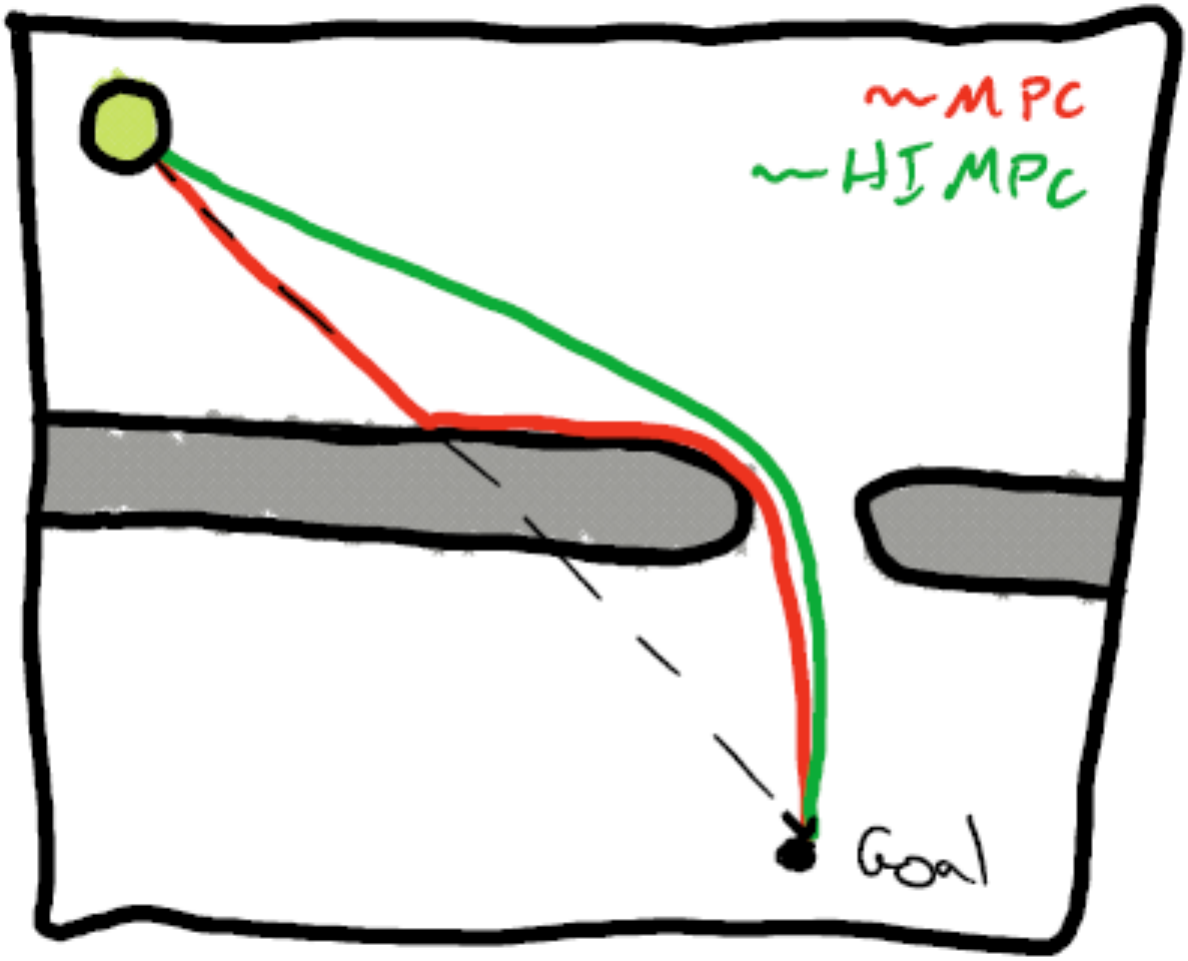


Make the online MPC mimic the hindsight plan!

