Learning to Control an Octopus Arm with Gaussian Process Temporal Difference Methods

Yaakov Engel

Joint work with Peter Szabo and Dmitry Volkinshtein (ex. Technion)

ALBERTA INGENUITY CENTRE FOR MACHINE LEARNING

UNIVERSITY OF ALBERTA

Why use GPs in RL?

- A Bayesian approach to value estimation
- Forces us to to make our assumptions explicit
- Non-parametric – priors are placed and inference is performed directly in **function** space (kernels).
- Domain knowledge intuitively coded into priors
- Provides full posterior, not just point estimates
- Efficient, on-line implementation, suitable for large problems

Markov Decision Processes

$\mathcal{X}$: state space

$\mathcal{U}$: action space

p: $\mathcal{X} \times \mathcal{X} \times \mathcal{U} \to [0,1], \quad \mathbf{x}_{t+1} \sim p(\cdot|\mathbf{x}_t, \mathbf{u}_t)$

q: $\mathbb{R} \times \mathcal{X} \times \mathcal{U} \to [0,1], \quad R(\mathbf{x}_t, \mathbf{u}_t) \sim q(\cdot|\mathbf{x}_t, \mathbf{u}_t)$

A Stationary policy:

μ: $\mathcal{U} \times \mathcal{X} \to [0,1], \quad \mathbf{u}_t \sim \mu(\cdot|\mathbf{x}_t)$

Discounted Return: $D^\mu(\mathbf{x}) = \sum_{i=0}^{\infty} \gamma^i R(\mathbf{x}_i, \mathbf{u}_i) | (\mathbf{x}_0 = \mathbf{x})$

Value function: $V^\mu(\mathbf{x}) = \mathbf{E}_\mu[D^\mu(\mathbf{x})]$

Goal: Find a policy μ^* maximizing $V^\mu(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{X}$

Bellman's Equation

For a fixed policy μ:

$$V^{\mu}(\mathbf{x}) = \mathbf{E}_{\mathbf{x}',\mathbf{u}|\mathbf{x}}\Big[R(\mathbf{x},\mathbf{u}) + \gamma V^{\mu}(\mathbf{x}')\Big]$$

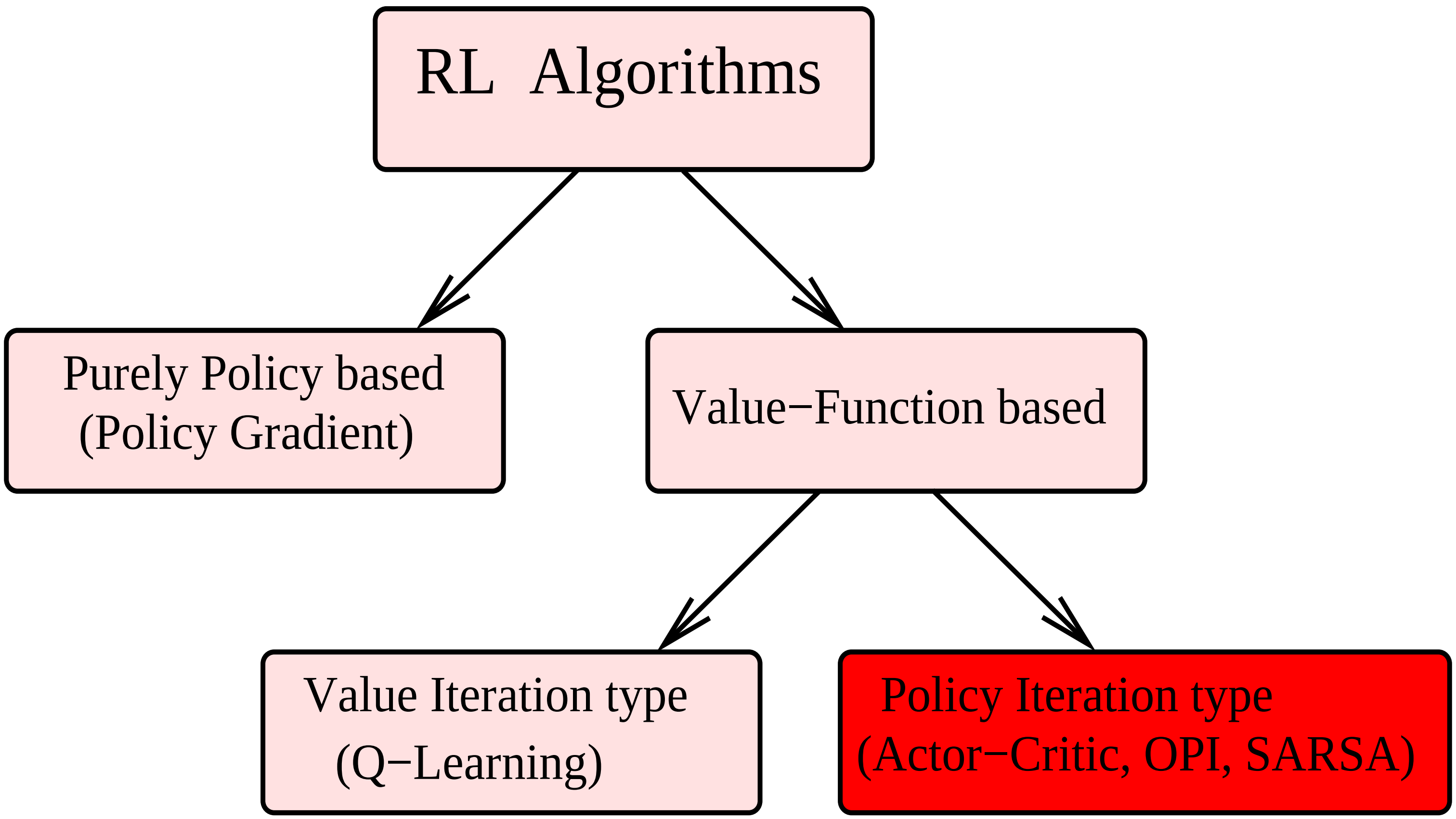
Optimal value and policy:

