CSE-571
AI-based Mobile Robotics

Reinforcement Learning
for
Active Sensing
Manipulator Control

Reinforcement Learning

- Same setting as MDP

- Passive:
  - Policy given, transition model and reward are unknown
  - Robot wants to learn value function for the given policy
  - Similar to policy evaluation, but without knowledge of transitions and reward

- Active:
  - Robot also has to learn optimal policy

Direct Utility Estimation

- Each trial gives a reward for the visited states

$$V_\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \; R(s_t) \mid s_0 = s, \pi\right]$$

- Determine average over many trials

- Problem: Doesn't use knowledge about state connections

$$V(s) = R(s) + \max_a \gamma \sum_{s'} p(s' \mid s, a) \, V(s')$$

Temporal Difference Learning

- Make use of observed transitions

- Uses difference in utilities between successive states

$$V(s) \leftarrow V(s) + \alpha\,(R(s) + \gamma V(s') - V(s))$$

- Learning rate *a* has to decrease with number of visits to a state
- Does not require / estimate explicit transition model
- Still assumes policy is given

Active Reinforcement Learning

- First learn model, then use Bellman equation

$$V(s) = R(s) + \max_a \gamma \sum_{s'} p(s' | s, a) V(s')$$

- Use this model to perform optimal policy

- Problem?

- Robot must trade off exploration (try new actions/states) and exploitation (follow current policy)

Q-Learning

- Model-free: learn action-value function
- Equilibrium for Q-values:

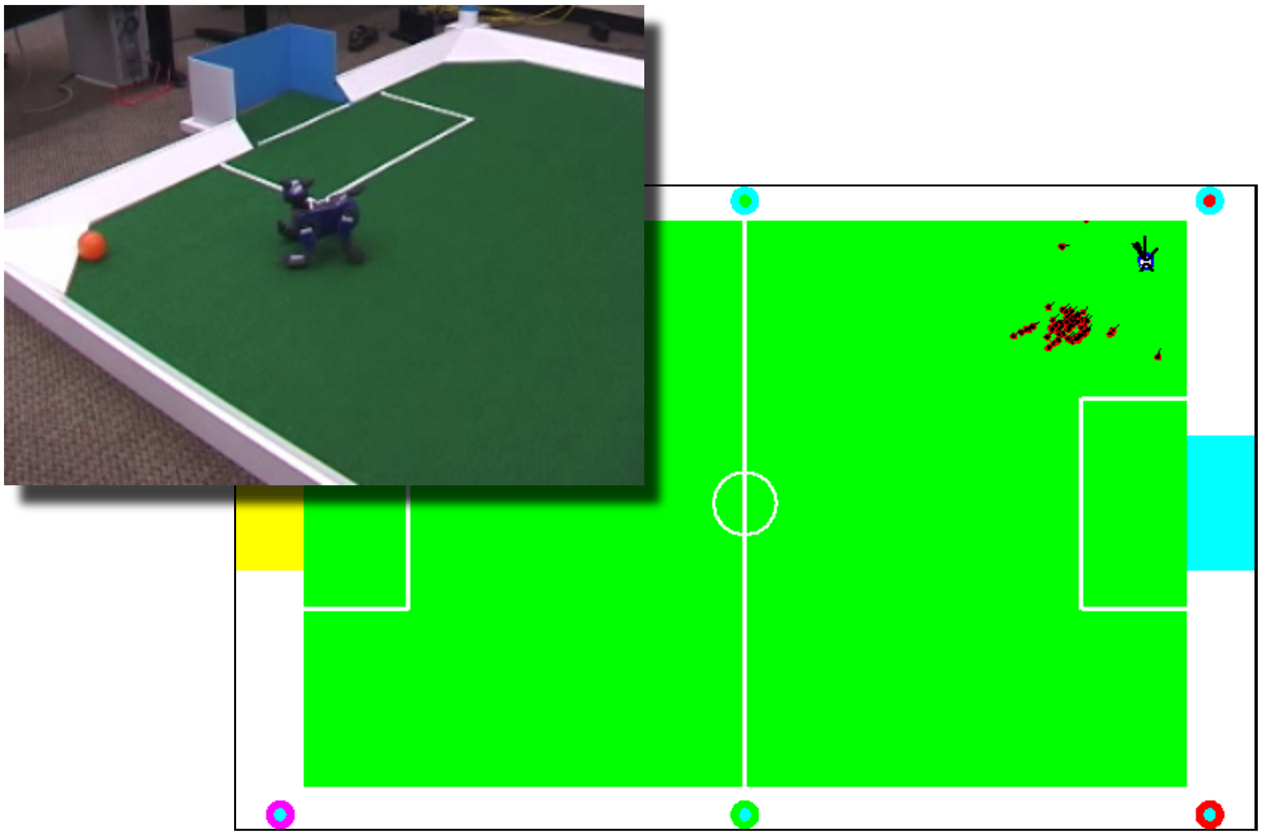
$$Q(a,s) = R(s) + \gamma \sum_{s'} p(s'|a,s) \max_{a'} Q(a',s')$$

$$V(s) = R(s) + \max_{a} \gamma \sum_{s'} p(s'|s,a) V(s')$$

- Updates:

$$Q(a,s) \leftarrow Q(a,s) + \alpha \left(R(s) + \gamma \max_{a'} Q(a',s') - Q(a,s)\right)$$