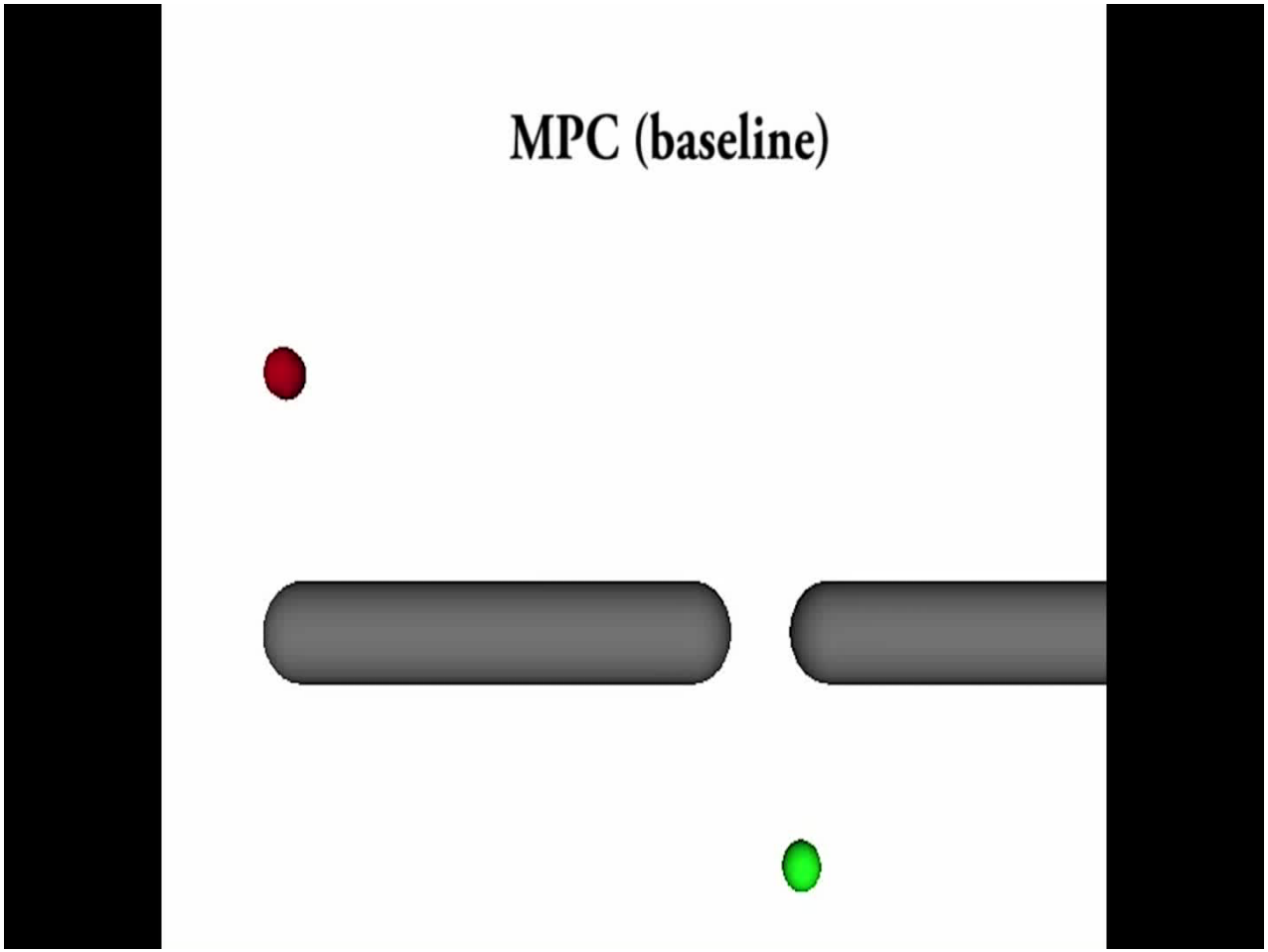
Illustrative Example



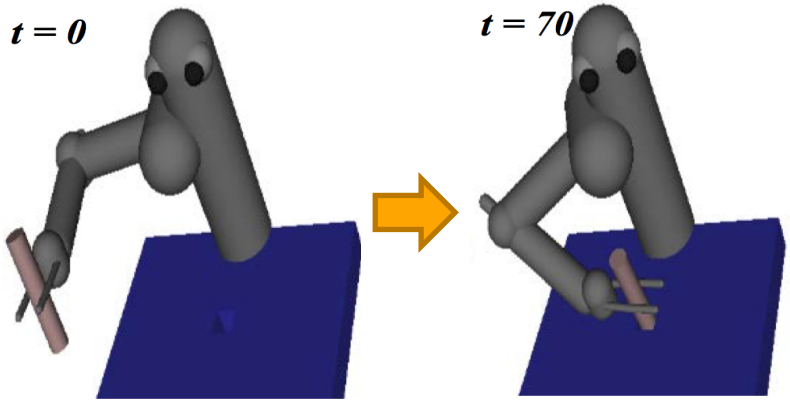
Illustrative Example

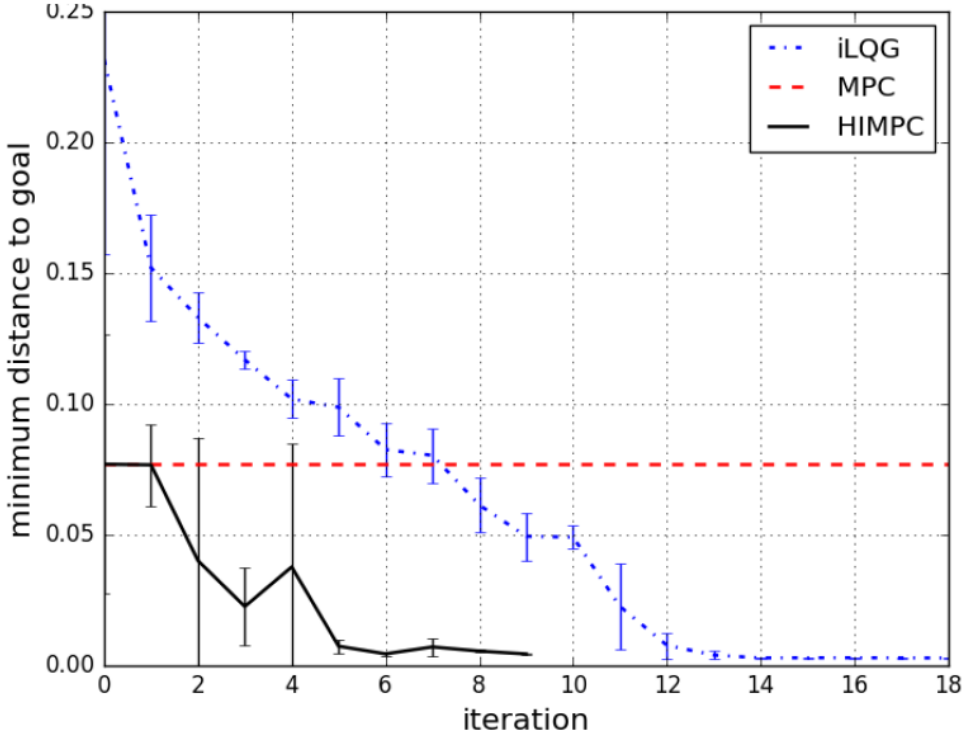