$$V^{*}(\mathbf{x}) = \max_{\mu} V^{\mu}(\mathbf{x}) \; , \quad \mu^{*} = \operatorname*{argmax}_{\mu} V^{\mu}(\mathbf{x})$$

How to solve it?

- Methods based on Value Iteration (e.g. Q-learning)
- Methods based on Policy Iteration (e.g. SARSA, OPI, Actor-Critic)

Solution Method Taxonomy



PI methods need a “subroutine” for policy evaluation

Gaussian Process Temporal Difference Learning

Model Equations:

$$\boxed{R(\mathbf{x}_i) = V(\mathbf{x}_i) - \gamma V(\mathbf{x}_{i+1}) + N(\mathbf{x}_i)}$$

Or, in compact form:

$$\boxed{R_t = \mathbf{H}_{t+1} V_{t+1} + N_t}$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & 0 & \dots & 0 \\ 0 & 1 & -\gamma & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 1 & -\gamma \end{bmatrix}.$$

Our (Bayesian) goal:

Find the posterior distribution of $V(\cdot)$, given a sequence of observed states and rewards.

The Posterior

General noise covariance: $\mathbf{Cov}[N_t] = \mathbf{\Sigma}_t$

Joint distribution:

$$\begin{bmatrix} R_{t-1} \\ V(\mathbf{x}) \end{bmatrix} \sim \mathcal{N}\left\{ \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{H}_t\mathbf{K}_t\mathbf{H}_t^\top + \mathbf{\Sigma}_t & \mathbf{H}_t\mathbf{k}_t(\mathbf{x}) \\ \mathbf{k}_t(\mathbf{x})^\top\mathbf{H}_t^\top & k(\mathbf{x},\mathbf{x}) \end{bmatrix} \right\}$$

Invoke Bayes' Rule:

$$\mathbf{E}[V(\mathbf{x})|R_{t-1}] = \mathbf{k}_t(\mathbf{x})^\top \boldsymbol{\alpha}_t$$

$$\mathbf{Cov}[V(\mathbf{x}), V(\mathbf{x}')|R_{t-1}] = k(\mathbf{x},\mathbf{x}') - \mathbf{k}_t(\mathbf{x})^\top \mathbf{C}_t \mathbf{k}_t(\mathbf{x}')$$

$$\mathbf{k}_t(\mathbf{x}) = (k(\mathbf{x}_1,\mathbf{x}), \ldots, k(\mathbf{x}_t,\mathbf{x}))^\top$$