RL for Active Sensing



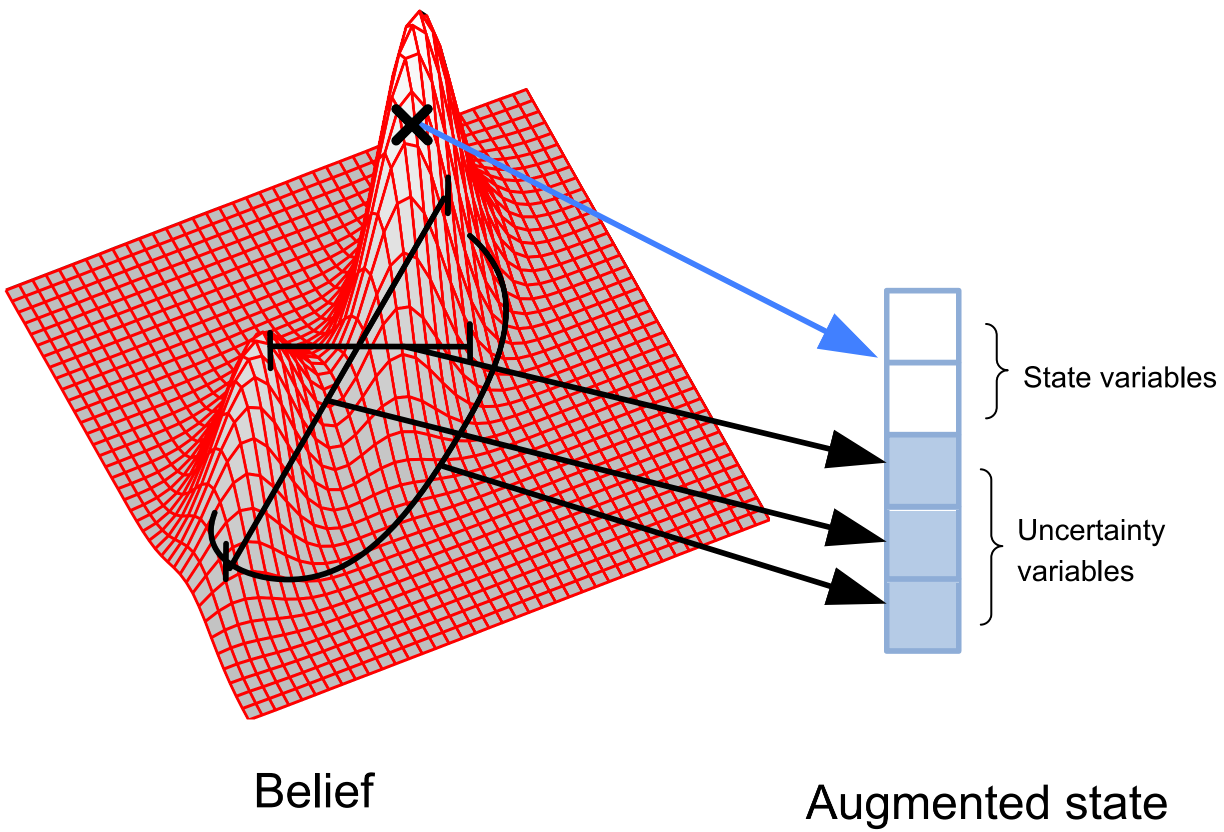
Active Sensing

- Sensors have limited coverage & range
- Question: Where to move / point sensors?
- Typical scenario: Uncertainty in only one type of state variable
  - Robot location [Fox et al., 98; Kroese & Bunschoten, 99; Roy & Thrun 99]
  - Object / target location(s) [Denzler & Brown, 02; Kreuchner et al., 04, Chung et al., 04]
- Predominant approach:
  Minimize expected uncertainty (entropy)

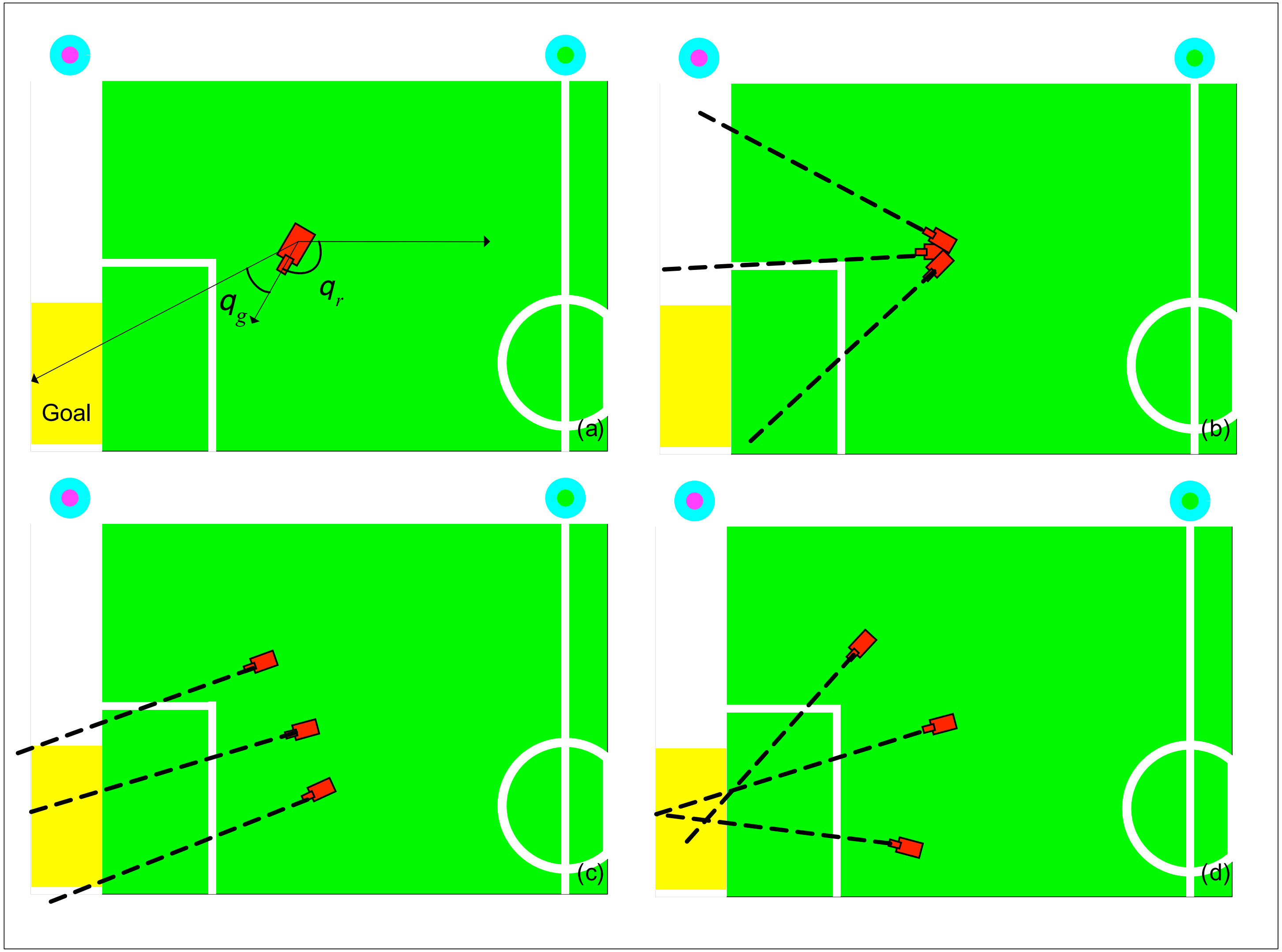
Active Sensing in Multi-State Domains

- Uncertainty in multiple, different state variables
  Robocup: robot & ball location, relative goal location, …
- Which uncertainties should be minimized?
- Importance of uncertainties changes over time.
  - Ball location has to be known very accurately before a kick.
  - Accuracy not important if ball is on other side of the field.
- Has to consider sequence of sensing actions!
- RoboCup: typically use hand-coded strategies.

