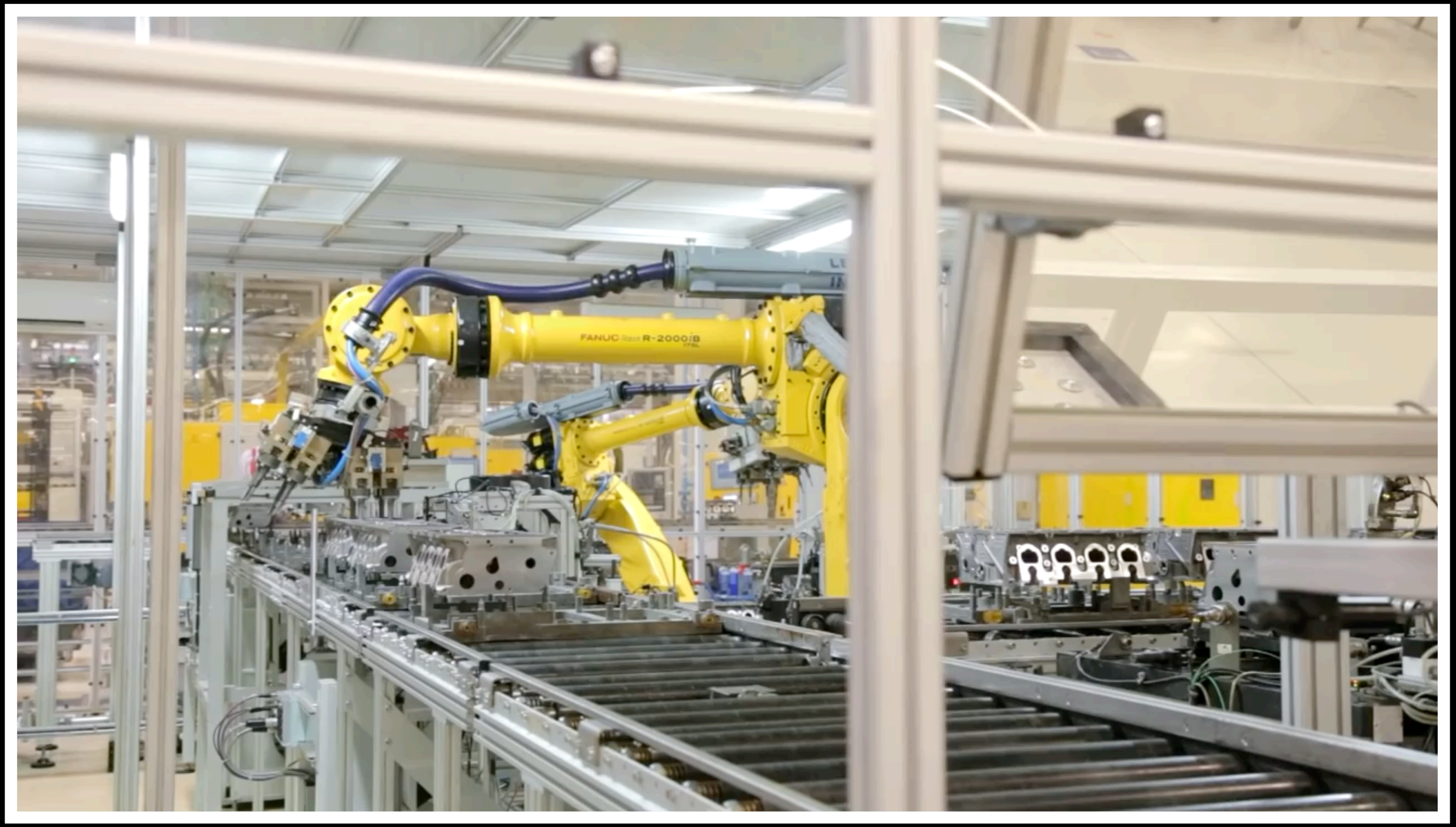
Inaccurate Models and How to Use Them

Anirudh Vemula

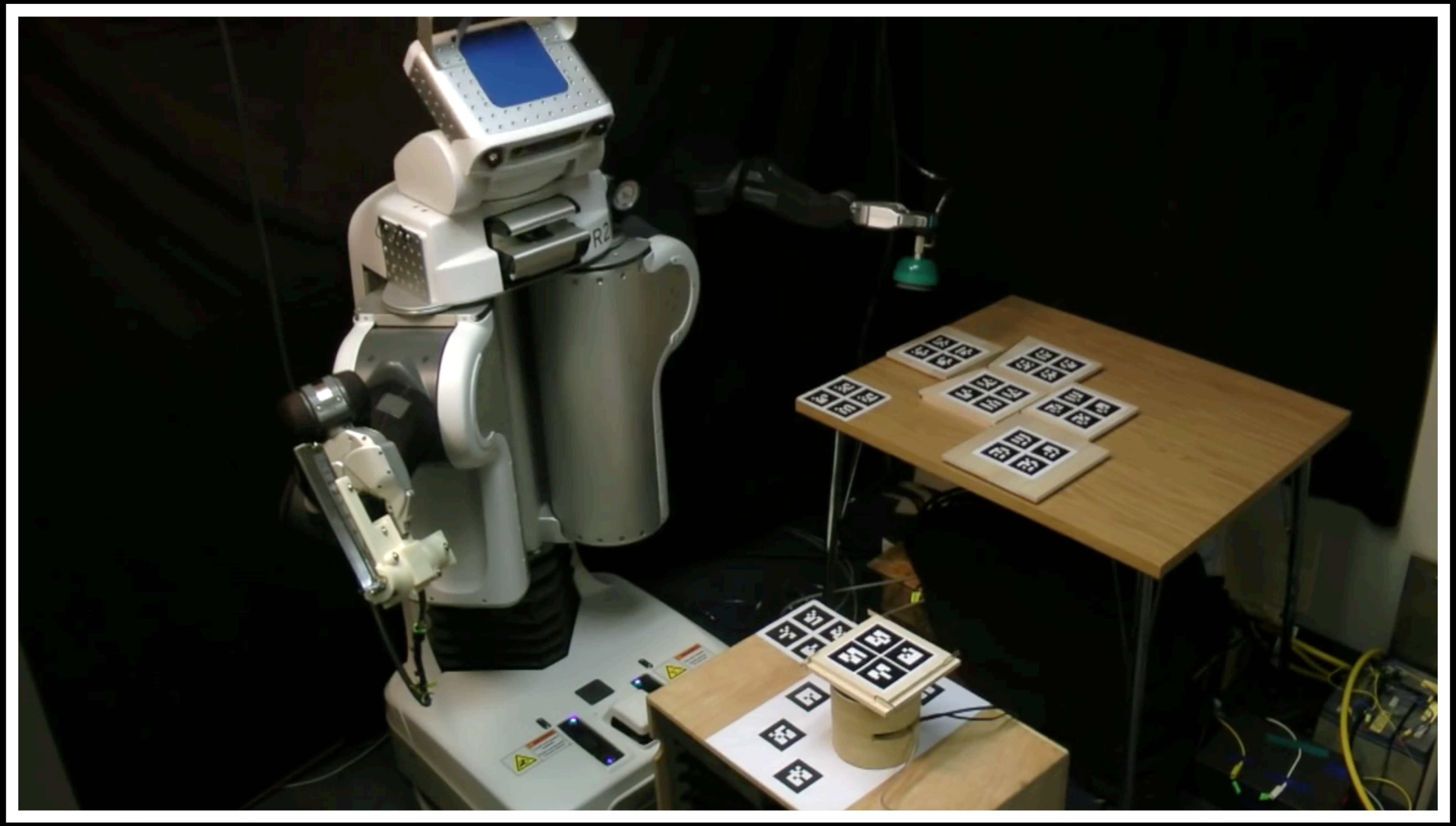
Thesis Defense

<u>Committee</u>:
Max Likhachev (Co-Chair)
Drew Bagnell (Co-Chair)
Oliver Kroemer
Leslie Kaelbling (MIT)

Planning in Structured Environments



Video from FANUC robotics

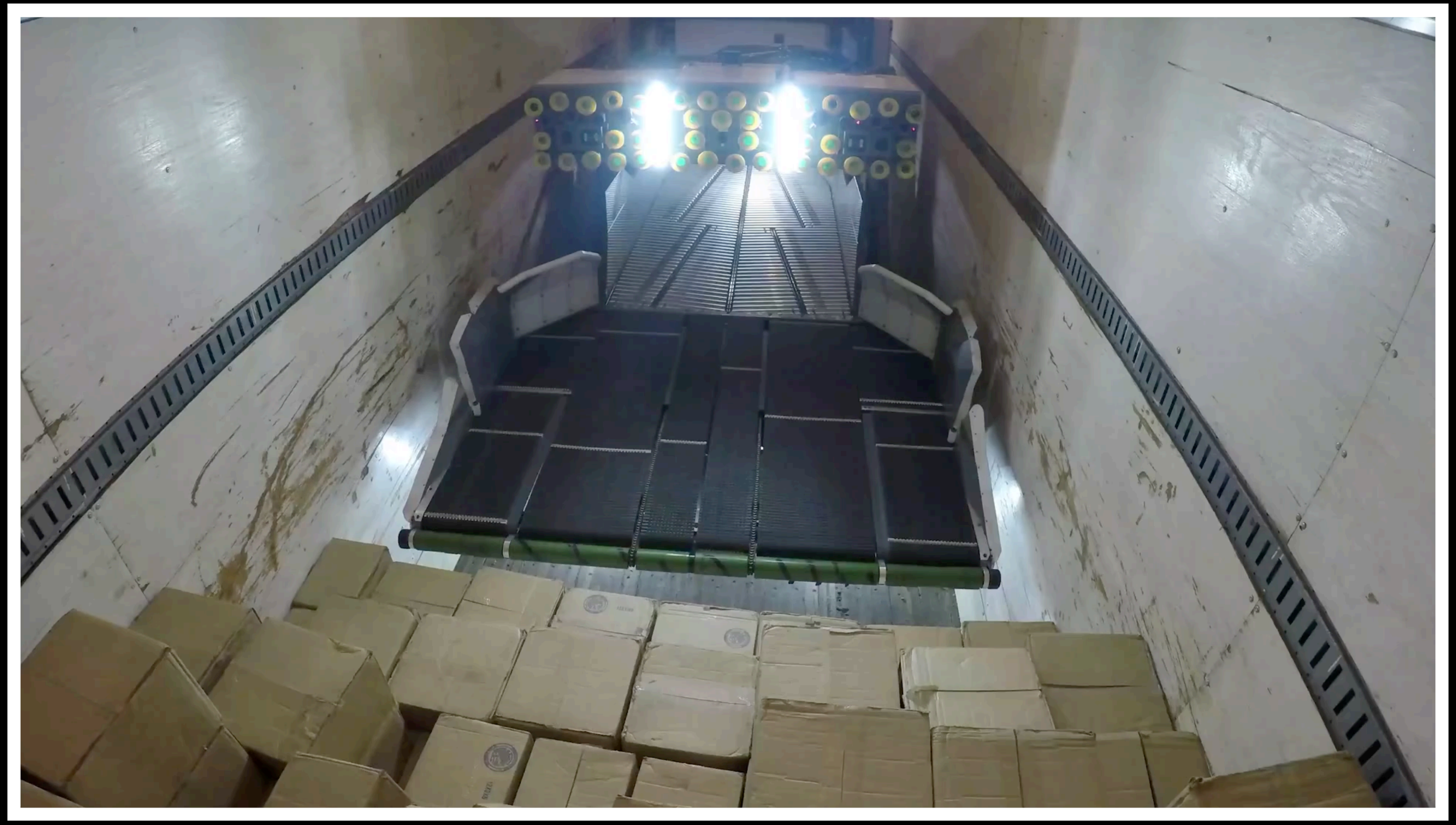


Video courtesy of SBPL

Access to accurate models of the robot and environment dynamics

But in unstructured environments, our models are almost always inaccurate

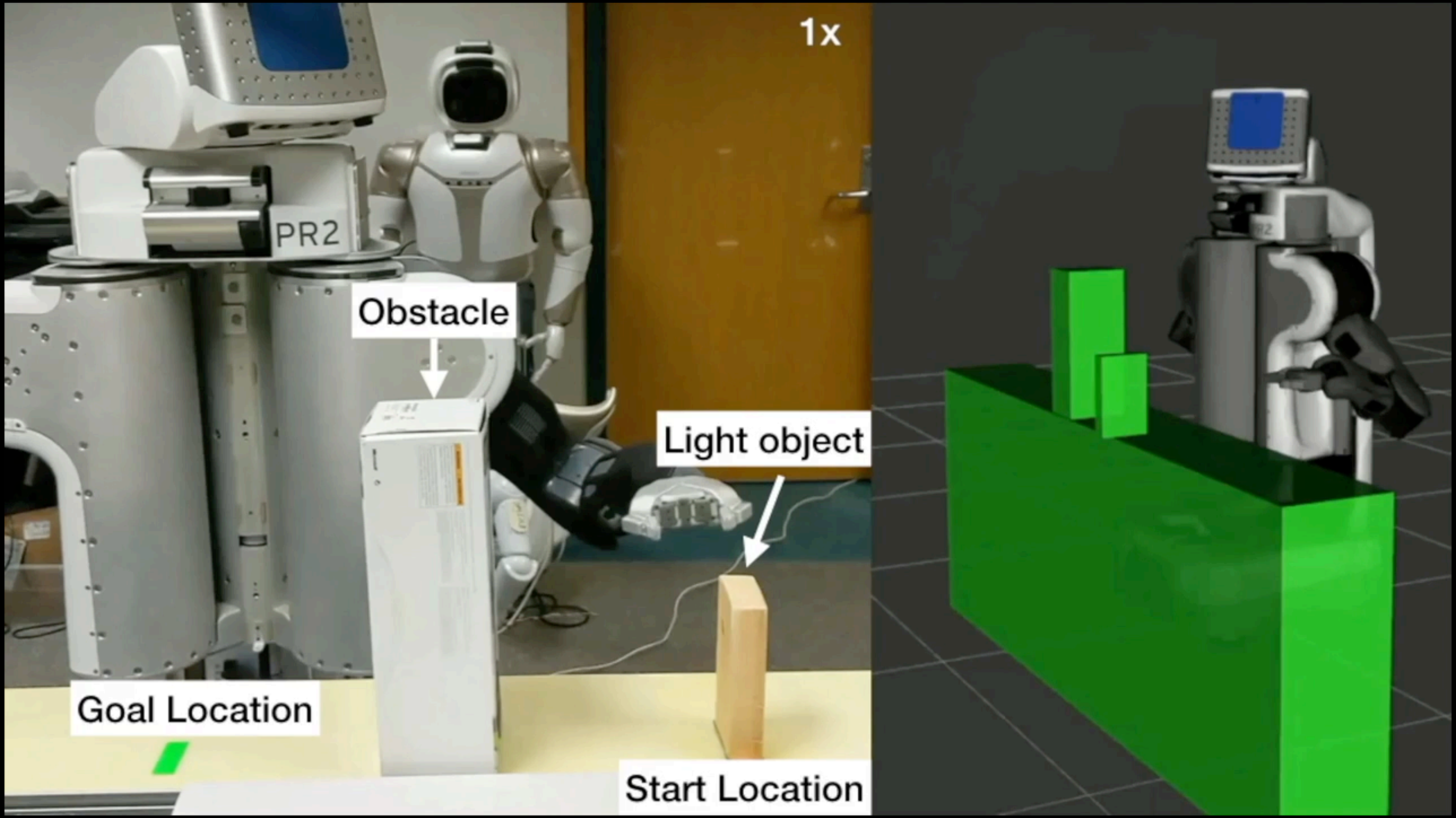
Video from [Miki et. al. 2022]



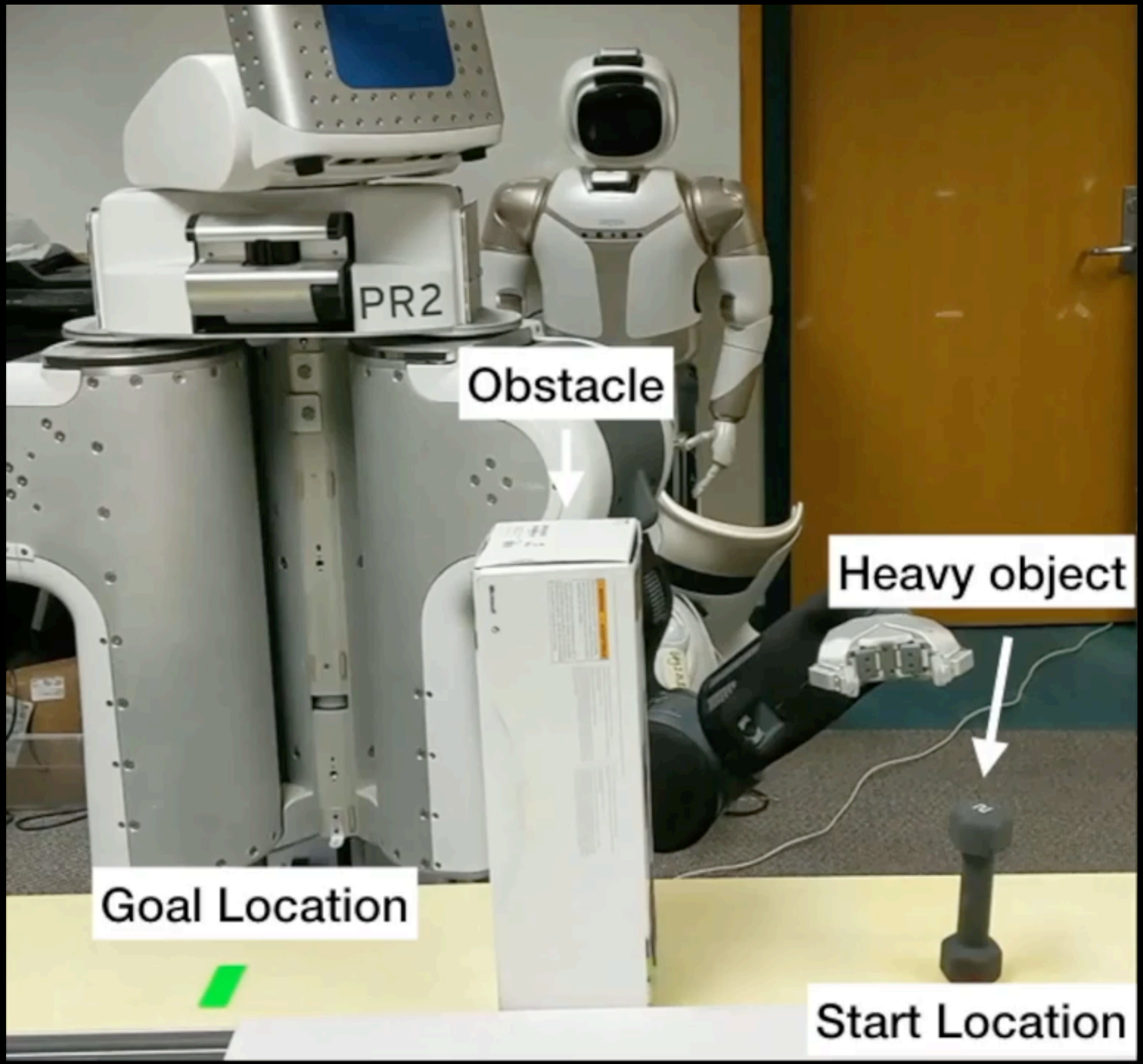
Video courtesy of Honeywell

Can we naively use inaccurate models and complete the task?

Motivating Example



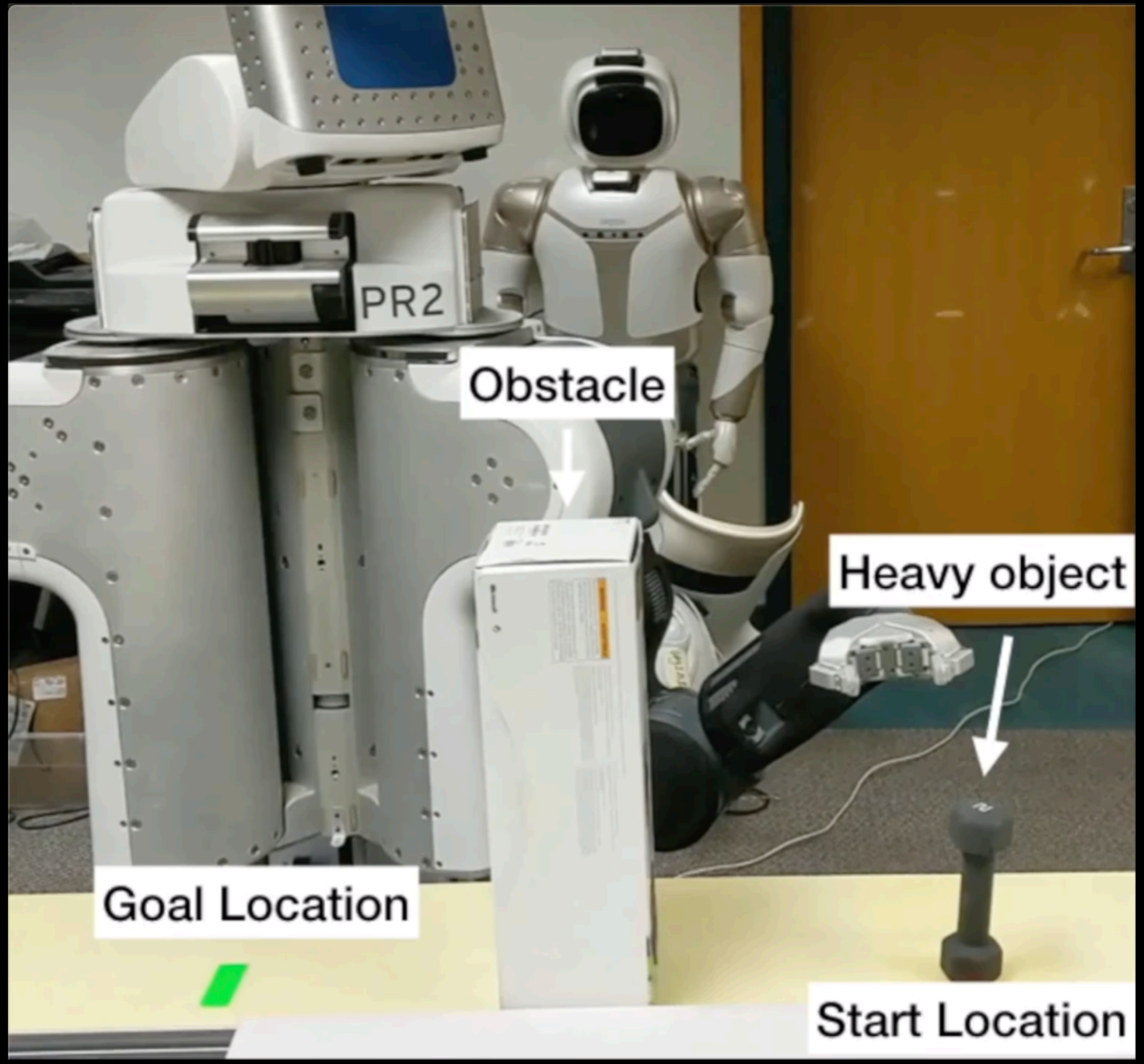
Naively Using Inaccurate Model Leads to Failure



We reach joint torque limits and cannot execute the same motion plan

The object is still modeled as light. Robot is stuck!

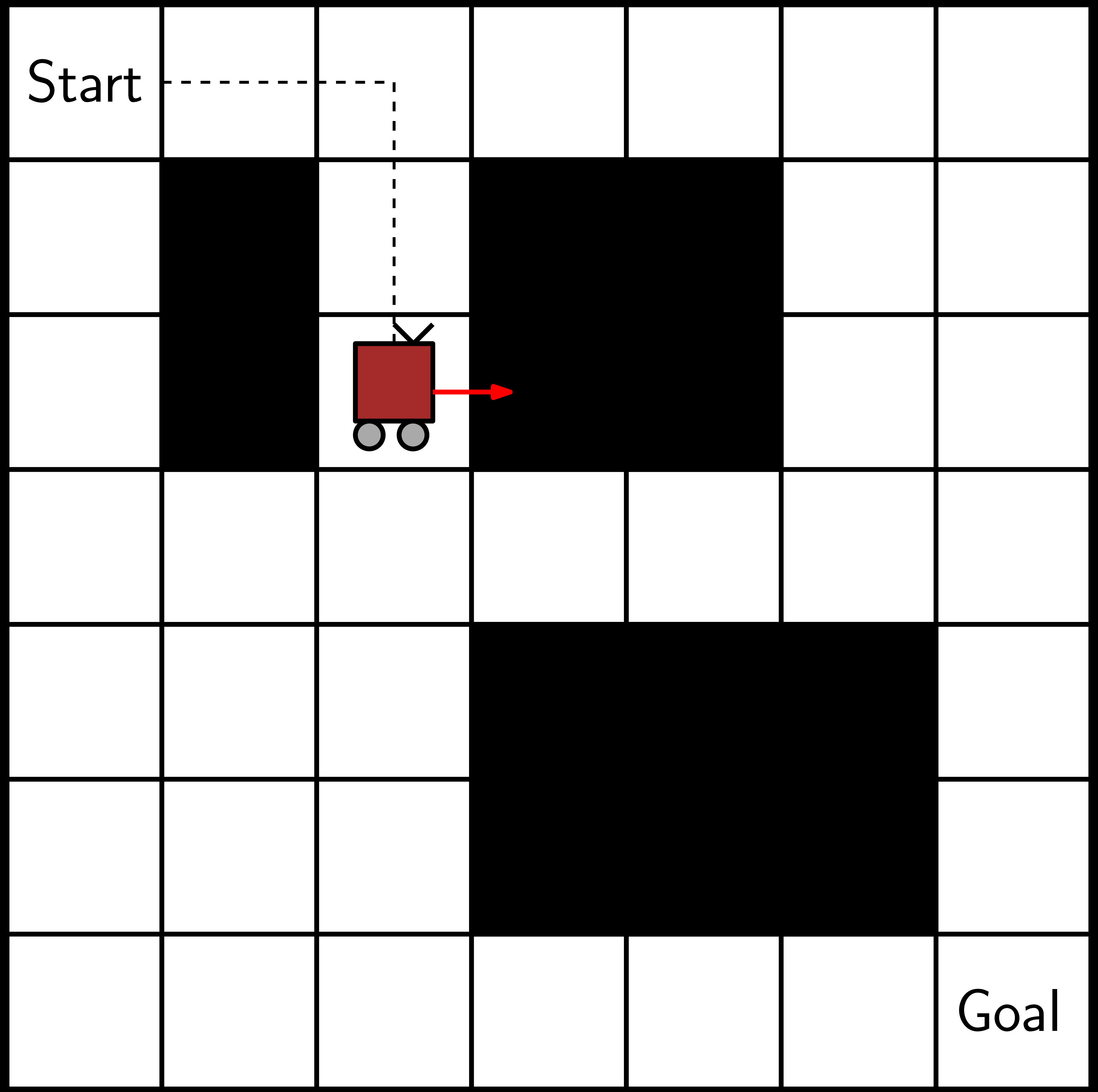
Objective



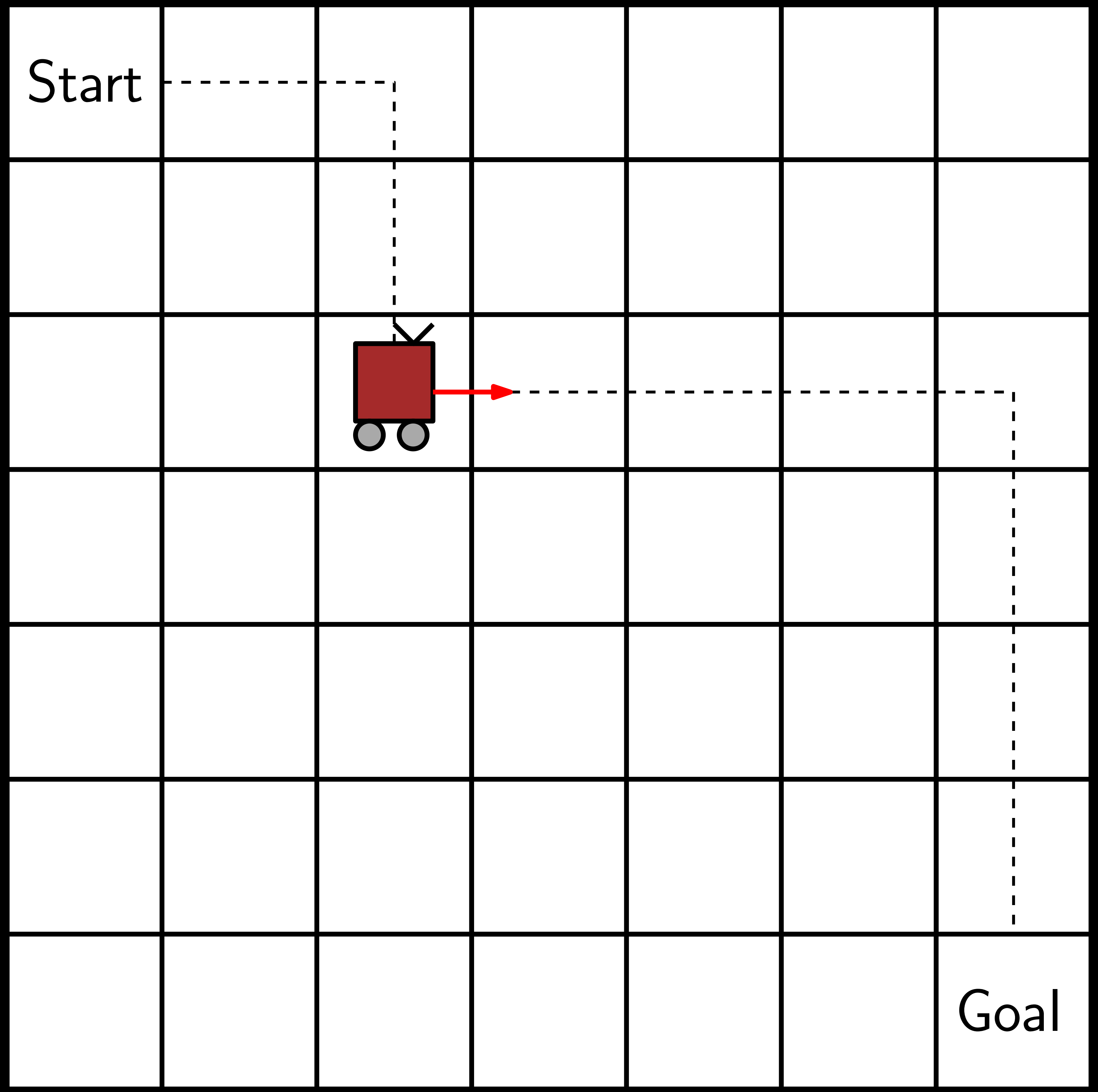
Provably reach the goal online, despite having an inaccurate dynamical model, without any resets

*Resets allow the robot to “reset” to a state, usually a previously visited state

Running Example

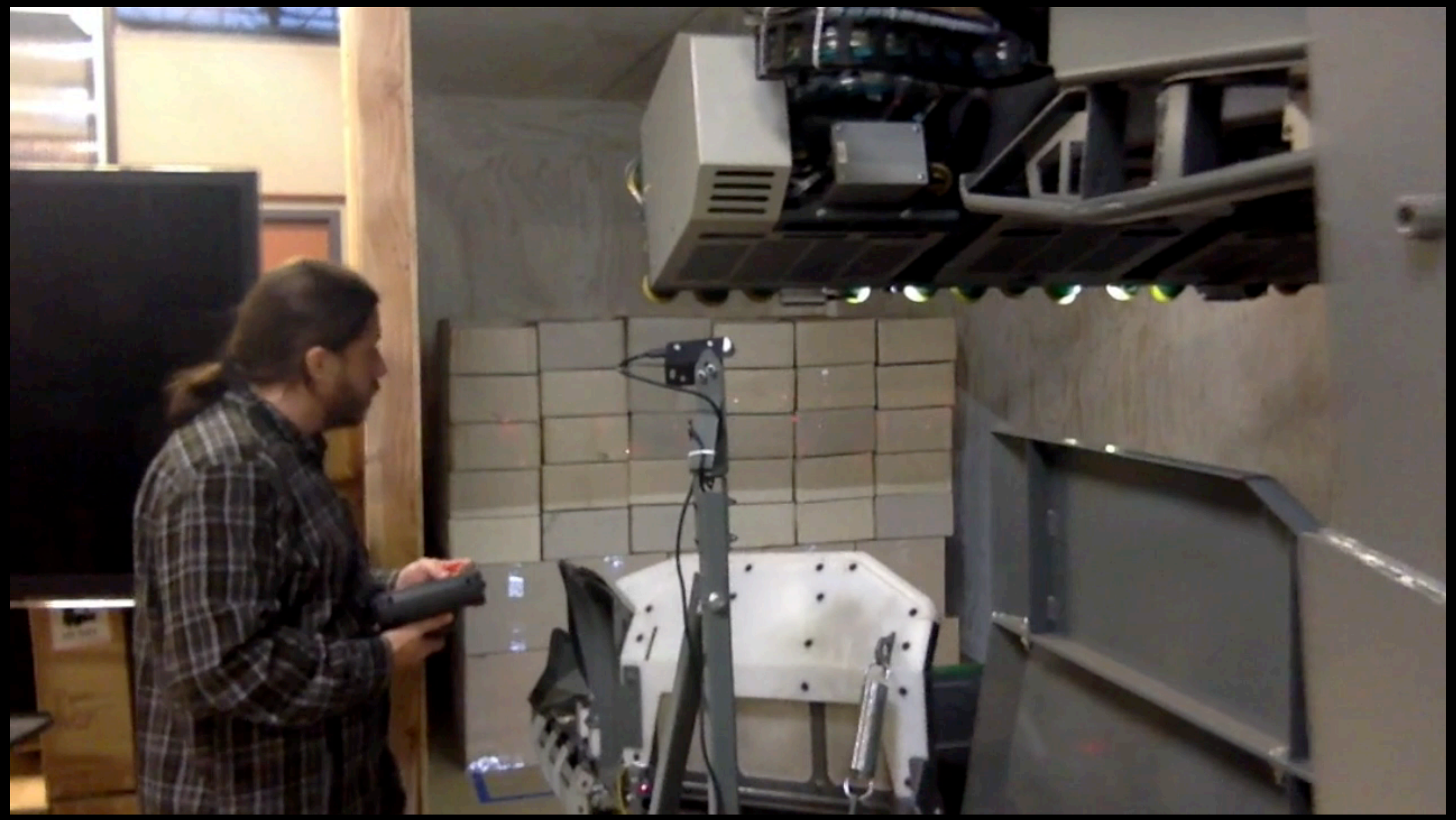


Environment



Approximate Model

Desired Characteristics



1. No access to resets
2. Needs small number of executions to reach goal
3. Needs no prior knowledge

Related Work: Planning in Unknown Environments

- Model-based RL
  - Use experience to update model
- Model-free RL
  - Use experience to update plan
- More data needed for model-free methods
  [Sun et. al. 2019, **Vemula et. al. 2019**]

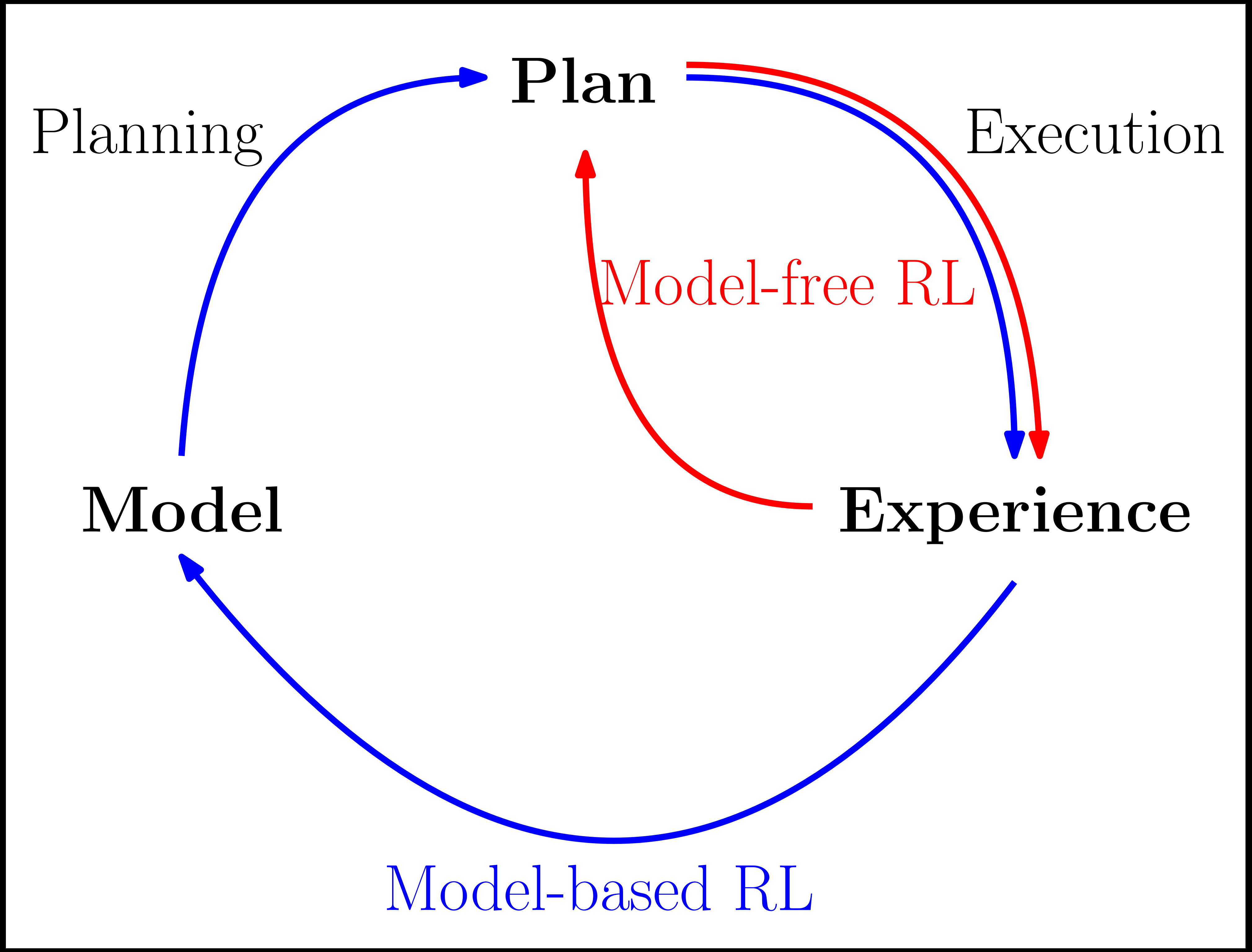
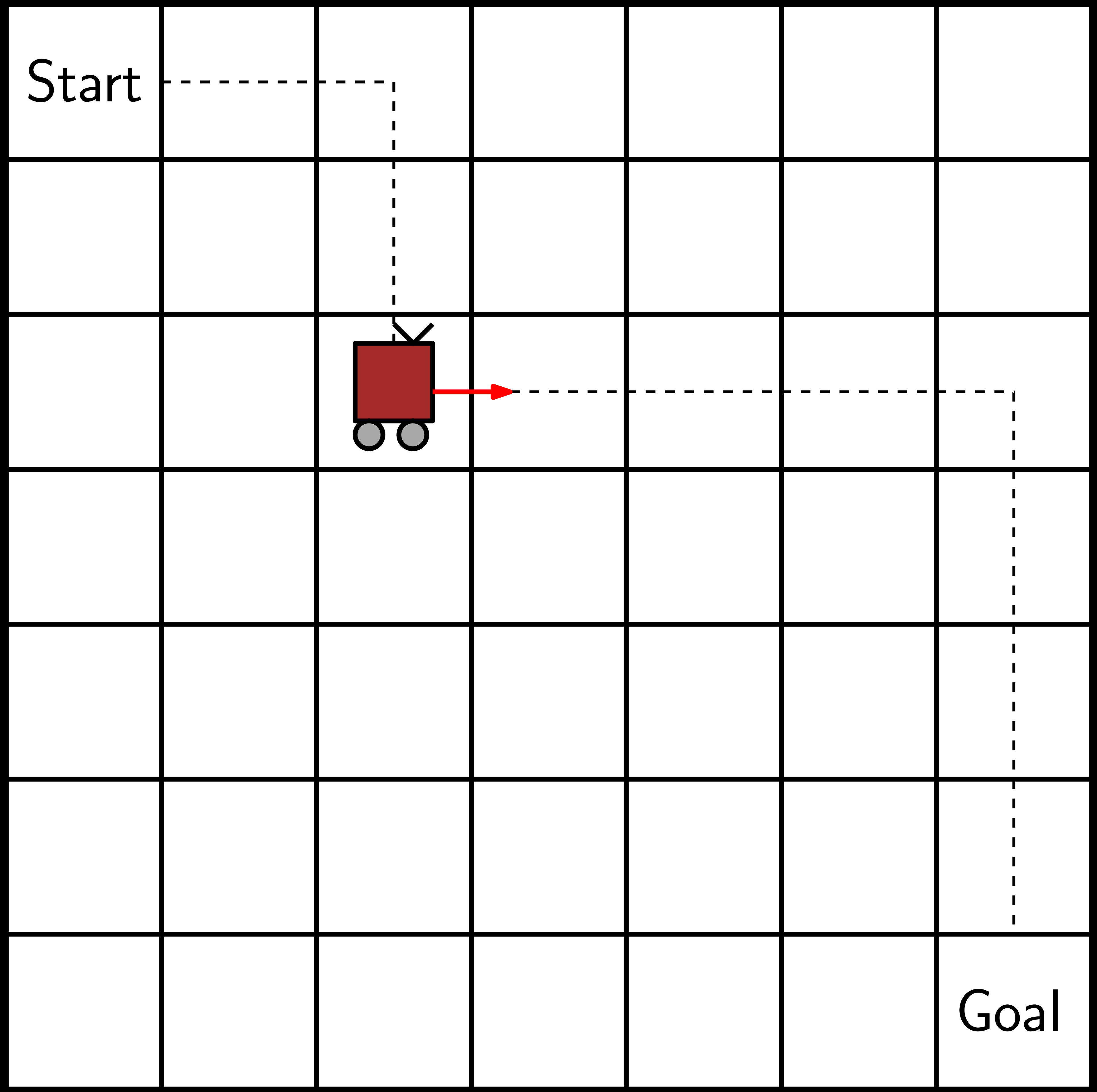
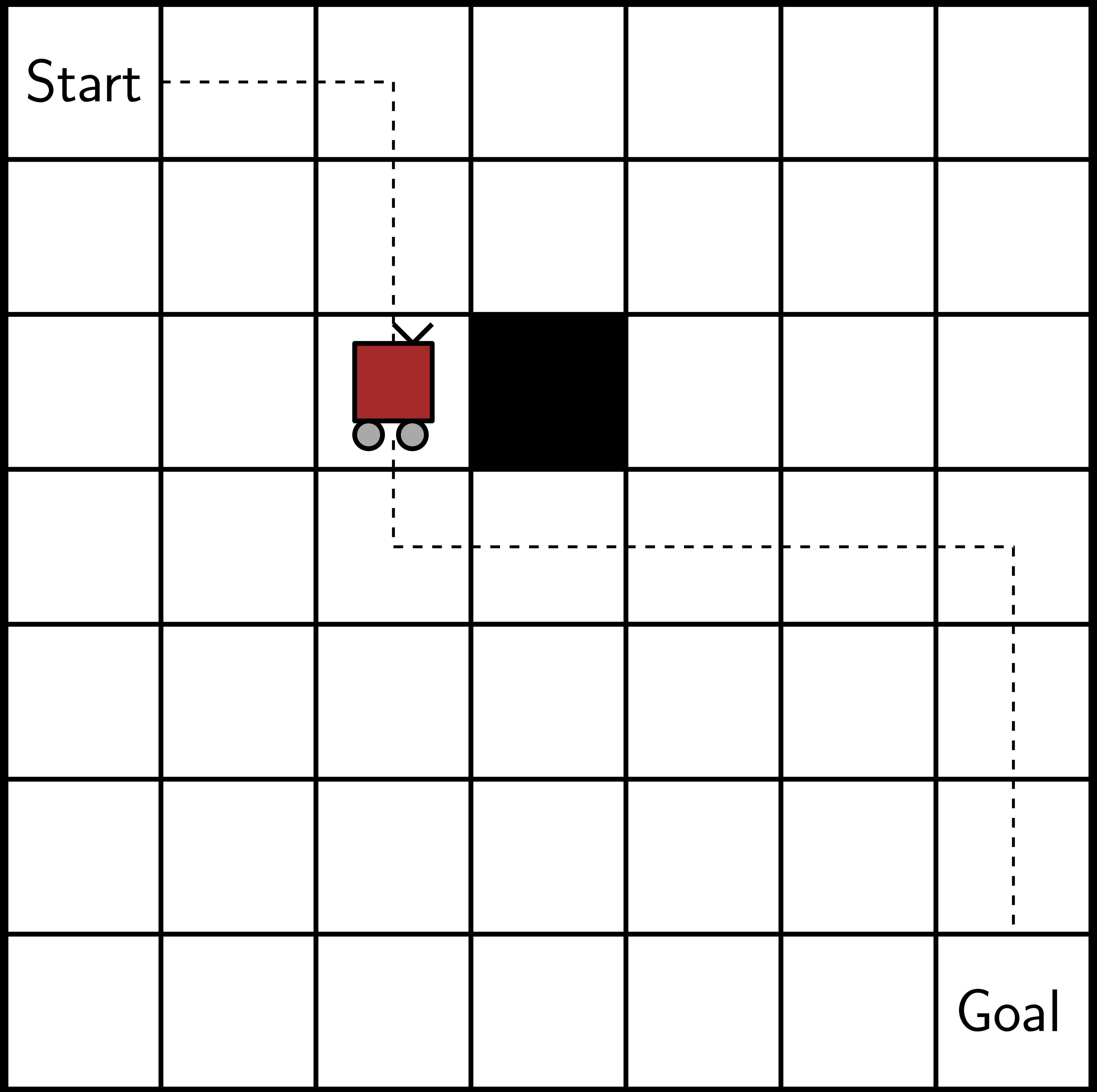