The Octopus Arm

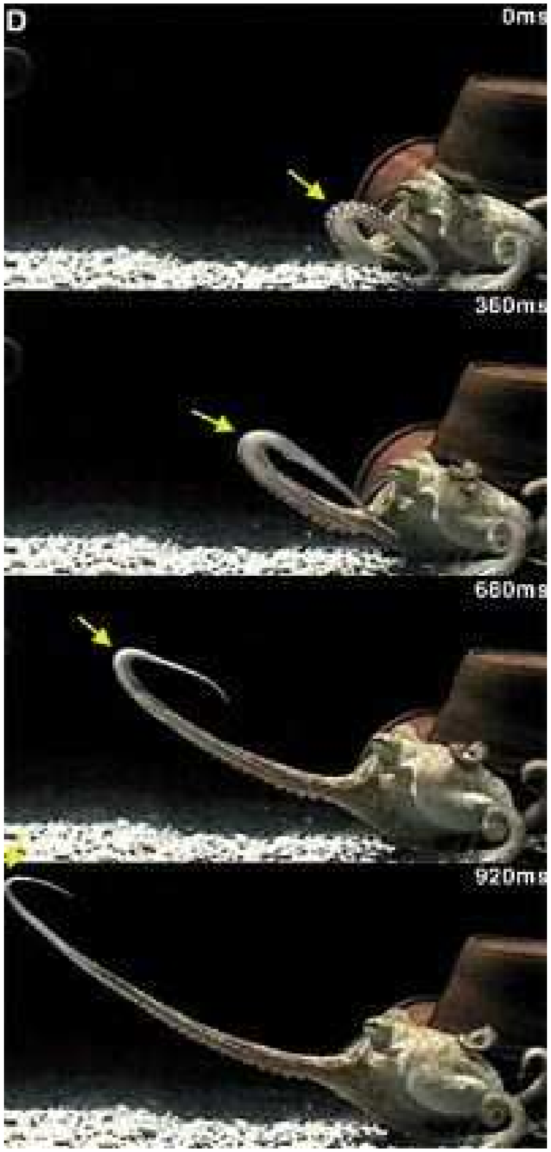
Can bend and twist at any point

Can do this in any direction

Can be elongated and shortened

Can change cross section

Can grab using any part of the arm



Virtually infinitely many DOF

The Muscular Hydrostat Mechanism

A constraint: Muscle fibers can only contract (actively)

In vertebrate limbs, two separate muscle groups - agonists and antagonists - are used to control each DOF of every joint, by exerting opposite torques.

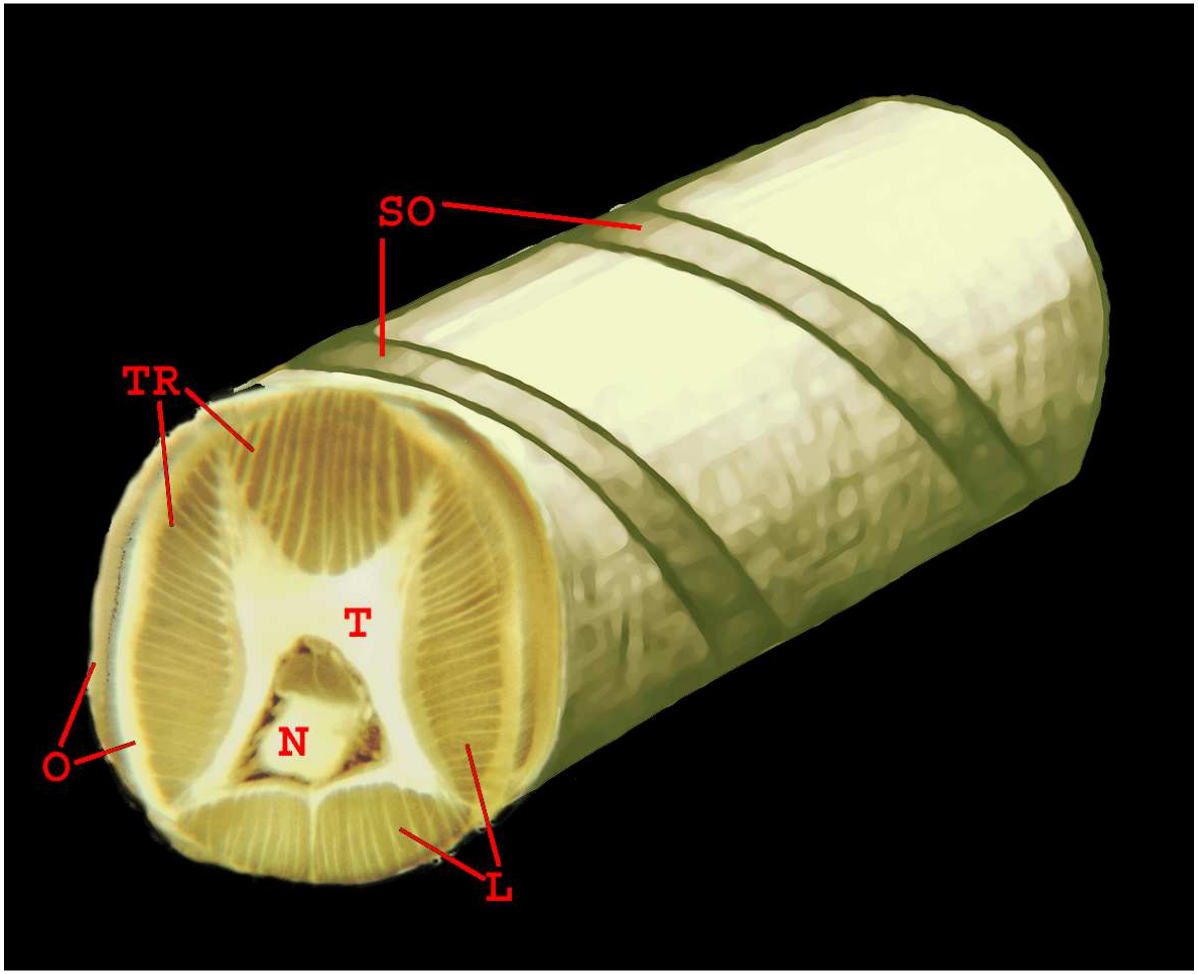
But the Octopus has no skeleton!