Converting Beliefs to Augmented States



Projected Uncertainty (Goal Orientation)



Why Reinforcement Learning?

- No accurate model of the robot and the environment.
- Particularly difficult to assess how (projected) entropies evolve over time.
- Possible to simulate robot and noise in actions and observations.

Least-squares Policy Iteration

- Model-free approach
- Approximates Q-function by linear function of state features

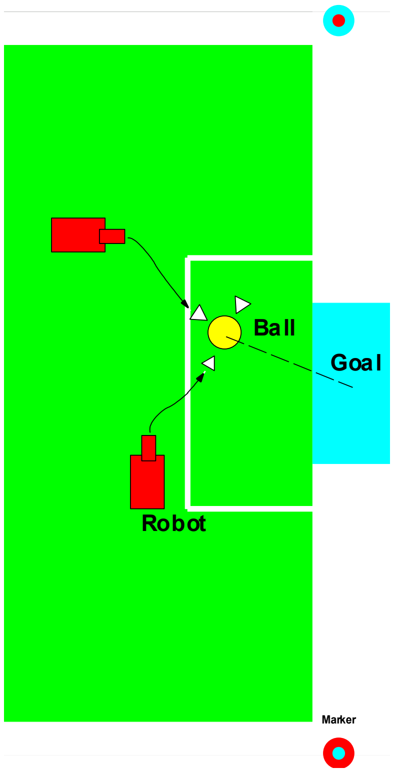
$$Q^{\pi}(s,a) \approx \hat{Q}^{\pi}(s,a;w) = \sum_{j=1}^{k} \phi_j(s,a)\, w_j$$

- No discretization needed
- No iterative procedure needed for policy evaluation
- Off-policy: can re-use samples

[Lagoudakis and Parr ’01,’03]

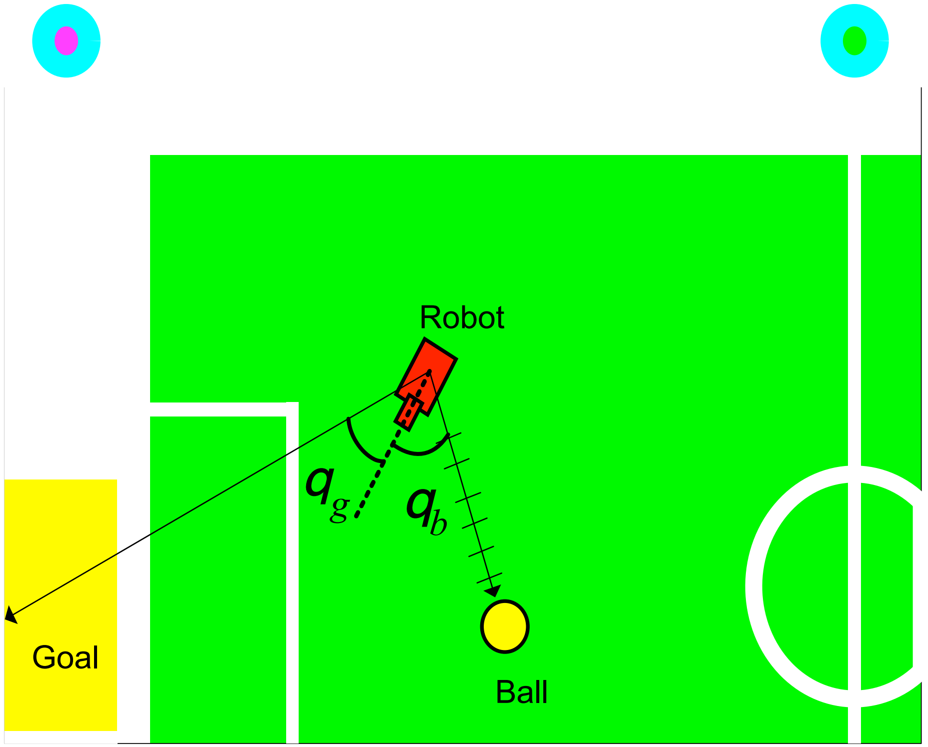
Application:
Active Sensing for Goal Scoring

- Task: AIBO trying to score goals
- Sensing actions: look at ball, or the goals, or the markers
- Fixed motion control policy: Uses most likely states to dock the robot to the ball, then kicks the ball into the goal.
- Find sensing strategy that "best" supports the given control policy.



Augmented State Space and Features

- State variables:
  - Distance to ball
  - Ball Orientation
- Uncertainty variables:
  - Ent. of ball location
  - Ent. of robot location
  - Ent. of goal orientation
- Features:

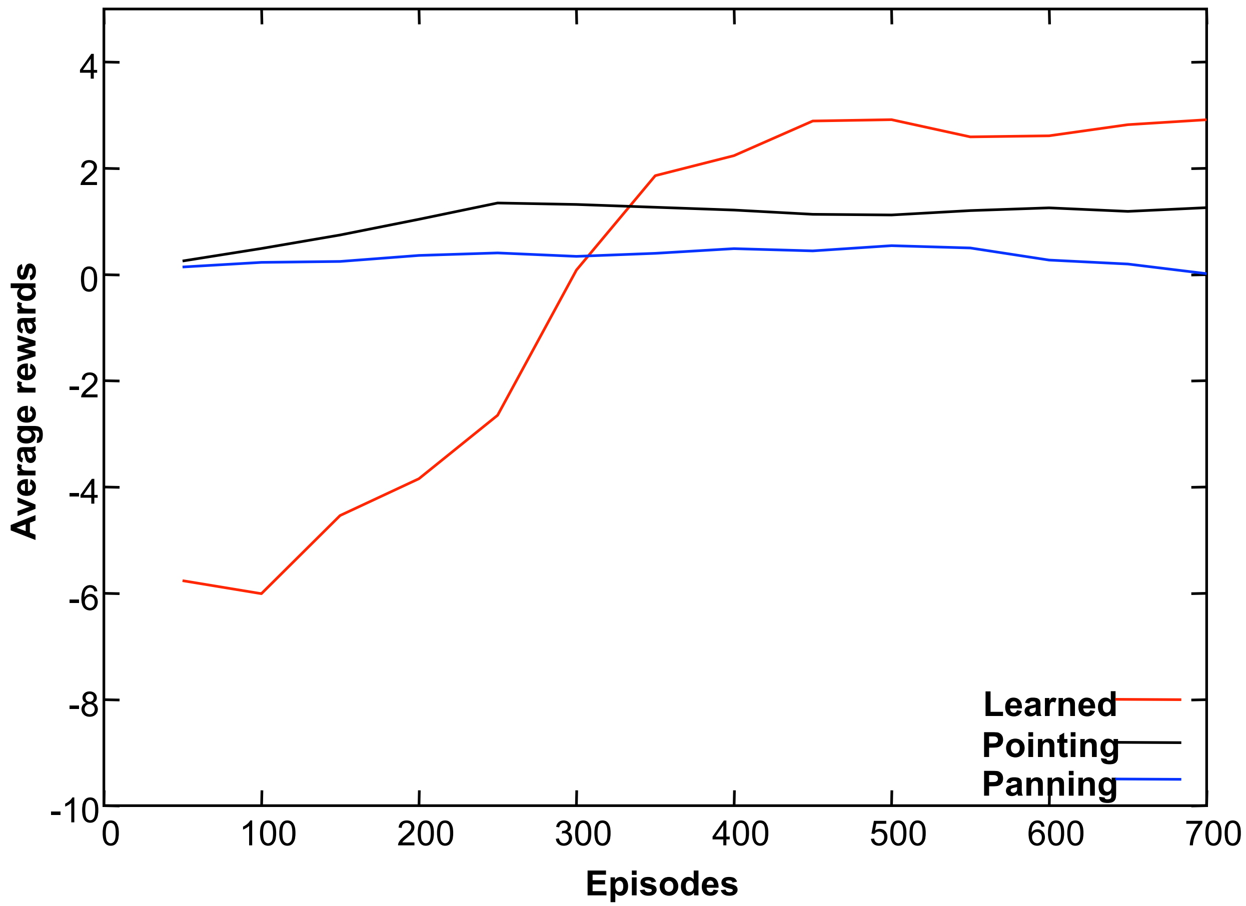


$$\phi(s,a,d_b) = \left\langle |\theta_b|, H_b, H_{\theta_g}, H_r, |\theta_a|, 1 \right\rangle$$

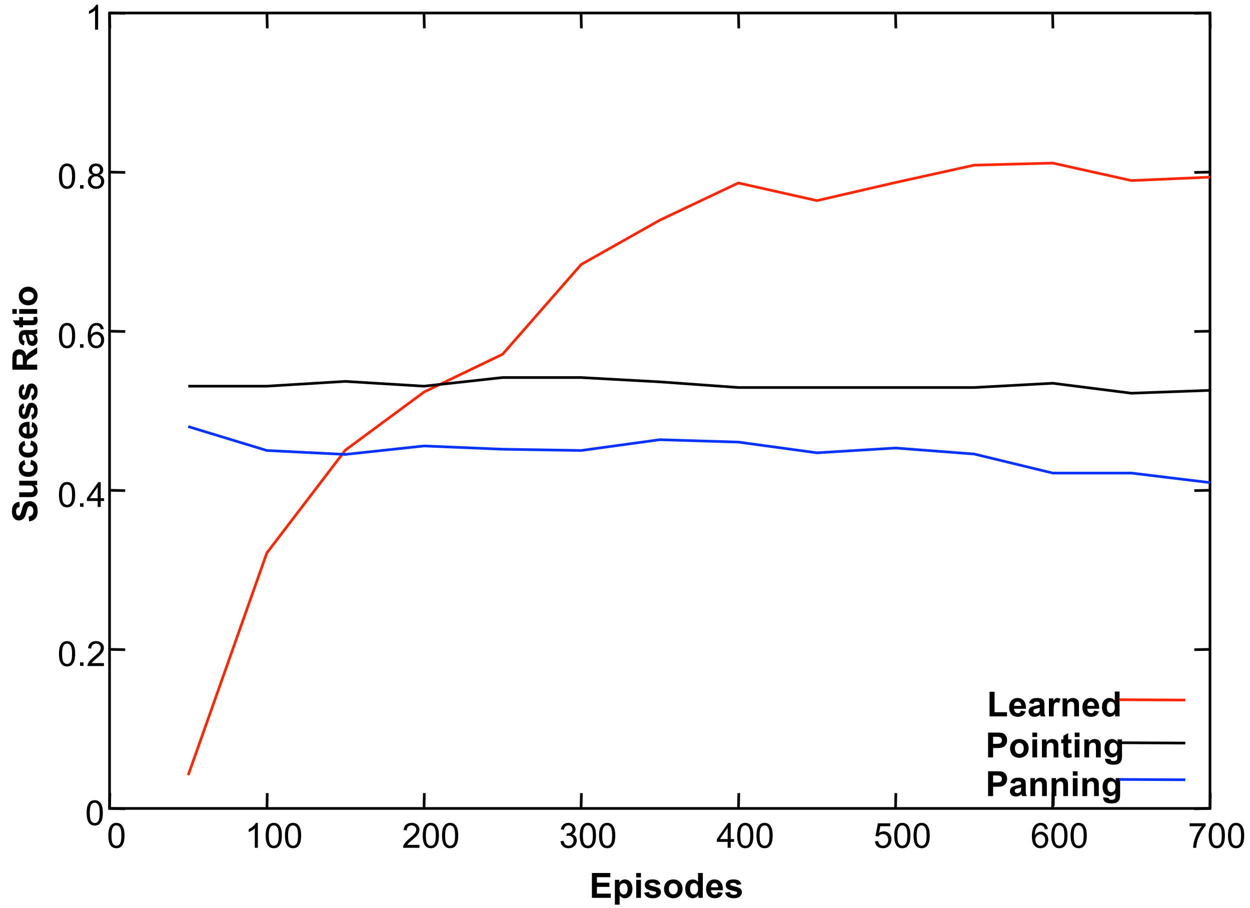
Experiments

- Strategy learned from simulation
- Episode ends when:
  - Scores (reward +5)
  - Misses (reward 1.5 – 0.1)
  - Loses track of the ball (reward -5)
  - Fails to dock / accidentally kicks the ball away (reward -5)
- Applied to real robot
- Compared with 2 hand-coded strategies
  - Panning: robot periodically scans
  - Pointing: robot periodically looks up at markers/goals