Figure inspired from DYNA [Sutton 1994]

Running Example : Model-based RL



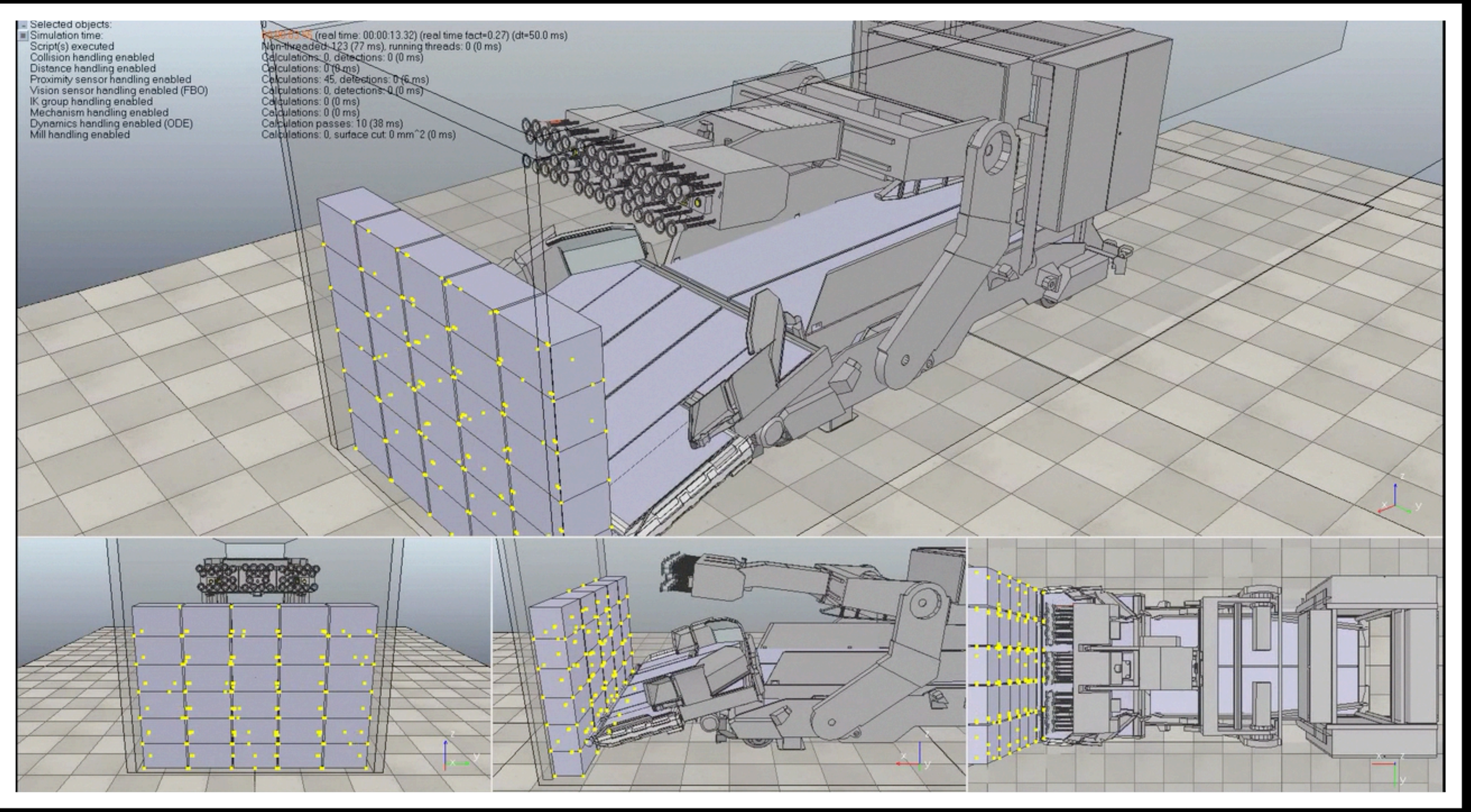
Model at time t



Updated Model at time t+1

Related Work: Updating Dynamics of Model Online

- Update approximate model using online experience [Sutton 1991, Barto et. al. 1995]
- Black-box simulators, interaction models, motion primitives
- Dynamics of such models cannot be changed arbitrarily online
- Need knowledge of how model is inaccurate to be efficient

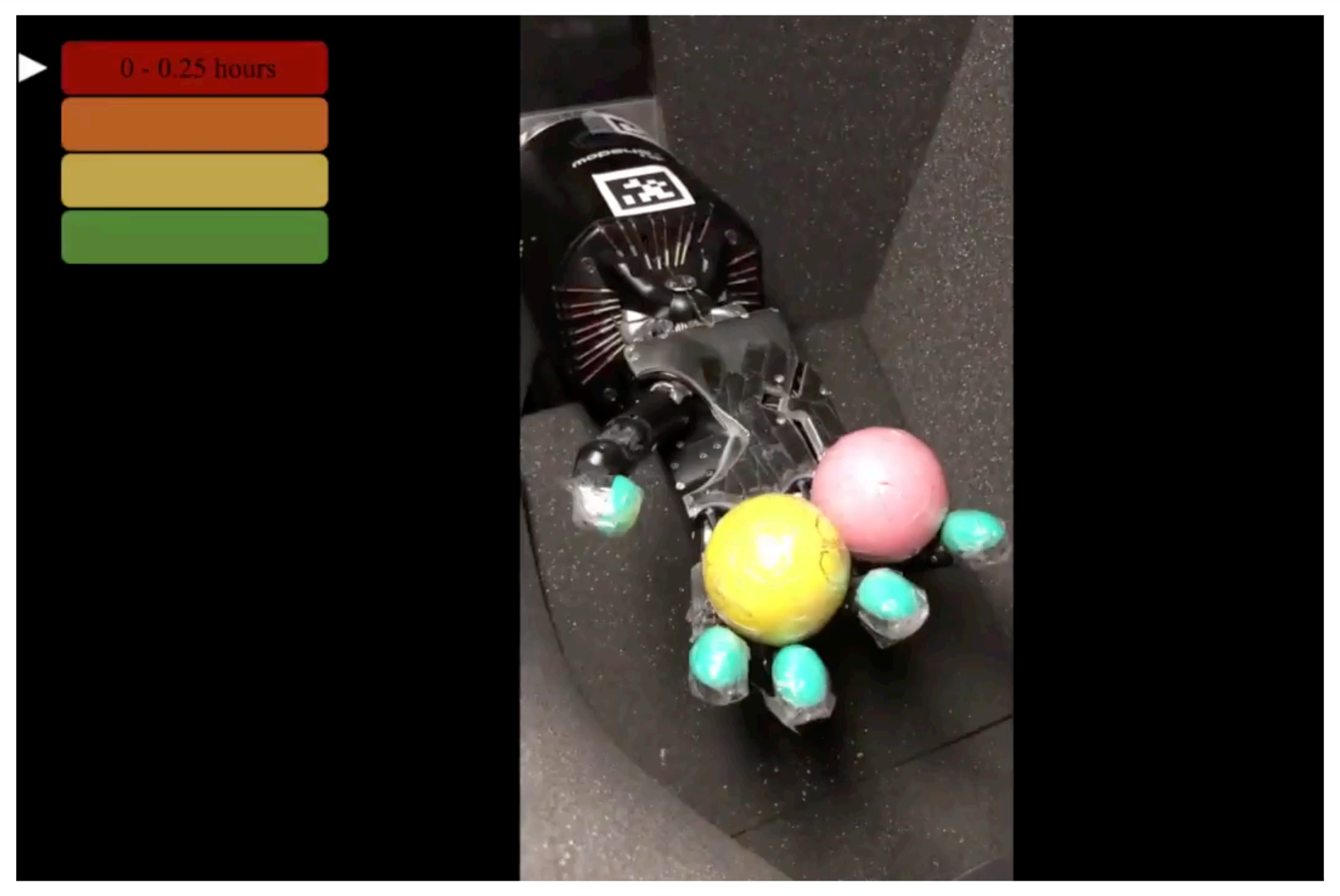


Video borrowed from [Vemula et. al. 2020]

Video borrowed from [Vemula et. al. 2017]

Related Work: Learning (Residual) Models from Executions

- Learn a model or a residual model [Nagabandi et. al. 2019, Saveriano et. al. 2017]
- Large number of samples, and access to resets required [Kearns and Singh 2002, Brafman et. al. 2002]
- No perfect model in model class [Joseph et. al. 2013]
- True dynamics are intractable to model e.g. deformable manipulation [McConachie et. al. 2020]



Videos borrowed from [Nagabandi et. al. 2019]

Related Work: Model-based Planning with Model-free Learning

- Fine-tune policy from model-based planning with model-free learning [Nagabandi et. al. 2017, Farshidian et. al. 2014]
- Use model-free learning in regions where model is inaccurate [Lee et. al. 2020, LaGrassa et. al. 2020]
- Relies on prior knowledge - inaccurately modeled region or expert demonstrations

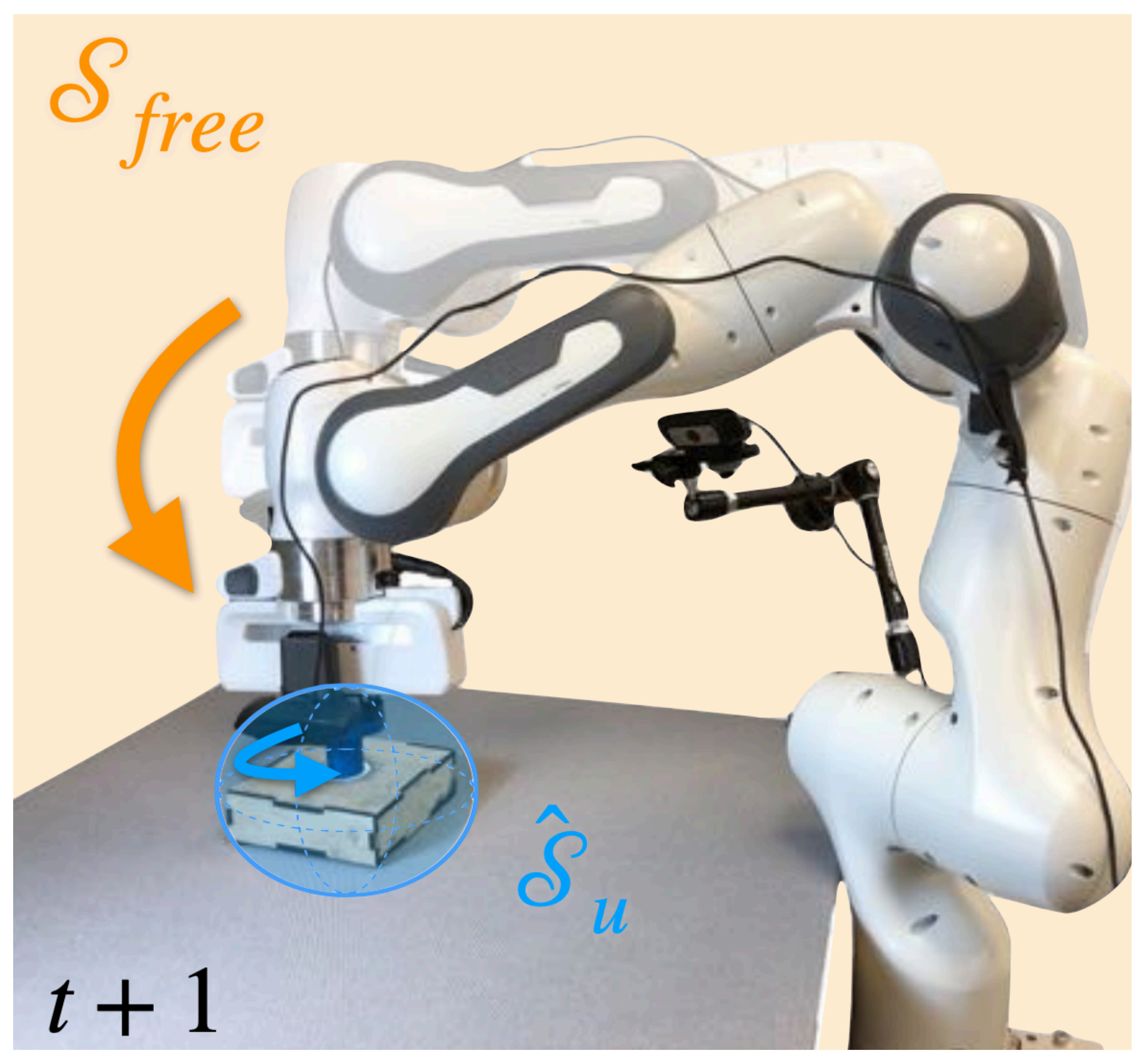


Figure and video borrowed from [Lee et. al. 2020]

Characteristics of our Algorithms - CMAX and CMAX++

✓ No updates to the dynamics of the model

✓ Use online experience to update behavior of planner

✓ Does not require access to resets

✓ Agnostic to how the model is inaccurate and require no prior knowledge

- Requires restrictive assumptions on the model

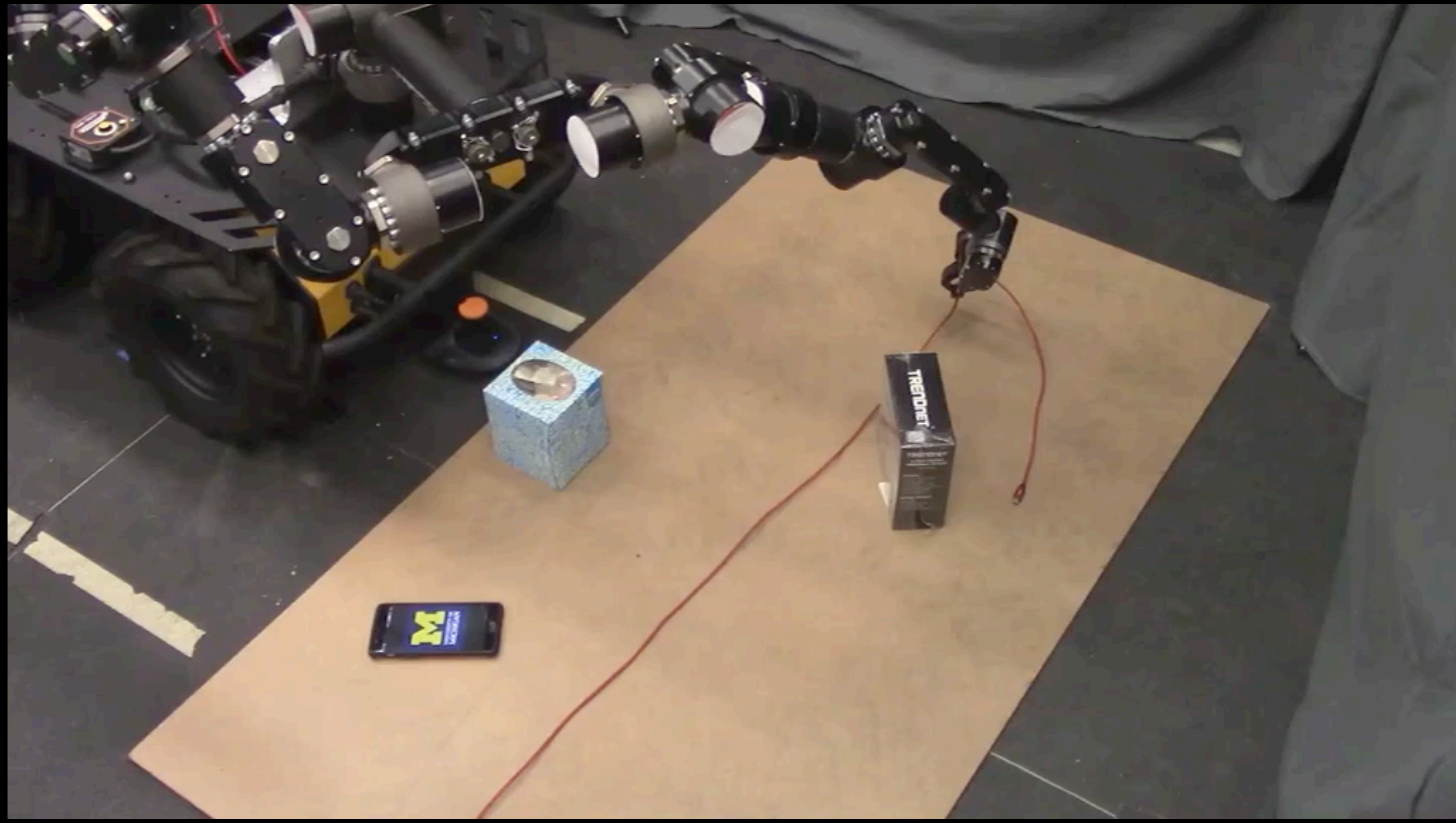
Updating the Behavior of the Planner

- Used in practice to deal with inaccurate modeling
- E.g. Poisoning (inflating) cost function
- Thesis answers when, why and how they work
- Extend to more general settings with less assumptions

Video borrowed from [Zucker et. al. 2011]

Concurrent and Follow-up Work

- CMAX for other domains such as deformable manipulation [McConachie et. al. 2020, Mitrano et. al. 2021]
- Penalize when planning using learned models [Power and Berenson 2021]
- Model-based Offline RL [Kidambi et. al. 2021]



Video borrowed from [Mitrano et. al. 2021]

Thesis Statement

By updating the behavior of the planner and not the dynamics of the model,

we can leverage simplified and potentially inaccurate models,

and significantly reduce the amount of experience required to complete the task

Interested in completing the task quickly and NOT in learning true dynamics

For real world tasks, there might be NO perfect model

Thesis Contributions

ANALYSIS

- Model-Free RL Requires Large Number of Samples [AISTATS 2019]
- Effectiveness of Using Inaccurate Models [Under review]

ALGORITHMS

- CMAX : Bias Planner Away From Inaccurately Modeled Regions [RSS 2020]
- CMAX++ : Learn to Exploit Inaccurately Modeled Regions [AAAI 2021]
- TOMS : Update Model to be Useful for Planning [Chapter 7 in Thesis]

Thesis Contributions

Model-Free RL Requires Large Number of Samples

[AISTATS 2019]

ANALYSIS

Effectiveness of Using Inaccurate Models

[Under review]

CMAX : Bias Planner Away From Inaccurately Modeled Regions

[RSS 2020]

CMAX++ : Learn to Exploit Inaccurately Modeled Regions

[AAAI 2021]

ALGORITHMS

TOMS : Update Model to be Useful for Planning

[Chapter 7 in Thesis]

Thesis Contributions

Model-Free RL Requires Large Number of Samples

[AISTATS 2019]

Effectiveness of Using Inaccurate Models

[Under review]

CMAX : Bias Planner Away From Inaccurately Modeled Regions

[RSS 2020]

CMAX++ : Learn to Exploit Inaccurately Modeled Regions

[AAAI 2021]

TOMS : Update Model to be Useful for Planning

[Chapter 7 in Thesis]

Problem Formulation

Can be formulated as a shortest path problem $M = (\mathbb{S}, \mathbb{A}, \mathbb{G}, f, c)$

$\mathbb{S}$: State space, $\mathbb{A}$: Discrete action space, $\mathbb{G}$: Goal space

Cost function: $c : \mathbb{S} \times \mathbb{A} \rightarrow [0,1]$

Unknown Deterministic True Dynamics: $f : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{S}$

Access to Approximate Dynamics: $\hat{f} : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{S}$

*State is fully observable

*Goal can be reached from any state (no dead-ends)

Incorrect Transitions

Transitions where true and approximate dynamics differ

$f(s, a) \neq \hat{f}(s, a)$ or $||f(s, a) - \hat{f}(s, a)|| > \xi$

$\mathcal{X} \subseteq \mathbb{S} \times \mathbb{A}$ = set of "incorrect" transitions

$\mathcal{X}$ is not known beforehand, and only discovered through online executions

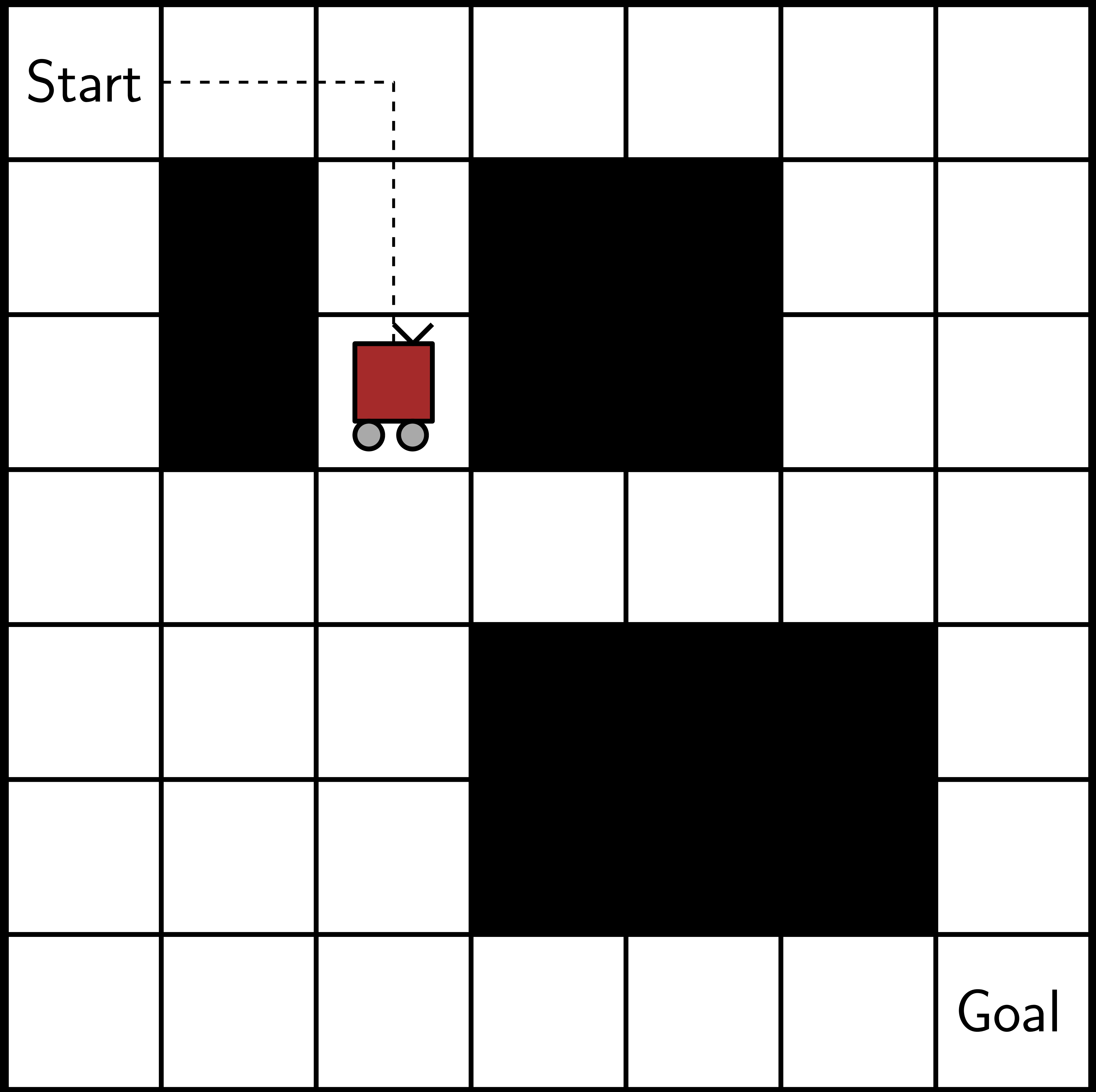
CMAX: Key Idea

Instead of learning the true dynamics,

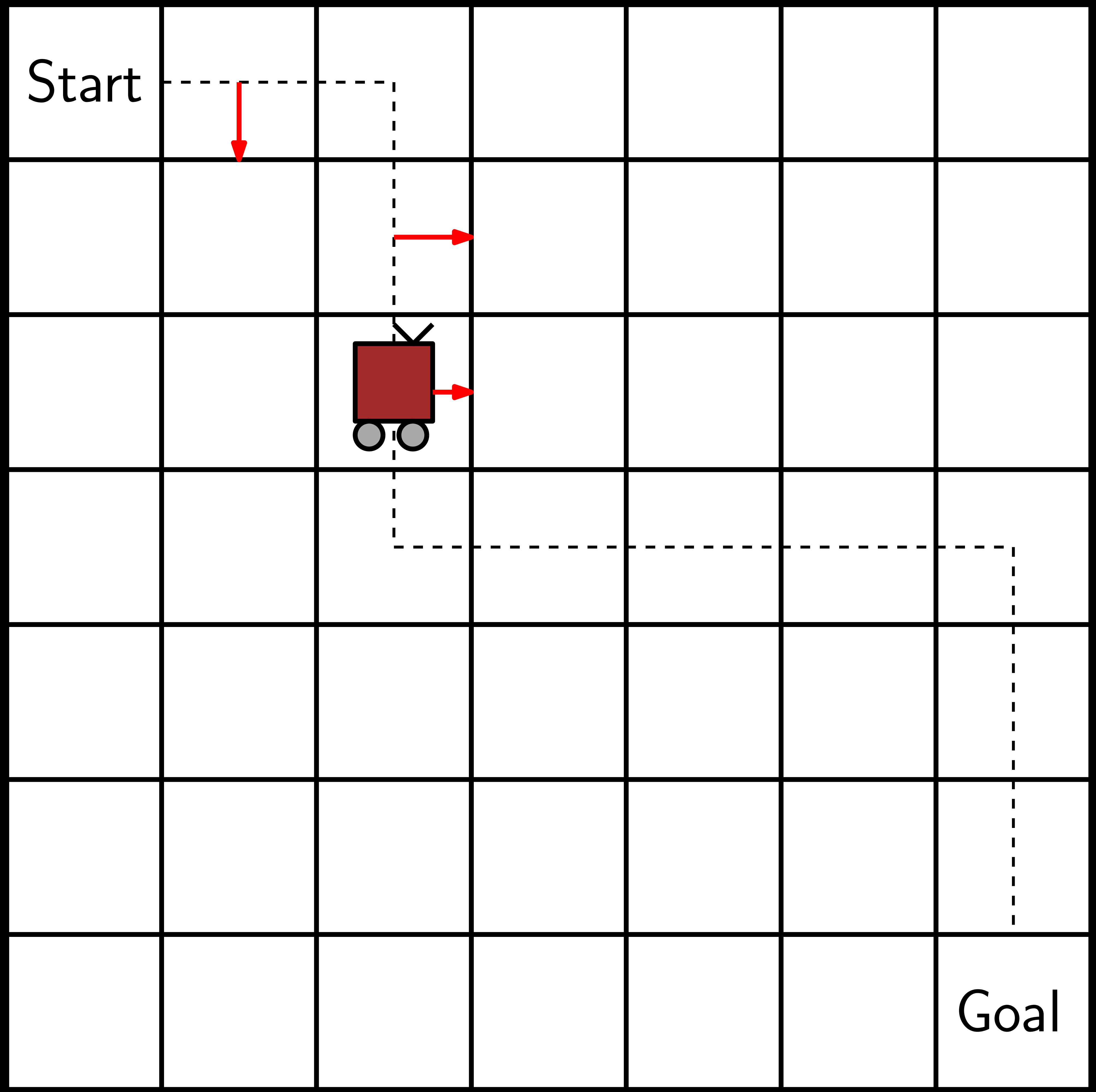
CMAX maintains a running estimate of the incorrect set and

biases the planner to avoid using incorrect transitions

Running Example : CMAX

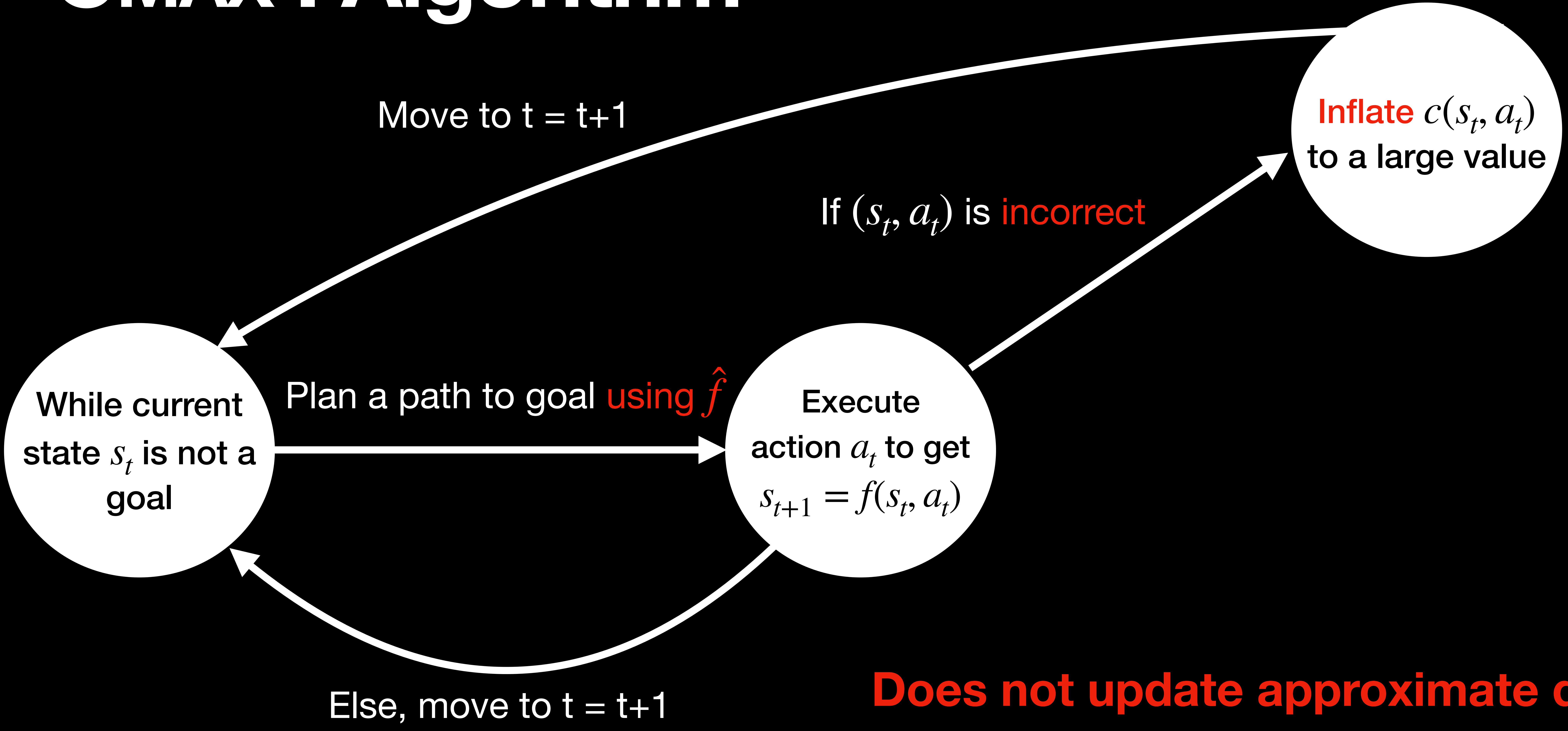


Environment



Approximate Model with incorrect transitions

CMAX : Algorithm



Does not update approximate dynamics $\hat{f}$!