Balloon example

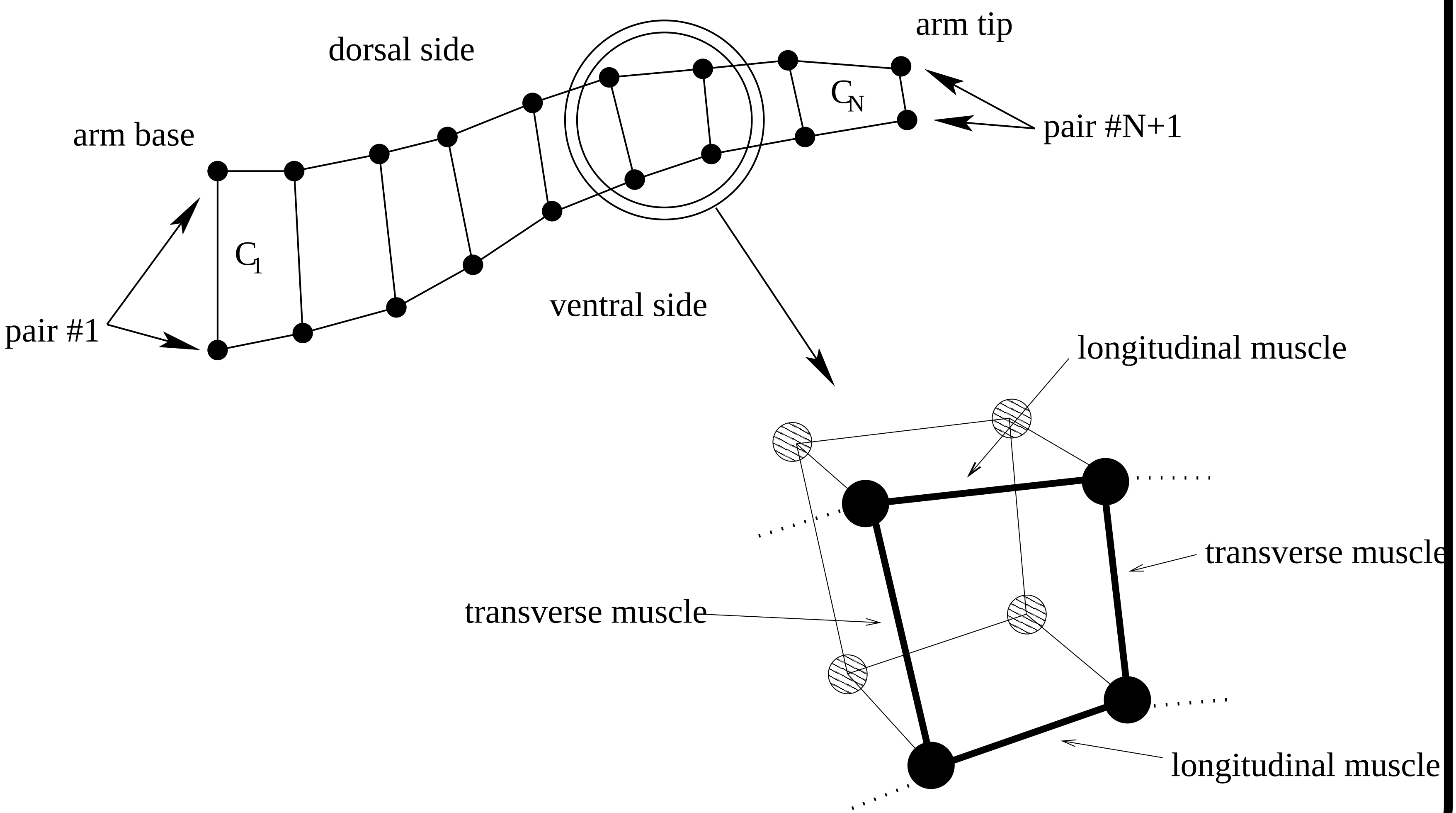
Muscle tissue is incompressible, therefore, if muscles are arranged such that different muscle groups are interleaved in perpendicular directions in the same region, contraction in one direction will result in extension in at least one of the other directions.