Rewards (simulation)



Success Ratio (simulation)



Learned Strategy

- Initially, robot learns to dock (only looks at ball)
- Then, robot learns to look at goal and markers

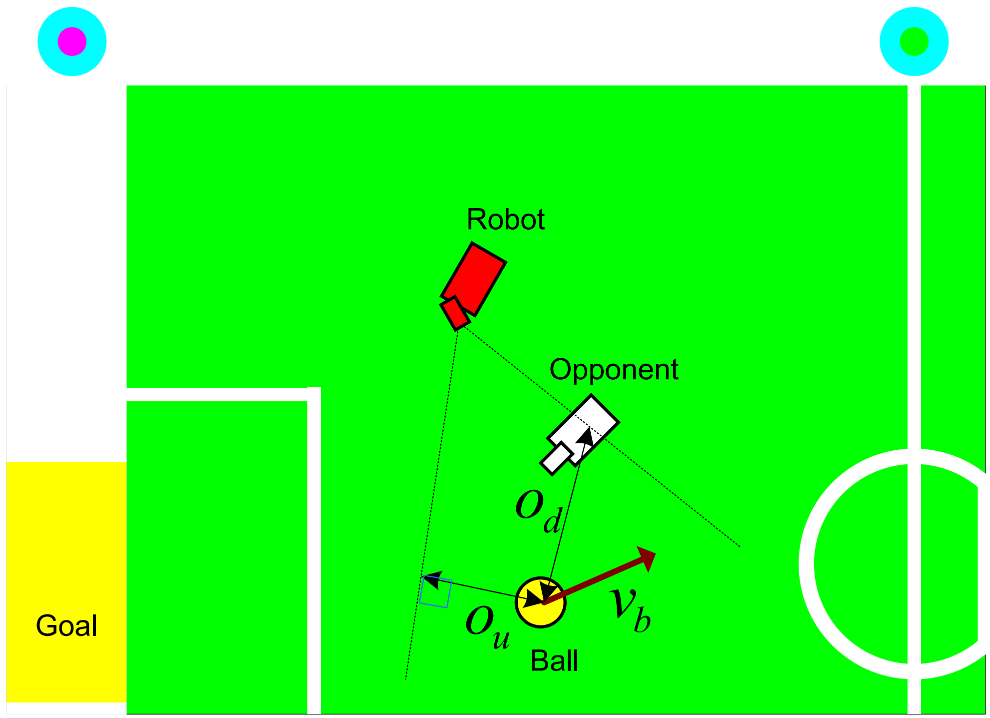
- Robot looks at ball when docking
- Briefly before docking, adjusts by looking at the goal
- Prefers looking at the goal instead of markers for location information

Results on Real Robots

- 45 episodes of goal kicking

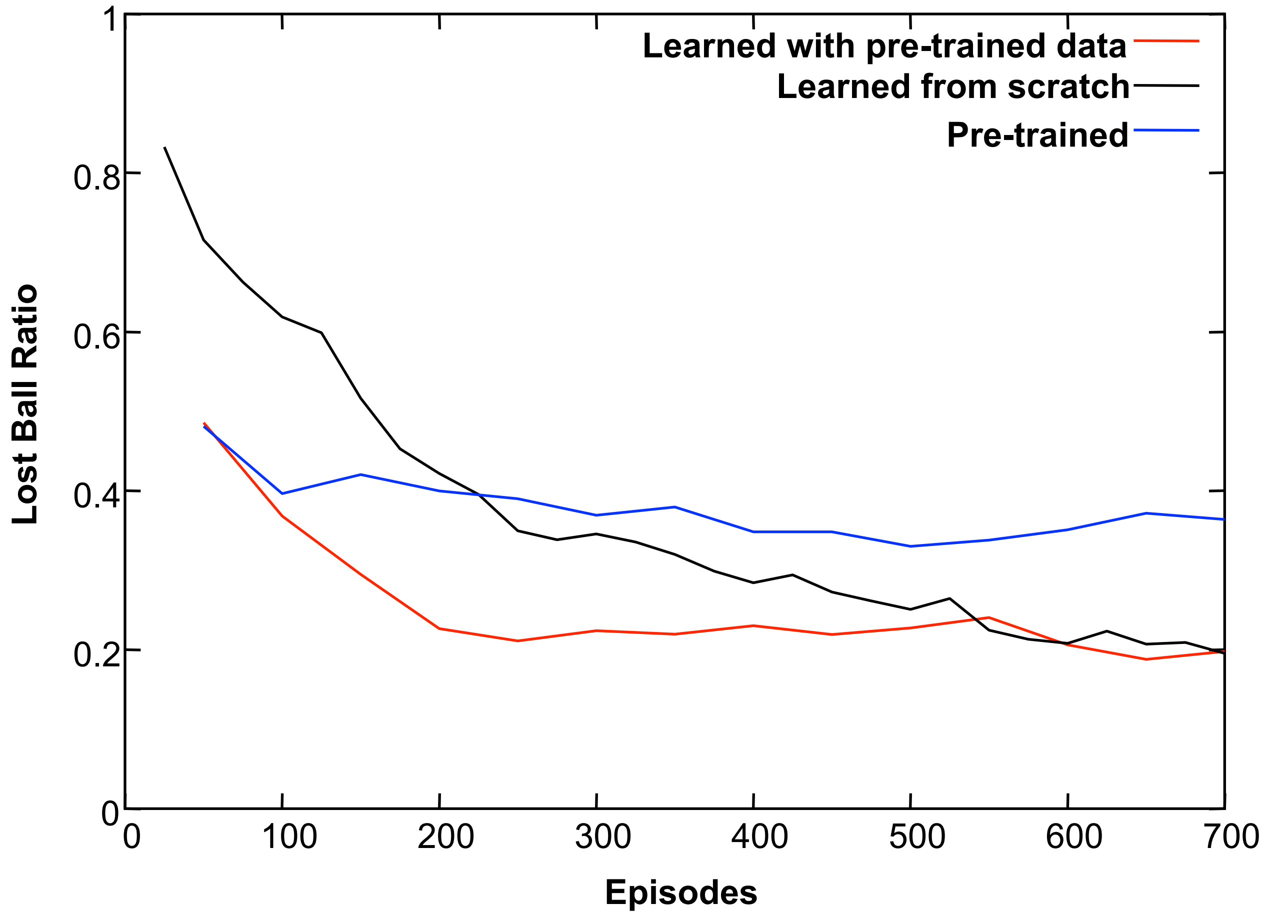
	Goals	Misses	Avg. Miss Distance	Kick Failures
Learned	31	10	6±0.3cm	4
Pointing	22	19	9±2.2cm	4
Panning	15	21	22±9.4cm	9

Adding Opponents



Additional features: ball velocity, knowledge about other robots

Learning With Opponents



- Robot learned to look at ball when opponent is close to it. Thereby avoids losing track of it.