CMAX: Task Completeness Guarantee

$\mathcal{X}_t$ - set of incorrect transitions discovered so far

Assumption: There always exists a path from s_t to a goal that does not contain any transition (s, a) known to be incorrect, i.e. $(s, a) \notin \mathcal{X}_t$

Under this assumption, the robot is guaranteed to reach a goal, i.e. CMAX is complete

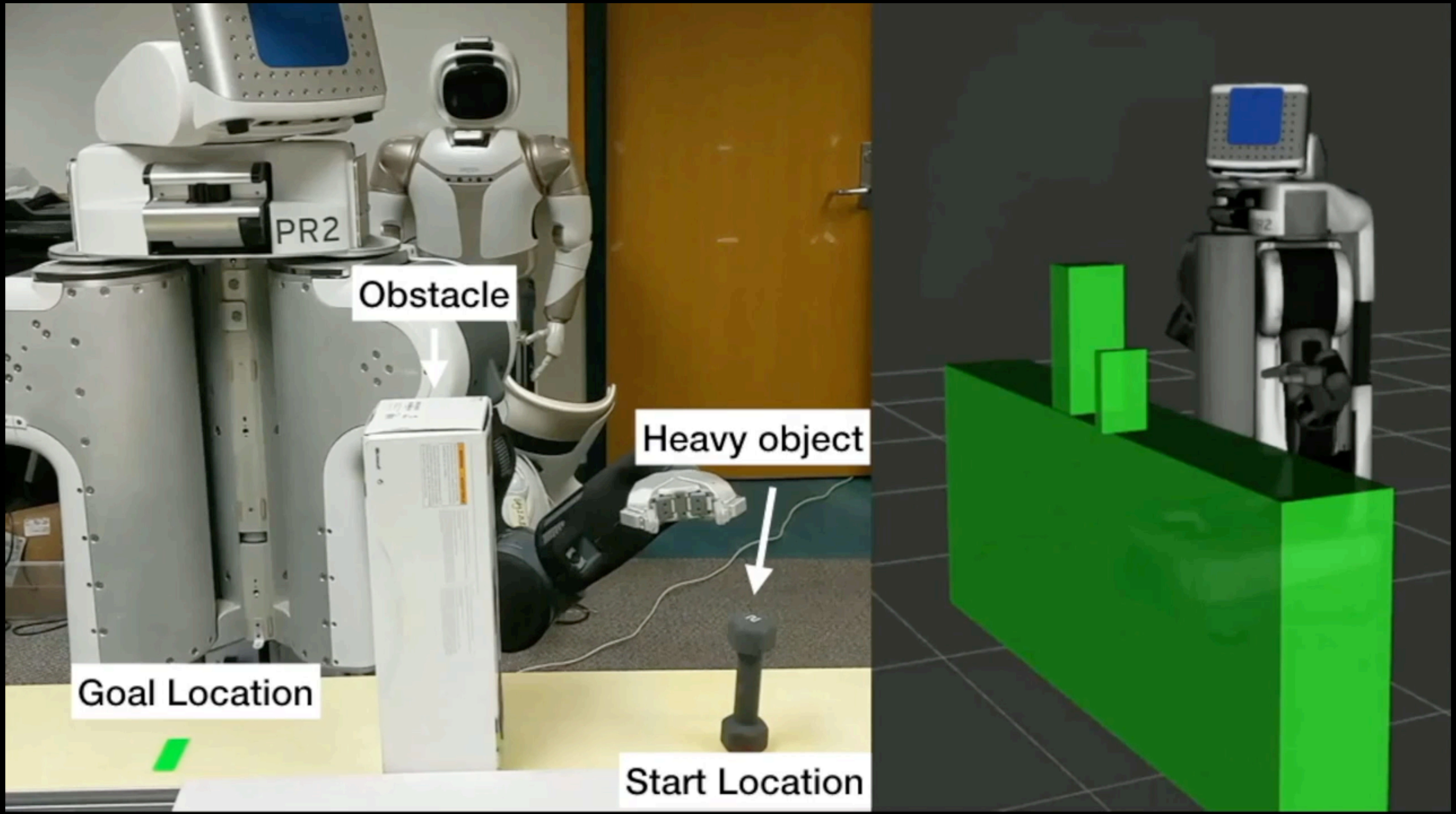
Summary

CMAX

1. Instead of updating dynamics, inflate cost of incorrect transitions
2. CMAX does not require updates to the dynamics of the model
3. *Use limited expansion search as planner to bound computation
4. *Use function approximation to scale CMAX to large state spaces

*refer to thesis for more details

CMAX : Goal-Driven Behavior



Outperforms Model-based and Model-Free Baselines

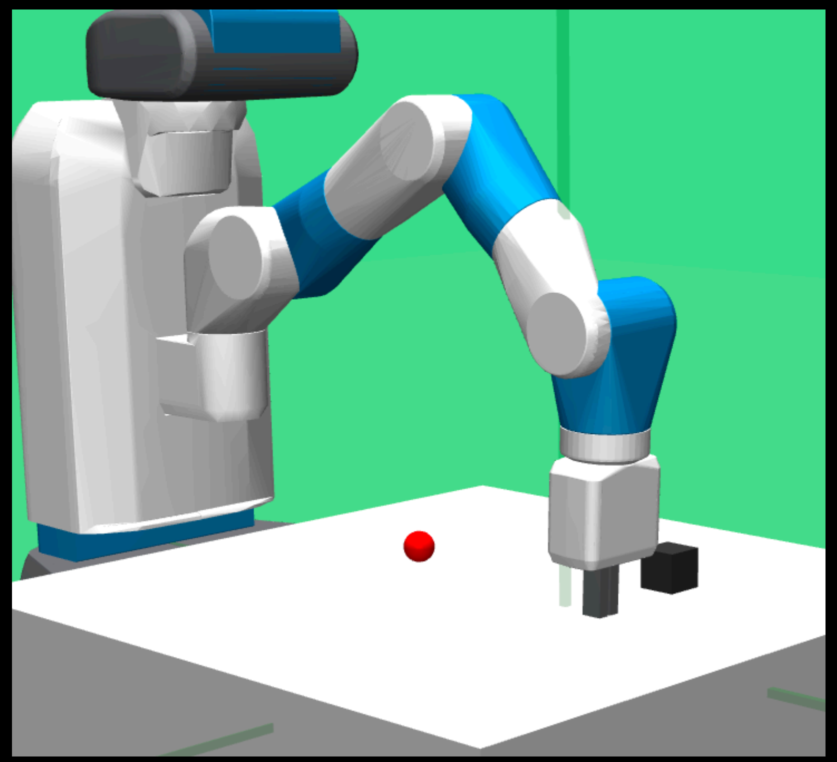
CMAX in large state spaces

Residual Model Learning using Neural Network

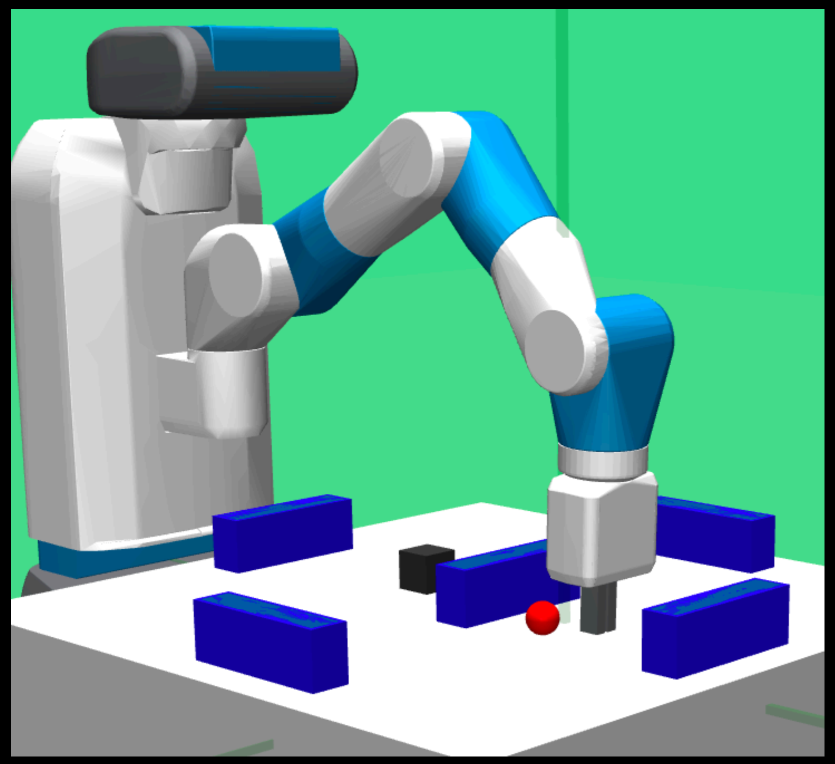
4D Planar Pushing in presence of obstacles

	Steps	% Success
CMAX		
Q-Learning		
Model NN		
Model KNN		

Residual Model Learning using K-Nearest Neighbor Regression

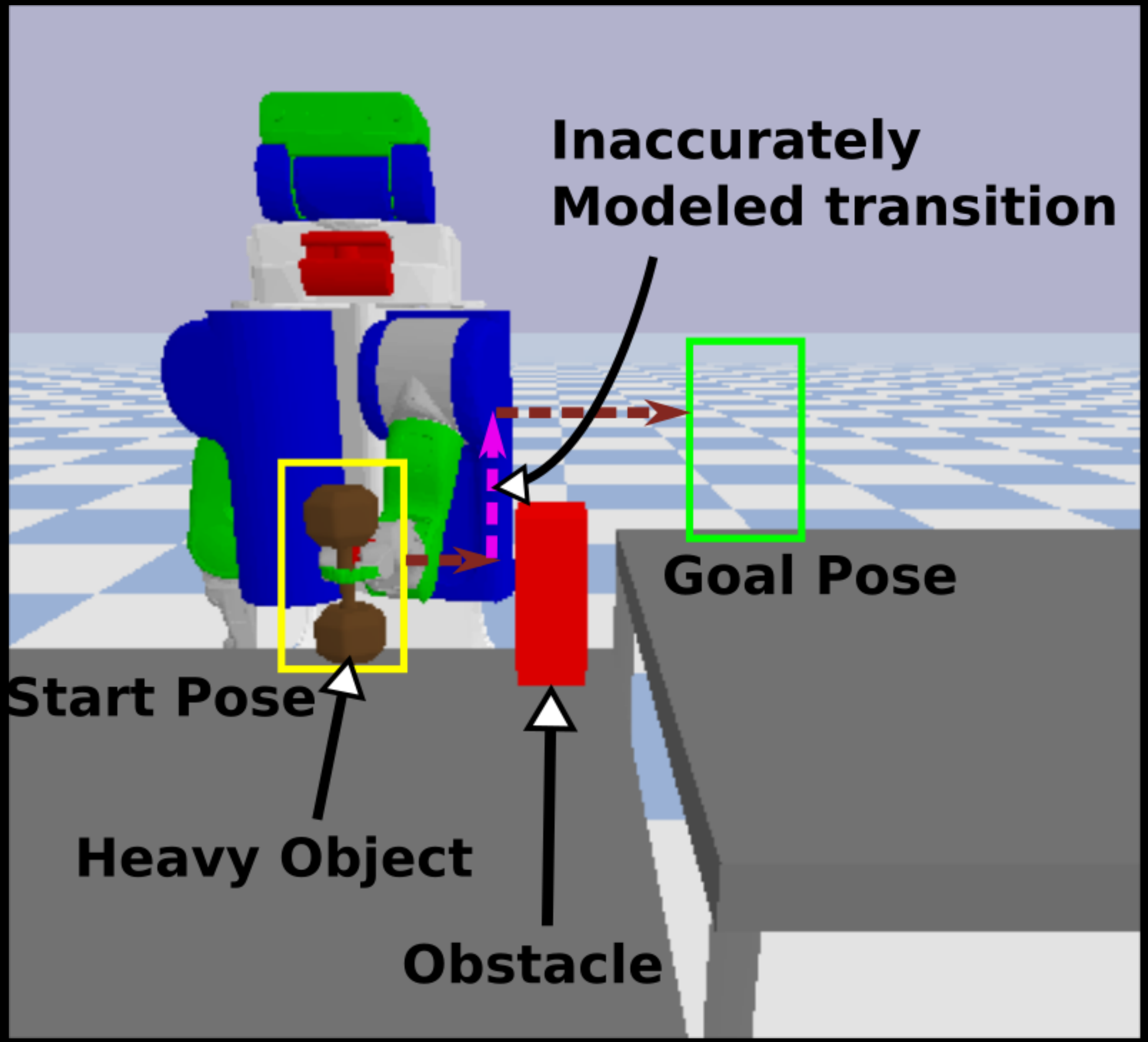


Approximate Model



Environment

CMAX fails in repetitive tasks



But by the 3rd repetition, CMAX takes more than 500 steps to reach the goal as previously executed incorrect transitions have inflated costs

Can we allow the planner to exploit incorrect transitions?

Thesis Contributions

Model-Free RL Requires Large Number of Samples

[AISTATS 2019]

Effectiveness of Using Inaccurate Models

[Under review]

CMAX : Bias Planner Away From Inaccurately Modeled Regions

[RSS 2020]

CMAX++ : Learn to Exploit Inaccurately Modeled Regions

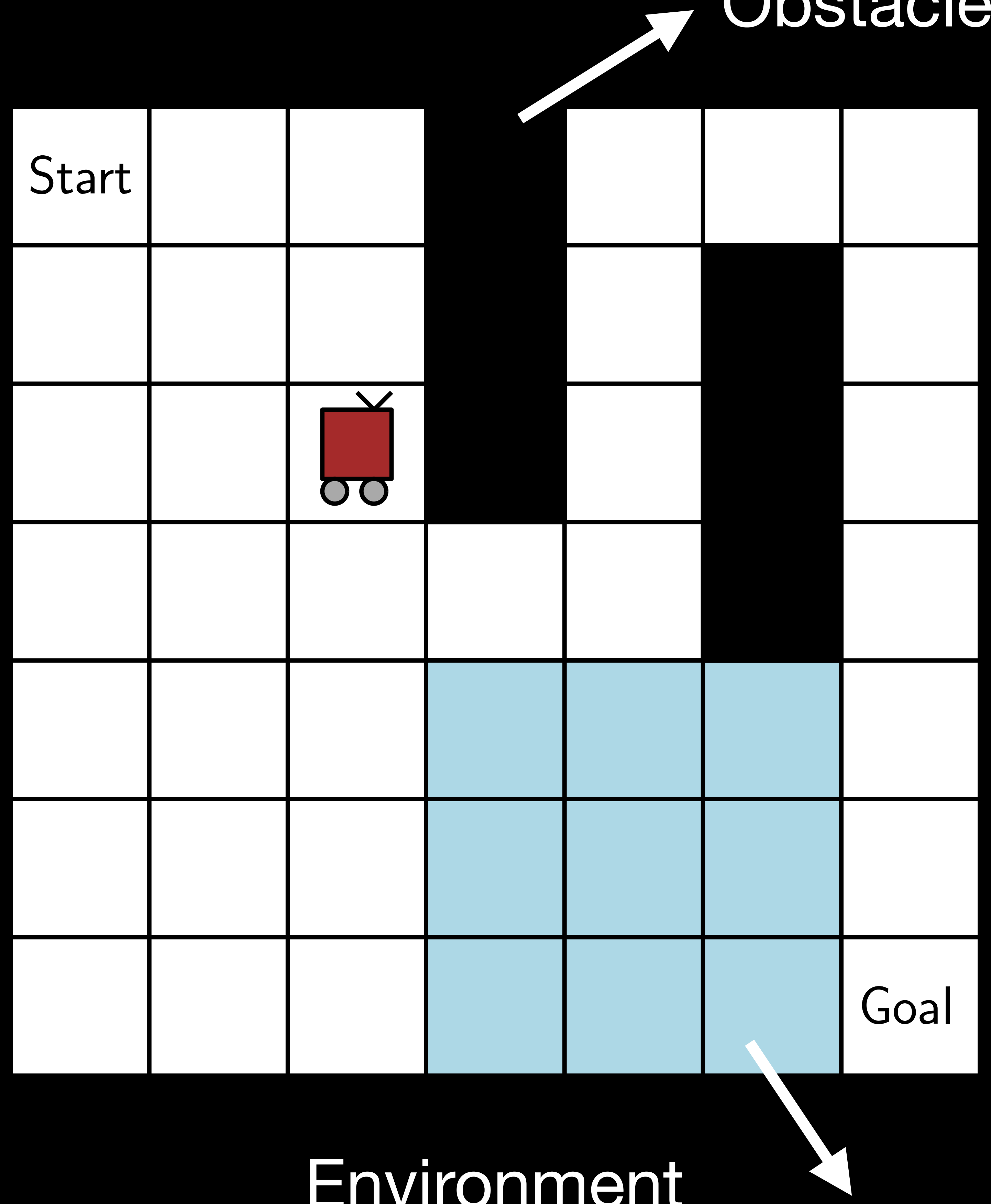
[AAAI 2021]

TOMS : Update Model to be Useful for Planning

[Chapter 7 in Thesis]

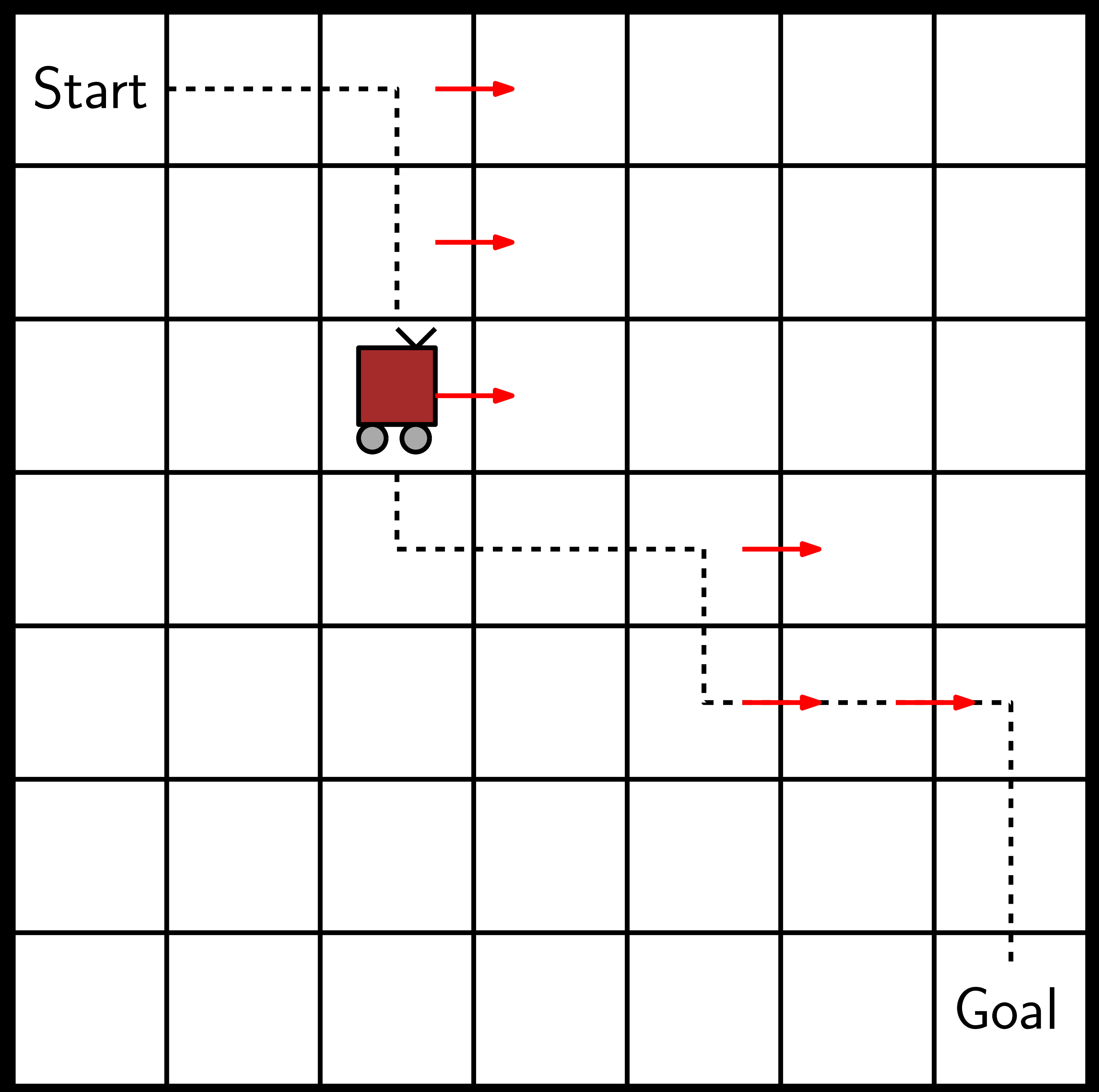
Modified Running Example

Obstacles

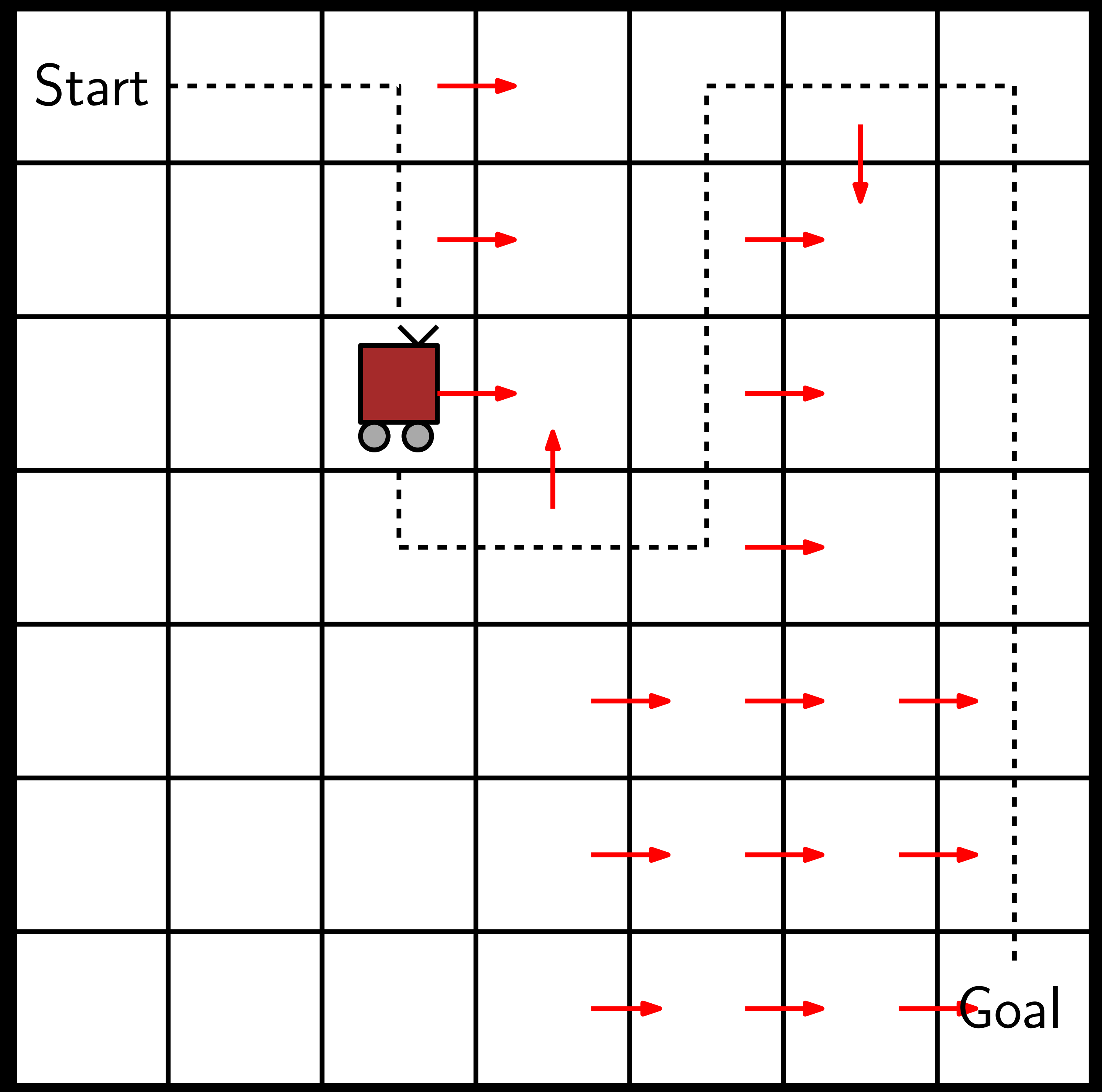


Environment

Ice (Robot slips)



CMAX at first repetition



CMAX at n-th repetition

Modified Objective

Provably reach the goal online in each repetition without any resets

allowing the path to contain incorrectly modeled transitions

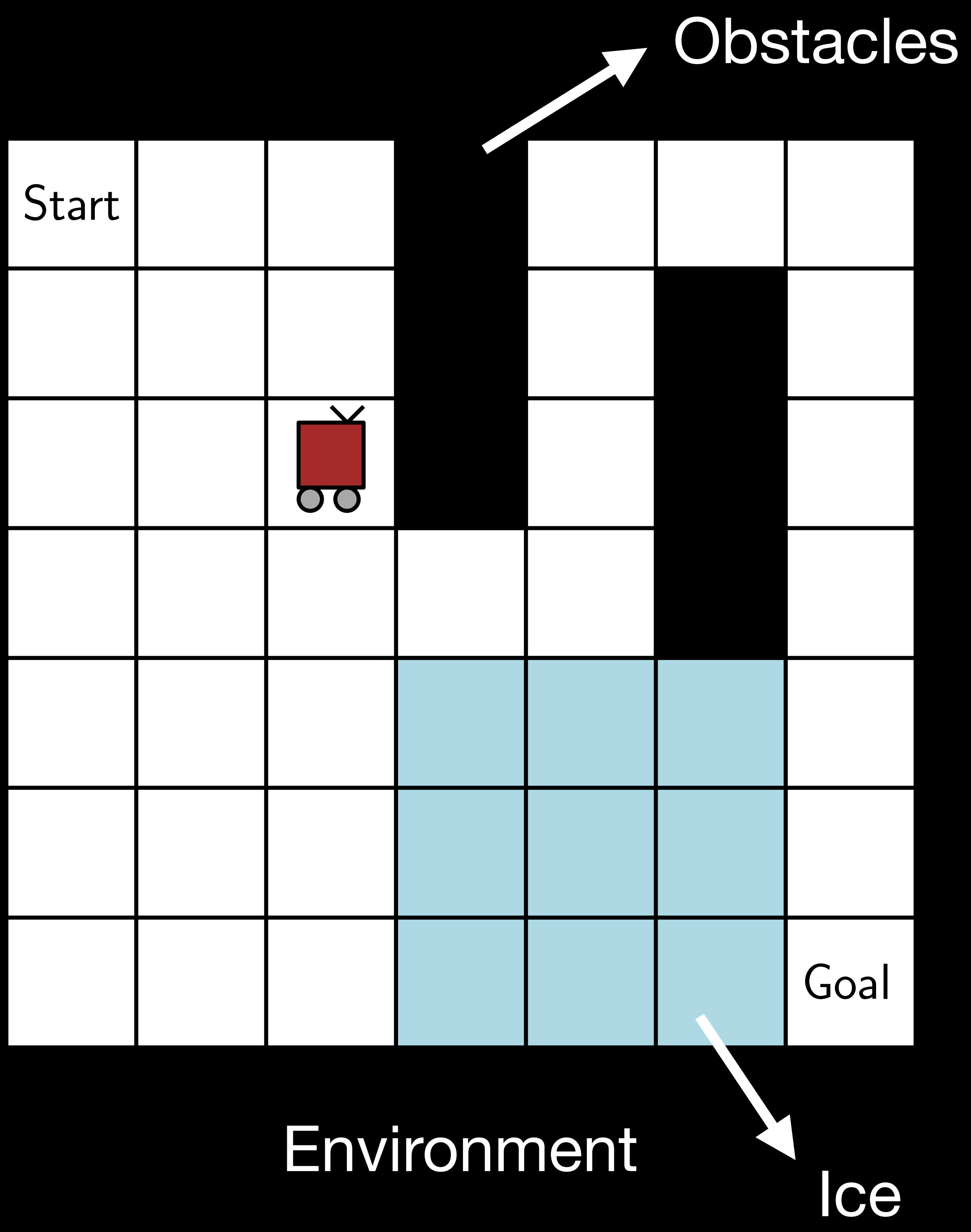
CMAX++: Key Idea

CMAX++ maintains model-free Q-value estimates of incorrect transitions

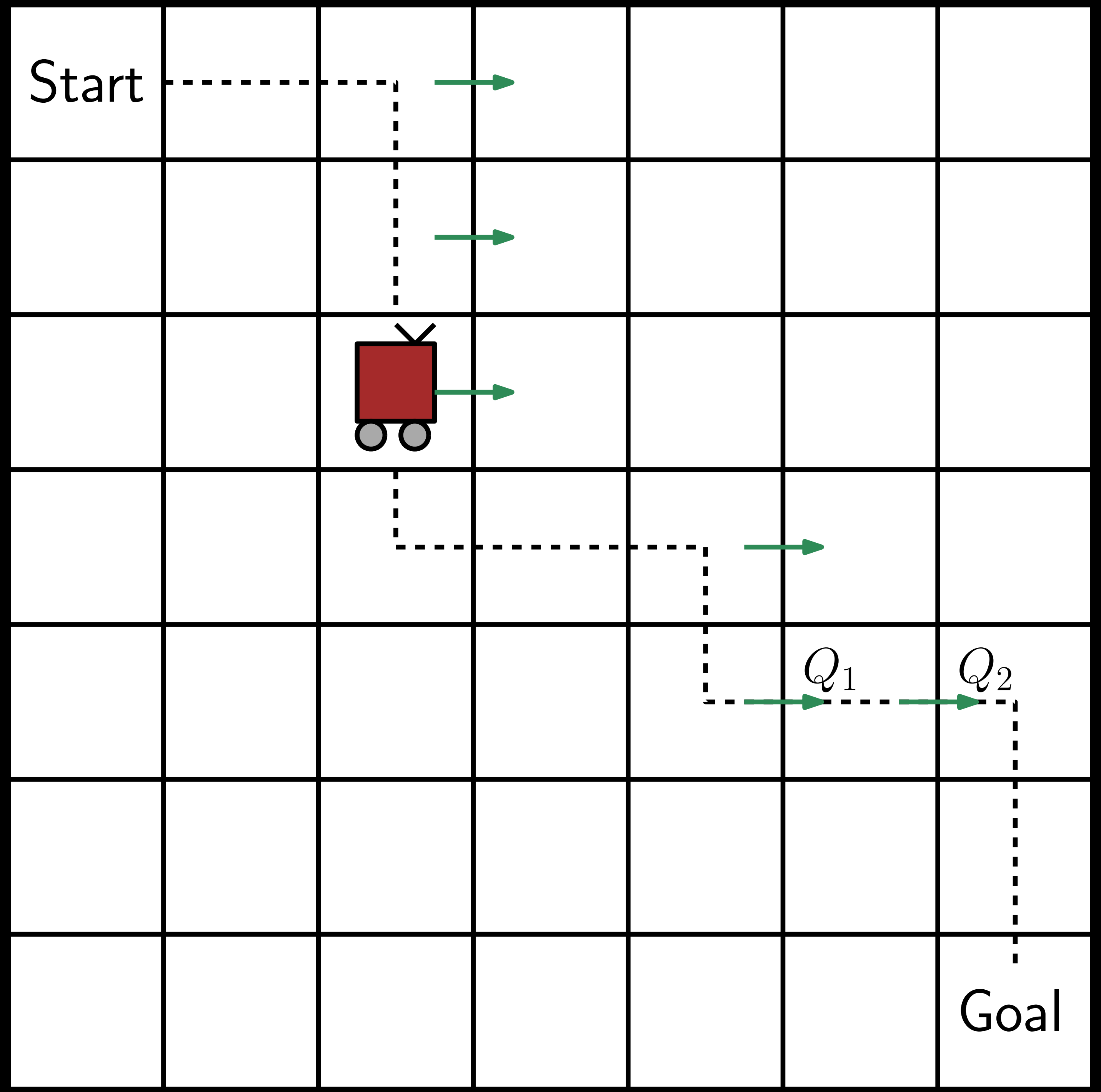
and

integrates them into model-based planning using the inaccurate model

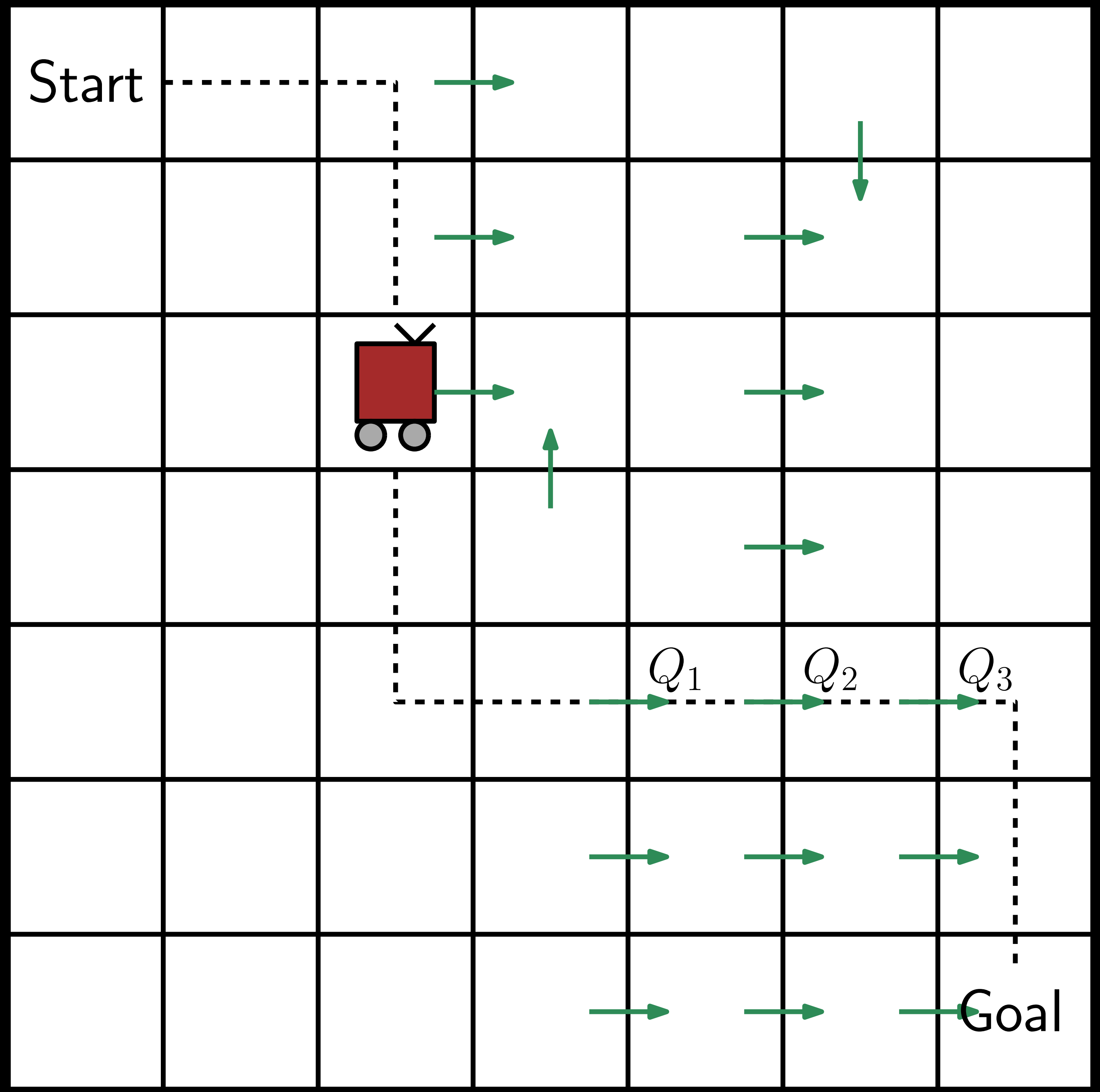
Running Example



Environment

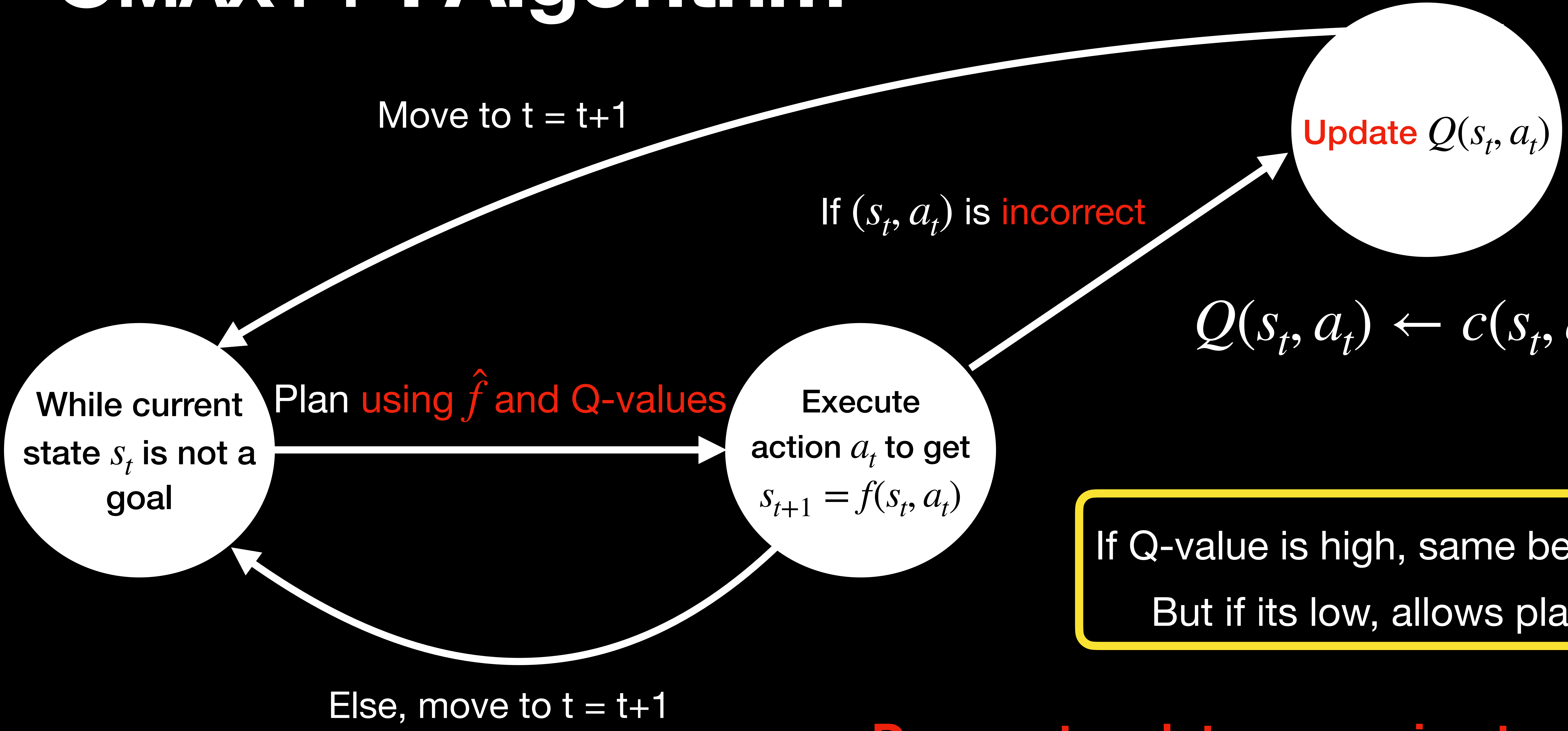


CMAX++ at first repetition



CMAX++ at n-th repetition

CMAX++ : Algorithm



$$Q(s_t, a_t) \leftarrow c(s_t, a_t) + V(s_{t+1})$$

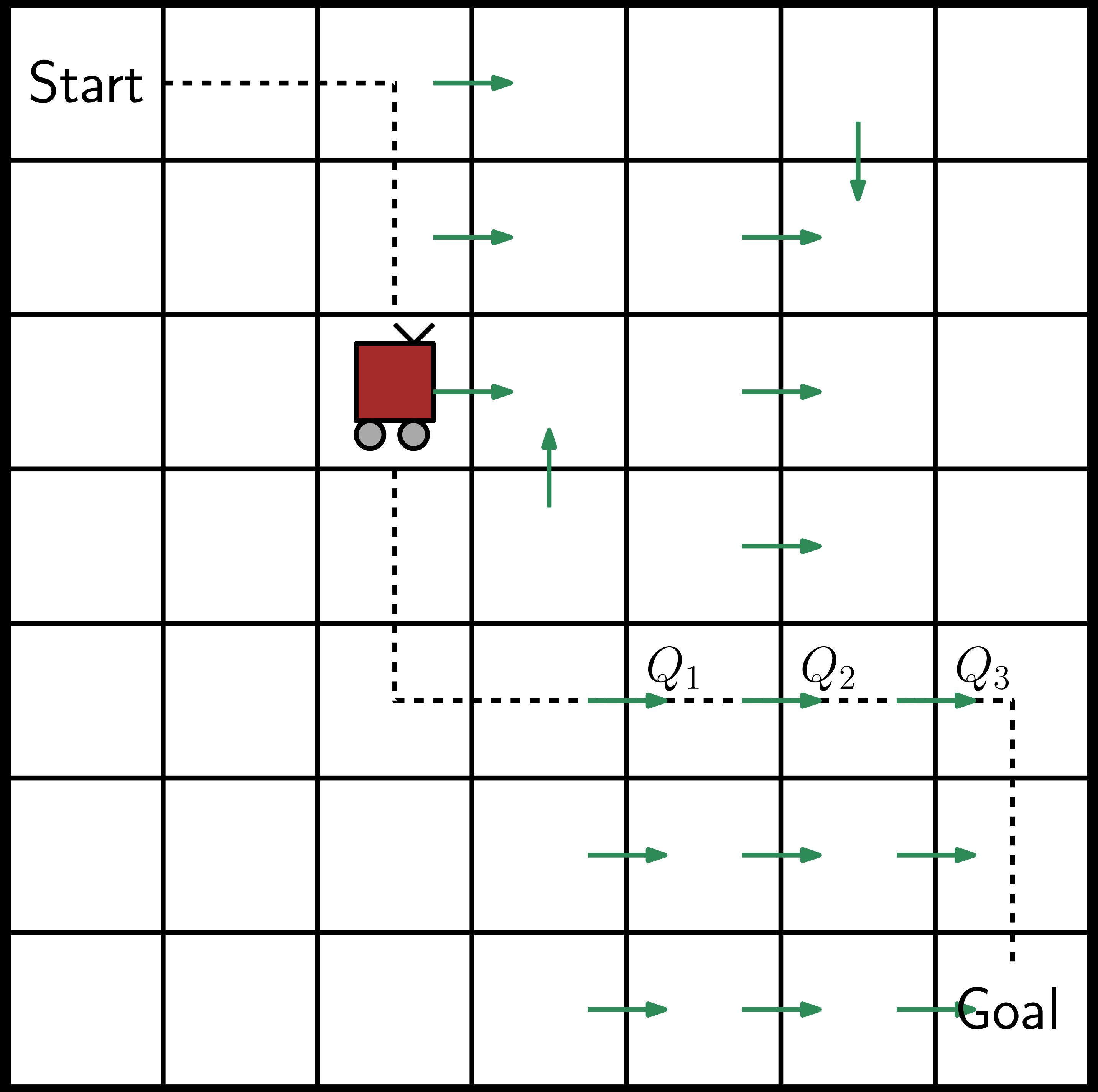
If Q-value is high, same behavior as CMAX

But if its low, allows planner to exploit

Does not update approximate dynamics $\hat{f}$!