This is the Muscular Hydrostat mechanism

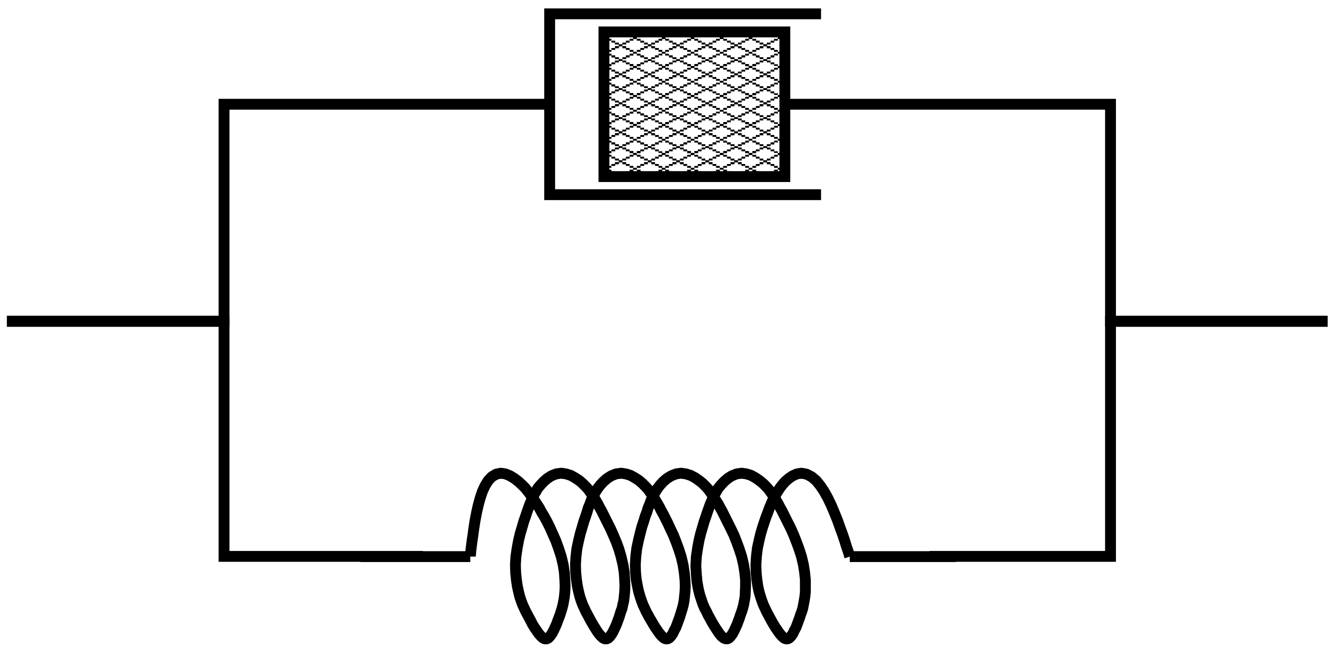
Octopus Arm Anatomy 101



Our Arm Model



THE MUSCLE MODEL



$$f(a) = (k_0 + a(k_{max} - k_0))\,(\ell - \ell_0) + \beta\frac{d\ell}{dt}$$

$$a \in [0, 1]$$

Other Forces

- Gravity
- Buoyancy
- Water drag
- Internal pressures (maintain constant compartmental volume)

Dimensionality

10 compartments $\Rightarrow$

22 point masses $\times$ $(x, y, \dot{x}, \dot{y})$

= 88 state variables

The Control Problem

Starting from a random position, bring {any part, tip} of arm into contact with a goal region, **optimally**.