Illustrative Example



Results – Peg Insertion





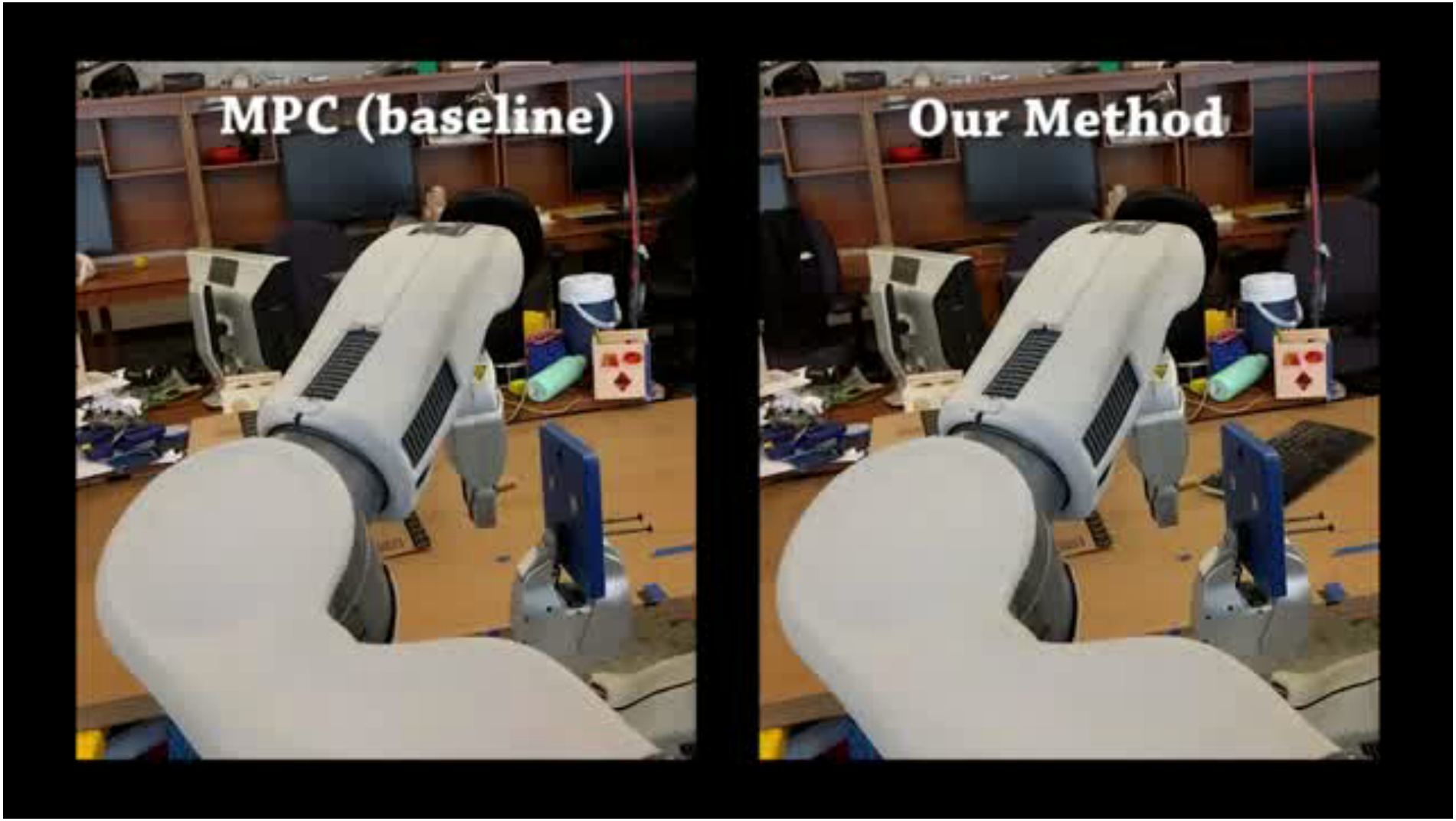
State space: 26D

- 7 DoF arm (7 angles + 7 ang. velocities)
- 2 end-effector points (6 positions + 6 velocities)

Dynamics: GMM

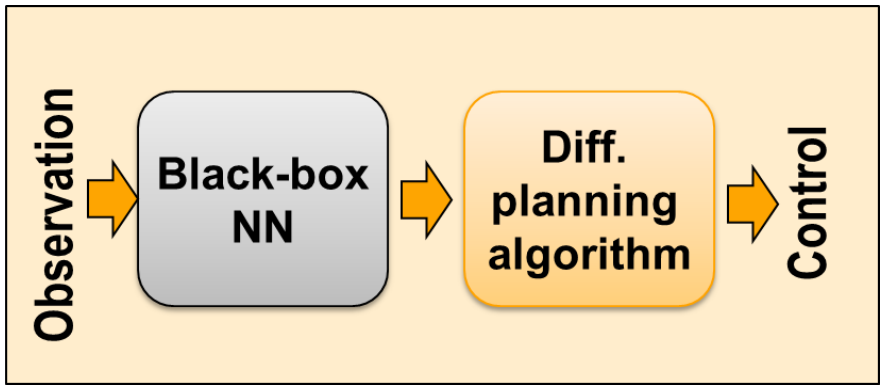
- Prior – free space dynamics
- Online adaptation