CMAX++ : Major Limitation of Model-Free Estimation

CMAX++ wastes executions estimating Q-values, and lacks goal-driven behavior like CMAX



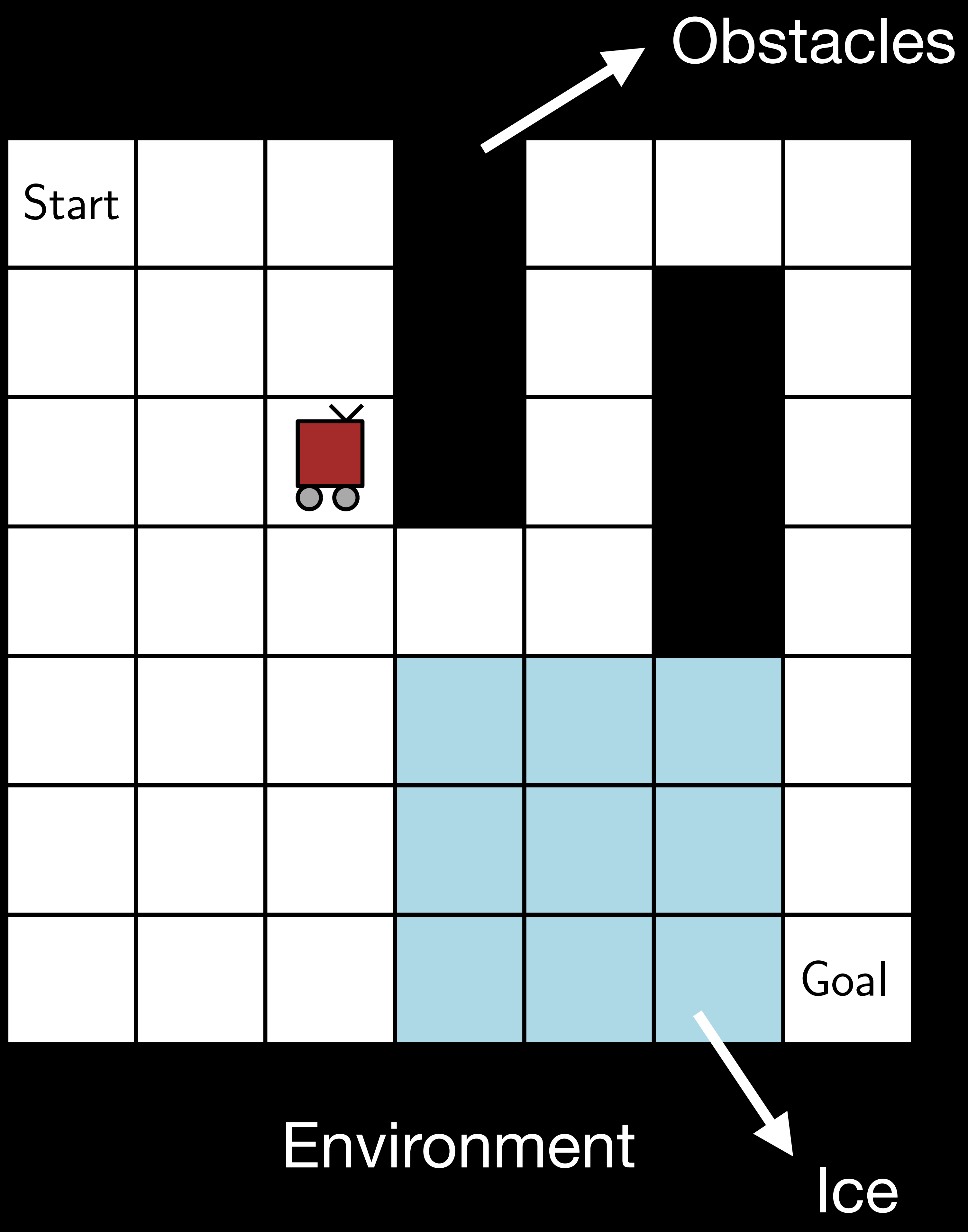
Adaptive-CMAX++: Key Idea

Intelligently switch between CMAX and CMAX++ during execution to combine advantages of both

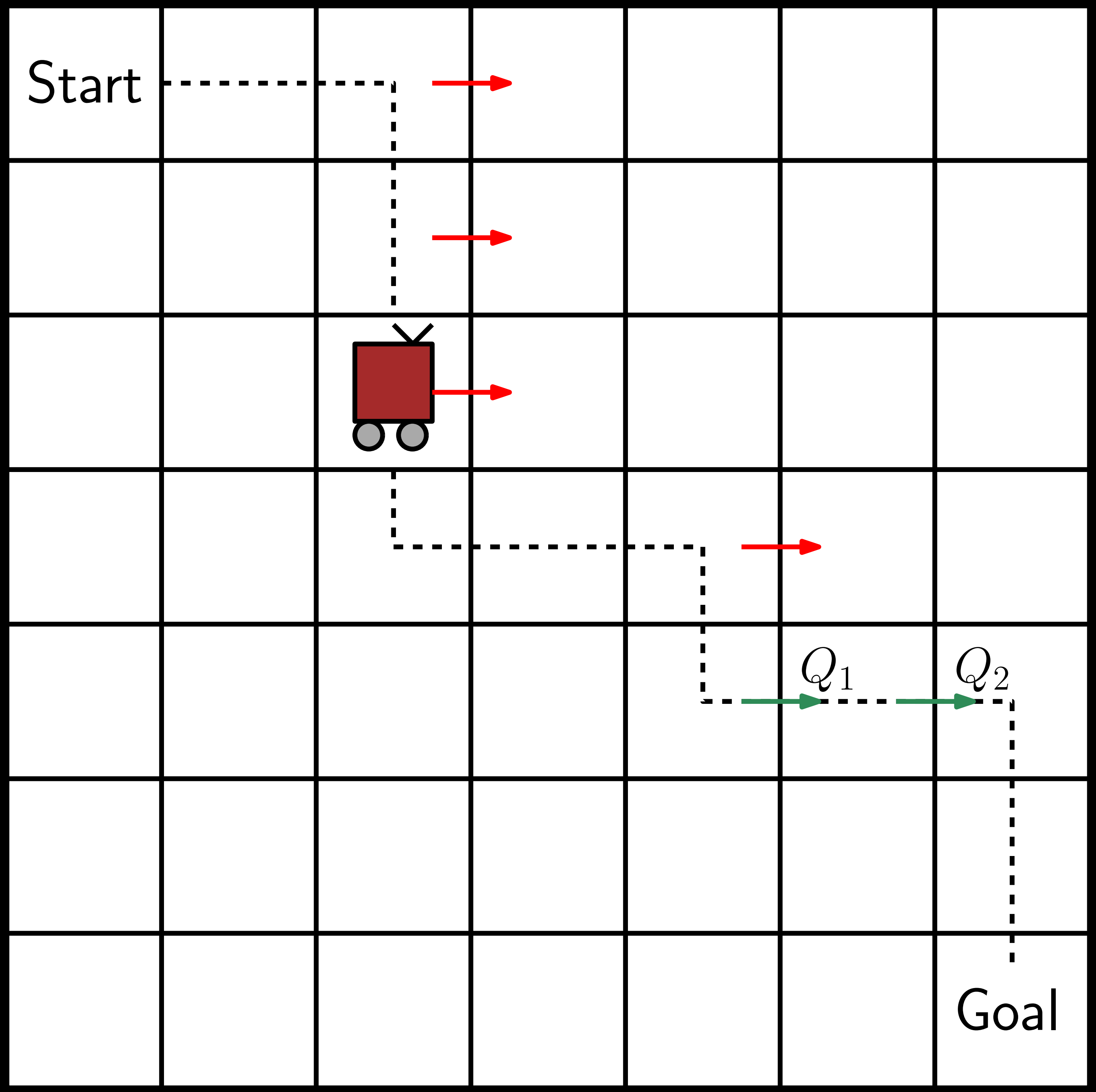
Adaptive-CMAX++ : Intuition

- If solution cost using CMAX is not far from CMAX++, prefer CMAX
- Anytime-like: Goal-driven in early repetitions, Optimal in later repetitions
- Executions to estimate Q-values spread across repetitions
- Strives to have good performance in every single repetition

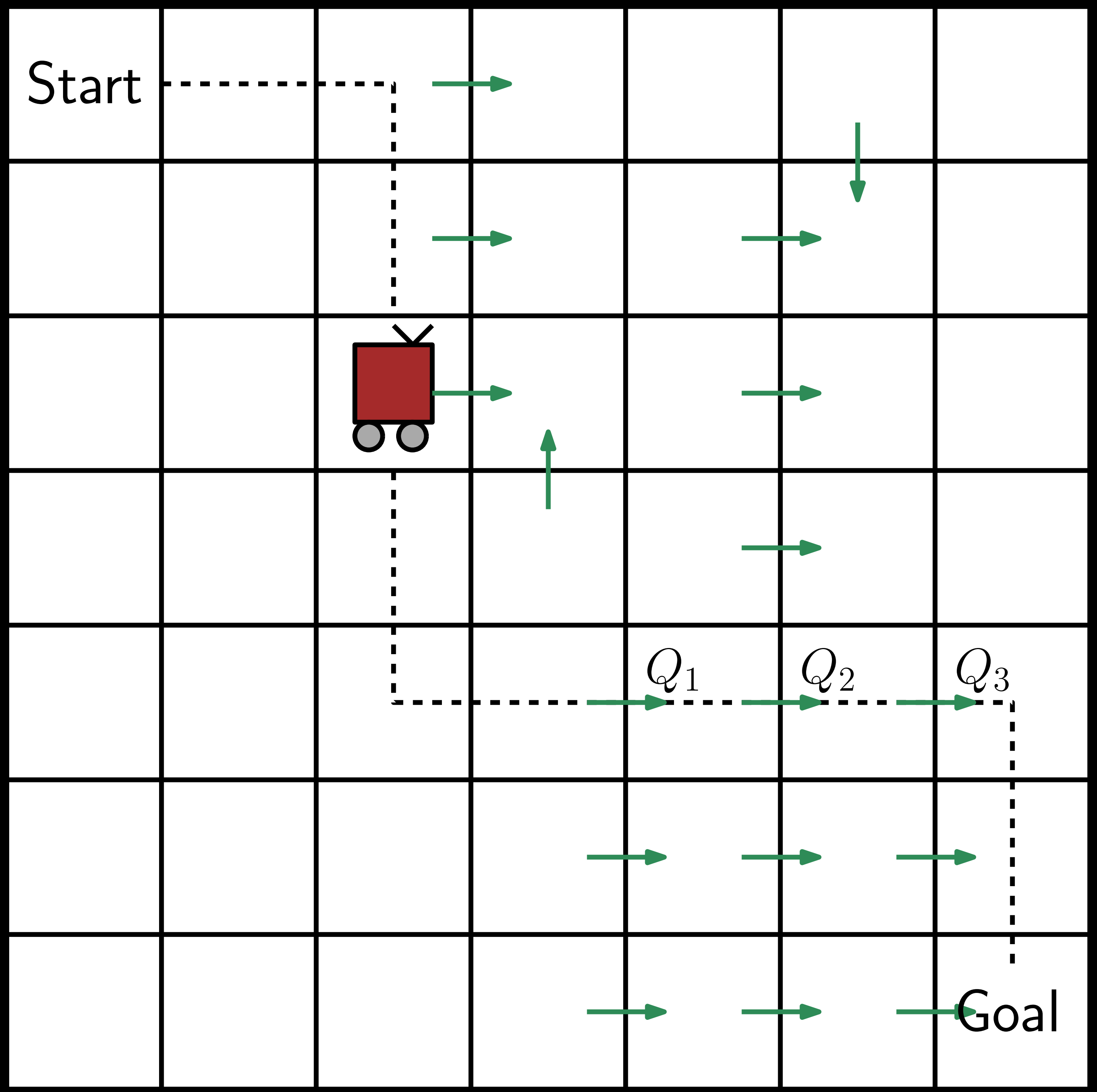
Running Example



Environment



A-CMAX++ at first repetition



A-CMAX++ at n-th repetition

Optimistic Model Assumption

Optimal value $\hat{V}*$ under approximate dynamics $\hat{f}$ underestimates the optimal value $V*$ under true dynamics f at all states

$$\hat{V}*(s) \leq V*(s), \forall s$$

Robot is never "pleasantly surprised" during execution

E.g. Free-space assumption in robot navigation [Nourbakhsh 1996]

Theoretical Guarantees

Completeness and Asymptotic Convergence

- Under Optimistic Model assumption, CMAX++ is guaranteed
  - to be complete in each repetition
  - to converge to the optimal path as number of repetitions grow

Summary

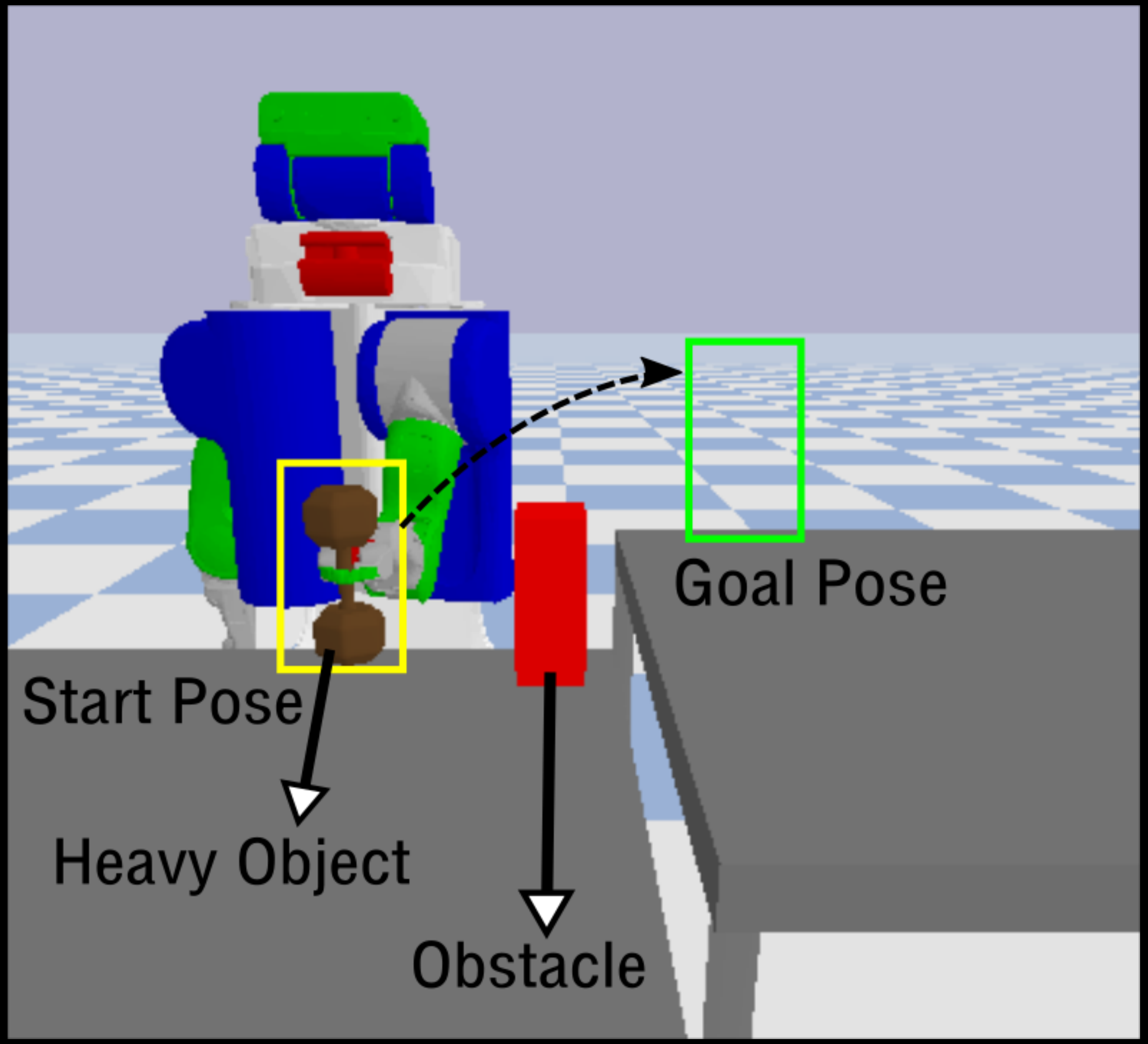
CMAX++ and Adaptive-CMAX++

1. Learn Q-value estimates for incorrect transitions
2. Integrate model-free Q-values into model-based planning
3. *Switch between CMAX and CMAX++ during execution
4. *Function approximation to scale algorithm to large state spaces

*refer to thesis for more details

7D Pick-and-Place with a Heavy Object

- Large state space - 7D arm configuration
- Object modeled as lightweight
- Can lift heavy object only in certain configurations
- Repetition is successful if robot reaches goal within 500 timesteps



Repetition→	1		5		20	
	Steps	*Success*	*Steps*	*Success*	*Steps*	*Success*
CMAX						
CMAX++						
A-CMAX++						
Model KNN						
Model NN						
Q-learning						

Model KNN : Residual Model learning using K-Nearest Neighbor Regression

Model NN : Residual Model learning using Neural Network Approximator

Q-learning: Model-free baseline with carefully initialized value estimates

Repetition→	1		5		20	
	Steps	*Success*	*Steps*	*Success*	*Steps*	*Success*
CMAX	**17.8 ± 3.4**	100%				
CMAX++	**17 ± 4.9**	100%				
A-CMAX++	**17.8 ± 3.4**	100%				
Model KNN	40.6 ± 7.3	100%				
Model NN	56 ± 16.2	100%				
Q-learning	172.4 ± 75	100%				

Lower is Better

Repetition→	1		5		20	
	Steps	*Success*	*Steps*	*Success*	*Steps*	*Success*
CMAX	$\mathbf{17.8 \pm 3.4}$	100%	13.6 ± 0.5	60%	15 ± 0	20%
CMAX++	$\mathbf{17 \pm 4.9}$	100%	14.2 ± 3.3	100%	$\mathbf{10.8 \pm 0.1}$	100%
A-CMAX++	$\mathbf{17.8 \pm 3.4}$	100%	$\mathbf{11.6 \pm 0.7}$	100%	$\mathbf{10.6 \pm 0.4}$	100%
Model KNN	40.6 ± 7.3	100%	12.8 ± 1.3	100%	12.4 ± 1.4	100%
Model NN	56 ± 16.2	100%	208.2 ± 92.1	80%	37.5 ± 20.1	40%
Q-learning	172.4 ± 75	100%	23.2 ± 10.3	80%	10.2 ± 0.6	80%

Performance in Last Repetition

CMAX++

CMAX

Thesis Contributions

Model-Free RL Requires Large Number of Samples

[AISTATS 2019]

Effectiveness of Using Inaccurate Models

[Under review]

CMAX : Bias Planner Away From Inaccurately Modeled Regions

[RSS 2020]

CMAX++ : Learn to Exploit Inaccurately Modeled Regions

[AAAI 2021]

TOMS : Update Model to be Useful for Planning

[Chapter 7 in Thesis]

What is the worst case performance of CMAX-like methods, given an inaccurate model?

and is it strictly better than naively using the model?

Iterative Learning Control (ILC) [Arimoto et. al. 1984]

A CMAX-like approach

✓ Uses inaccurate model for control

✓ Does not update model dynamics

✓ Updates control inputs/plan directly

- Requires access to resets

Easier for worst case performance analysis

Simplified Problem Setting

- Discrete-time Linearized Systems with fixed start x_0

$$x_{t+1} = Ax_t + Bu_t$$

- Approximate dynamics $\hat{A}, \hat{B}$ (e.g. from sysID)

$$||\hat{A} - A||_2 \leq \epsilon_A \text{ and } ||\hat{B} - B||_2 \leq \epsilon_B$$

- Minimize sum of quadratic costs, $J = \sum_{t=0}^{H-1} x_t^T Q x_t + u_t^T R u_t$

- Linear Quadratic Regulator (LQR) [Bertsekas 2005]

Optimal Controller in Closed Form

- For true dynamics A, B the optimal control is given by $u_t^\star = K_t^\star x_t$
- Optimal cost-to-go from time t is given by $x_t^T P_t^\star x_t$
- **Takeaway:** Optimal Linear Controller $K_t^\star$ and Quadratic Cost-to-go $P_t^\star$
- But we do not know A, B to compute this!

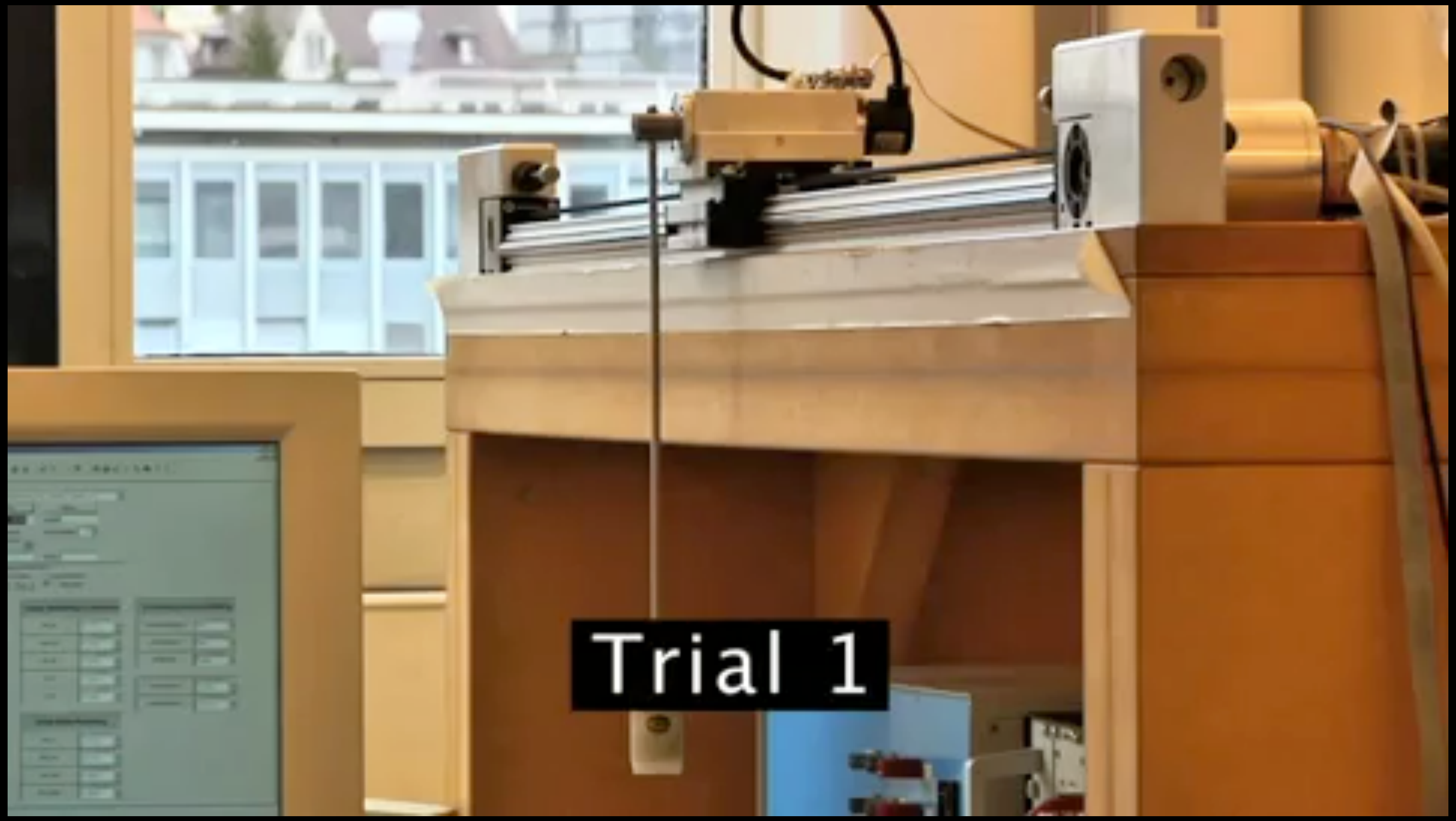
Naively Using Misspecified Model (Naive)

- Approximate dynamics $\hat{A}, \hat{B}$ also linear
- Results in a linear controller $\hat{K}_t$ and quadratic cost-to-go $\hat{P}_t$
- But suboptimal as $\hat{A}, \hat{B}$ are approximate
- Sub-optimality gap:

$$\hat{J} - J^\star = \sum_{t=0}^{H-1} c(\hat{x}_t, \hat{u}_t) - c(x_t^\star, u_t^\star)$$

Iterative Learning Control

Initialize controls $u_{0:H-1}$ using $\hat{A}, \hat{B}$
while not converged **do**
    Rollout $u_{0:H-1}$ **on real system** to get trajectory $x_{0:H}$
    Compute $\arg\min_{\Delta x, \Delta u} J(\Delta x, \Delta u)$ subject to $\underline{\hat{A}\Delta x_t + \hat{B}\Delta u_t = \Delta x_{t+1}}$
    Update $u_{0:H-1} = u_{0:H-1} + \alpha \Delta u_{0:H-1}$



Video from [Schoellig and D'Andrea 2009]

Iterative Learning Control

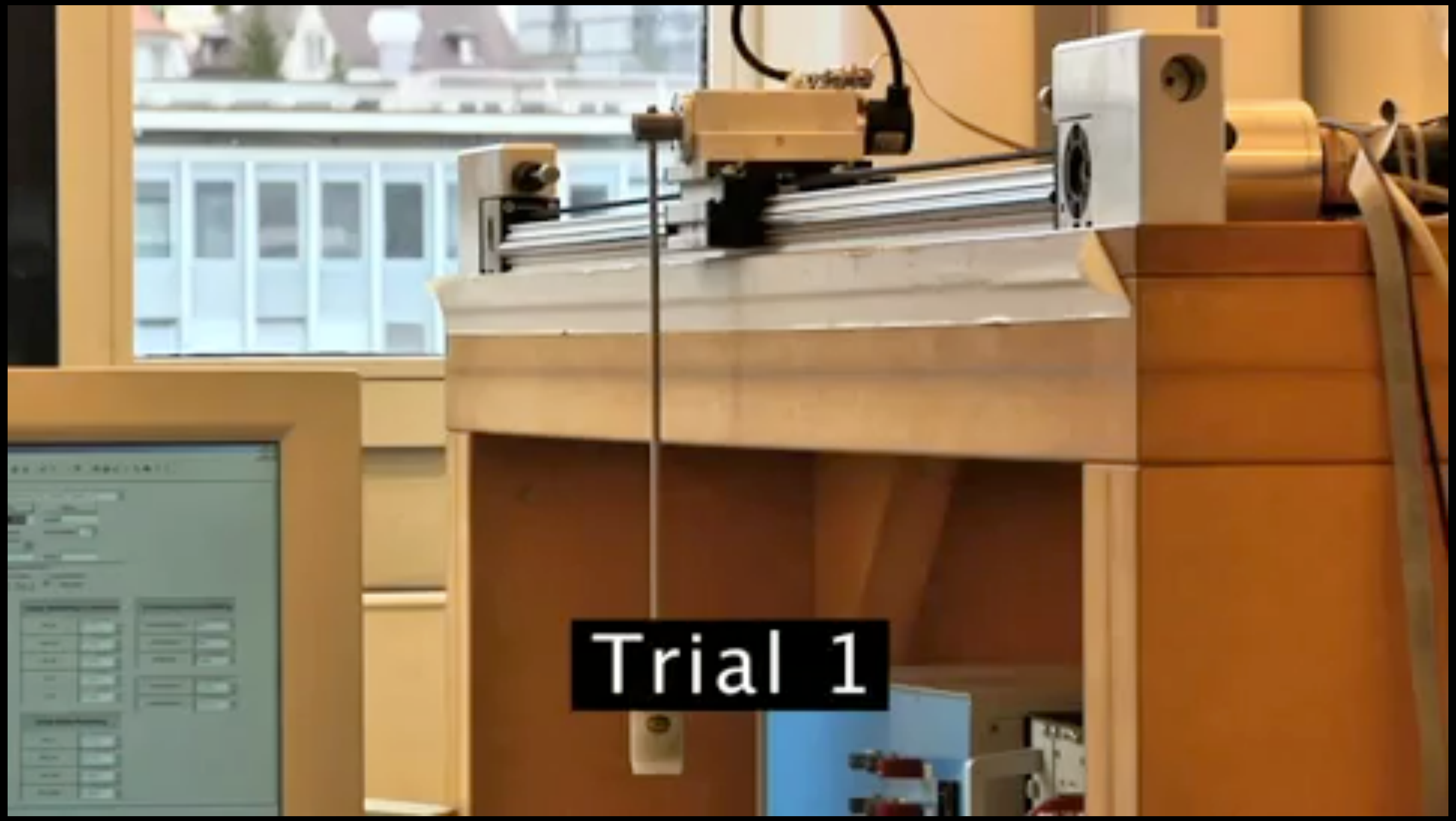
Initialize controls $u_{0:H-1}$ using $\hat{A}, \hat{B}$

while not converged **do**

  Rollout $u_{0:H-1}$ **on real system** to get trajectory $x_{0:H}$

  Compute $\arg\min_{\Delta x, \Delta u} J(\Delta x, \Delta u)$ subject to $\underline{\hat{A}\Delta x_t + \hat{B}\Delta u_t = \Delta x_{t+1}}$

  Update $u_{0:H-1} = u_{0:H-1} + \alpha \Delta u_{0:H-1}$



Video from [Schoellig and D'Andrea 2009]

Iterative Learning Control

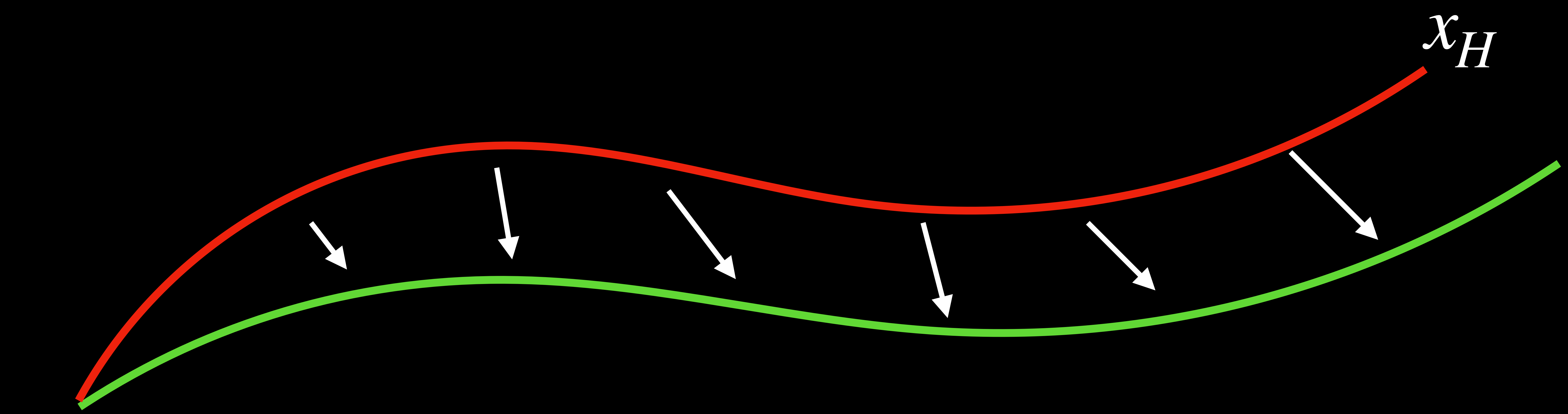
Initialize controls $u_{0:H-1}$ using $\hat{A}, \hat{B}$

while not converged **do**

  Rollout $u_{0:H-1}$ **on real system** to get trajectory $x_{0:H}$

  Compute $\arg\min_{\Delta x, \Delta u} J(\Delta x, \Delta u)$ subject to $\underline{\hat{A}\Delta x_t + \hat{B}\Delta u_t = \Delta x_{t+1}}$

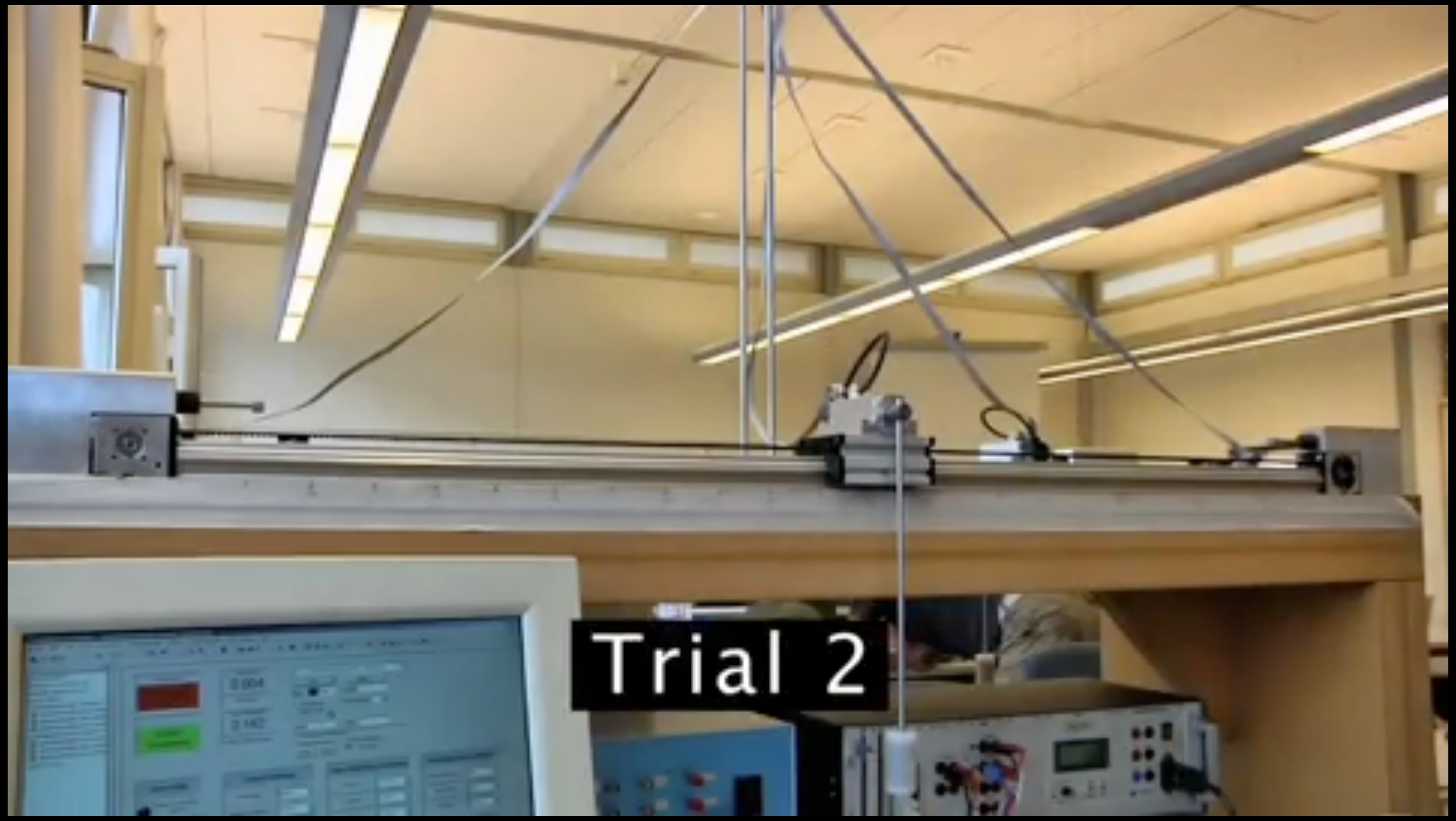
  Update $u_{0:H-1} = u_{0:H-1} + \alpha \Delta u_{0:H-1}$



Updates computed using **approximate** model around **real** trajectory

Iterative Learning Control

Initialize controls $u_{0:H-1}$ using $\hat{A}, \hat{B}$
while not converged **do**
    Rollout $u_{0:H-1}$ **on real system** to get trajectory $x_{0:H}$
    Compute $\arg\min_{\Delta x, \Delta u} J(\Delta x, \Delta u)$ subject to $\underline{\hat{A}\Delta x_t + \hat{B}\Delta u_t = \Delta x_{t+1}}$
    Update $u_{0:H-1} = u_{0:H-1} + \alpha \Delta u_{0:H-1}$



Video from [Schoellig and D'Andrea 2009]

Never updating the model!

ILC Controller

- Converges to a linear controller $\tilde{K}_t$ with quadratic cost-to-go $\tilde{P}_t$
- Still suboptimal as we rely on model to compute updates

In the worst case, is ILC as bad as naively using approximate model?

Recursive Bounds

- Coarsely, $\hat{J} - J^\star \leq O(1)\max\{\epsilon_A^2, \epsilon_B^2, \|\hat{P}_0 - P_0^\star\|^2\}$ and similar for ILC
- Naively using inaccurate model

$$\|\hat{P}_t - P_t^\star\| \leq O(\epsilon_A + \epsilon_B + \epsilon_A^2 + \epsilon_B^2) + O(1 + \epsilon_A + \epsilon_A^2)\|P_{t+1}^\star - \hat{P}_{t+1}\|$$

- Iterative Learning Control

$$\|\tilde{P}_t - P_t^\star\| \leq O(\epsilon_A + \epsilon_B) + O(1 + \epsilon_A)\|P_{t+1}^\star - \tilde{P}_{t+1}\|$$

- **Takeaway**: Higher-order terms are significant when ϵ_A, ϵ_B are large

Case Study 1: Small Modeling Errors

- Small ϵ_A, ϵ_B
- Can ignore higher-order terms in Naive approach's upper bound
- Similar worst case performance: $\hat{J} - J^\star \approx \tilde{J} - J^\star$
- Model is a very good approximation of real dynamics

Case Study 2: Highly Damped Systems

- Small $||A||$, i.e. state goes down to zero quickly
- The sub-optimality gap for ILC shrinks significantly:

$$||\tilde{P}_t - P_t^\star|| \leq O(1)$$

- The Naive approach incurs significant error in higher-order terms:

$$||\hat{P}_t - P_t^\star|| \leq \boxed{O(\epsilon_A^2)} + O(1)||\hat{P}_{t+1} - P_{t+1}^\star||$$

- ILC focuses on minimizing cost in the first few time-steps

Case Study 3: Weakly Controlled Systems

- Small $\|B\|$, i.e. controls do not affect dynamics
- ILC error does not depend on ϵ_B - robust to modeling errors in B

$$\|\tilde{P}_t - P_t^\star\| \leq O(\epsilon_A) + O(1 + \epsilon_A)\|\tilde{P}_{t+1} - P_{t+1}^\star\|$$

- But the naive approach pays penalty in ϵ_B^2

$$\|\hat{P}_t - P_t^\star\| \leq O(\epsilon_B^2 + \epsilon_A + \epsilon_A^2) + O(1 + \epsilon_A + \epsilon_A^2)\|\hat{P}_{t+1} - P_{t+1}^\star\|$$

- ILC realizes inefficacy of controls and minimizes control costs

Experiments

1. Toy Linear Dynamical System with Approximate Model
2. Nonlinear Inverted Pendulum with Misspecified Mass
3. Nonlinear Quadrotor Control in Wind

Small Modeling Errors - ILC $\approx$ Naive

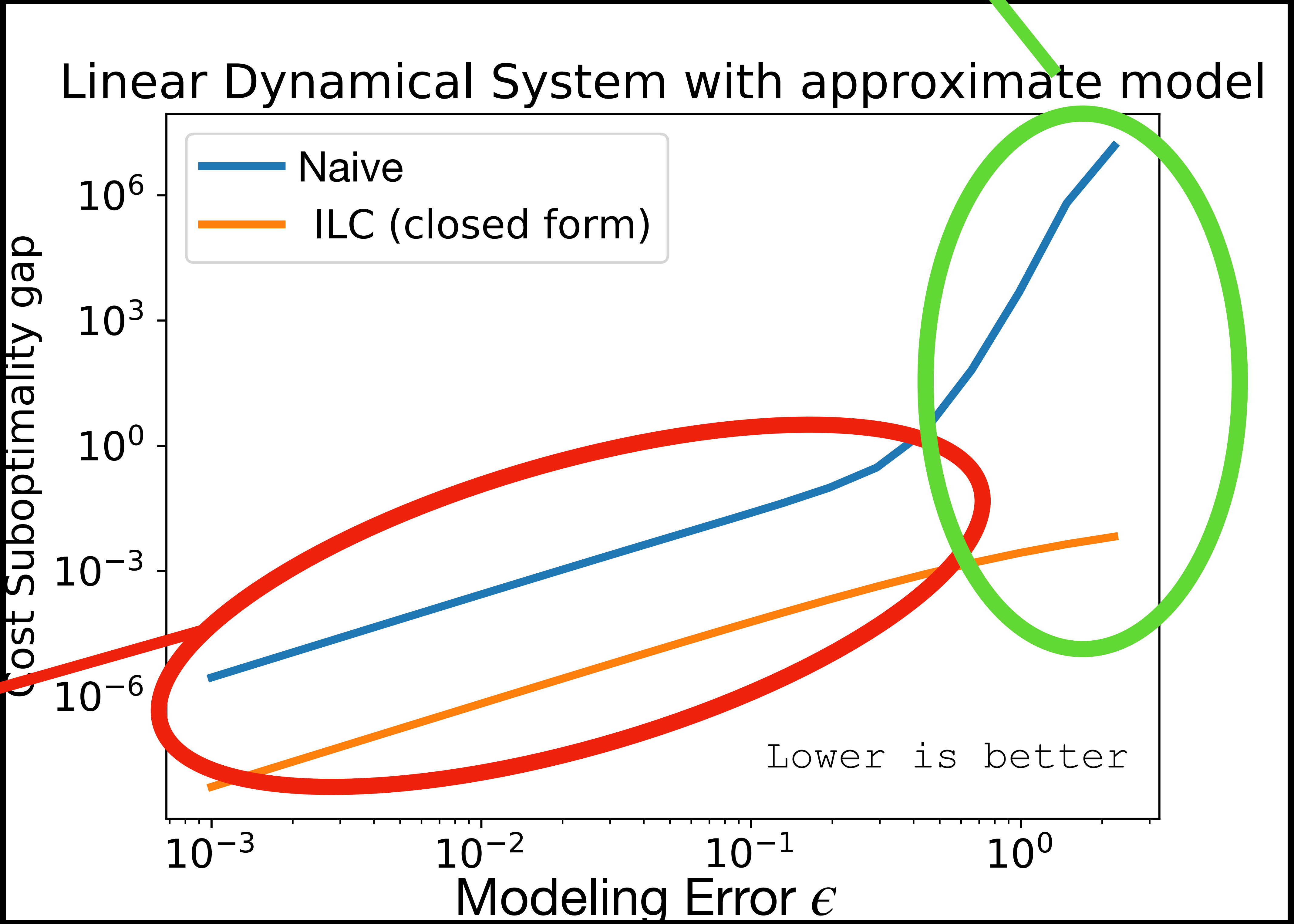
Large Modeling Errors - ILC significantly better than Naive

Experiment 1

Toy Linear Dynamical System

- $x \in \mathbb{R}^2, u \in \mathbb{R}$
- $\hat{A} = A + \epsilon I, \hat{B} = B + \epsilon \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Large modeling errors - significant improvement



Small modeling errors - constant improvement

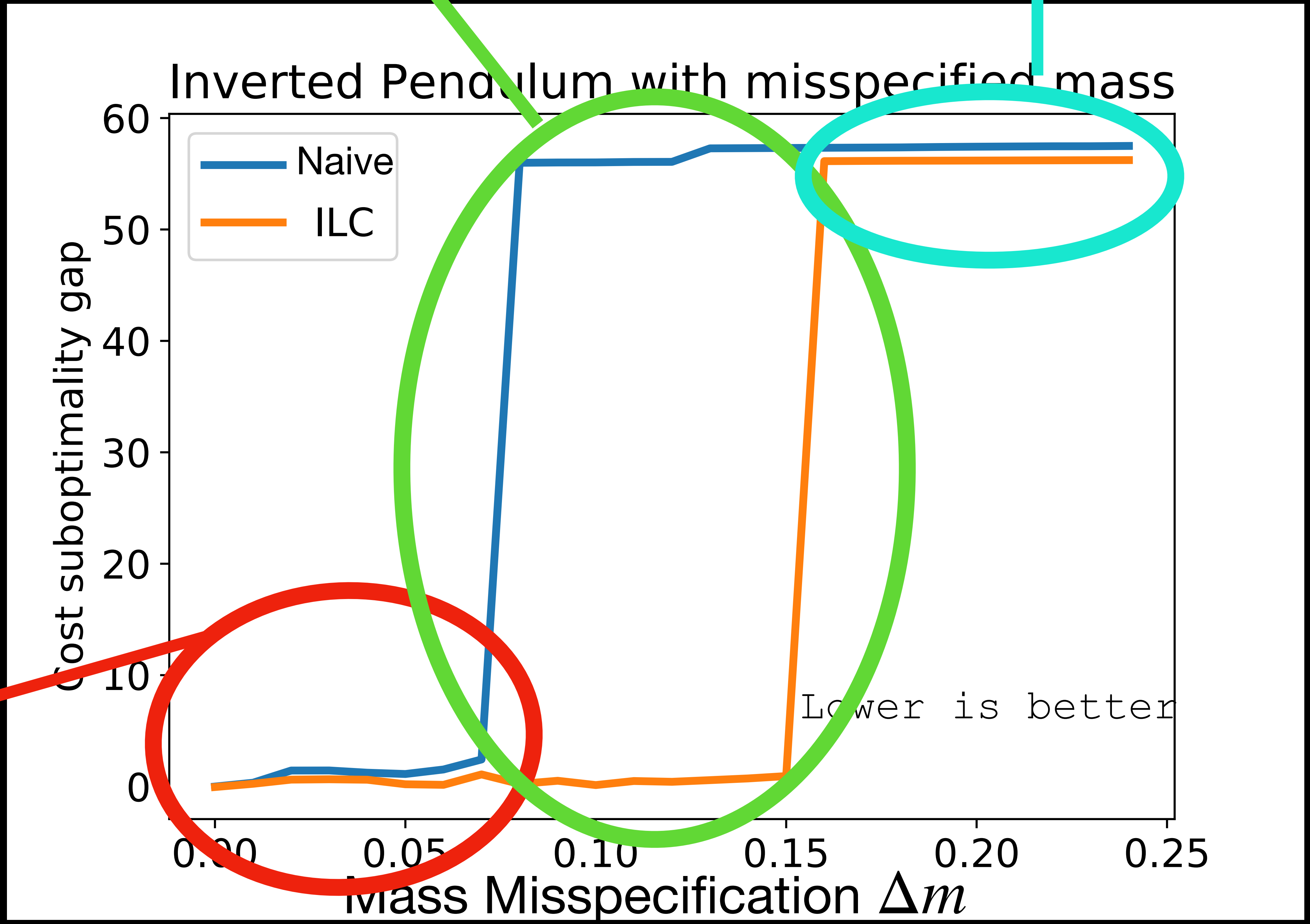
Experiment 2

Inverted Pendulum with Misspecified Mass

- Nonlinear dynamics
- Unknown mass m
- Access to model with $\hat{m} = m + \Delta m$ (misspecification)
- Same trend in nonlinear systems!