Optimality criteria:

Time, energy, obstacle avoidance

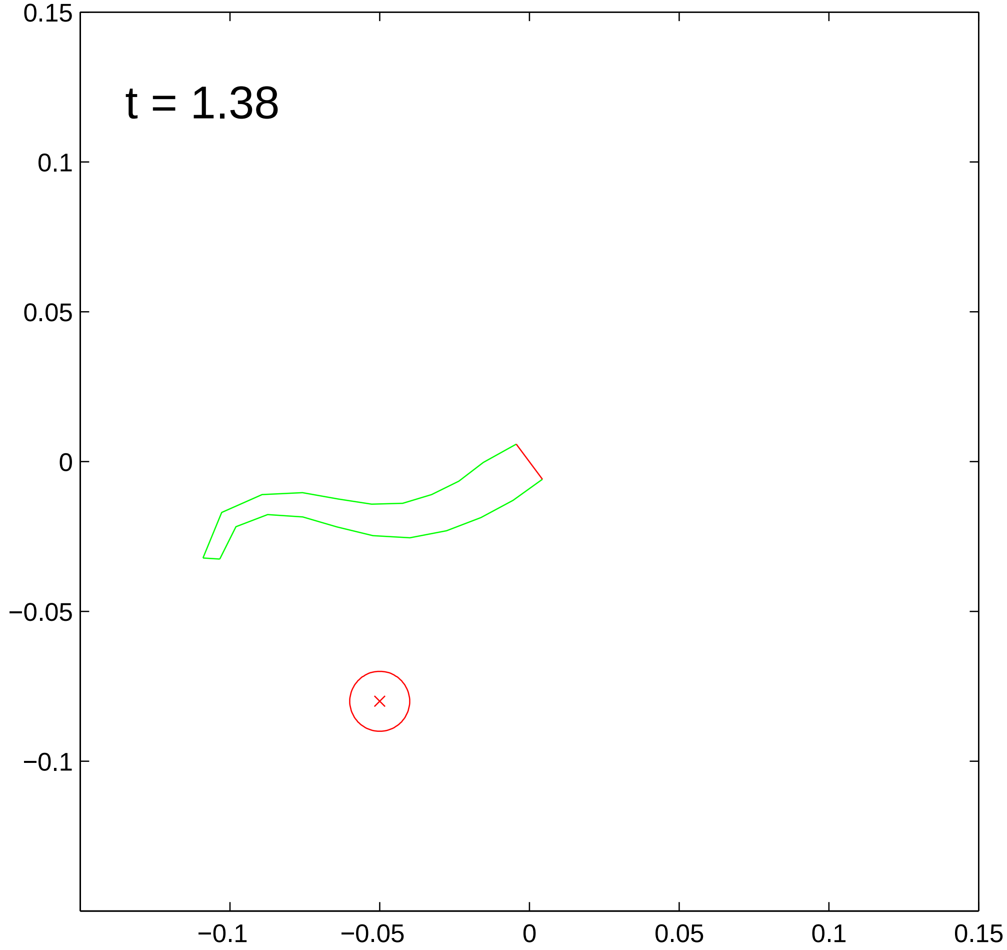
Constraint:

We only have access to sampled trajectories

Our approach:

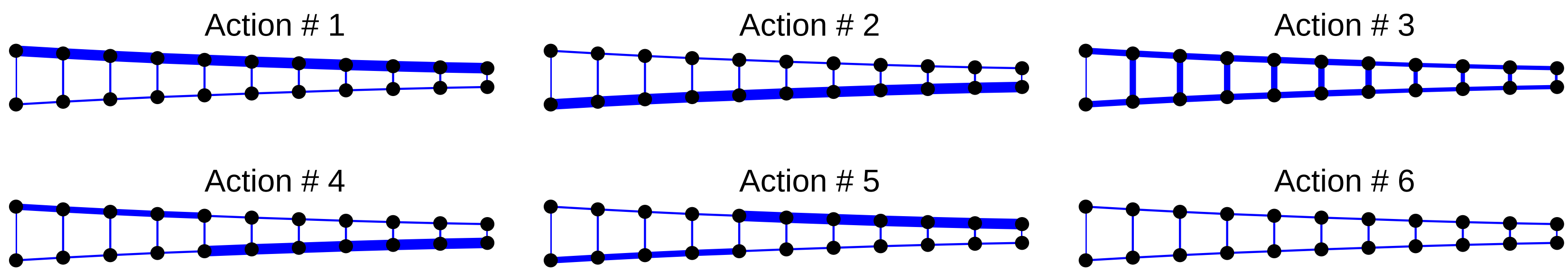
Define problem as a MDP
Apply Reinforcement Learning algorithms