Summary

- Learned effective sensing strategies that make good trade-offs between uncertainties
- Results on a real robot show improvements over carefully tuned, hand-coded strategies
- Augmented-MDP (with projections) good approximation for RL
- LSPI well suited for RL on augmented state spaces

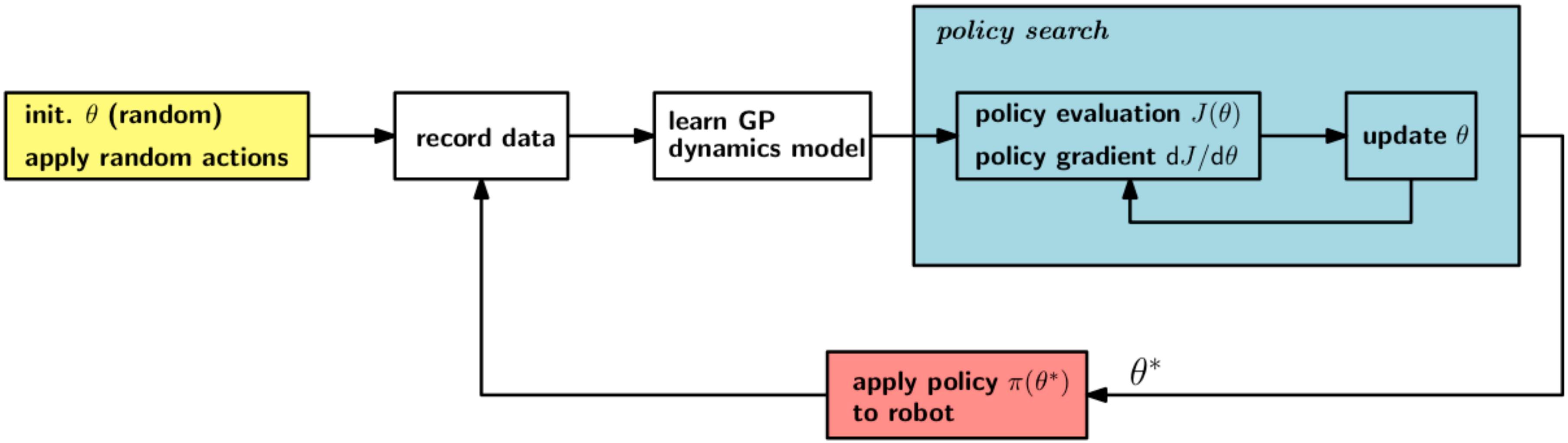
Policy Search

- Works directly on parameterized representation of policy
- Compute gradient of expected reward wrt. policy parameters
- Get gradient analytically or empirical via sampling (simulation)

PILCO: Probabilistic Inference for Learning Control

- Model-based policy search
- Learn Gaussian process dynamics model
- Goal-directed exploration
  → no "motor babbling" required
- Consider model uncertainties
  → robustness to model errors
- Extremely data efficient

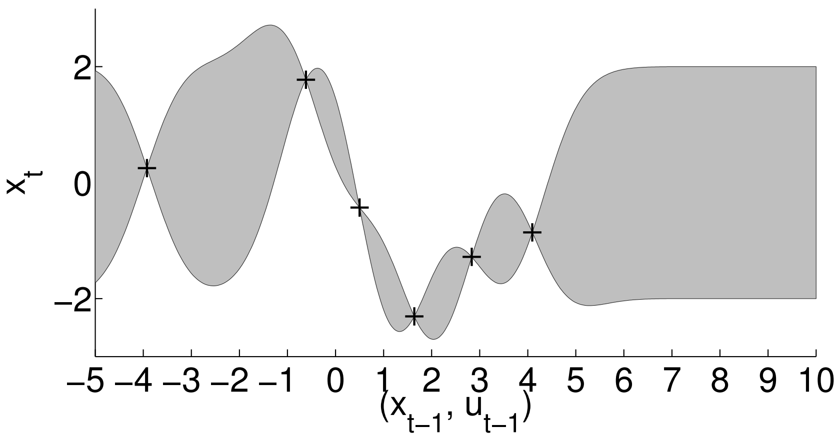
PILCO: Overview



- Cost function given
- Policy: mapping from state to control
- Rollout: plan using current policy and GP model
- Policy parameter update via CG/BFGS

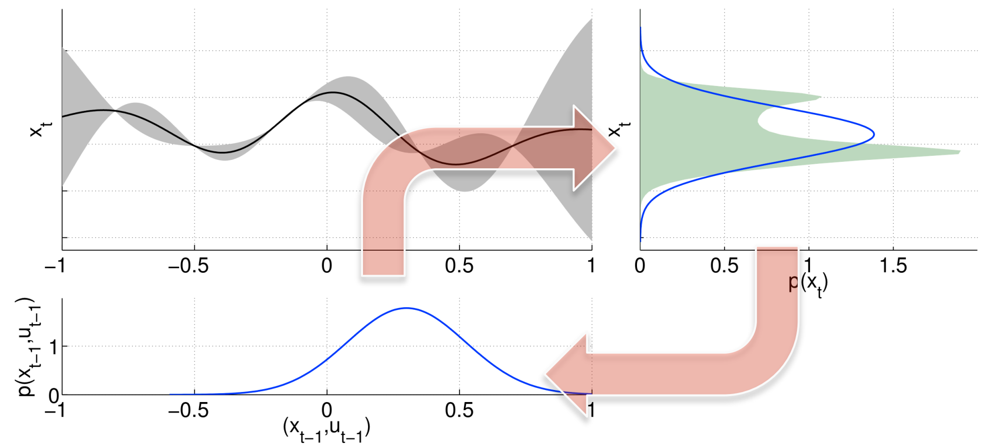
Model Learning and Approximate Inference