Too large modeling errors

Large modeling errors - significant improvement



Small modeling errors - constant improvement

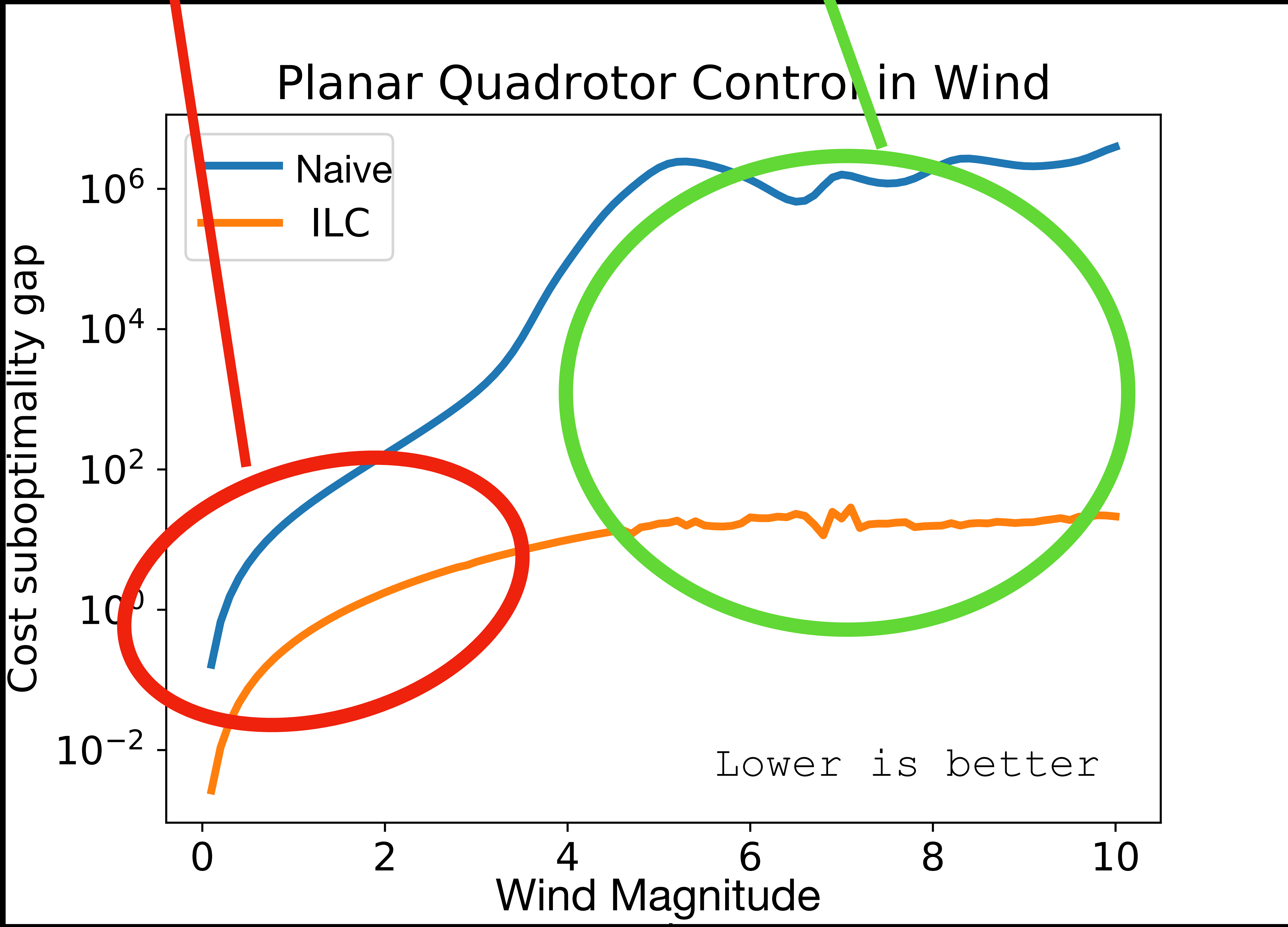
Experiment 3

Planar Quadrotor Control in Wind

Large modeling errors - significant improvement

Small modeling errors - constant improvement

- Nonlinear Dynamics
- Dynamics affected by a wind force field (external disturbance)



Video captured using simulator from Alex Spitzer

Summary

On the Effectiveness of Inaccurate Models

- Naive use can result in highly suboptimal performance
- ILC cancels out errors by evaluating on real system
- Absence of significant higher-order terms
- For highly damped and weakly controlled systems
  - ILC is provably more efficient than naively using inaccurate models

Thesis Contributions

ANALYSIS

Effectiveness of Using Inaccurate Models

[Under review]

Model-Free RL Requires Large Number of Samples

[AISTATS 2019]

ALGORITHMS

CMAX : Bias Planner Away From Inaccurately Modeled Regions

[RSS 2020]

CMAX++ : Learn to Exploit Inaccurately Modeled Regions

[AAAI 2021]

TOMS : Update Model to be Useful for Planning

[Chapter 7 in Thesis]

Future Work Directions

1. Unified Framework for Planning and Execution
2. Extending CMAX and CMAX++ to Stochastic Dynamics
3. Finite Data Performance Analysis

Future Work Direction 1

Unified Framework for Planning and Execution

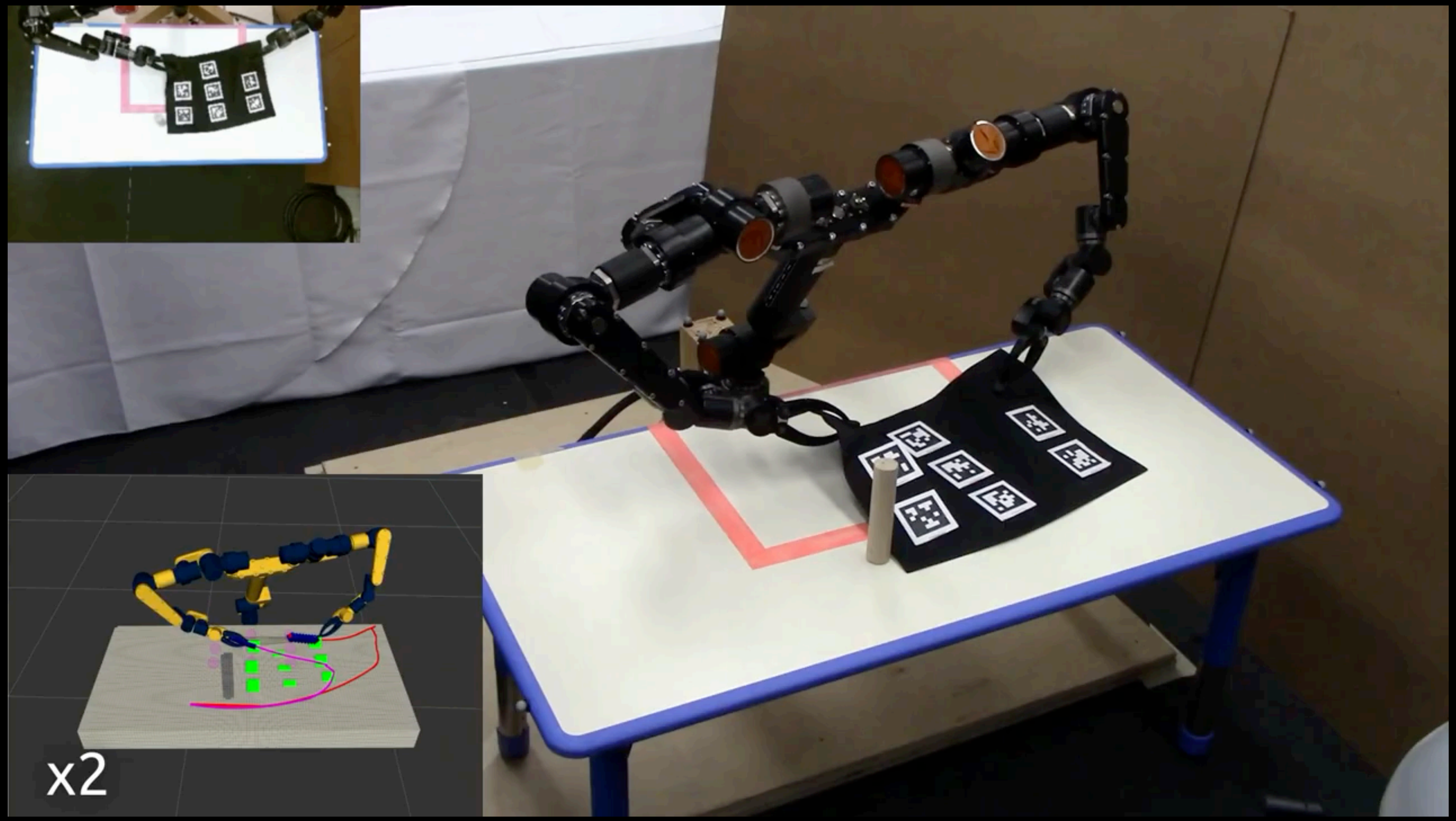
- Challenges:

  1. Model Learning : Build models that help future planning [Chapter 7 in Thesis]

  2. Completeness with learned models

     [Chapter 7 in Thesis]

- Switch between CMAX, CMAX++ and updating the model, during execution

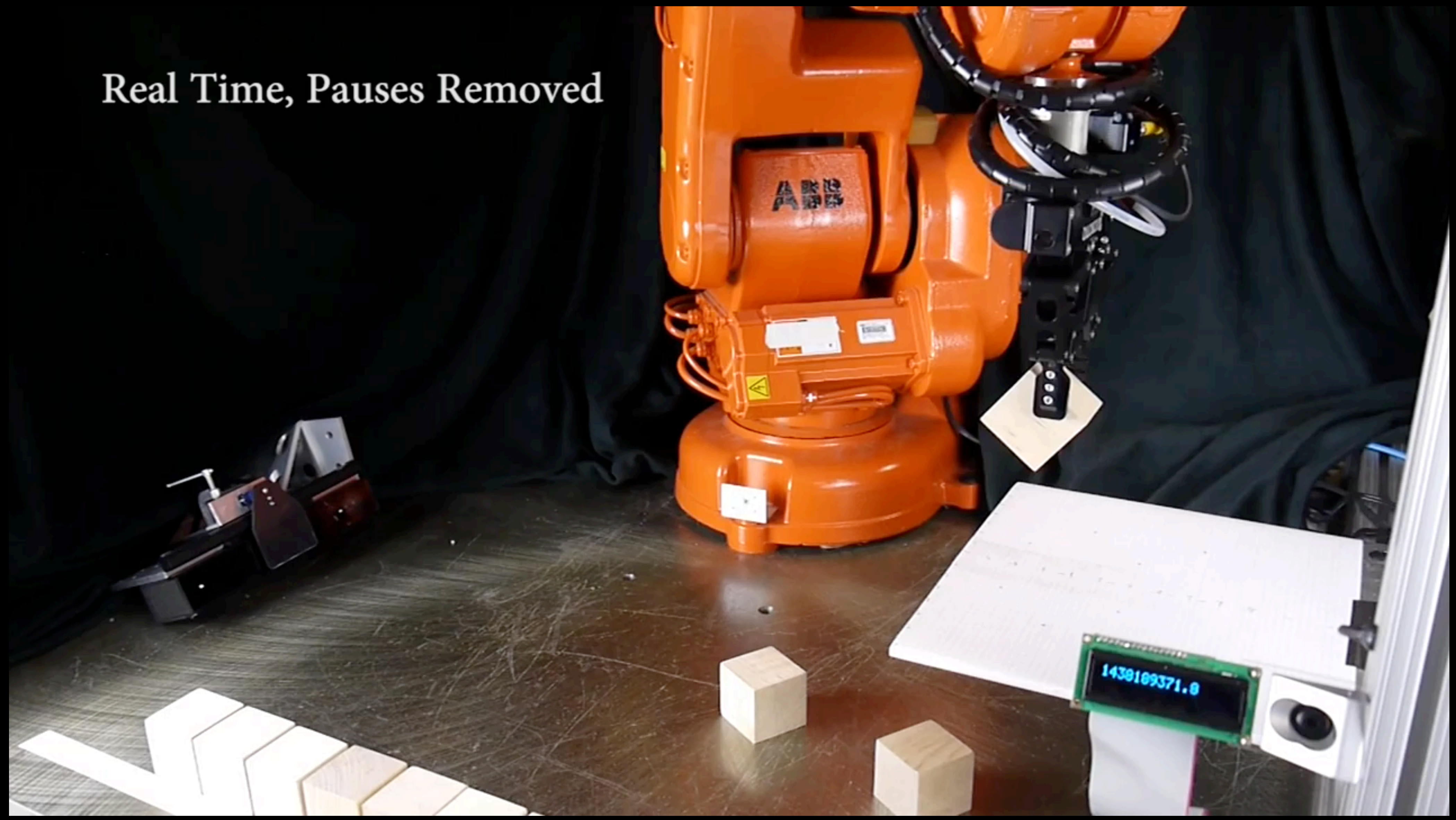


Video from [McConcachie et. al. 2020]

Future Work Direction 2

Extending CMAX and CMAX++ to Stochastic Dynamics

- Challenges:
    1. Planning: MDP planners, Stochastic motion roadmaps [Alterovitz et. al. 2007]
    2. Inaccurate Transitions: Maintain uncertainty estimates [Kidambi et. al. 2020, Yu et. al. 2020]



Video from [Paolini and Mason 2016]

Future Work Direction 3

Finite Data Performance Analysis

What performance can we expect using approximate dynamics and finite amount of experience from N rollouts?

- Regret w.r.t optimal robust controller K^* across N rollouts [Dean et. al. 2019]

$$\text{Regret} = \sum_{i=1}^{N} J_i - \sum_{i=1}^{N} J(K^*)$$

Conclusion

By updating the behavior of the planner and not the dynamics of the model,

we can leverage simplified and potentially inaccurate models,

and significantly reduce the amount of experience required to complete the task

Acknowledgements

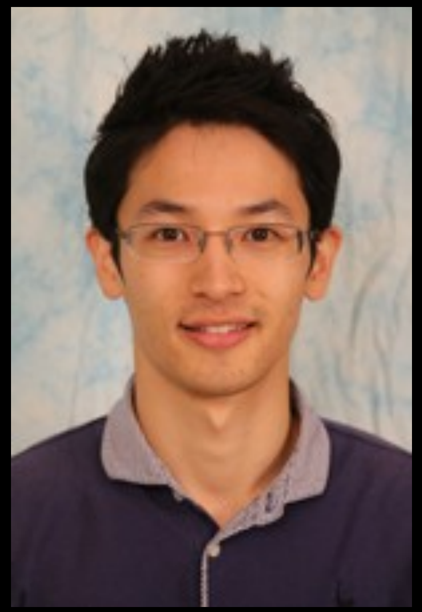
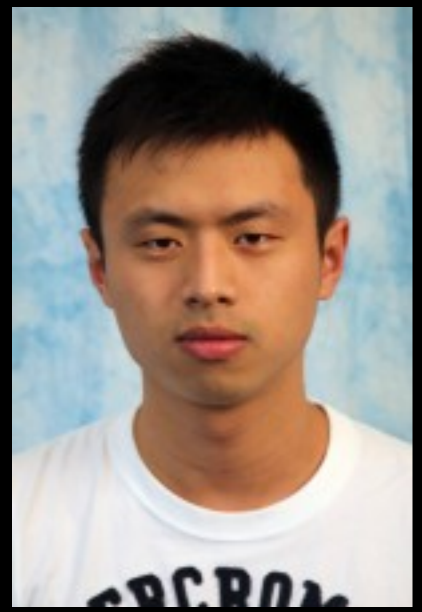
Advisors


Committee


Labs

Collaborators

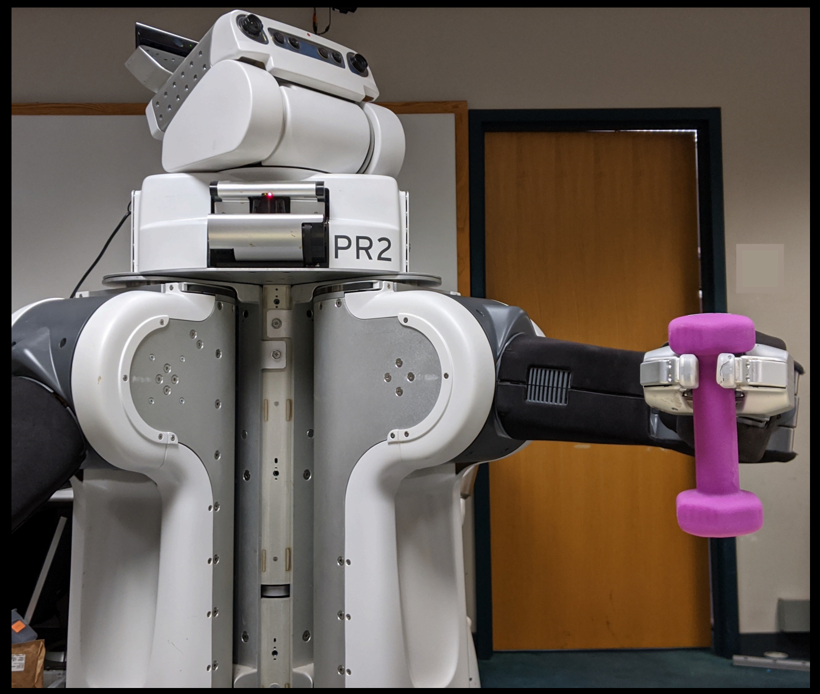







Robots



Thesis Contributions

ANALYSIS

Model-Free RL Requires Large Number of Samples

[AISTATS 2019]

Effectiveness of Using Inaccurate Models

[Under review]

ALGORITHMS

CMAX : Bias Planner Away From Inaccurately Modeled Regions

[RSS 2020]

CMAX++ : Learn to Exploit Inaccurately Modeled Regions

[AAAI 2021]

TOMS : Update Model to be Useful for Planning

[Chapter 7 in Thesis]

References

- Sutton, Richard S. "Dyna, an integrated architecture for learning, planning, and reacting." *ACM Sigart Bulletin* 2.4 (1991): 160-163.
- Akkaya, Ilge, et al. "Solving Rubik's Cube with a Robot Hand." *arXiv preprint arXiv:1910.07113* (2019).
- Phillips, Mike, and Maxim Likhachev. "Sipp: Safe interval path planning for dynamic environments." *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011.
- Zhu, Shaojun, et al. "Fast model identification via physics engines for data-efficient policy search." *arXiv preprint arXiv:1710.08893* (2017).
- Schrittwieser, Julian, et al. "Mastering atari, go, chess and shogi by planning with a learned model." *arXiv preprint arXiv:1911.08265* (2019).
- Nagabandi, Anusha, et al. "Deep Dynamics Models for Learning Dexterous Manipulation." *arXiv preprint arXiv:1909.11652* (2019).
- Zeng, Andy, et al. "Tossingbot: Learning to throw arbitrary objects with residual physics." *arXiv preprint arXiv:1903.11239* (2019).
- Janner, Michael, et al. "When to trust your model: Model-based policy optimization." *Advances in Neural Information Processing Systems*. 2019.
- Saveriano, Matteo, et al. "Data-efficient control policy search using residual dynamics learning." *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.
- Rastogi, Divyam, Ivan Koryakovskiy, and Jens Kober. "Sample-efficient reinforcement learning via difference models." *Machine Learning in Planning and Control of Robot Motion Workshop at ICRA*. 2018.
- Mordatch, Igor, Kendall Lowrey, and Emanuel Todorov. "Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids." *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015.
- Bagnell, J. Andrew, and Jeff G. Schneider. "Autonomous helicopter control using reinforcement learning policy search methods." *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 2. IEEE, 2001.
- Brafman, Ronen I., and Moshe Tennenholtz. "R-max-a general polynomial time algorithm for near-optimal reinforcement learning." *Journal of Machine Learning Research* 3.Oct (2002): 213-231.
- Zucker, Matt, et al. "Optimization and learning for rough terrain legged locomotion." *The International Journal of Robotics Research* 30.2 (2011): 175-191.
- Koenig, Sven, and Maxim Likhachev. "Real-time adaptive A." Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. 2006.
- Silver, David, et al. "Mastering the game of go without human knowledge." *Nature* 550.7676 (2017): 354-359

References

- Optimization-based iterative learning control for trajectory tracking A. P. Schoellig and R. D'Andrea in Proc. of the European Control Conference (ECC), 2009, pp. 1505-1510.
- Manipulating Deformable Objects by Interleaving Prediction, Planning, and Control Dale McConachie, Andrew Dobson, Mengyao Ruan, and Dmitry Berenson International Journal of Robotics Research (IJRR), vol. 39, no. 8, pp. 957–982, July 2020.
- Perception and Motion Planning for Pick-and-Place of Dynamic Objects A. Cowley, B. Cohen, W. Marshall, C.J. Taylor, and M. Likhachev Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), 2013.
- Williams, G., Wagener, N., Goldfain, B., Drews, P., Rehg, J. M., Boots, B., & Theodorou, E. A. (2017, May). Information theoretic MPC for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1714-1721). IEEE.
- Sun, W., Jiang, N., Krishnamurthy, A., Agarwal, A., & Langford, J. (2019, June). Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory* (pp. 2898-2933).
- Barto, A. G., Bradtke, S. J., & Singh, S. P. (1995). Learning to act using real-time dynamic programming. *Artificial intelligence*, *72*(1-2), 81-138.
- Nagabandi, A., Kahn, G., Fearing, R. S., & Levine, S. (2018, May). Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 7559-7566). IEEE.
- Farshidian, F., Neunert, M., & Buchli, J. (2014, September). Learning of closed-loop motion control. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1441-1446). IEEE.
- Lee, M. A., Florensa, C., Tremblay, J., Ratliff, N., Garg, A., Ramos, F., & Fox, D. (2020). Guided Uncertainty-Aware Policy Optimization: Combining Learning and Model-Based Strategies for Sample-Efficient Policy Learning. *arXiv preprint arXiv:2005.10872*.
- LaGrassa, A., Lee, S., & Kroemer, O. (2020). Learning Skills to Patch Plans Based on Inaccurate Models. *arXiv preprint arXiv:2009.13732*.
- Meier, F., Hennig, P., & Schaal, S. (2014). Incremental local gaussian regression. In *Advances in Neural Information Processing Systems* (pp. 972-980).
- Farahmand, A. M. (2018). Iterative value-aware model learning. In *Advances in Neural Information Processing Systems* (pp. 9072-9083).
- Agarwal, N., Bullins, B., Hazan, E., Kakade, S. M., & Singh, K. (2019). Online control with adversarial disturbances. *arXiv preprint arXiv:1902.08721*.
- Dean, S., Mania, H., Matni, N., Recht, B., & Tu, S. (2019). On the sample complexity of the linear quadratic regulator. *Foundations of Computational Mathematics*, 1-47.

References

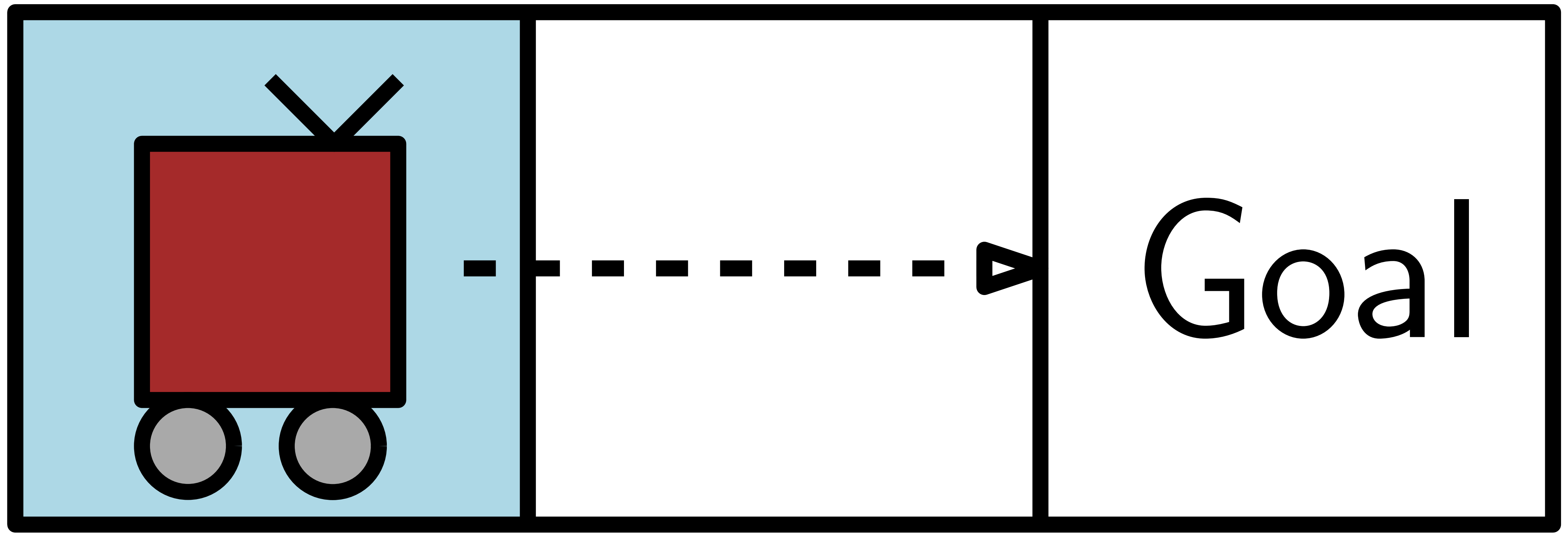
- Miki, Takahiro, et al. "Learning robust perceptive locomotion for quadrupedal robots in the wild." *Science Robotics* 7.62 (2022): eabk2822.
- McConachie, Dale, et al. "Learning when to trust a dynamics model for planning in reduced state spaces." *IEEE Robotics and Automation Letters* 5.2 (2020): 3540-3547.
- Mitrano, P., D. McConachie, and D. Berenson. "Learning where to trust unreliable models in an unstructured world for deformable object manipulation." *Science Robotics* 6.54 (2021): eabd8170.
- Power, Thomas, and Dmitry Berenson. "Keep It Simple: Data-Efficient Learning for Controlling Complex Systems With Simple Models." *IEEE Robotics and Automation Letters* 6.2 (2021): 1184-1191.
- Kidambi, Rahul, et al. "Morel: Model-based offline reinforcement learning." *arXiv preprint arXiv:2005.05951* (2020).
- Yu, Tianhe, et al. "MOPO: Model-based Offline Policy Optimization." *Advances in Neural Information Processing Systems* 33 (2020): 14129-14142.
- Paolini, Robert, and Matthew T. Mason. "Data-driven statistical modeling of a cube regrasp." *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.

Spectrum of approaches

Our goal-driven approach CMAX

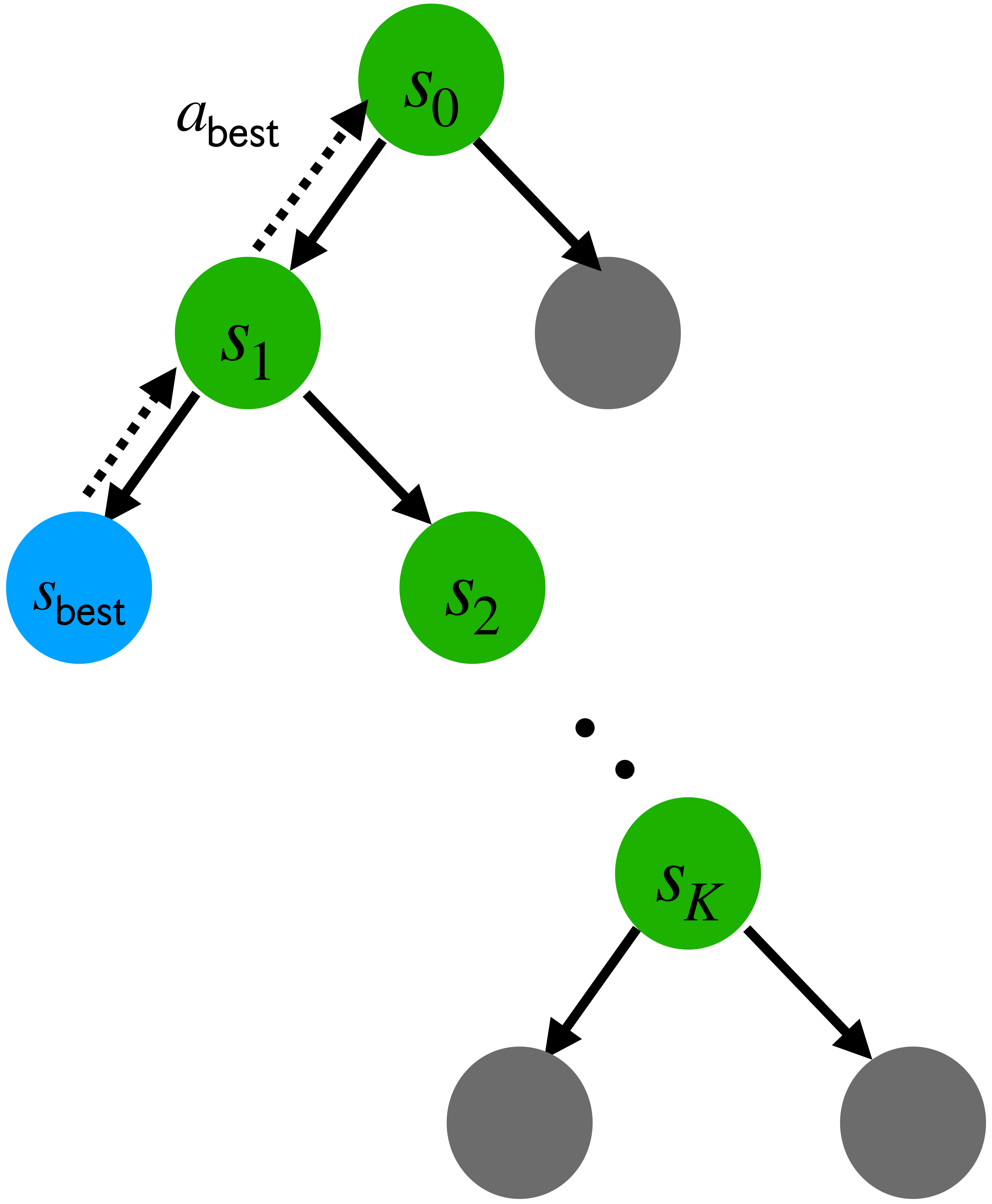
Update approximate dynamical model	Learn a residual dynamical model	Learn a dynamical model from scratch
Ensemble-CIO [Mordatch et. al. 2015]	TossingBot [Zeng et. al. 2019]	MBPO [Janner et. al. 2019]
[Bagnell and Schneider 2001]	PI-REM [Saveriano et. al. 2017]	PDDM [Nagabandi et. al. 2019]
DYNA [Sutton 1991]	[Rastogi et. al. 2018]	RMAX [Brafman et. al. 2002]

Subtle case for CMAX assumption



CMAX: Practical Algorithm for Large State Spaces

- **<u>Challenge 1</u>: Planning to goal is expensive**
  - Limited-expansion search as a planner
  - Best action by backtracking from best leaf after K expansions
  - Update value estimates of expanded states