The Task



Actions

Each action specifies a set of fixed activations – one for each muscle in the arm.



Base rotation adds duplicates of actions 1,2,4 and 5 with positive and negative torques applied to the base.

REWARDS

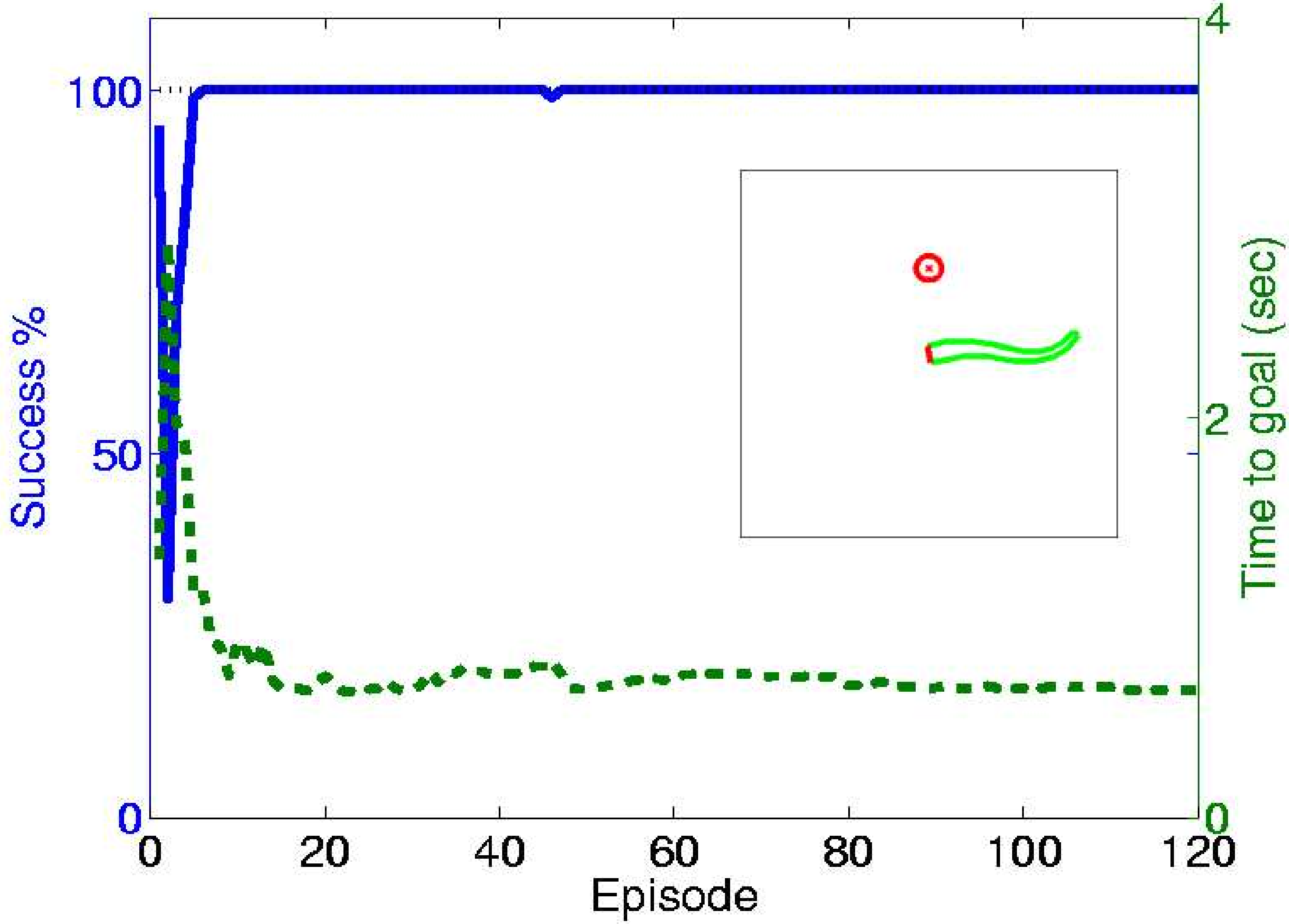
Deterministic rewards:

+10 for a goal state,
Large negative value for obstacle hitting,
-1 otherwise.