Results – PR2 Peg Insertion



Conclusion

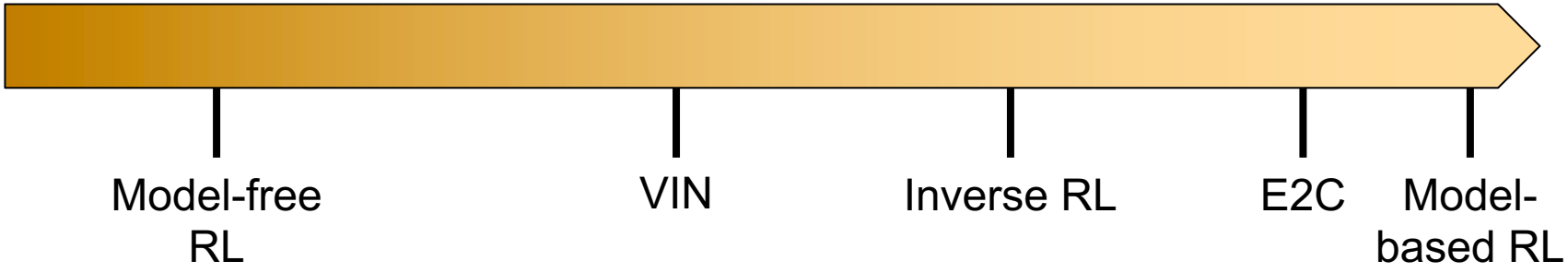
- Decades of research on planning/control
- Control algorithm → inductive bias



- Principled approach to network design
- Better generalization (change in task)
- New algorithmic ideas

Conclusion

- Inductive bias



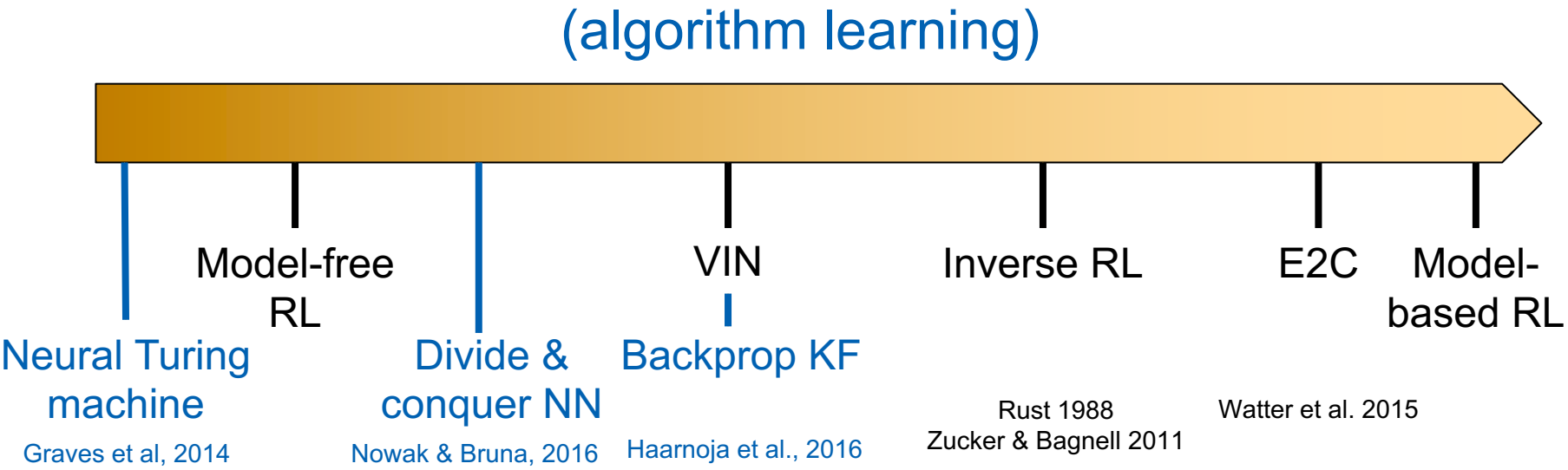
Mnih et al, 2015
Oh et al, 2016

Rust 1988
Zucker & Bagnell 2011

Watter et al. 2015

Conclusion

- Inductive bias



Thanks

- Collaborators
  - Pieter Abbeel (UCB), Yinlam Chow (Stanford), Mohammad Ghavamzadeh (Adobe), Yonatan Glassner (Technion), Sergey Levine (UCB), Shie Mannor (Technion), Marco Pavone (Stanford), Garrett Thomas (UCB), Yi Wu (UCB), Tianhao Zhang (UCB)
- Funding
  - Siemens
  - Viterbi Scholarship

Berkeley
Artificial Intelligence Research Laboratory