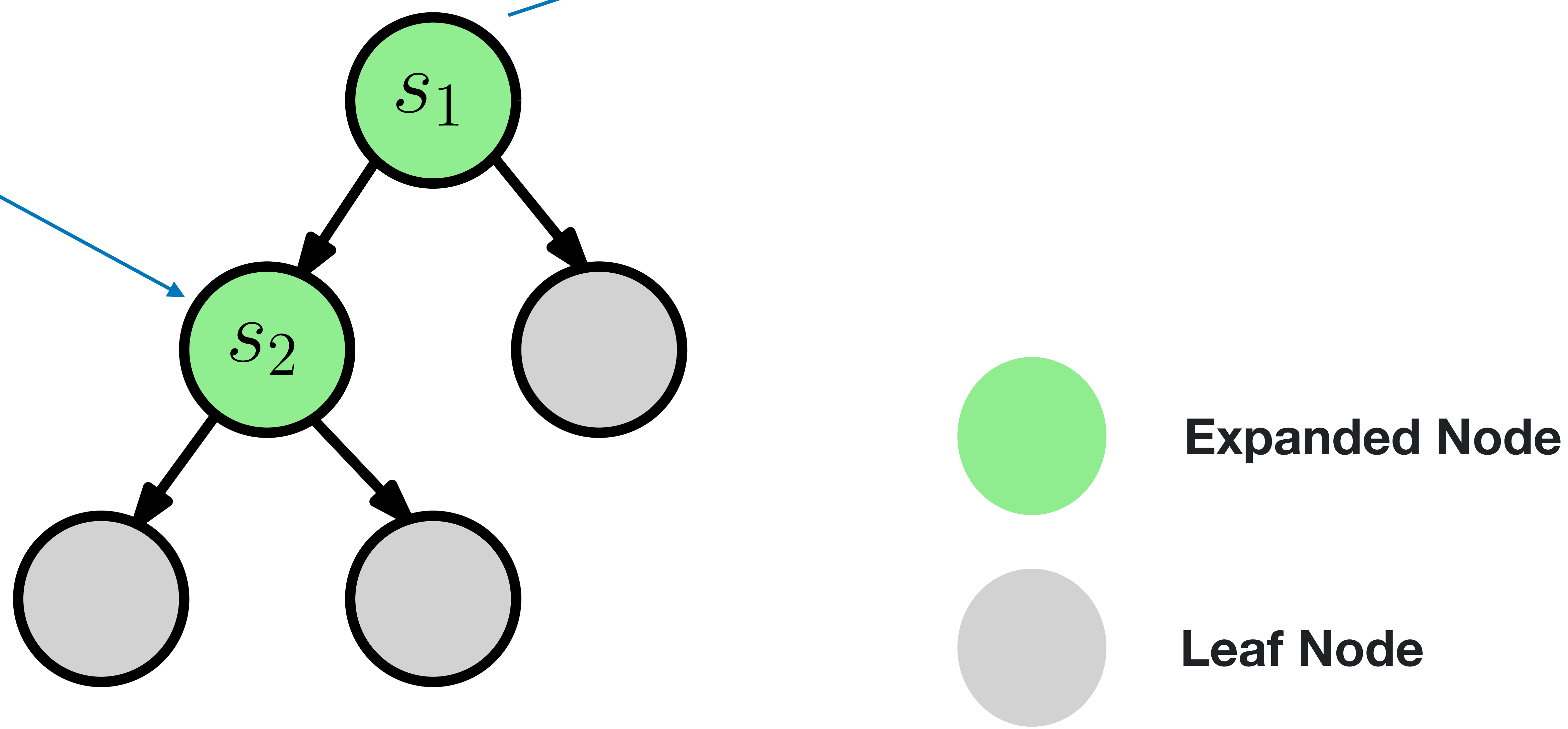


Limited-Expansion Search
Stage I: compute best action

Choose the next node to expand acc. to

$$\min_{s \in \text{leaves}} g(s) + V(s)$$

Current State



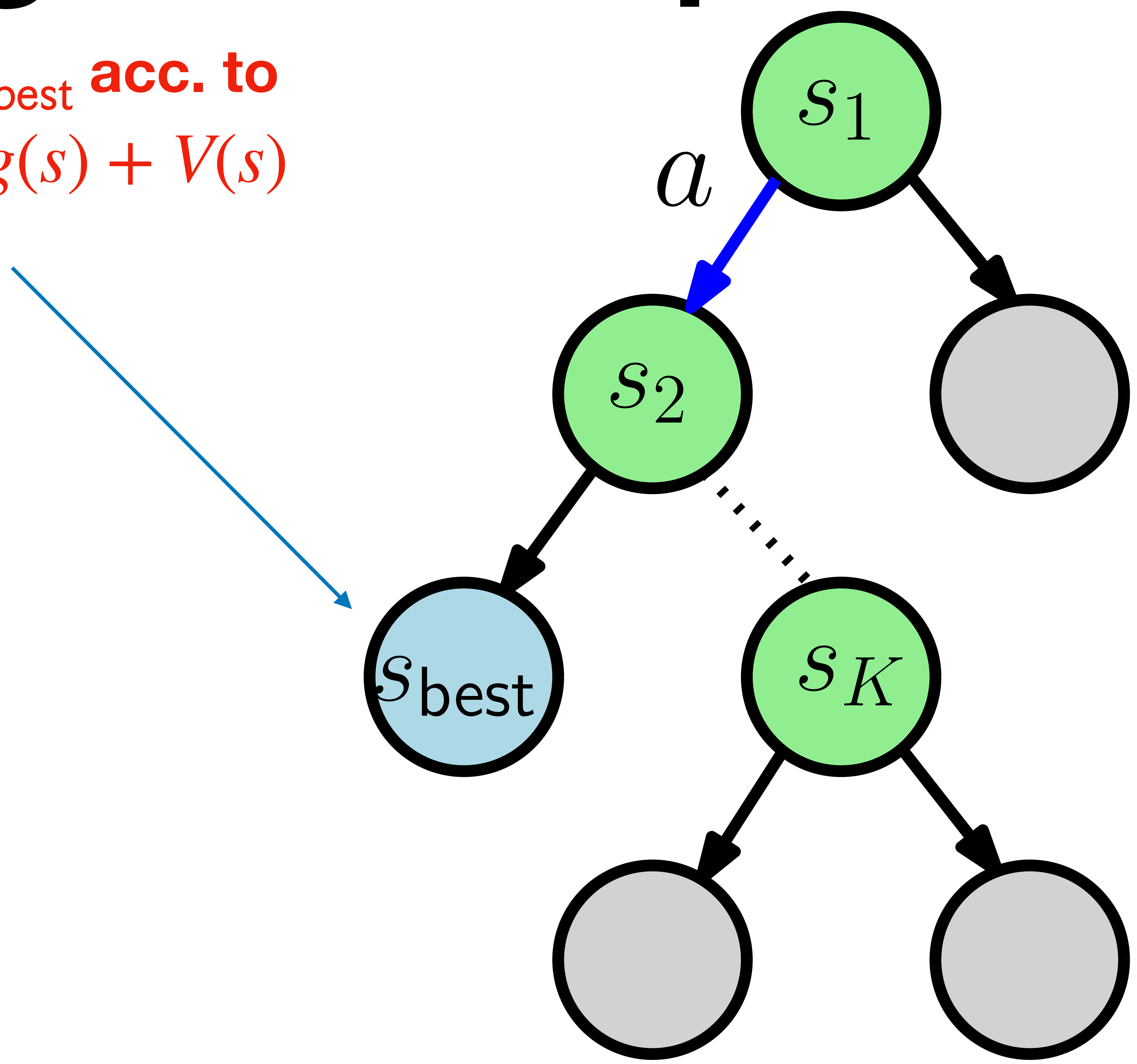
$g(s)$ = **Cost-to-come to s from s_1**

$V(s)$ = **Estimate of cost-to-go (or value/heuristic) from s to any goal**

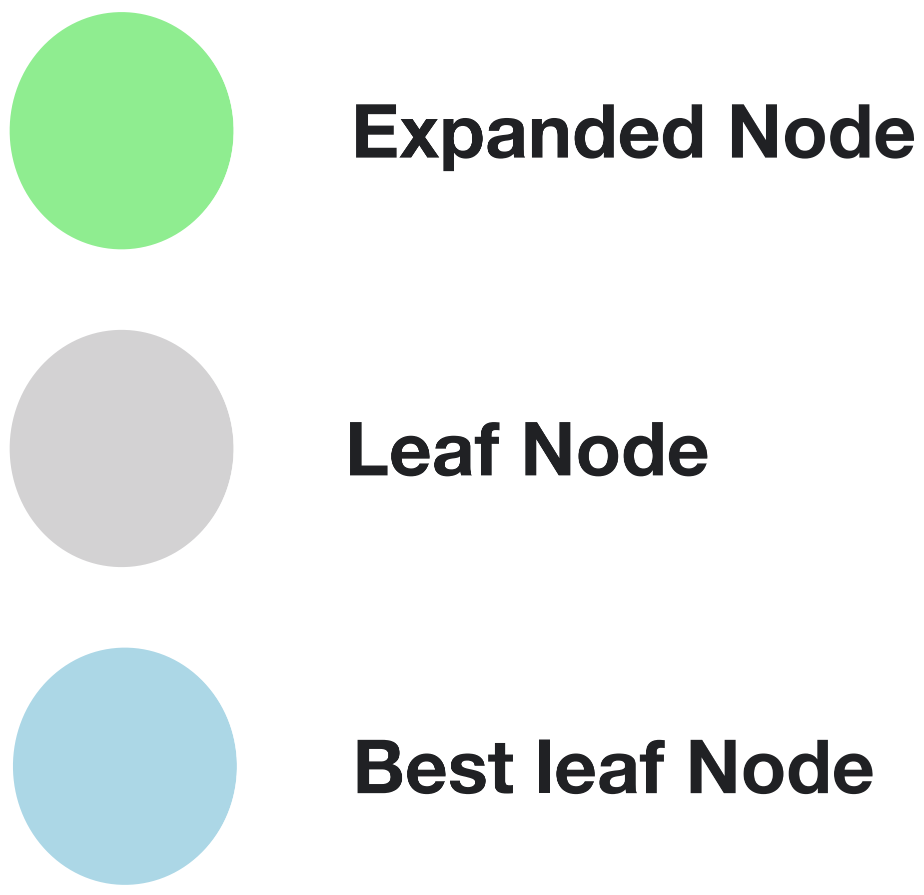
Limited-Expansion Search
Stage I: compute best action

Choose s_{best} acc. to

$$\min_{s \in \textbf{leaves}} g(s) + V(s)$$



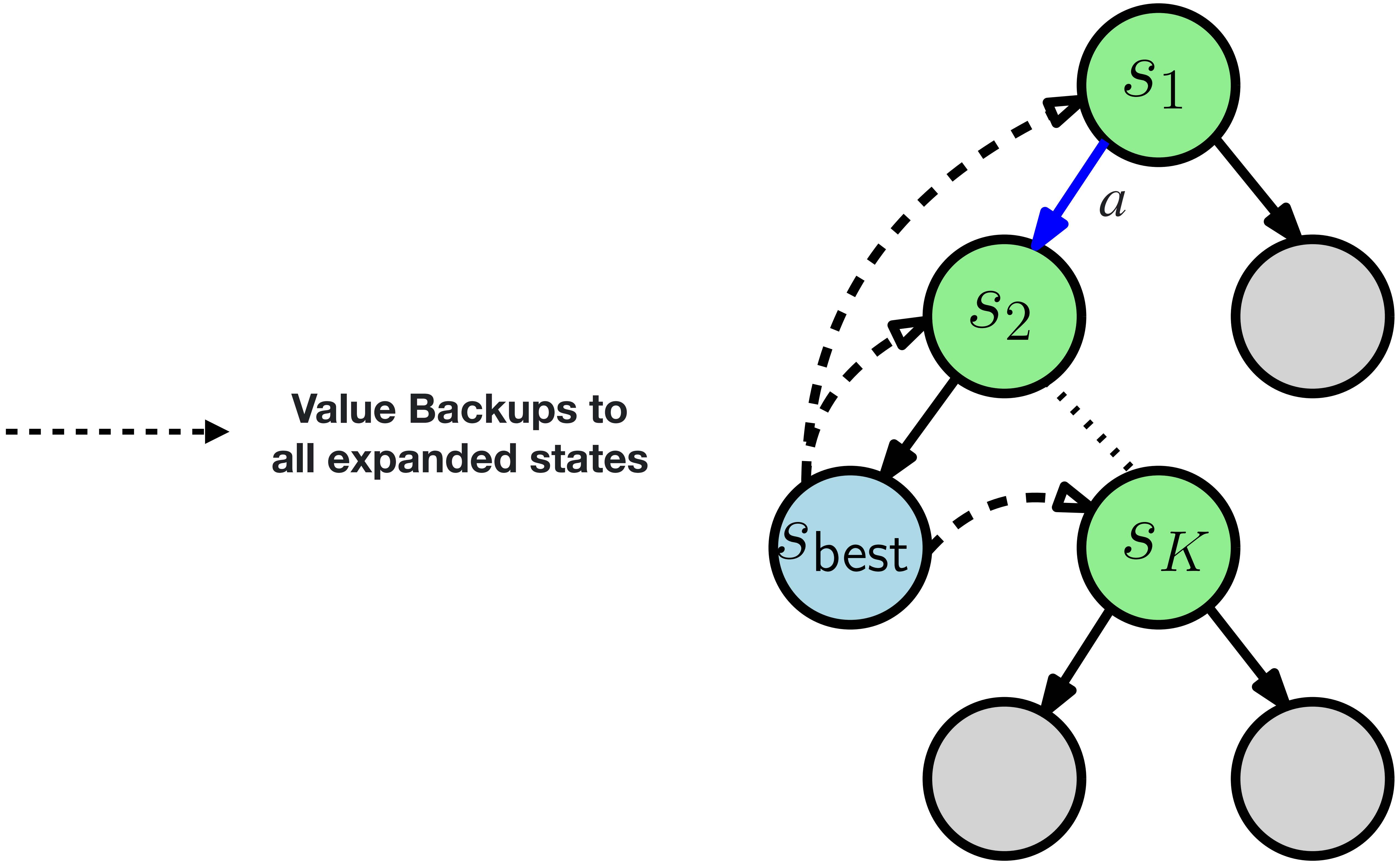
K - Number of Expansions



Expand K states and choose the best leaf node

Backtrack from s_{best} to s_1 to get the best action a

Limited-Expansion Search
Stage II: update value estimates

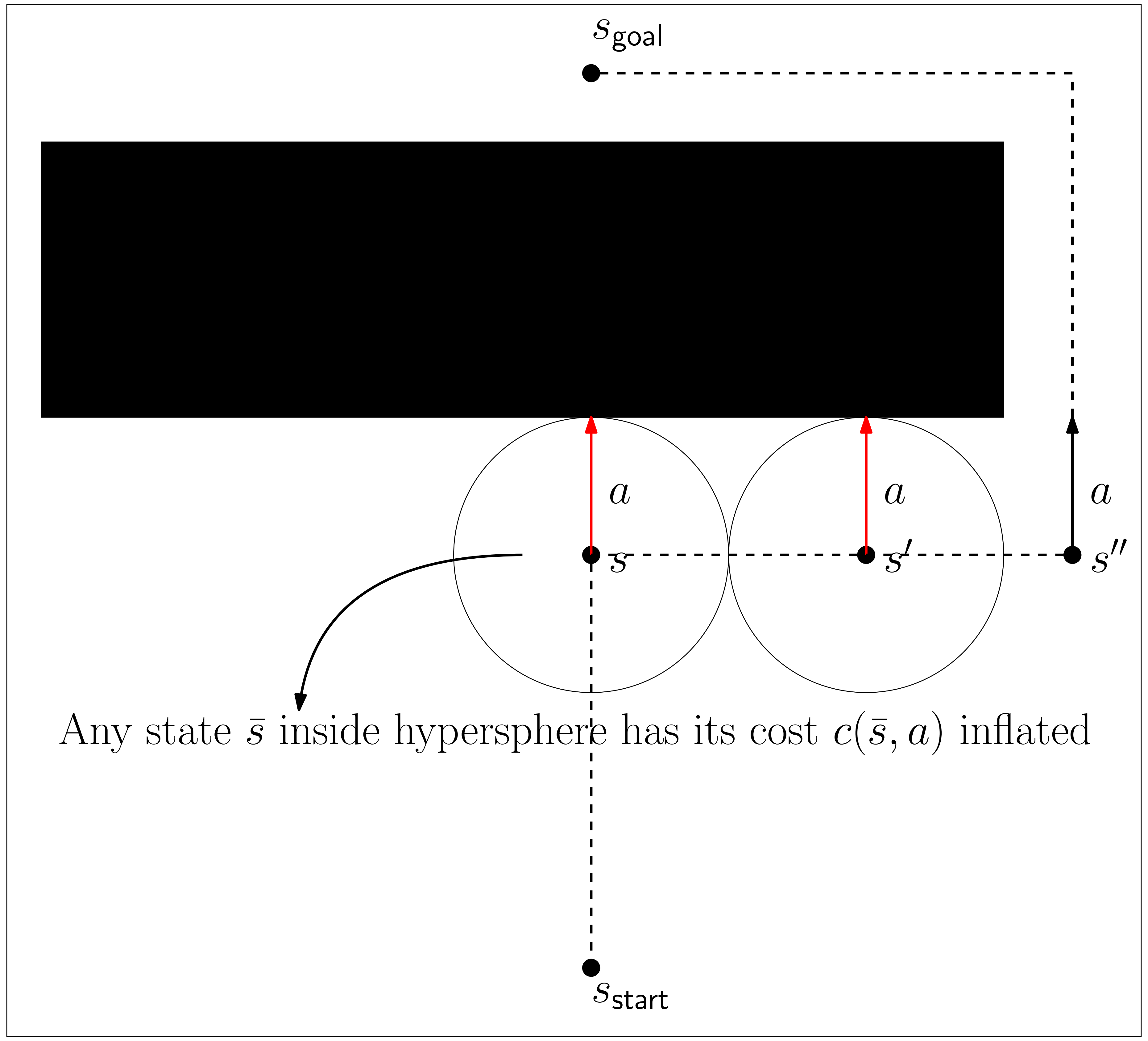


Update value estimates of all the expanded states

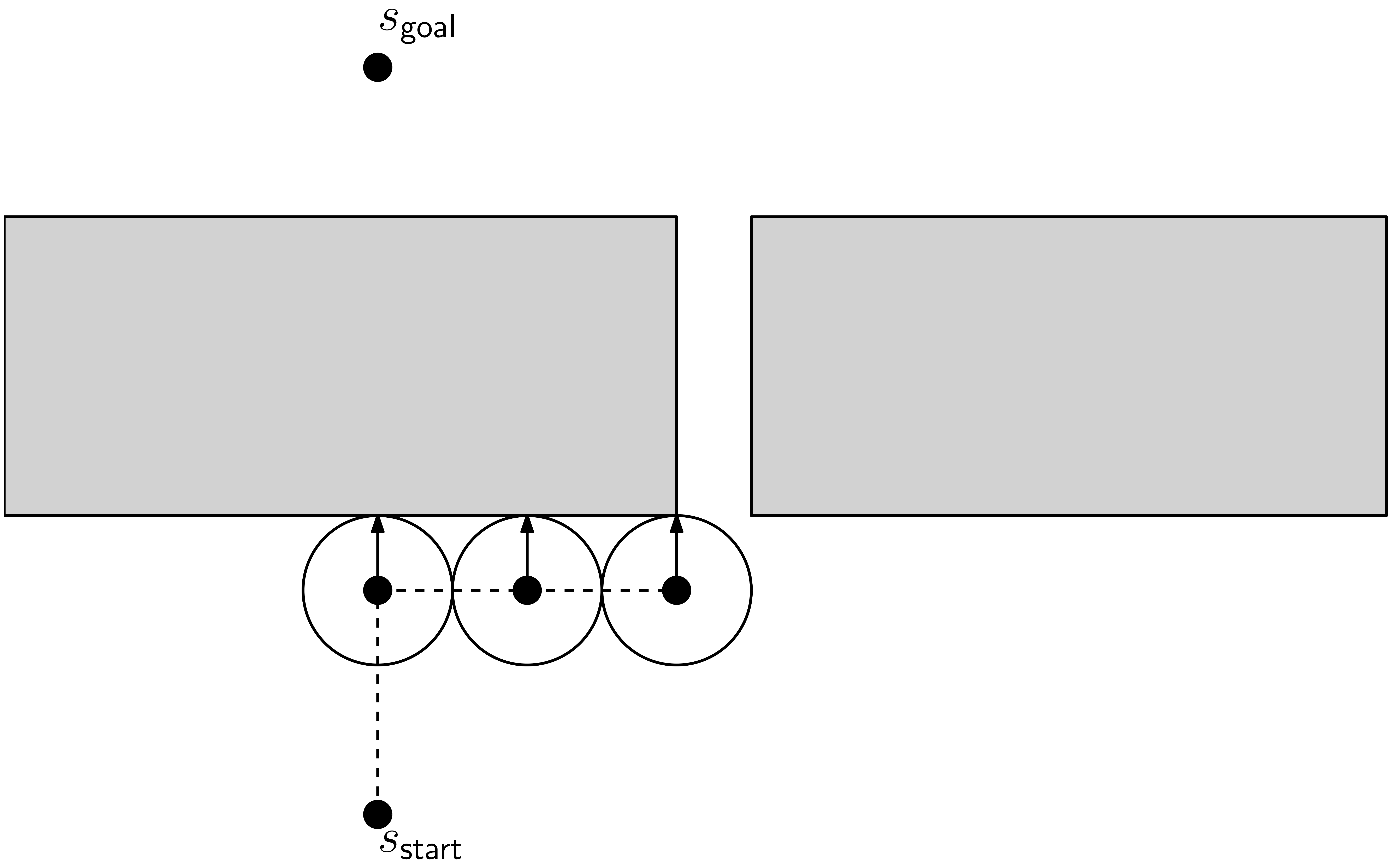
$$V(s_i) \leftarrow g(s_{\text{best}}) + V(s_{\text{best}}) - g(s_i)$$

CMAX: Practical Algorithm for Large State Spaces

- **<u>Challenge 2</u>: Cannot maintain values and incorrect set $\mathcal{X}_t$ as table**
  - Global Function Approximation for values: $V_\theta : \mathbb{S} \to \mathbb{R}, \theta \in \mathbb{R}^n$
  - Local Function Approximation for $\mathcal{X}_t$: Hyperspheres and KD-Trees in $\mathbb{S}$



Failure Case



Theoretical Guarantees under exact planning

- **<u>Assumption</u>**: There always exists a path from current state s_t to a goal state that is δ distance away from any transition that is known to be ξ-incorrect i.e. $(s, a) \in \mathcal{X}_t^\xi$

- **<u>Guarantee</u>:** If initial value estimates are admissible and consistent, the robot is guaranteed to reach a goal state in at most $|\mathbb{S}|^2$ time steps. (Completeness)

- If we do $K = |\mathbb{S}|$ expansions then, the robot is guaranteed to reach in $|\mathbb{S}|(\mathscr{C}(\delta) + 1)$ time steps

Proof Sketch

- RTAA* is guaranteed to reach the goal state
- Assumption ensures that there always exists a path from the current state to goal state in penalized model $\tilde{M}_{\mathscr{X}}$
- Thus, CMAX is also guaranteed to reach the goal
- Number of steps to discover all incorrect transitions is $|\mathbb{S}||\mathscr{X}|$
- Once we discover all, it will take a maximum of $|\mathbb{S}|$ steps to reach the goal

Real-time statistics

- Robot takes 25 seconds to reach the goal with heavy object (compared to 22 seconds for light object)
- Robot takes 32 seconds to reach goal with broken joint (compared to 25 seconds for operational joint)

Experiment Details

- 4D planar pushing: $\delta = 0.02, \xi = 0.01$, euclidean distance, K = 5, N = 5 planning updates, Batch size 64, adam optimizer (20 random seeds)

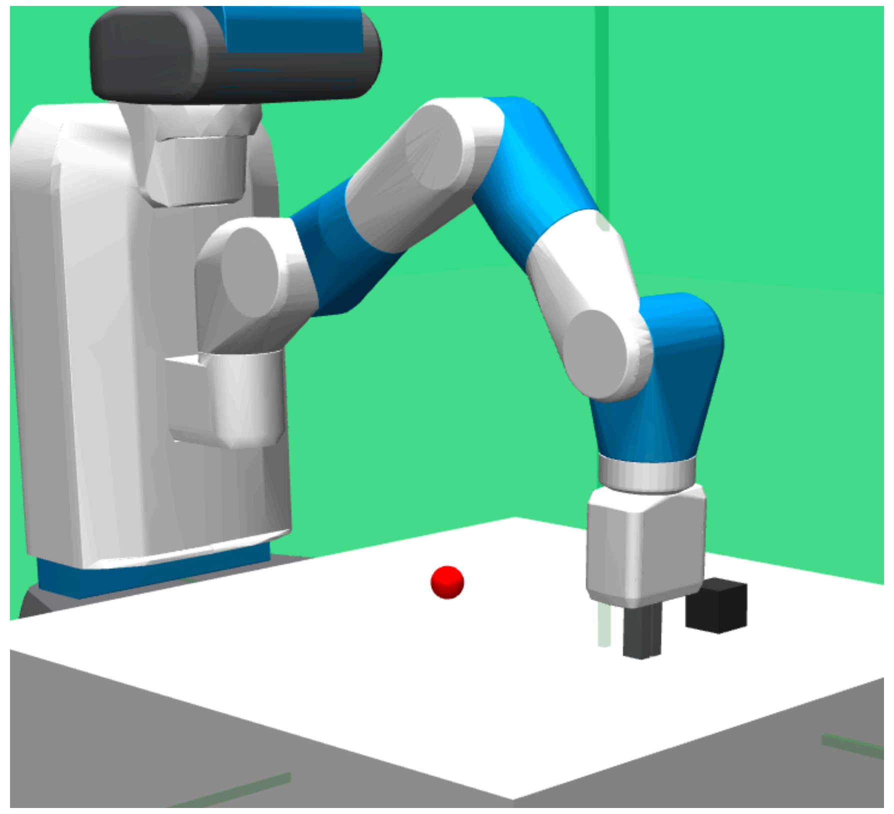
- 3D pick-and-place: K=3, 20x20x20 state space, 6 actions

- 7D arm planning: $\delta = 1, \xi = 1, \gamma = 10$ length scale, 10^7 state space, 14 actions (10 random trials)

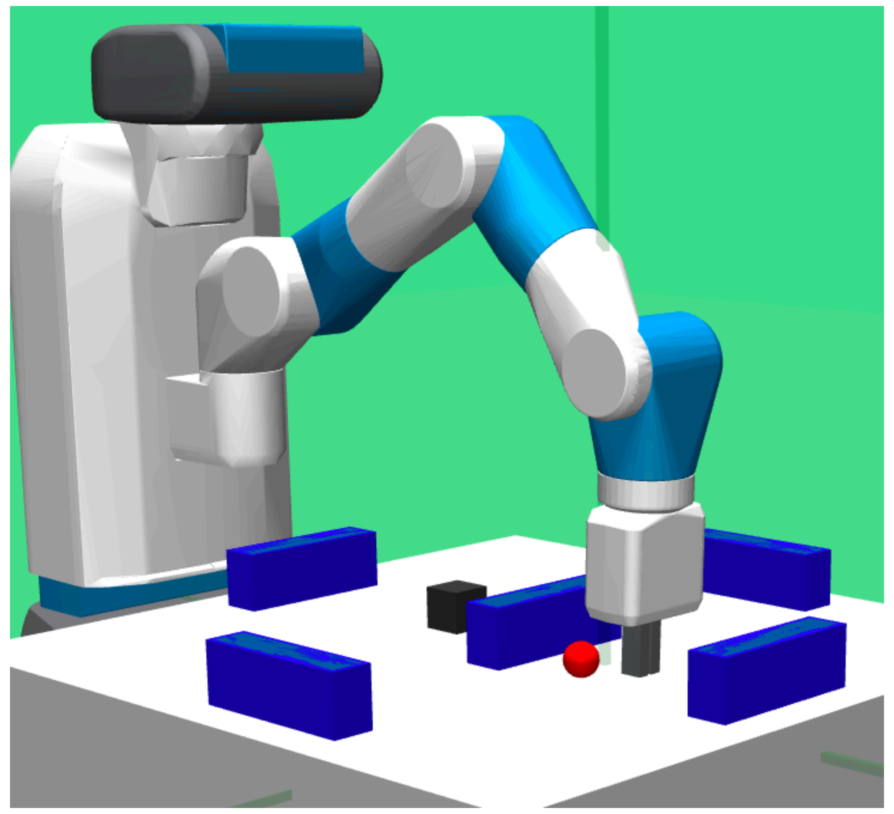
- 2D gridworld: 100x100 grid size (50 random seeds)

Simulated 4D Planar Pushing with Obstacles

	Accurate Model		Inaccurate Model	
	Steps	% Success	Steps	% Success
CMAX	63 ± 22	90%	192 ± 40	80%
Q-Learning	34 ± 5	90%	441 ± 100	45%
Model NN	62 ± 26	90%	348 ± 82	15%
Model KNN	106 ± 34	95%	533 ± 118	50%
Plan with Acc. Model	63 ± 22	90%	364 ± 53	85%

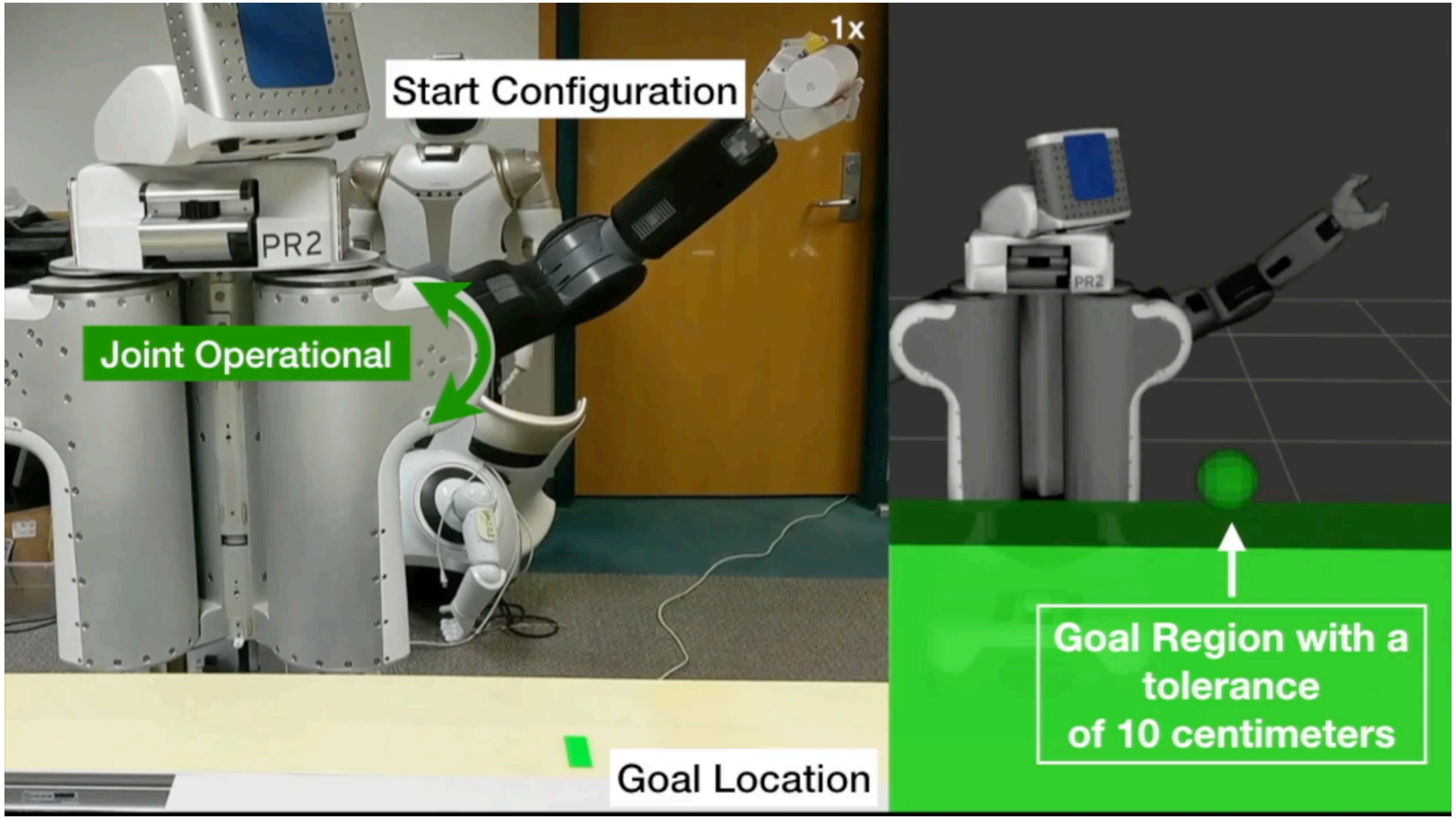


te Model



Environment

7D Arm Planning with a non-operational joint



7D Arm Planning with a non-operational joint

7D Arm Planning with a broken joint

	Steps	% Success
CMAX	47 ± 6	100%
RTAA*	138 ± 65	30%

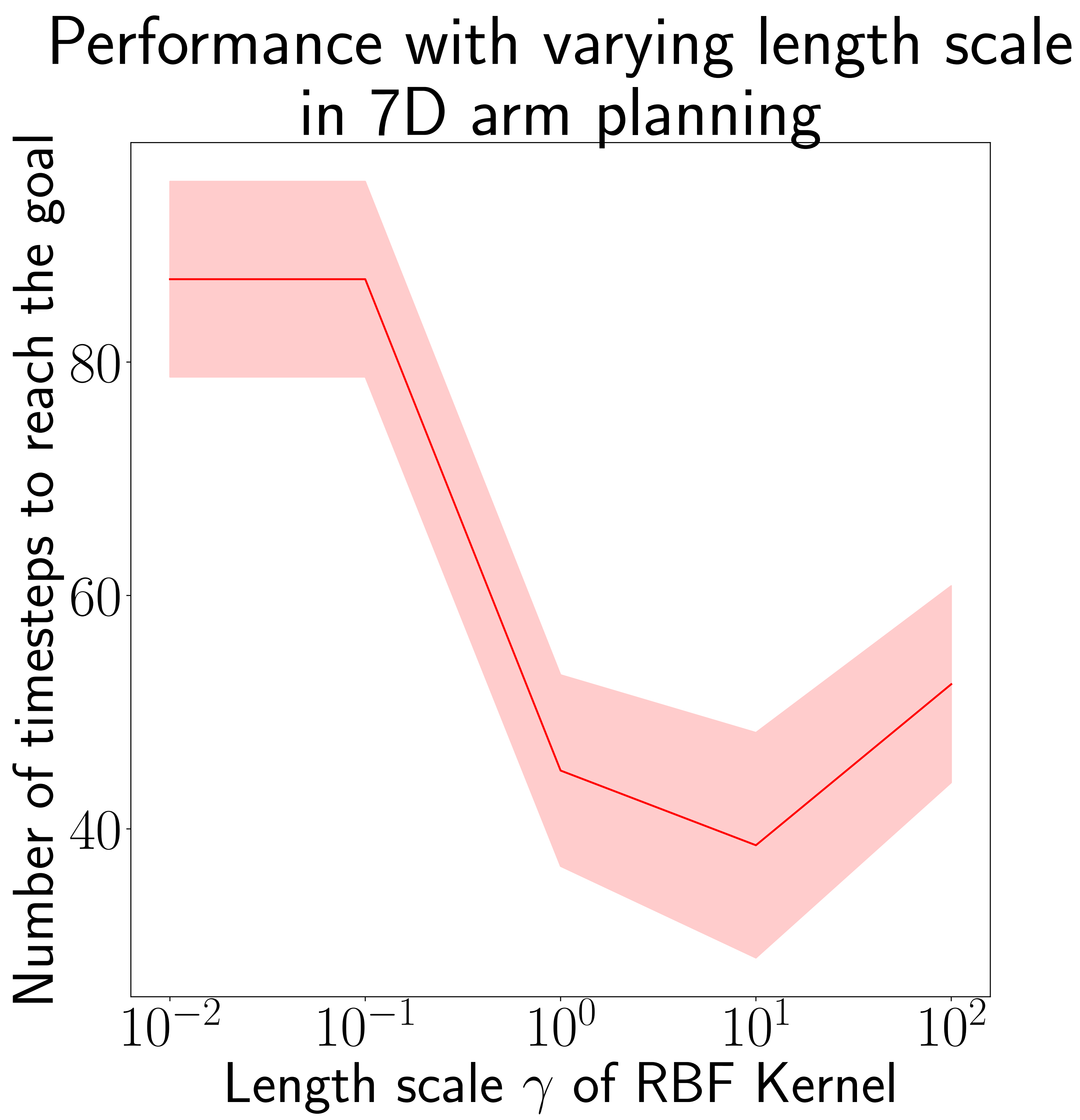
Exact Planning without any function approximation

Does Global Value Function Approximation Help?

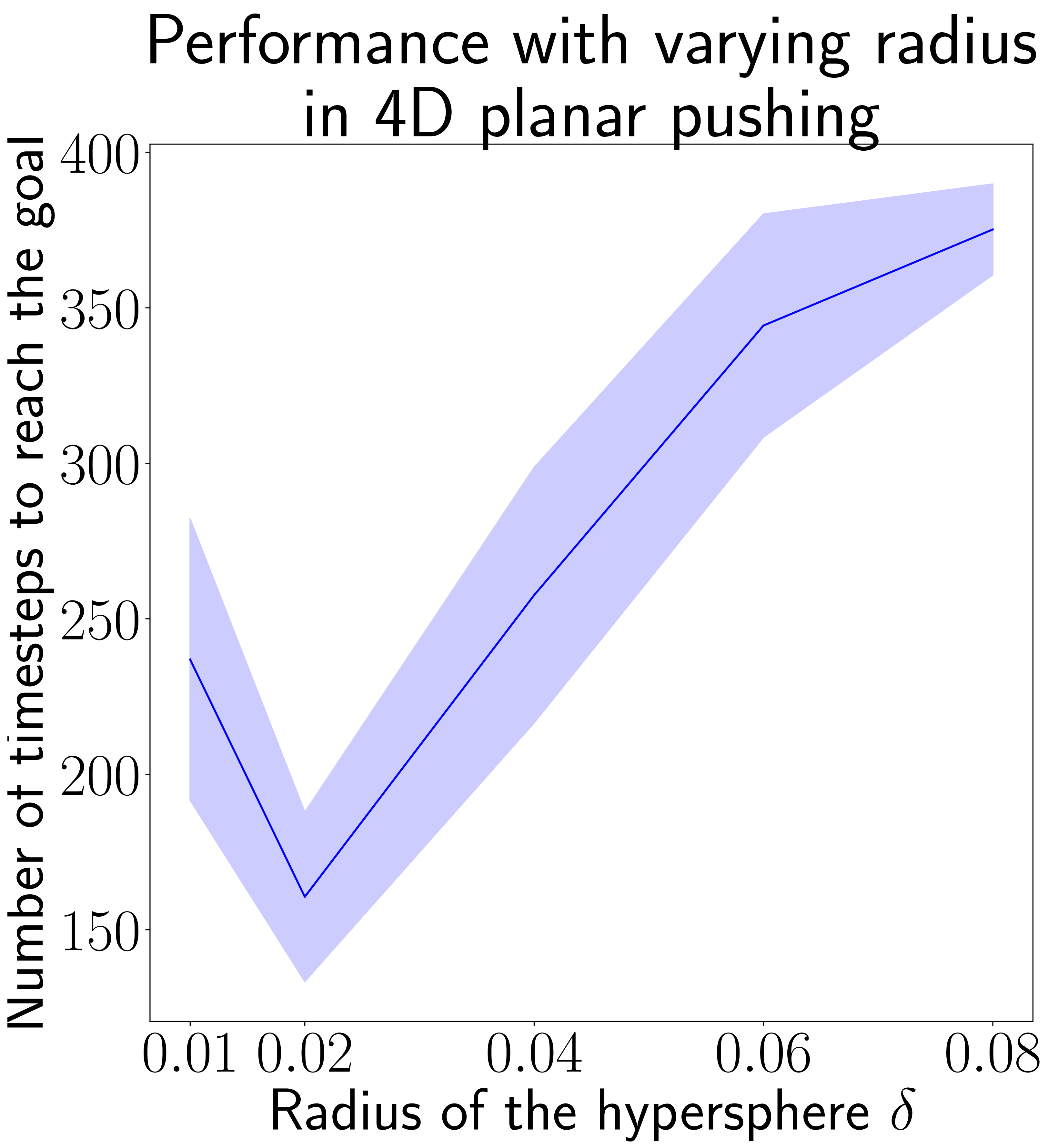
7D Arm Planning with Random Start and Goal Configurations

	Steps	% Success
CMAX	47 ± 6	100%
RTAA*	138 ± 65	30%

Exact Planning without any function approximation



Effect of the radius of hypersphere on performance



Experiment: 2D Grid World
efficient model update is possible

Model becomes more inaccurate

% obstacles	0%	40%	80%
CMAX	78 ± 4	1551 ± 373	138 ± 10
Adaptive RTAA*	78 ± 4	1015 ± 230	137 ± 10
Q-Learning	3879 ± 305	11803 ± 2542	510 ± 36

2D Grid World Navigation in the presence of obstacles

Model learning baseline

Model becomes more inaccurate

% Ice	0%	40%	80%
CMAX	78 ± 4	231 ± 18	2869 ± 331
Adaptive RTAA*	78 ± 4	219 ± 18	2185 ± 249
Q-Learning	3914 ± 303	1220 ± 103	996 ± 108

2D Grid World Navigation in the presence of ice

Concurrent work in Offline RL

- “Pessimism” based approaches
  - MoREL (Rajeswaran et. al. 2020), MOPO (Yu et. al. 2020)
  - Importance of Pessimism (Buckman et. al. 2020)
- Interpreting offline dataset as an approximate model

Advantages of CMAX

- Does not rely on knowledge of how model is inaccurate
- No need for approximate model to be flexible
- Applicable even in situations where modeling true dynamics is intractable
- Empirically requires significantly less number of online executions to reach the goal

Shortcomings of CMAX

- Assumption is restrictive and is not valid in some realistic tasks
- E.g. task of opening a spring-loaded door which is not modeled as spring-loaded. There is discrepancy in every transition and CMAX as is cannot solve it
- Fails to improve quality of solution for repetitive tasks

Advantages of CMAX++ and Adaptive-CMAX++

- Exploit incorrect transitions without wasting executions to learn true dynamics
- Useful in domains where modeling true dynamics is intractable, e.g. deformable manipulation, or vary over time due to wear and tear
- Optimistic model assumption easier to satisfy and performance of CMAX++ degrades gracefully with accuracy of model reducing to Q-learning

Limitations of CMAX++ and Adaptive-CMAX++

- Sequence $\{\alpha_i\}$ requires tuning, but performance is reasonably robust to a wide range of choices
- Assumption can be restrictive to satisfy in domains where designing an optimistic initial model is difficult
- However, infeasible to relax this assumption without resorting to global undirected exploration

Related work : Model-based planning and model-free learning

- After 120 training episodes (and 90 minutes of training), GUAPO is able to achieve 93% insertion rate
-

# Expert Demos:	1	5	10	20
Success Rate:	7/21	15/21	16/21	19/21

Fig. 9. Overall success of our method on the shape insertion task depending on the number of training samples. The first row is the number of training samples used and the second row is the rate of success for the 21 trials. Success and the experimental trials performed are explained in V-B

7D Pick-and-Place with a Heavy Object

Repetition→	1		5		10		15		20	
	Steps	*Success*	*Steps*	*Success*	*Steps*	*Success*	*Steps*	*Success*	*Steps*	*Success*
CMAX	$\mathbf{17.8 \pm 3.4}$	100%	13.6 ± 0.5	60%	18 ± 0	20%	15 ± 0	20%	15 ± 0	20%
CMAX++	$\mathbf{17 \pm 4.9}$	100%	14.2 ± 3.3	100%	$\mathbf{10.6 \pm 0.3}$	100%	$\mathbf{11 \pm 0}$	100%	$\mathbf{10.8 \pm 0.1}$	100%
A-CMAX++	$\mathbf{17.8 \pm 3.4}$	100%	$\mathbf{11.6 \pm 0.7}$	100%	17 ± 6	100%	$\mathbf{10.4 \pm 0.3}$	100%	$\mathbf{10.6 \pm 0.4}$	100%
Model KNN	40.6 ± 7.3	100%	12.8 ± 1.3	100%	29.6 ± 16.1	100%	15.8 ± 2.9	100%	12.4 ± 1.4	100%
Model NN	56 ± 16.2	100%	208.2 ± 92.1	80%	124.5 ± 81.6	40%	28 ± 7.7	40%	37.5 ± 20.1	40%
Q-learning	172.4 ± 75	100%	23.2 ± 10.3	80%	26.5 ± 6.7	80%	18 ± 2.8	80%	10.2 ± 0.6	80%

Model KNN : Local model learning approach using KNN regression

Model NN : Global model learning approach using a neural network

Q-learning: Model-free baseline with carefully initialized value estimates

Exploration in Model-Free Policy Search

- Uses random exploration to estimate gradient

$$\nabla_\theta J(\theta) = \nabla_\pi J(\theta)\, \nabla_\theta \pi$$

Jacobian of policy

Estimate using parameter space exploration

Estimate using action space exploration

- Require $O\left(\frac{1}{\epsilon^4}\right)$ samples to converge to a ϵ-suboptimal policy
- Exponential gap between model-free and model-based [Sun et. al. 2019]
- Cannot be practically used without combining with a model-based procedure

Exploration in Model-Free Policy Search

- Number of samples required to reach θ such that $\|\nabla_\theta J(\theta)\|_2^2 \leq \epsilon$
  - Parameter space exploration = $\mathcal{O}\left(\frac{d^2}{\epsilon^3}\right)$ samples
  - Action space exploration = $\mathcal{O}\left(\frac{p^2 H^4}{\epsilon^4}\right)$ samples
- For tasks with long horizon, exploration in parameter space is preferred
- If parametric complexity required is large, exploration in action space is preferred
- Sample complexity requirement for model-free methods is very large and precludes them from being applied on robots naively

Exploration in Model-Free Policy Search

Parameter Space Exploration

- Find a direction of improvement directly in parameter space through random exploration
- Purely zeroth order approach
- Eg: Cross-entropy method, Evolutionary strategies, Augmented Random search etc.