Gaussian Process Forward Model



- Probabilistic GP model consistently describes model uncertainties

Approximate Inference for Policy Learning



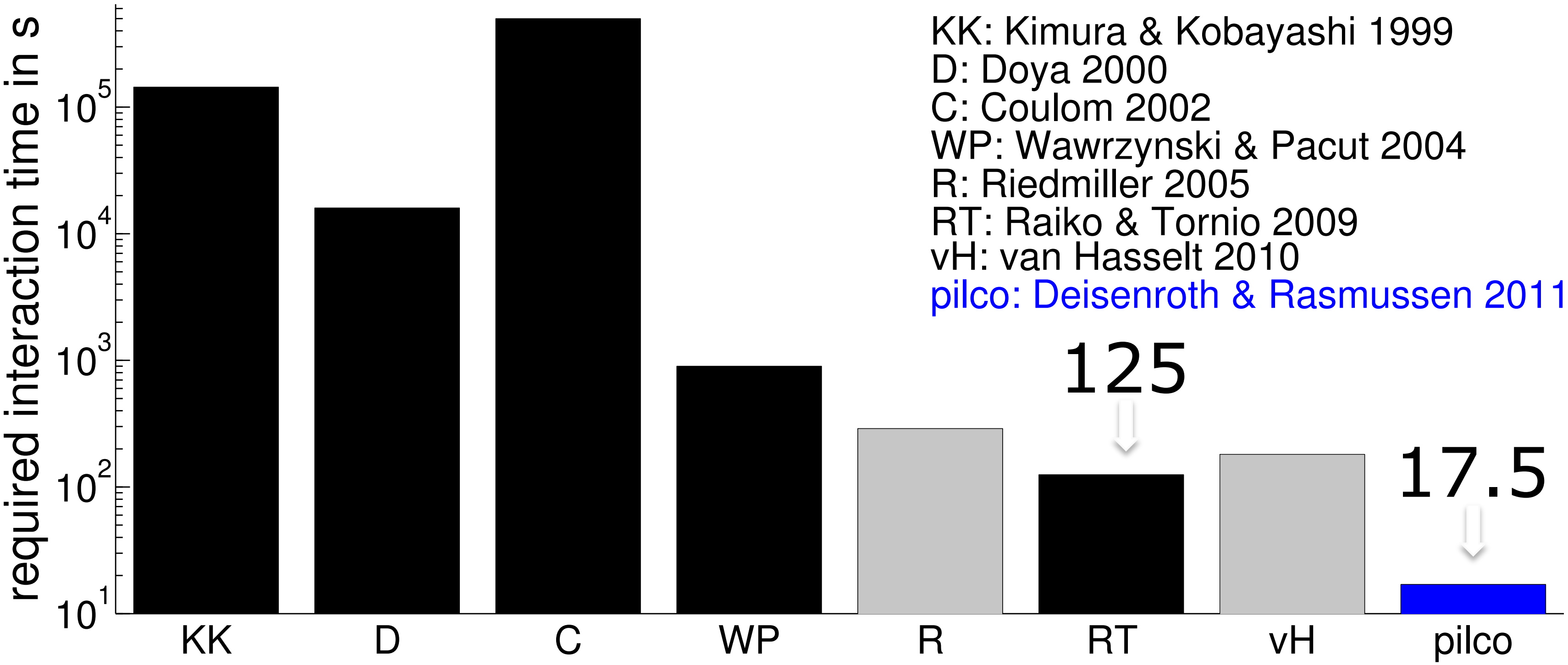
- Long-term planning requires approximate inference: **moment matching**
- **Model uncertainties are integrated out** analytically (opposed toMC [Bagnell-00])

$$E[x_t] = E_{x_{t-1}}\Big[E_f[f(x_{t-1},\pi(x_{t-1},\theta))|x_{t-1}]\Big]$$
$$V[x_t] = E_{x_{t-1}}\Big[V_f[f(x_{t-1},\pi(x_{t-1},\theta))|x_{t-1}]\Big]+V_{x_{t-1}}\Big[E_f[f(x_{t-1},\pi(x_{t-1},\theta))|x_{t-1}]\Big]$$

Demo: Standard Benchmark Problem

- Swing pendulum up and balance in inverted position
- Learn nonlinear control from scratch
- 4D state space, 300 controll parameters
- **7 trials/17.5 sec experience**
- Control freq.: 10 Hz

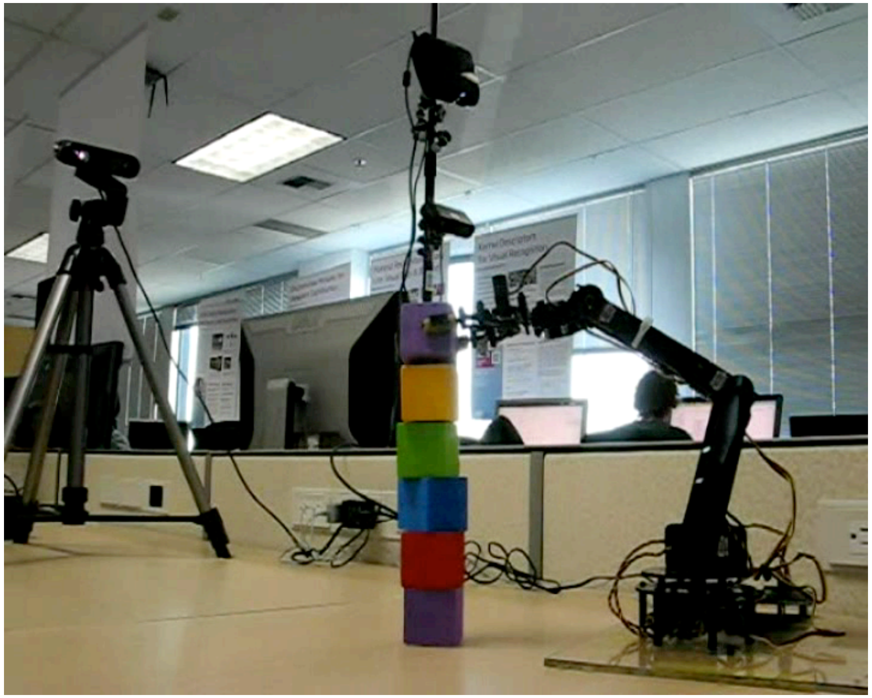
Data Efficiency in Comparison



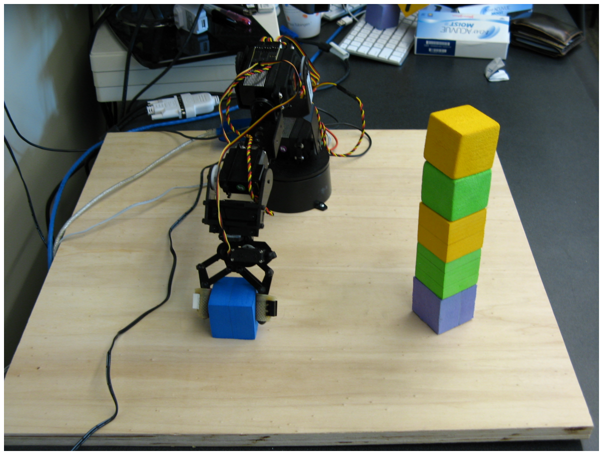
- Gray bars: balancing only
- Learning from scratch
- **Also applied to unicycle, double pendulum**