Action Space Exploration

- Find a direction of improvement in action space through random exploration
- **Leverage Jacobian of policy** to update parameters
- A combination of zeroth and first order approach
- Eg: REINFORCE and its extensions

$$\nabla_\theta J(\theta) = \nabla_\pi J(\theta) \nabla_\theta \pi$$

Directly estimate using a zeroth order approach e.g. finite differencing → $\nabla_\theta J(\theta)$

Jacobian of the policy → $\nabla_\theta \pi$

Estimate using a zeroth order approach → $\nabla_\pi J(\theta)$

Analysis

	Linear Contextual Bandit	Model-Free RL
Parameter space	$\mathcal{O}(\frac{d^2}{\epsilon^2})$	$\mathcal{O}\left(\frac{d^2 Q \sigma^3}{\epsilon^3}\right)$
Action space	$\mathcal{O}(\frac{1}{\epsilon^4})$	$\mathcal{O}(\frac{p^2 H^4}{\epsilon^4}(Q^3 + \sigma^2 Q))$

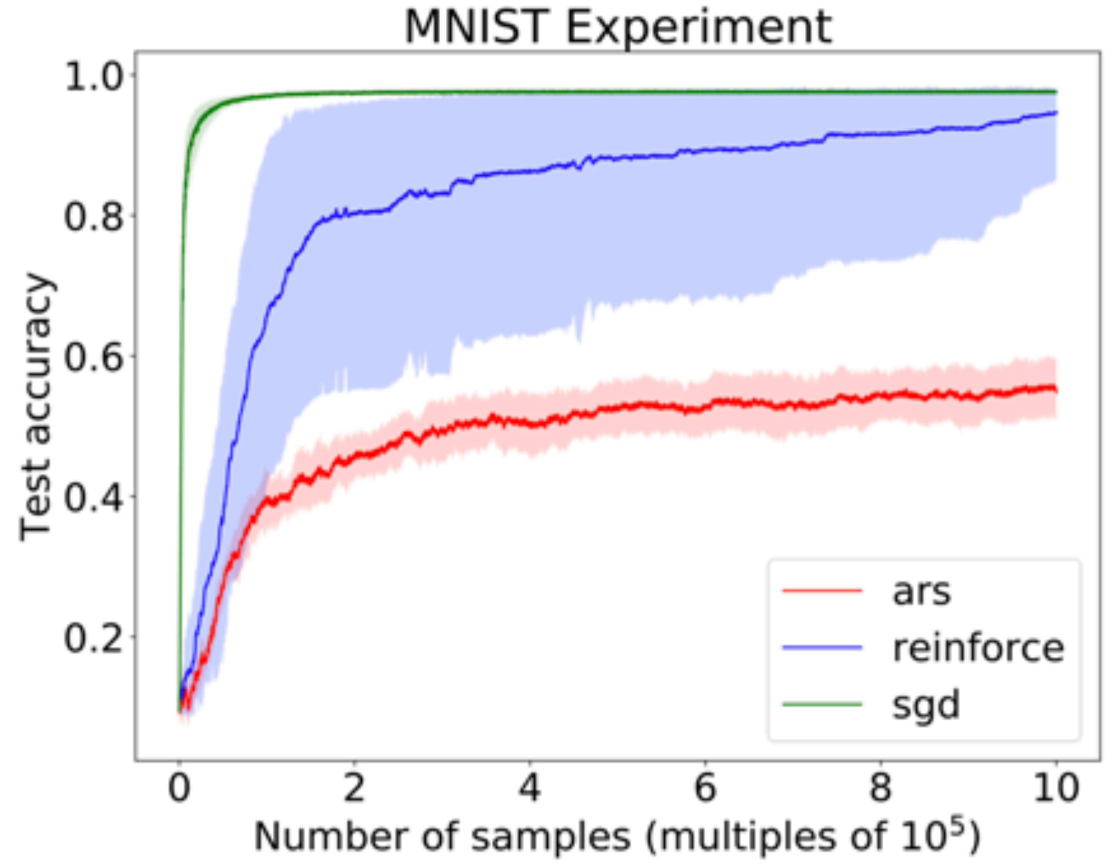
Linear Contextual Bandit : $\text{Avg. Regret} = \frac{1}{T}(\mathbb{E}[\sum_{t=1}^{T} c_i(\theta_i)] - \min_\theta \sum_{t=1}^{T} c_i(\theta))$

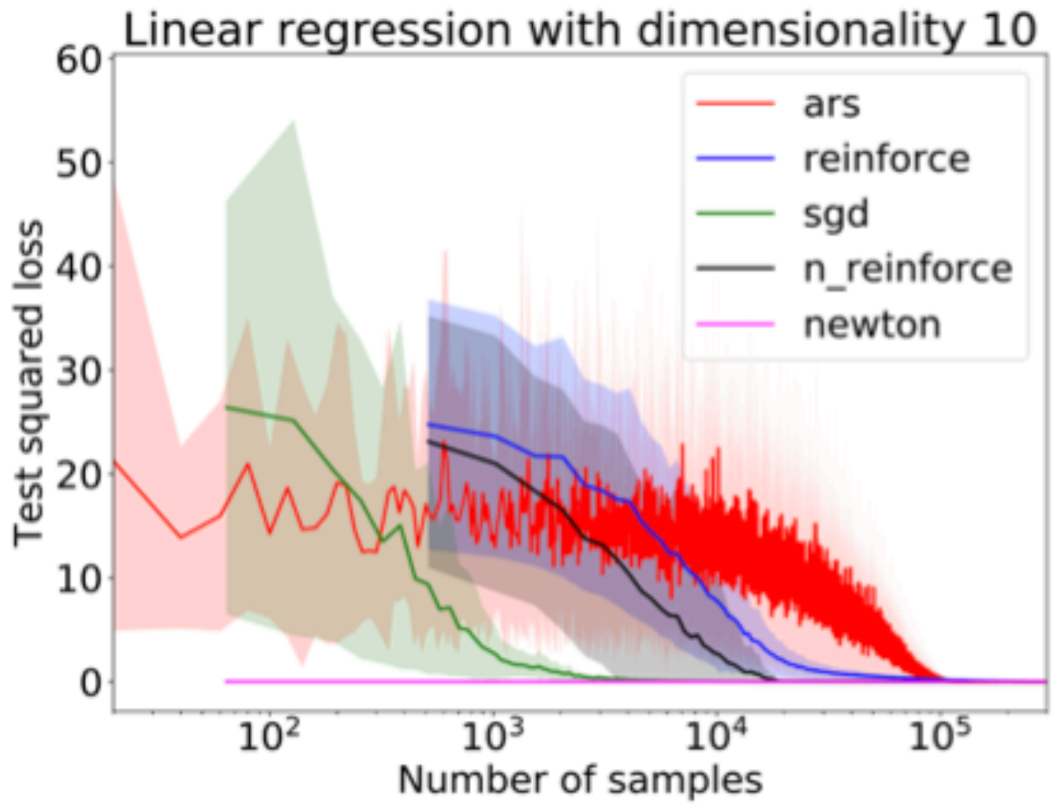
Model-Free RL : $\|\nabla_\theta J(\theta)\|_2^2 \leq \epsilon$ eps-stationary point

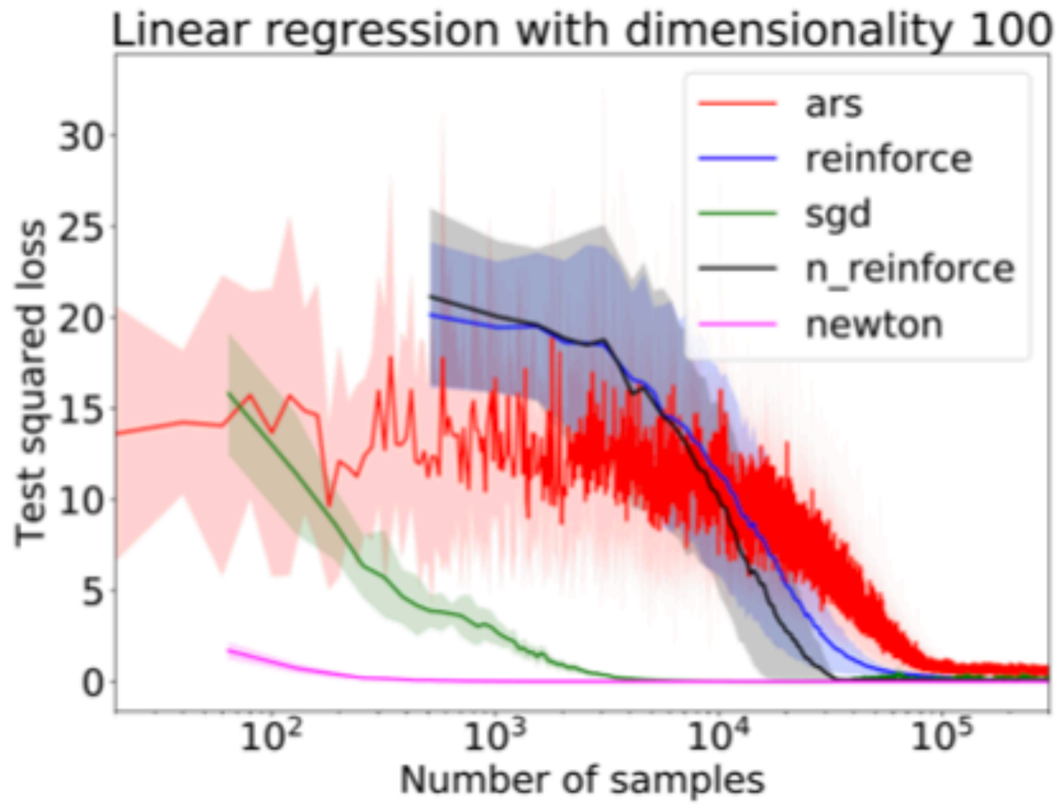
Dependence on parameter dimensionality
Independent of horizon length

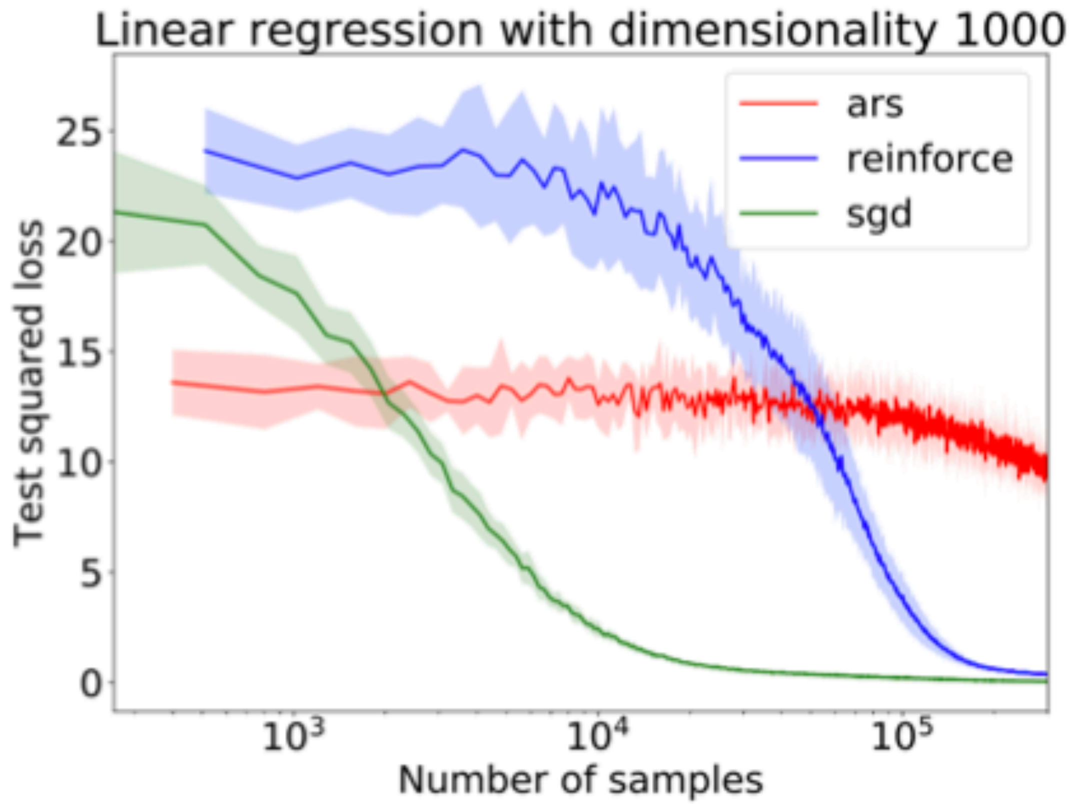
Dependence on horizon length
Dependence on action dimensionality
Independent of parameter dimensionality

Experiments

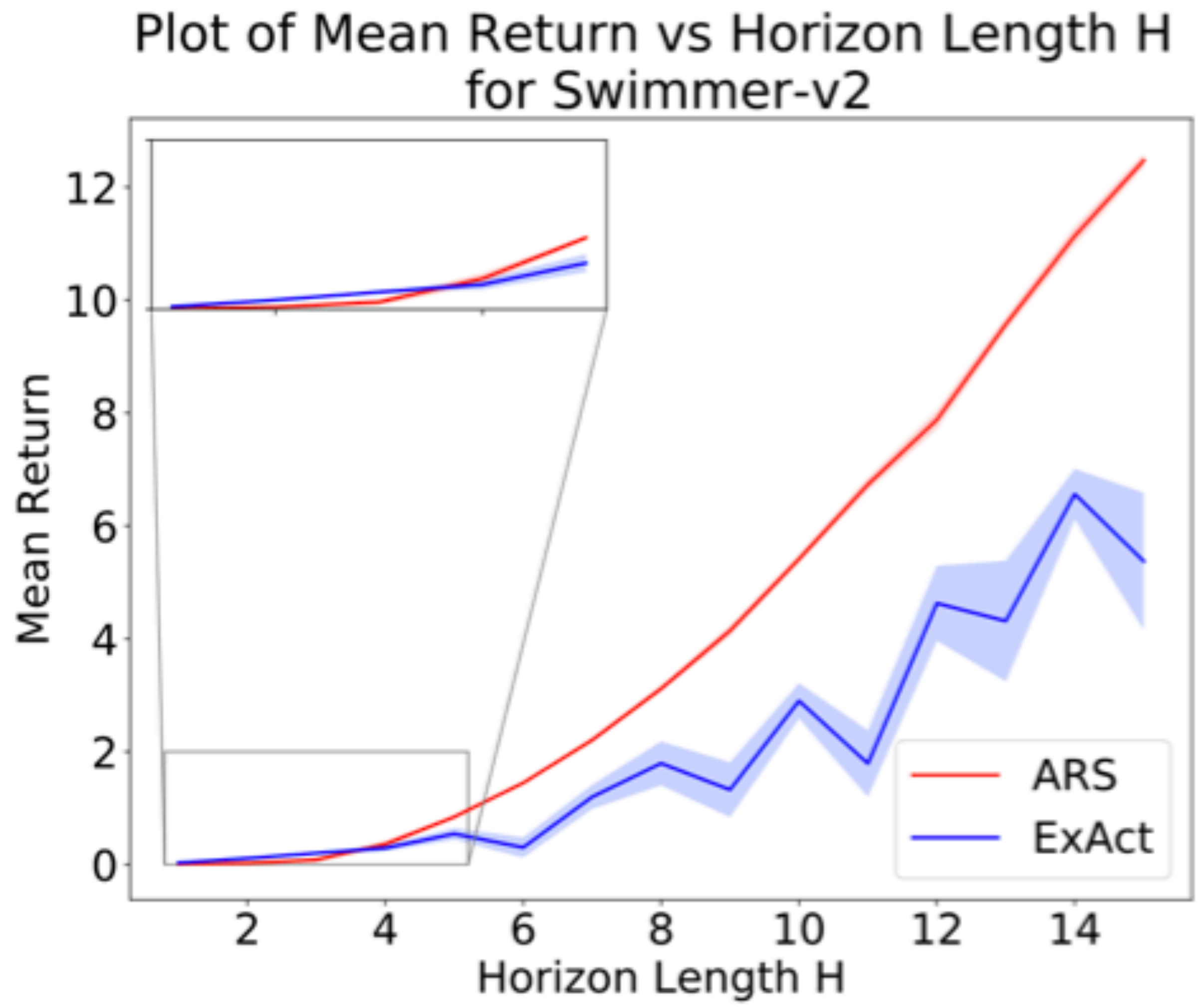


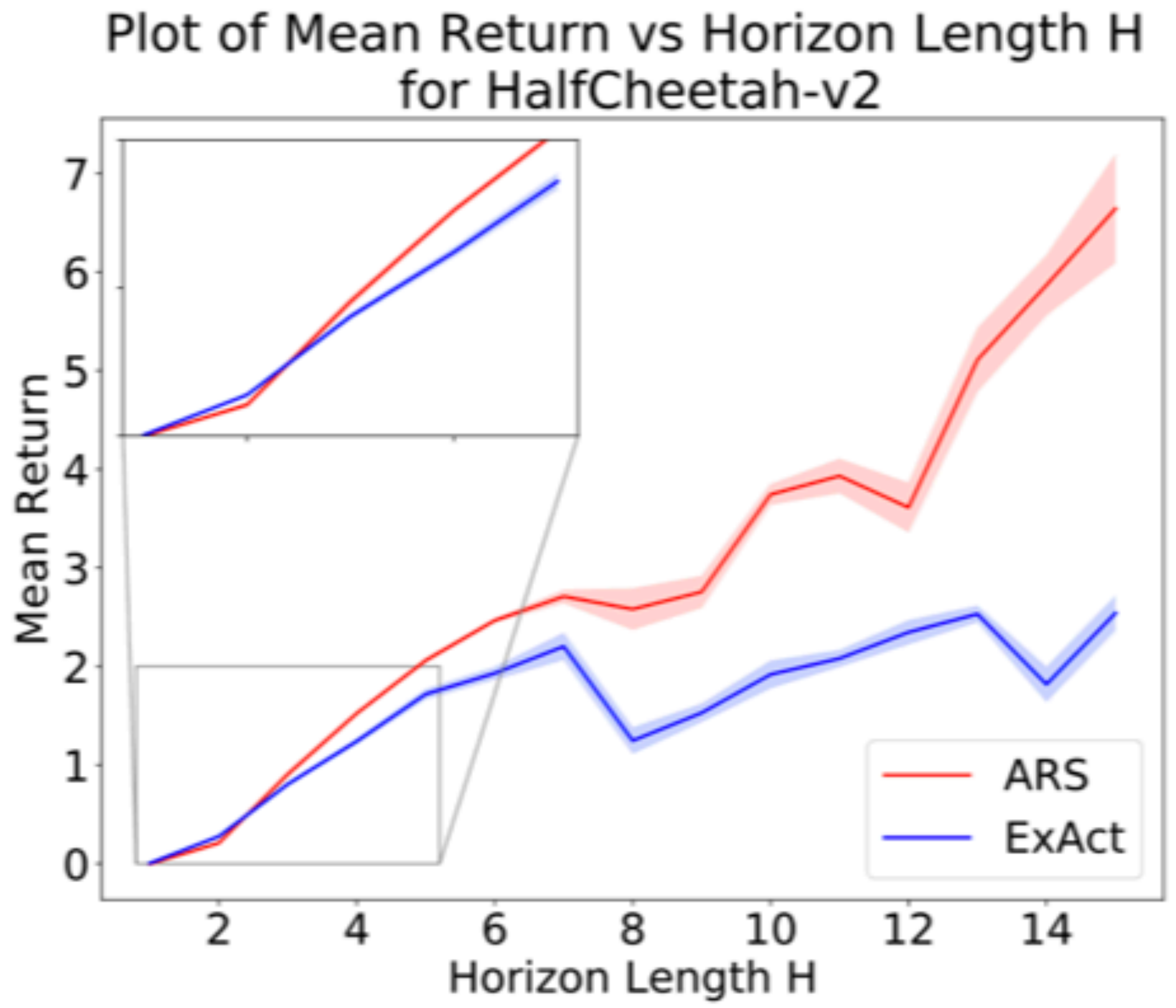






- Our analysis explains the success of black-box policy search methods like random search and evolutionary strategies in RL *(OpenAI gym tasks have very long horizons)*
- **For tasks with long horizons, exploration in parameter space should be preferred**
- **If the parametric complexity required is large, exploration in action space is better**





CMAX++ Algorithms

Algorithm 1 Hybrid Limited-Expansion Search

1: **procedure** SEARCH$(s, \hat{M}, V, Q, \mathcal{X}, K)$
2: Initialize $g(s) = 0$, min-priority open list O, and closed list C
3: Add s to open list O with priority $p(s) = g(s) + V(s)$
4: **for** $i = 1, 2, \cdots, K$ **do**
5: Pop s_i from O
6: **if** s_i is a dummy state or $s_i \in \mathbb{G}$ **then**
7: Set $s_{\text{best}} \leftarrow s_i$ and go to Line 22
8: **for** $a \in \mathbb{A}$ **do** ▷ *Expanding state* s_i
9: **if** $(s_i, a) \in \mathcal{X}$ **then** ▷ *Incorrect transition*
10: Add a dummy state s' to O with priority $p(s') = g(s_i) + Q(s_i, a)$
11: **continue**
12: Get successor $s' = \hat{f}(s_i, a)$
13: If $s' \in C$, **continue**
14: **if** $s' \in O$ and $g(s') > g(s_i) + c(s_i, a)$ **then**
15: Set $g(s') = g(s_i) + c(s_i, a)$ and recompute $p(s')$
16: Reorder open list O
17: **else if** $s' \notin O$ **then**
18: Set $g(s') = g(s_i) + c(s_i, a)$
19: Add s' to O with priority $p(s') = g(s') + V(s')$
20: Add s_i to closed list C
21: Pop s_{best} from open list O
22: **for** $s' \in C$ **do**
23: Update $V(s') \leftarrow p(s_{\text{best}}) - g(s')$
24: Backtrack from s_{best} to s, and set a_{best} as the first action on path from s to s_{best} in the search tree
return a_{best}

Algorithm 2 CMAX++ and A-CMAX++ in small state spaces

Require: Model $\hat{M}$, start state s, initial value estimates V, Q, number of expansions K, $t \leftarrow 1$, incorrect set $\mathcal{X} \leftarrow \{\}$, Number of repetitions N, Sequence $\{\alpha_i \geq 1\}_{i=1}^{N}$, initial penalized value estimates $\tilde{V} = V$, penalized model $\tilde{M} \leftarrow \hat{M}$
1: **for** each repetition $i = 1, \cdots, N$ **do**
2: $t \leftarrow 1$, $s_1 \leftarrow s$
3: **while** $s_t \notin \mathbb{G}$ **do**
4: Compute $a_t = \text{SEARCH}(s_t, \hat{M}, V, Q, \mathcal{X}, K)$
5: Compute $\tilde{a}_t = \text{SEARCH}(s_t, \tilde{M}, \tilde{V}, Q, \{\}, K)$
6: If $\tilde{V}(s_t) \leq \alpha_i V(s_t)$, assign $a_t = \tilde{a}_t$
7: Execute a_t in environment to get $s_{t+1} = f(s_t, a_t)$
8: **if** $s_{t+1} \neq \hat{f}(s_t, a_t)$ **then**
9: Add (s_t, a_t) to the set: $\mathcal{X} \leftarrow \mathcal{X} \cup \{(s_t, a_t)\}$
10: Update: $Q(s_t, a_t) = c(s_t, a_t) + V(s_{t+1})$
11: Update penalized model $\tilde{M} \leftarrow \tilde{M}_{\mathcal{X}}$
12: $t \leftarrow t + 1$

Adaptive-CMAX++ : Maintain two sets of value estimates

- **CMAX++ Value Estimates:** V obtained without inflating costs and using model-free Q-values
- **CMAX Value Estimates:** $\tilde{V}$ obtained by inflating costs

Adaptive-CMAX++ : Algorithm

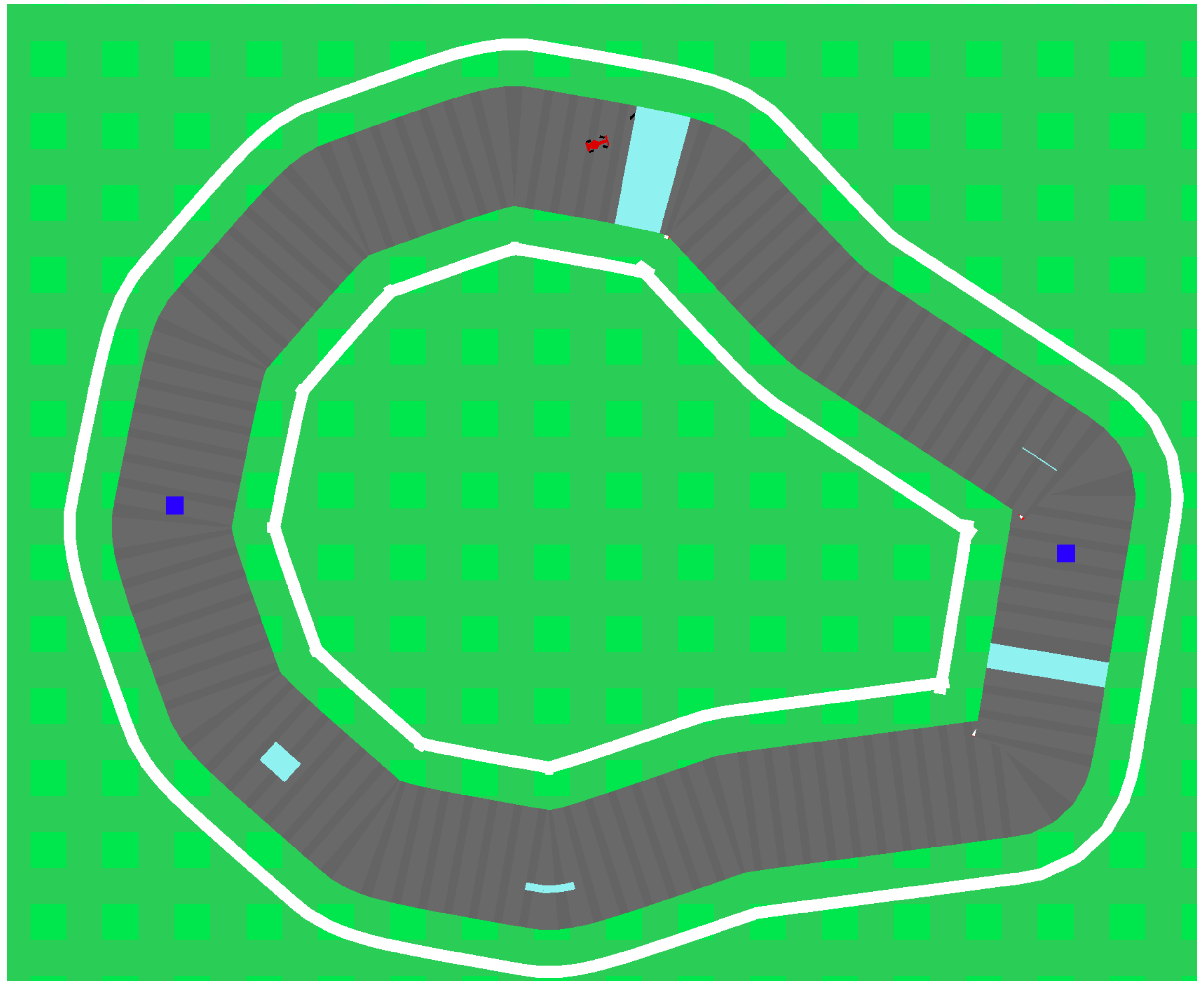
- Given a sequence $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_N \geq 1$ where N is the number of repetitions
- At time step t in repetition i,
  - If $\tilde{V}(s_t) \leq \alpha_i V(s_t)$
    - Execute CMAX action
  - Else
    - Execute CMAX++ action

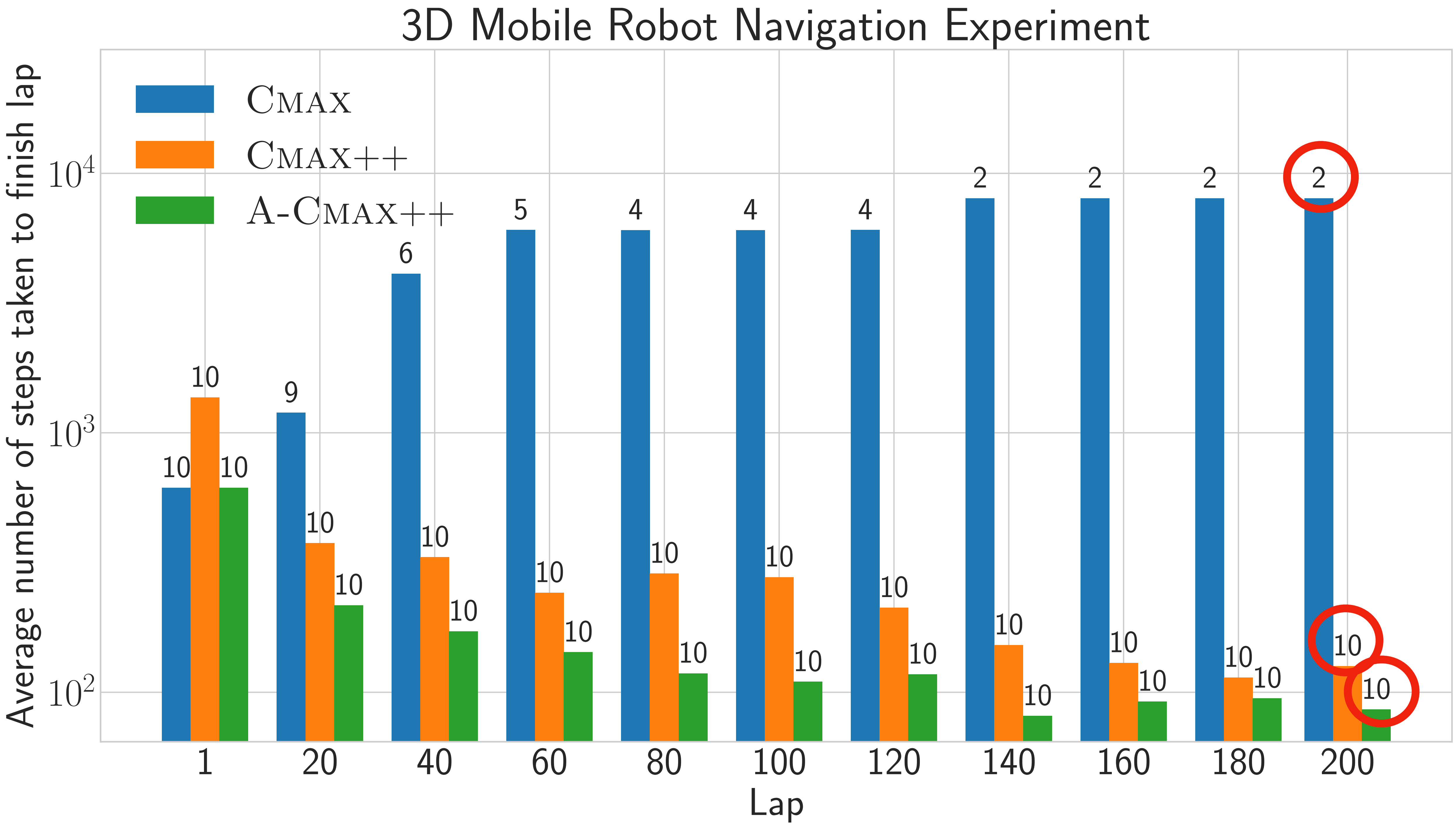
Goal-driven in early repetitions

Optimal in later repetitions

3D Mobile Robot Navigation with Icy Patches

- Small state space (x, y, θ)
- Model has no icy patches and the robot slips on ice





10 randomly generated tracks

CMAX++ Experiment details

- **Car Experiment**
- 100 x 100 x 16 grid with 66 motion primitives
- $\alpha_i = 1 + \beta_i$ with $\beta_1 = 100$ and β_i is decreased by 2.5 after every 5 repetitions
- K = 100 expansions
- **PR2 Experiment**
- 14 discrete actions, two in each dimension - 6DOF gripper pose + 1 redundant joint (forearm roll)
- 10^7 states
- K = 5 expansions, $\delta = 3$, $\xi = 0$
- $\alpha_i = 1 + \beta_i$ with $\beta_1 = 4$ and $\beta_i = 0.5\beta_{i-1}$

Significance of Optimistic Model Assumption

- Completeness guarantees require use of admissible and consistent value estimates

- The above requirement needs to hold every time we plan/replan

- Never discard a path as being too expensive when it is cheap in reality

- Optimistic model assumption ensures that planning in the model always keeps value estimates admissible and consistent

- E.g. Free space assumption in navigation

CMAX++ Proof Sketch : Completeness

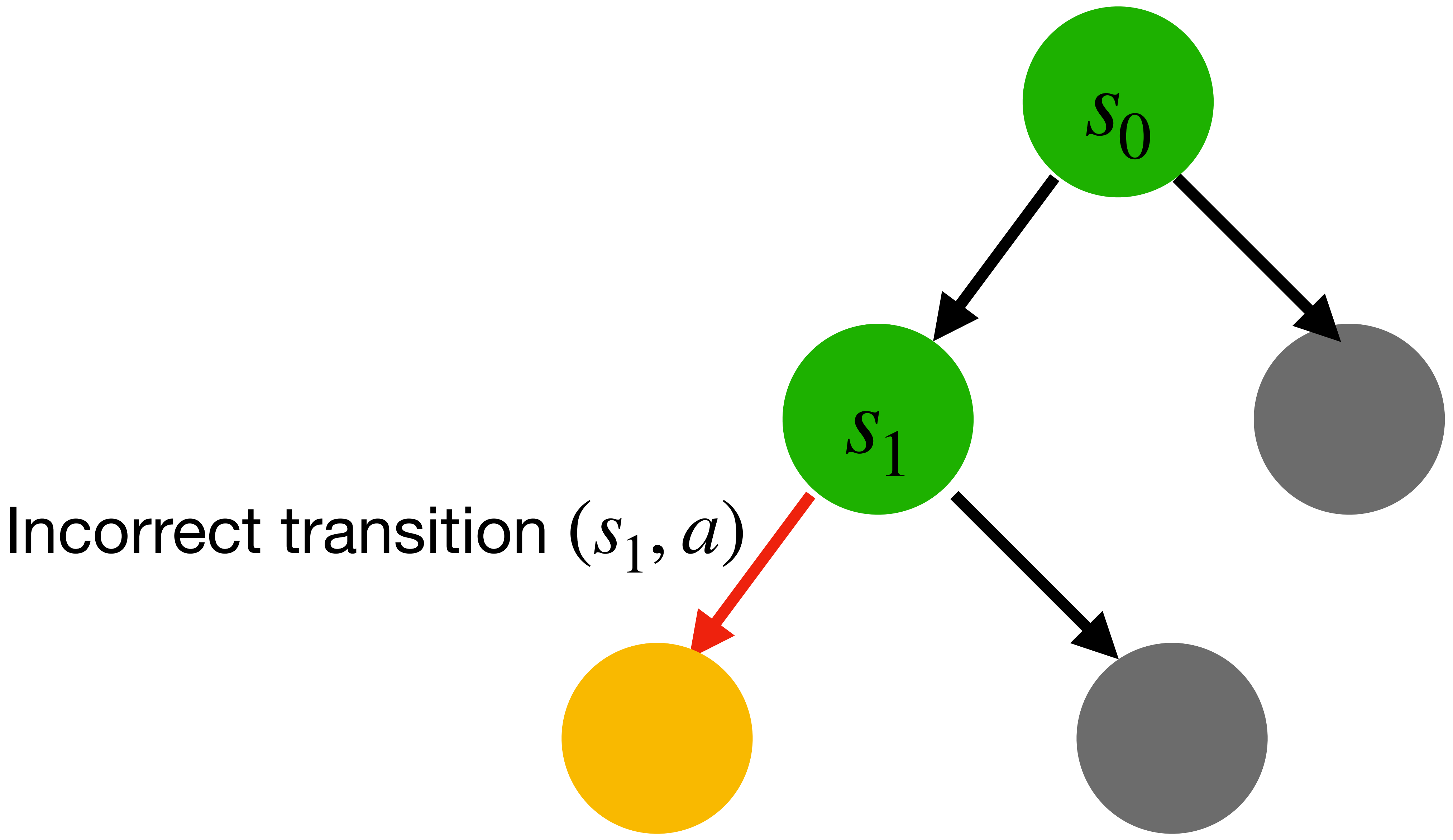
- Use worst case bounds of Q-learning
- RTAA* with optimistic model assumption is guaranteed to be complete
- Model being inaccurate everywhere reduces to Q-learning
- Under CMAX assumption, the bound is tighter

CMAX++ Proof Sketch : Asymptotic Convergence

- Again, derived from Q-learning and LRTA* asymptotic convergence proofs
- Under CMAX assumption, only guaranteed to converge to the optimal path in penalized model

Planning using Inaccurate Model and Q-values

- Create a dummy state for the successor of an incorrect transition
- Compute priority of dummy state as $g(s_1) + Q(s_1, a)$ where s_1 is the parent node
- If dummy state is ever chosen as the next state to be expanded, then terminate search and return dummy state as best node



$$p(s) = g(s_1) + Q(s_1, a)$$

Incremental Model Learning

- LWR, LWPR, LGR - local incremental methods
- Promise of model-KNN
- Dealing with discrete and continuous state spaces - so far have dealt primarily in continuous
- Right state space for planning and learning dynamics
- GP for model uncertainty - optimize for mean dynamics

Value-Aware Model Learning

Minimize planning error and not prediction error

$Q_{k+1} \leftarrow T^*_{P*} Q_k = r + \gamma P^* V_k$ - value iteration where $V_k(s) \leftarrow \max_a Q_k(s, a)$

Start from $\hat{Q}_0 \leftarrow r$, and for each iteration k, solve $\hat{P}_k = \arg\min_{P \in M} ||(P - P^*)\hat{V}_k||_2^2$

Use $\hat{P}_k$ to compute $\hat{Q}_{k+1}, \hat{V}_{k+1}$ using $\hat{Q}_{k+1} \leftarrow T^*_{\hat{P}_k} \hat{Q}_k$ and $\hat{V}_{k+1}(s) = \max_a \hat{Q}_{k+1}(s, a)$

Can be extended to approximate value iteration

replace population version with empirical version from samples

Initial ideas on guarantees and combining methods

- Learn local incremental models on-the-fly
- Early repetitions - not enough samples -> approximation errors
- Later repetitions - enough samples -> can use updated model
- Guarantees using multi-heuristic framework, so that we ultimately only rely on optimistic model assumption
- Switch similar to A-CMAX++ based on predicted cost-to-go with the bias as shown above

ILC for continuous linearized systems

$$\min_{K_1,\cdots,K_{T-1}} \max_{\substack{\|\Delta_t^A\|_2 \le \epsilon_t^A \\ \|\Delta_t^B\|_2 \le \epsilon_t^B}} J(K)$$

$$\text{Subject to} \quad x_{t+1} = (\hat{A}_t + \Delta_t^A)x_t + (\hat{B}_t + \Delta_t^B)u_t$$

- Dynamic game between player and adversary

- Player tries to minimize regularized cost using $K_1, \cdots, K_{T-1}$ while adversary maximizes regularized cost using Δ_t^A, Δ_t^B

Potential Domains

- Deformable Manipulation (Rope Dragging) and Rearrangement planning

[Yan et. al. 2019]

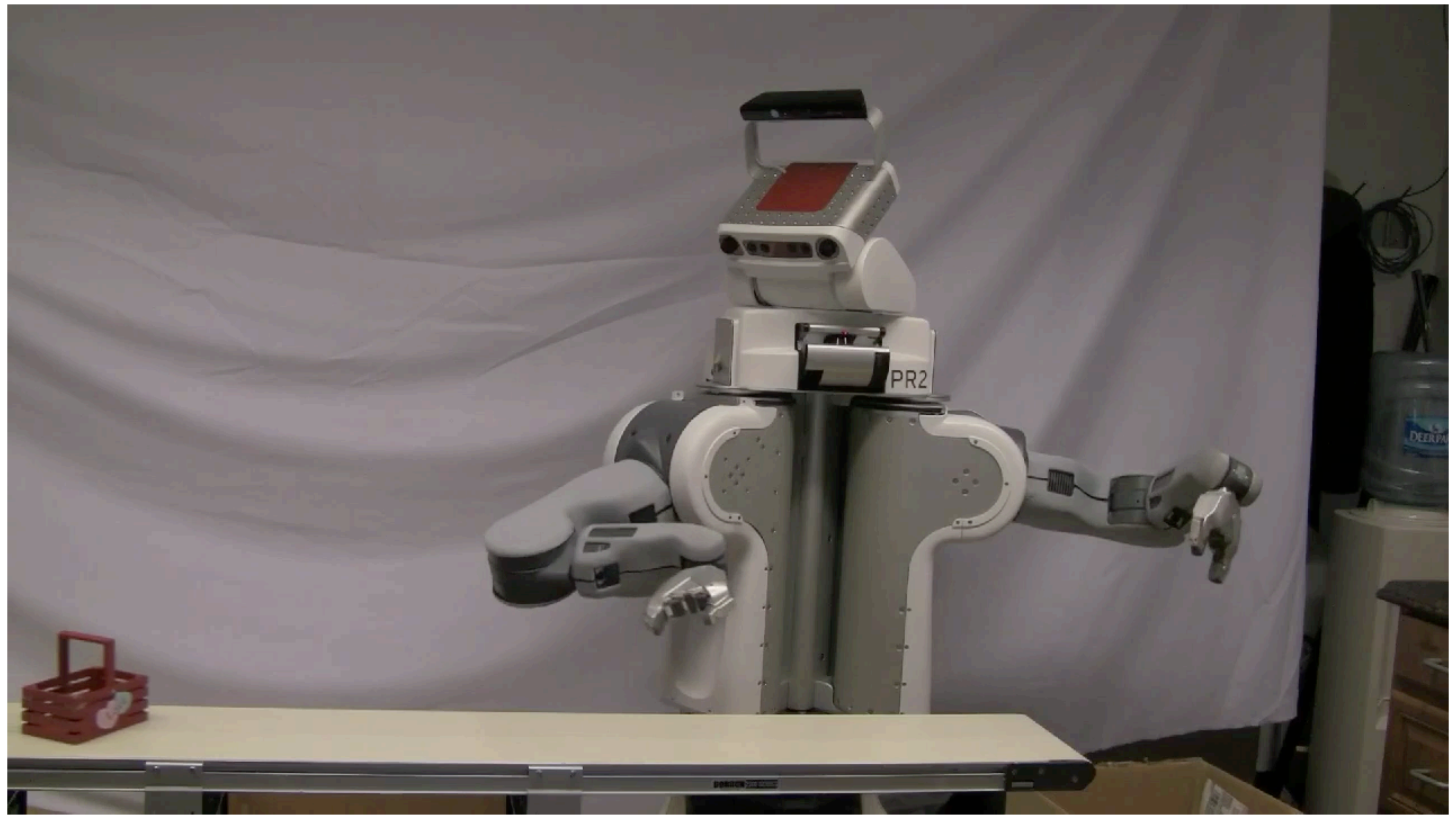
[King et. al. 2015]

Proposed Work #1 : Unified Framework

- Best of worlds: Update dynamics of model + CMAX + CMAX++

Proposed Work #1 : Unified Framework

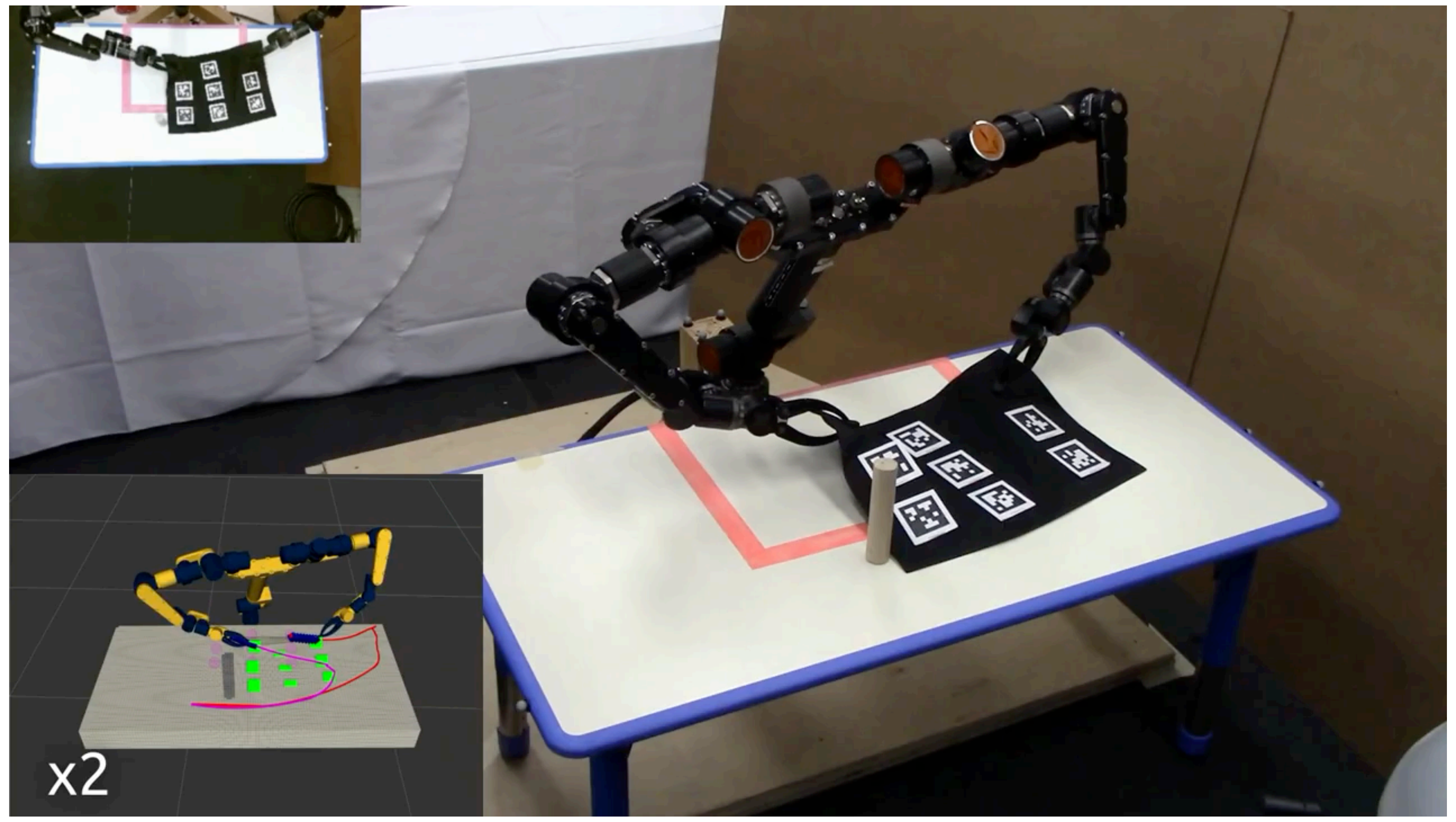
- Best of both worlds: Update dynamics of model + CMAX + CMAX++
- Several challenges:
  1. **Data efficiency** : Need incremental local model learning [Meier et. al. 2014]



Video from [Cowley et. al. 2013]

Proposed Work #1 : Unified Framework

- Best of both worlds: Update dynamics of model + CMAX + CMAX++
- Several challenges:
  1. **Data efficiency** : Need incremental local model learning [Meier et. al. 2014]
  2. **Task-aware model learning** : NOT learn true dynamics but build models that help future planning [Farahmand 2018]



Video from [McConcachie et. al. 2020]

Proposed Work #1 : Unified Framework

- Best of both worlds: Update dynamics of model + CMAX + CMAX++
- Several challenges:
    1. **Data efficiency** : Need incremental local model learning [Meier et. al. 2014]
    2. **Task-aware model learning** : NOT learn true dynamics but build models that help future planning [Farahmand 2018]
    3. **Completeness using learned models** : what assumptions are required?

Proposed Work #1 : Unified Framework

- Best of both worlds: Update dynamics of model + CMAX + CMAX++
- Several challenges:
  1. **Data efficiency** : Need incremental local model learning [Meier et. al. 2014]
  2. **Task-aware model learning** : NOT learn true dynamics but build models that help future planning [Farahmand 2018]
  3. **Completeness using learned models** : what assumptions are required?
- Switch between CMAX, CMAX++ and updating the model, during execution

Proposed Work #2: Continuous Linearized Systems

- Discrete systems allow optimal planning but only asymptotic analysis
- Continuous domain allows more fine-grained analysis

$$x_{t+1} = A_t x_t + B_t u_t$$

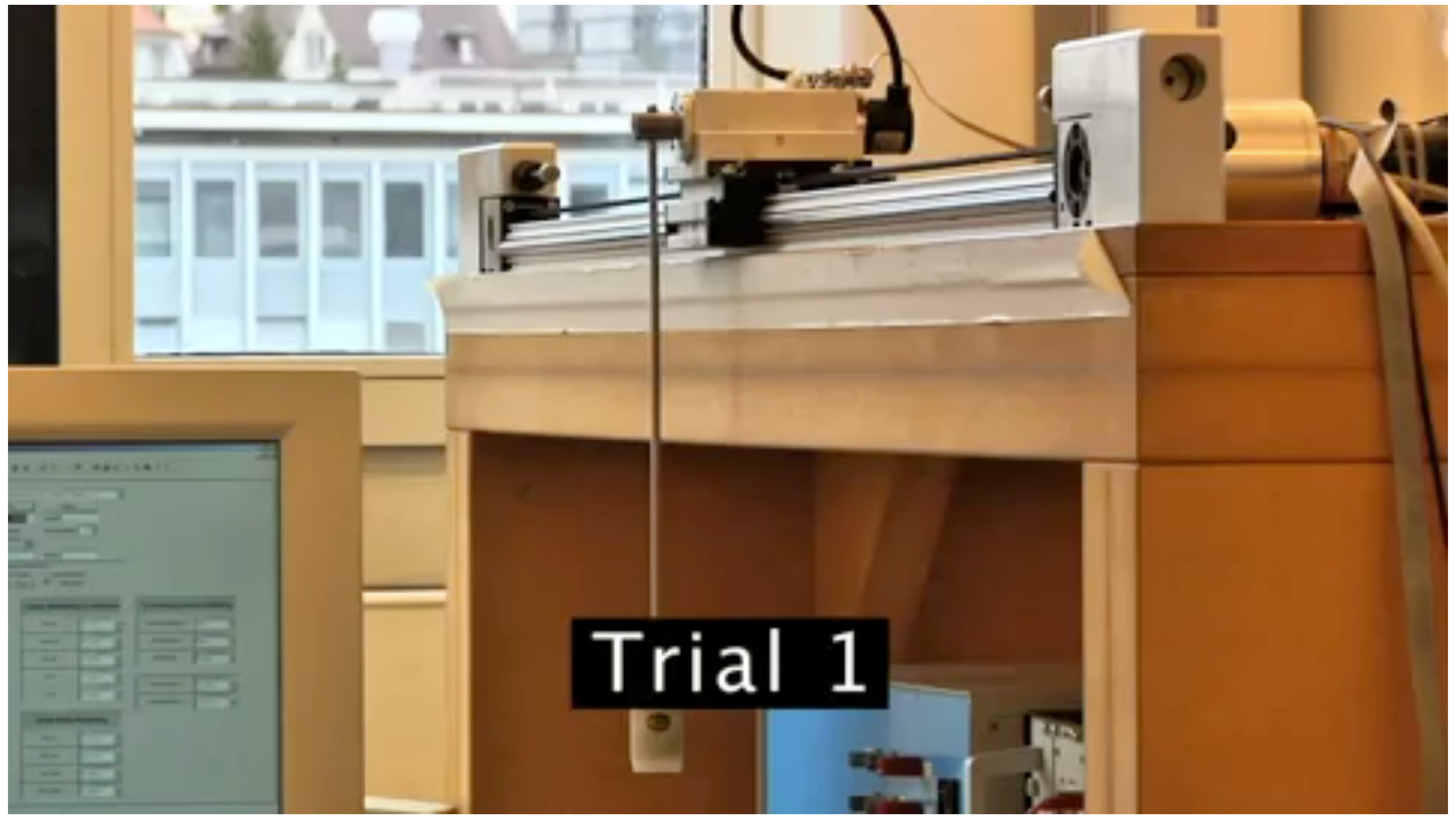
- Nominal approximate dynamics $\hat{A}_t, \hat{B}_t$ such that

$$||A_t - \hat{A}_t||_2 \leq \epsilon_t^A \text{ and } ||B_t - \hat{B}_t||_2 \leq \epsilon_t^B$$

- Minimize sum of convex costs along a finite horizon, $J = \sum_{t=1}^{T} c_t(x_t, u_t)$

Proposed Work #2: Continuous Linearized Systems

- **Iterative Learning Control (ILC)**
  - Compute update using nominal model gradients
  - Evaluate using real world rollouts



Video from [Schoellig and D'Andrea 2009]

Proposed Work #2: Continuous Linearized Systems

What performance can we expect using approximate dynamics and finite amount of experience from N rollouts?

- Past work on infinite horizon, known dynamics [Agarwal et. al. 2019]
- Our setting is ILC with approximate dynamics
- Regret w.r.t optimal robust controller K^* across N rollouts [Dean et. al. 2019]

$$\text{Regret} = \sum_{i=1}^{N} J_i - \sum_{i=1}^{N} J(K^*)$$

Timeline

- **Spring 2021:**
  - Finish analysis of iterative learning control in continuous linearized systems
  - Design and implement incremental task-aware model learning algorithm
- **Summer 2021:**
  - Combine model learning algorithm with CMAX and CMAX++ creating the unified framework
  - Demonstrate framework on simulated and real robot experiments
- **Fall 2021:**
  - Write and defend thesis

Thesis Contributions

Completed Work

- **Sample Complexity of Exploration in Model-free RL** [AISTATS 2019]
- **CMAX : Biasing Planner Away From Inaccurately Modeled Regions** [RSS 2020]
- **CMAX++ : Leveraging Model-free Values in Model-based Planner** (Under review)

Proposed Work

- **Combining model learning methods with CMAX and CMAX++**
- **Robust Control in continuous linearized systems with model uncertainty**

ILC and MM Controllers

$$\tilde{K}_t = -(R + \hat{B}_t^T \tilde{P}_{t+1} B_t)^{-1} \hat{B}_t^T \tilde{P}_{t+1} A_t$$

$$\tilde{P}_{t+1} = Q + \hat{A}_t^T \tilde{P}_{t+1} (I + B_t R^{-1} \hat{B}_t^T \tilde{P}_{t+1})^{-1} A_t$$

$$\hat{K}_t = -(R + \hat{B}_t^T \hat{P}_{t+1} \hat{B}_t)^{-1} \hat{B}_t^T \tilde{P}_{t+1} \hat{A}_t$$

$$\hat{P}_{t+1} = Q + \hat{A}_t^T \hat{P}_{t+1} (I + \hat{B}_t R^{-1} \hat{B}_t^T \hat{P}_{t+1})^{-1} \hat{A}_t$$

ILC Analysis Assumptions

- Assumption 1: Assume Q, Q_f, R are P.D matrices, and smallest singular value of R, $\sigma_1(R) \geq 1$
- Assumption 2: Optimal controller $K^\star$ satisfies $||A_t + B_t K_t^\star|| \leq 1 - \delta$ for some $0 < \delta \leq 1$ and all $t = 0, \cdots, H-1$
- Assumption 3: The matrix $B_t R^{-1} \hat{B}_t^T$ has eigenvalues that have non-negative real parts for all $t = 0, \cdots, H-1$

ILC Analysis Lemmas

Theorem 6.3.1. *Suppose* $d \leq n$. *Denote* $\Gamma = 1 + \max_t\{||A_t||, ||B_t||, ||P_t^\star||, ||K_t^\star||\}$. *Then under Assumption 6.2.2 and if* $||K_t^\star - \hat{K}_t|| \leq \frac{\delta}{2||B_i||}$ *for all* $t = 0, \cdots, H-1$, *we have*

$$\hat{V}_0(x_0) - V_0^\star(x_0) \leq d\Gamma^3 \|x_0\|^2 \sum_{t=0}^{H-1} e^{-\delta t} \|K_t^\star - \hat{K}_t\|^2 \tag{6.3}$$

Lemma 6.3.1. *If* $||A_t - \hat{A}_t|| \leq \epsilon_A$ *and* $||B_t - \hat{B}_t|| \leq \epsilon_B$ *for* $t = 0, \cdots, H-1$, *and we have* $||P_{t+1}^\star - P_{t+1}^{\mathsf{MM}}|| \leq f_{t+1}^{\mathsf{MM}}(\epsilon_A, \epsilon_B)$ *for some function* f_{t+1}^{MM}. *Then we have under Assumption 6.2.1 for all* $t = 0, \cdots, H-1$,

$$||K_t^\star - K_t^{\mathsf{MM}}|| \leq 14\Gamma^3 \epsilon_t \tag{6.4}$$

where $\Gamma = 1 + \max_t\{||A_t||, ||B_t||, ||P_t^\star||, ||K_t^\star||\}$ *and* $\epsilon_t = \max\{\epsilon_A, \epsilon_B, f_{t+1}^{\mathsf{MM}}(\epsilon_A, \epsilon_B)\}$.

ILC Analysis Lemmas

Theorem 6.3.2. *If the cost-to-go matrices for the optimal controller and MM controller are specified by* $\{P_t^\star\}$ *and* $\{P_t^{\mathsf{MM}}\}$ *such that* $P_H^\star = P_H^{\mathsf{MM}} = Q_f$ *then,*

$$\begin{aligned} ||P_t^\star - P_t^{\mathsf{MM}}|| \leq & \|A_t\|^2 \|P_{t+1}^\star\|^2 (2\|B_t\| \|R^{-1}\| \epsilon_B + \|R^{-1}\| \epsilon_B^2) \\ & + 2\|A_t\| \|P_{t+1}^\star\| \epsilon_A + \|P_{t+1}^\star\| \epsilon_A^2 \\ & + c_{P_{t+1}^\star} (\|A_t\| + \epsilon_A)^2 ||P_{t+1}^\star - P_{t+1}^{\mathsf{MM}}|| \end{aligned} \tag{6.5}$$

for $t = 0, \cdots, H-1$ *where* $c_{P_{t+1}^\star} \in \mathbb{R}^+$ *is a constant that is dependent only on* $P_{t+1}^\star$ *if* ϵ_A, ϵ_B *are small enough such that* $\|P_{t+1}^\star - P_{t+1}^{\mathsf{MM}}\| \leq \|P_{t+1}^\star\|^{-1}$*. Furthermore, the upper bound (6.5) is tight up to constants that only depend on the true dynamics* A_t, B_t*, cost matrix* R*, and* $P_{t+1}^\star$*.*

Lemma 6.3.2. *If* $||A_t - \hat{A}_t|| \leq \epsilon_A$ *and* $||B_t - \hat{B}_t|| \leq \epsilon_B$ *for* $t = 0, \cdots, H-1$*, and we have* $||P_{t+1} - P_{t+1}^{\mathsf{ILC}}|| \leq f_{t+1}^{\mathsf{ILC}}(\epsilon_A, \epsilon_B)$ *for some function* f_{t+1}^{ILC}*. Then we have under Assumption 6.2.1 for all* $t = 0, \cdots, H-1$*,*

$$||K_t^\star - K_t^{\mathsf{ILC}}|| \leq 6\Gamma^3 \epsilon_t \tag{6.6}$$

where $\Gamma = 1 + \max_t\{||A_t||, ||B_t||, ||P_t^\star||, ||K_t^\star||\}$ *and* $\epsilon_t = \max\{\epsilon_A, \epsilon_B, f_{t+1}^{\mathsf{ILC}}(\epsilon_A, \epsilon_B)\}$*.*

ILC Analysis Lemmas

Theorem 6.3.3. *If the cost-to-go matrices for the optimal controller and iterative learning control are specified by* $\{P_t^\star\}$ *and* $\{P_t^{\mathsf{ILC}}\}$ *such that* $P_H^\star = P_H^{\mathsf{ILC}} = Q_f$ *then we have under Assumption 6.2.3,*

$$\begin{aligned}||P_t^\star - P_t^{\mathsf{ILC}}|| \leq \|A_t\|^2 \|P_{t+1}^\star\|^2 \|B_t\| \|R^{-1}\| \epsilon_B + \|A_t\| \|P_{t+1}^\star\| \epsilon_A \\ + c_{P_{t+1}^\star} ||A_t|| (\|A_t\| + \epsilon_A) ||P_{t+1}^\star - P_{t+1}^{\mathsf{ILC}}|| \end{aligned} \quad (6.7)$$

for $t = 0, \cdots, H-1$ *where* $c_{P_{t+1}^\star} \in \mathbb{R}^+$ *is a constant that is dependent only on* $P_{t+1}^\star$ *if* ϵ_A, ϵ_B *are small enough that* $\|P_{t+1}^\star - P_{t+1}^{\mathsf{ILC}}\| \leq \|P_{t+1}^\star\|^{-1}$*. Furthermore, the upper bound (6.7) is tight upto constants that depend only on the true dynamics* A_t, B_t*, cost matrix* R*, and* $P_{t+1}^\star$*.*

Modeling Error only at first time step

$||\hat{A}_1 - A_1|| \leq \epsilon_A, ||\hat{B}_1 - B_1|| \leq \epsilon_B \qquad A_t = \hat{A}_t, B_t = \hat{B}_t, t = 2,\cdots, H-1$

$$\hat{J} - J^\star \leq \mathcal{O}(1)(\epsilon_A + \epsilon_A^2 + \epsilon_B + \epsilon_B^2)^2$$

$$\tilde{J} - J^\star \leq \mathcal{O}(1)(\epsilon_A + \epsilon_B)^2$$

When modeling errors ϵ_A, ϵ_B are large, higher order terms like $\epsilon_A^2\epsilon_B, \epsilon_A^3$ are significant

Inverted Pendulum Dynamics

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \qquad u = \tau \qquad \ddot{\theta} = \frac{\bar{\tau}}{m\ell^2} - \frac{g\sin(\theta)}{\ell}$$

$$\bar{\tau} = \max(\tau_{min}, \min(\tau_{max}, \tau))$$

For Naive, run iLQR using model for both forward and backward pass

For ILC, run iLQR using model for backward pass and the real system for forward pass

TOMS: Task-Aware Online Model Search

- Updates dynamics of model to optimize task performance, rather than prediction error
- Environment M with unknown dynamics $f : \mathbb{S} \times \mathbb{A} \to \mathbb{S}$
- Access to misspecified model class $\mathscr{F} = \{\hat{f}_\theta : \mathbb{S} \times \mathbb{A} \to \mathbb{S} \,|\, \theta \in \Theta\}$
- Misspecified => $f \notin \mathscr{F}$ - usually true in real world tasks
- Access to a planner P when given model $\hat{f}_\theta$ results in a policy π_θ that optimizes cost-to-go in the model

TOMS: Task-Aware Online Model Search

- Crucially, we assume access to an optimistic model f_{opt}
- We would like to search in the space of model parameters Θ to find the model $\hat{f}_\theta$ that results in policy π_θ with planner P that optimizes the true cost-to-go $V^{\pi_\theta}(s_0)$

$$\theta \leftarrow \theta - \frac{\partial V^{\pi_\theta}(s_0)}{\partial \theta}$$

- Infeasible to compute the gradient as $\theta \rightarrow V^{\pi_\theta}(s_0)$ is highly nonlinear and unknown

Online Model Search Framework

Algorithm 16 Online Model Search Framework

Require: Initial state s_1, Planner P, Initial Model θ, Dataset $\mathcal{D} = \{\}$, Model update frequency $\nu \in \mathbb{Z}$

$t \leftarrow 1$, $\pi_\theta \leftarrow P(\hat{f}_\theta)$
while $s_t \notin \mathbb{G}$ **do**
    Compute $a_t \leftarrow \pi_\theta(s_t)$
    Execute a_t in M to get $s_{t+1} = f(s_t, a_t)$
    Update $\mathcal{D} = \mathcal{D} \cup \{(s_t, a_t, s_{t+1})\}$
    if t is a multiple of ν **then**
        Update $\theta \leftarrow \mathsf{MODELSEARCH}(\mathcal{D})$
        Update $\pi_\theta \leftarrow P(\hat{f}_\theta)$

Optimistic Off-Policy Evaluation

- To evaluate $V^{\pi_\theta}(s_0)$ given a dataset $\mathscr{D} = \{(s_t, a_t, s_{t+1})\}$of executed transitions in M is done as follows:
-

Algorithm 17 Optimistic Off-Policy Evaluation

Require: Policy π_θ, Dataset $\mathcal{D}$, start state s_1, horizon H, Distance metric Δ, Distance threshold $\mu \geq 0$

```
Initialize s̃ ← s_1, V̂^{π_θ}(s_1) ← 0
for t = 1 to H do
    Compute ã ← π_θ(s̃)
    Find (s_t, a_t, s_{t+1}) ← argmin_{(s,a,s')∈D} Δ((s̃, ã), (s, a))
    if Δ((s̃, ã), (s_t, a_t)) ≤ μ then
        D ← D \ (s_t, a_t, s_{t+1})
    else
        Compute s_{t+1} ← f_opt(s_t, a_t)
    V̂^{π_θ}(s_1) ← V̂^{π_θ}(s_1) + c(s_t, a_t), s̃ ← s_{t+1}
    if s̃ ∈ G then
        break
return V̂^{π_θ}(s_1)
```

Derivative-Free Model Search

Algorithm 15 Model Search Using Derivative-Free Optimization [Jos+13]

procedure MODELSEARCH($\mathcal{D}$)
  Initial perturbation δ^{init}, minimum perturbation δ^{min}, start parameters θ, Initial state s_1, $\delta \leftarrow \delta^{init}$, planner P
  while $\delta > \delta^{min}$ **do**
    for each dimension of Θ **do**
      while True **do**
        Compute $\{\pi_{\theta-}, \pi_{\theta}, \pi_{\theta+}\} \leftarrow \{P(\hat{f}_{\theta-\delta}), P(\hat{f}_{\theta}), P(\hat{f}_{\theta+\delta})\}$
        Evaluate $\{V^{\pi_{\theta-}}, V^{\pi_{\theta}}, V^{\pi_{\theta+}}\}$
        if $\min(V^{\pi_{\theta-}}(s_1), V^{\pi_{\theta+}}(s_1)) > V^{\pi_{\theta}}(s_1)$ **then**
          break
        if $V^{\pi_{\theta-}}(s_1) < V^{\pi_{\theta+}}(s_1)$ **then**
          $\theta \leftarrow \theta - \delta$
        else
          $\theta \leftarrow \theta + \delta$
    $\delta \leftarrow \frac{\delta}{2}$
return θ

TOMS: Theoretical Guarantees

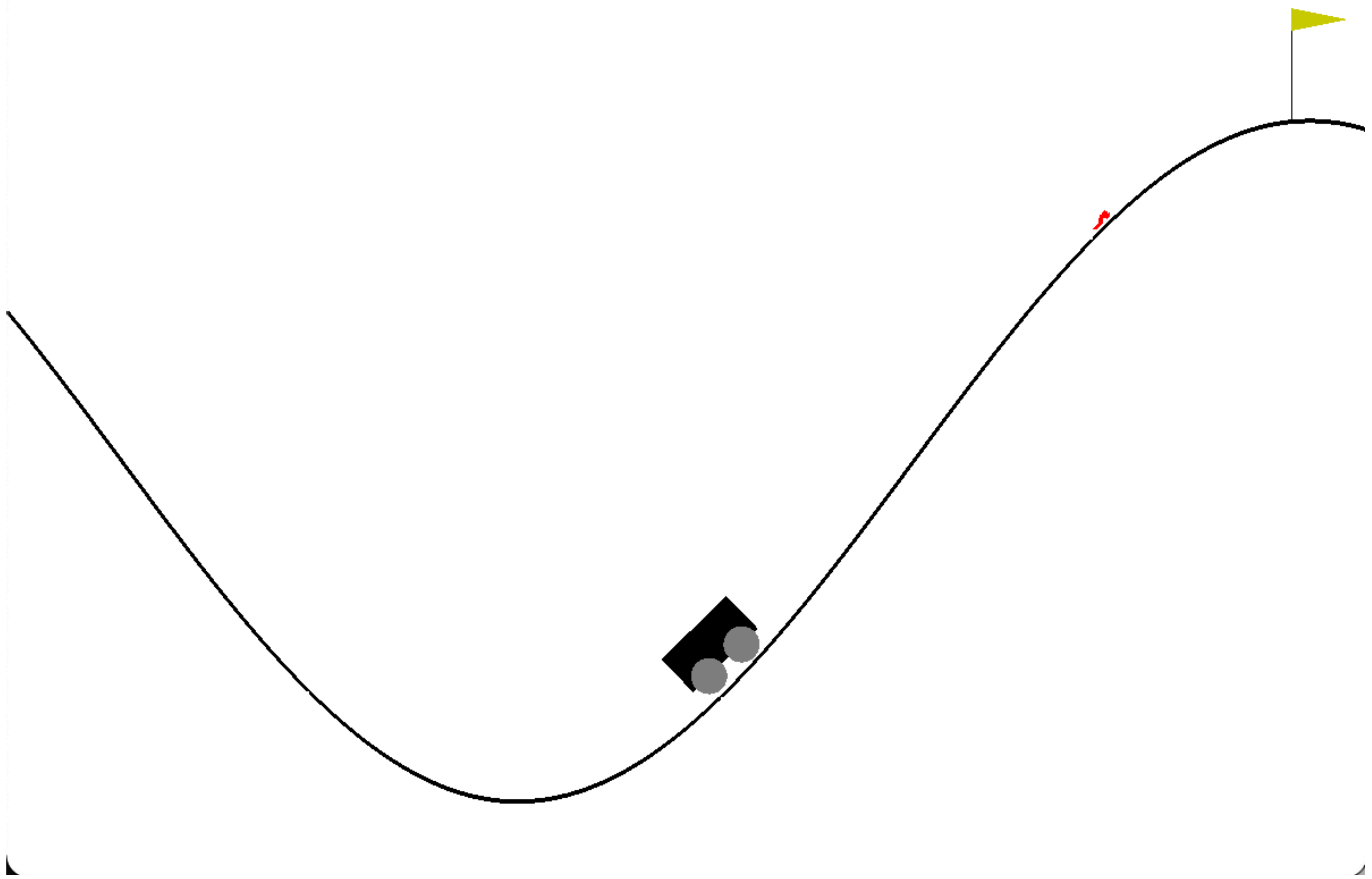
- **Evaluation Guarantee:** If state-action value function Q^{π_θ} is $L-$lipschitz under distance metric Δ for any policy π_θ then we have that the estimate $\hat{V}^{\pi_\theta}$ satisfies

$$\hat{V}^{\pi_\theta}(s_0) \leq V^{\pi_\theta}(s_0) + LH\mu$$

- **Task Completeness Guarantee:** With $\mu = 0$ and unlimited computation, TOMS is guaranteed to reach a goal state if there exists at least a single model in the model class $\mathscr{F}$ that is good enough to result in a policy that can reach a goal in M

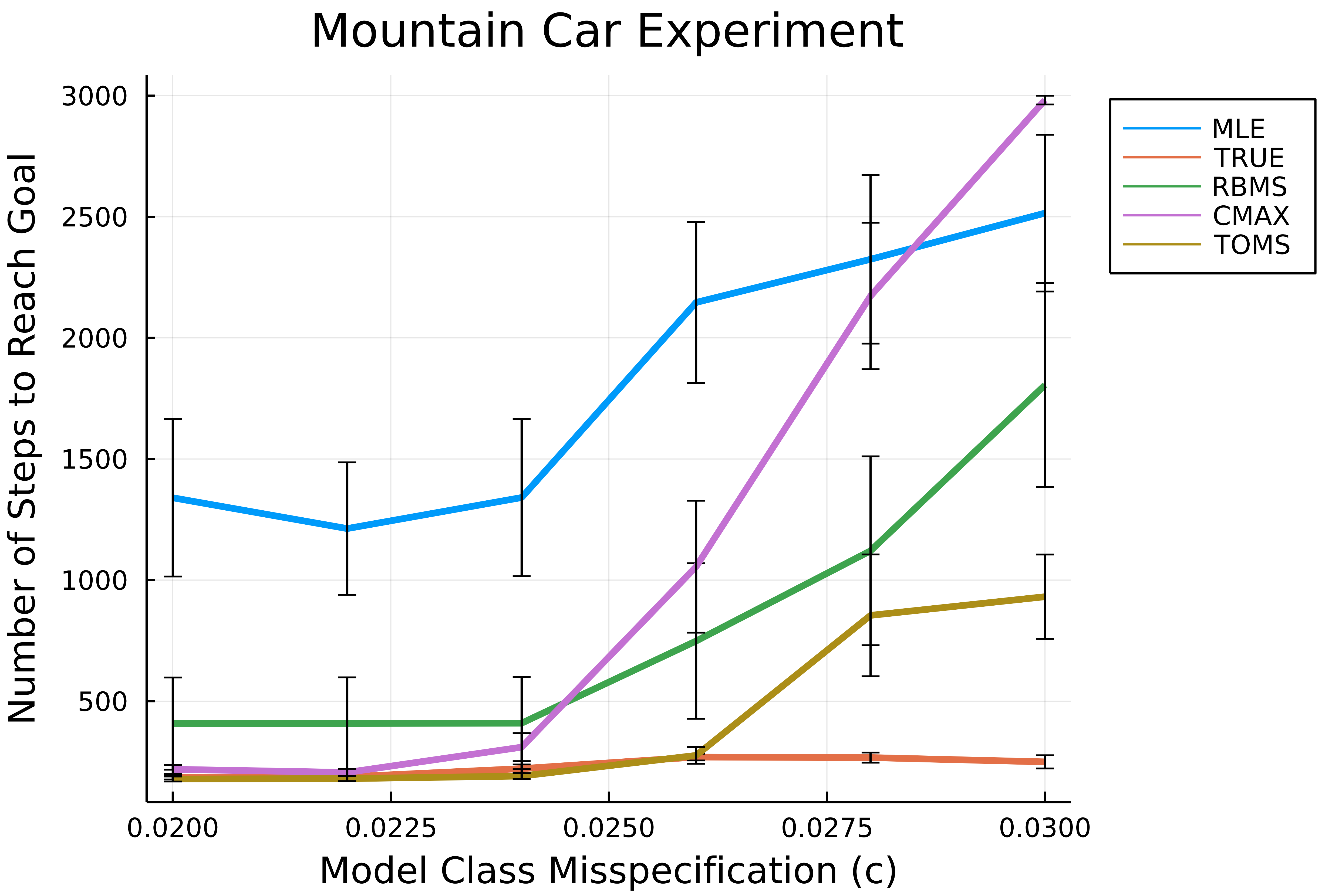
Mountain Car Domain

- Rock that decreases speed by c
- Dynamics: $x_{t+1} = x_t + \dot{x}_t$ and $\dot{x}_{t+1} = \dot{x}_t + u + \theta_1 \cos(\theta_2 x_t)$
- Control $u \in \{-0.001, 0.001\}$
- Model Class $\mathcal{F} = \{(\theta_1, \theta_2) \mid \theta_1, \theta_2 \in \mathbb{R}\}$
- M uses $\theta_1 = -0.0025, \theta_2 = 3$



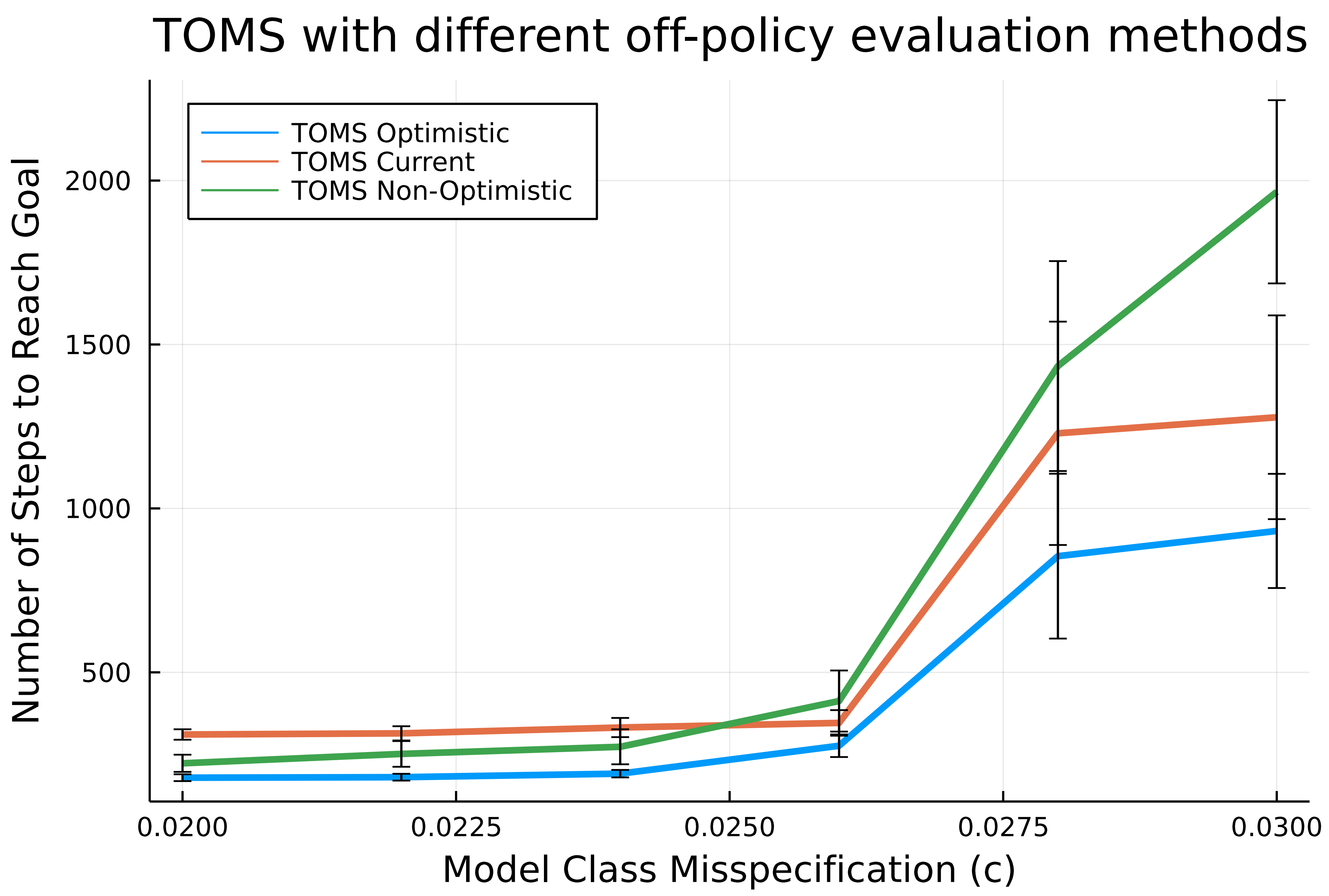
TOMS Experiment 1

- MLE - optimizes prediction error
- TRUE - Uses true dynamics
- RBMS - Uses only the dataset $\mathscr{D}$ for evaluation
- CMAX - penalizes any incorrect transition
- TOMS - Our approach that uses optimistic evaluation
- f_{opt} uses $\theta_1 = -0.0025, \theta_2 = 3$ but does not have rock



TOMS Experiment 2

- Optimistic - our approach
- Current - Uses the model evaluated as the fallback model for evaluation
- Non-optimistic - Uses a non-optimistic model as fallback for evaluation



Simulation Lemma

Lemma 8.1.1 (Undiscounted Deterministic Dynamics Simulation Lemma)**.** *Let M, M' be two Markov Decision Processes with the same cost function. If we have a fixed start state s_0, a deterministic policy $\pi : \mathbb{S} \to \mathbb{A}$, and M, M' have deterministic dynamics $f, f' : \mathbb{S} \times \mathbb{A} \to \mathbb{S}$. Then we have,*

$$
\begin{aligned}
J_M(\pi) &= J_{M'}(\pi) + \sum_{t=0}^{\infty} c(s_t^M, \pi(s_t^M)) + V_{M'}^{\pi}(s_{t+1}^M) - V_{M'}^{\pi}(s_t^M) && (8.1) \\
&= J_{M'}(\pi) + \sum_{t=0}^{\infty} V_{M'}^{\pi}(s_{t+1}^M) - V_{M'}^{\pi}(f'(s_t^M, \pi(s_t^M))) && (8.2)
\end{aligned}
$$

$$
\begin{aligned}
V_{M'}^{\pi}(s_{t+1}^M) - V_{M'}^{\pi}(f'(s_t^M, \pi(s_t^M))) &\leq L\|s_{t+1}^M - f'(s_t^M, \pi(s_t^M))\| \\
&\leq L\|f(s_t^M, \pi(s_t^M)) - f'(s_t^M, \pi(s_t^M))\|
\end{aligned}
$$