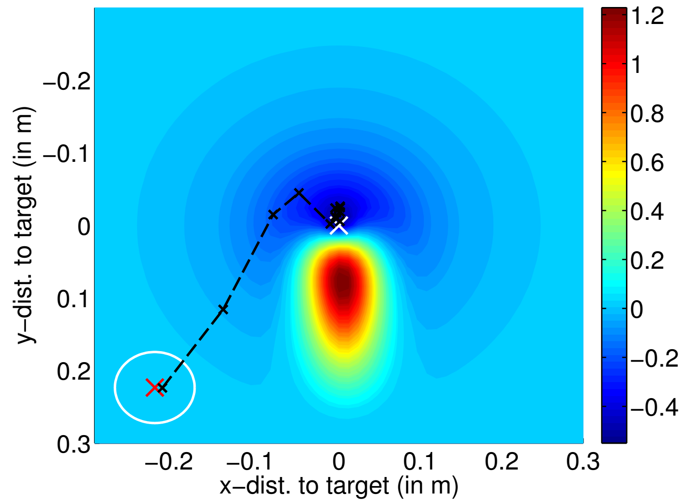
Controlling a Low-Cost Robotic Manipulator

- Low-cost system ($500 for robot arm and Kinect)
- Very noisy
- No sensor information about robot's joint configuration used
- **Goal:** Learn to stack tower of 5 blocks from scratch
- Kinect camera for tracking block in end-effector
- State: coordinates (3D) of block center (from Kinect camera)
- 4 controlled DoF
- 20 learning trials for stacking 5 blocks (5 seconds long each)
- Account for system noise, e.g.,
  - Robot arm
  - Image processing



Collision Avoidance

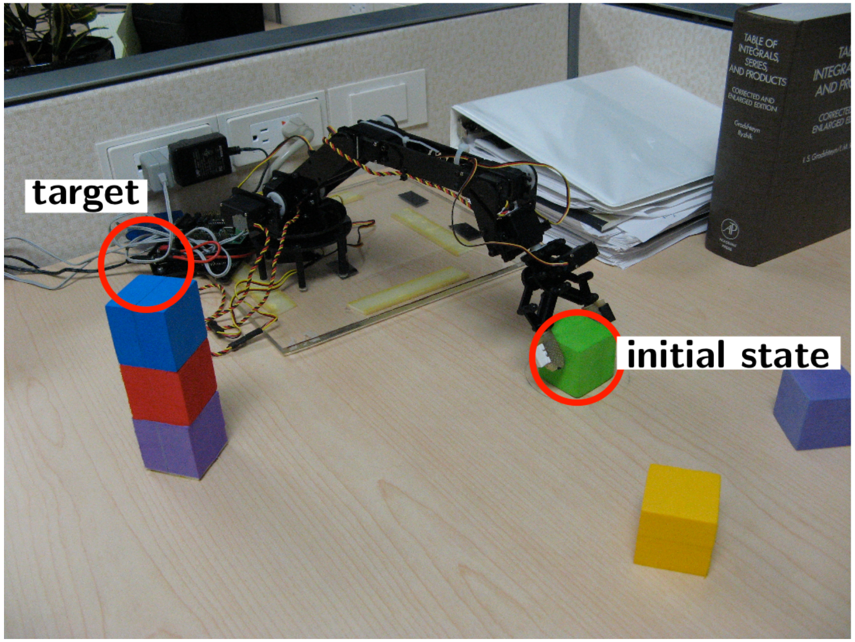




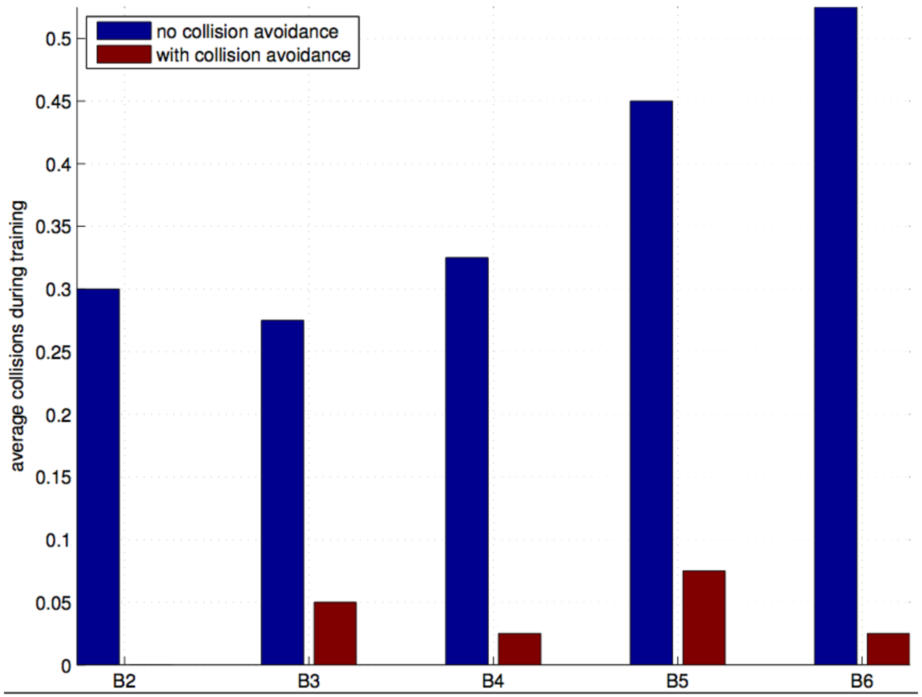
- Use valuable prior information about obstacles if available
- Incorporation into planning → penalize in cost function

Collision Avoidance Results

Experimental Setup



Training runs (during learning) with collisions



- Cautious learning and exploration (rather safe than risky-successful)
- Learning slightly slower, but with significantly fewer collisions during training
- Average collision reduction (during training): 32.5% → 0.5%