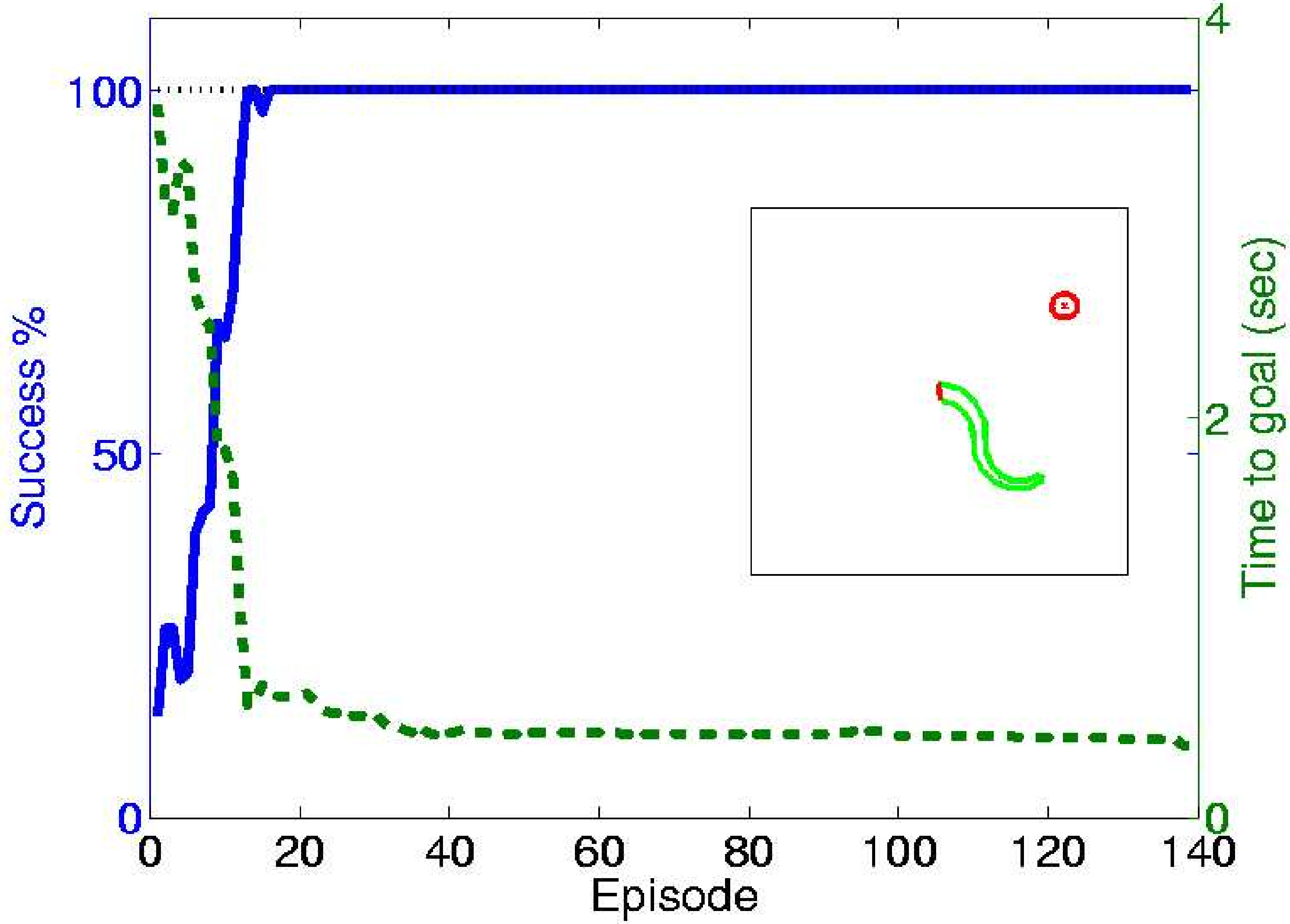
Energy economy:

A constant multiple of the energy expended by the muscles in each action interval was deducted from the reward.

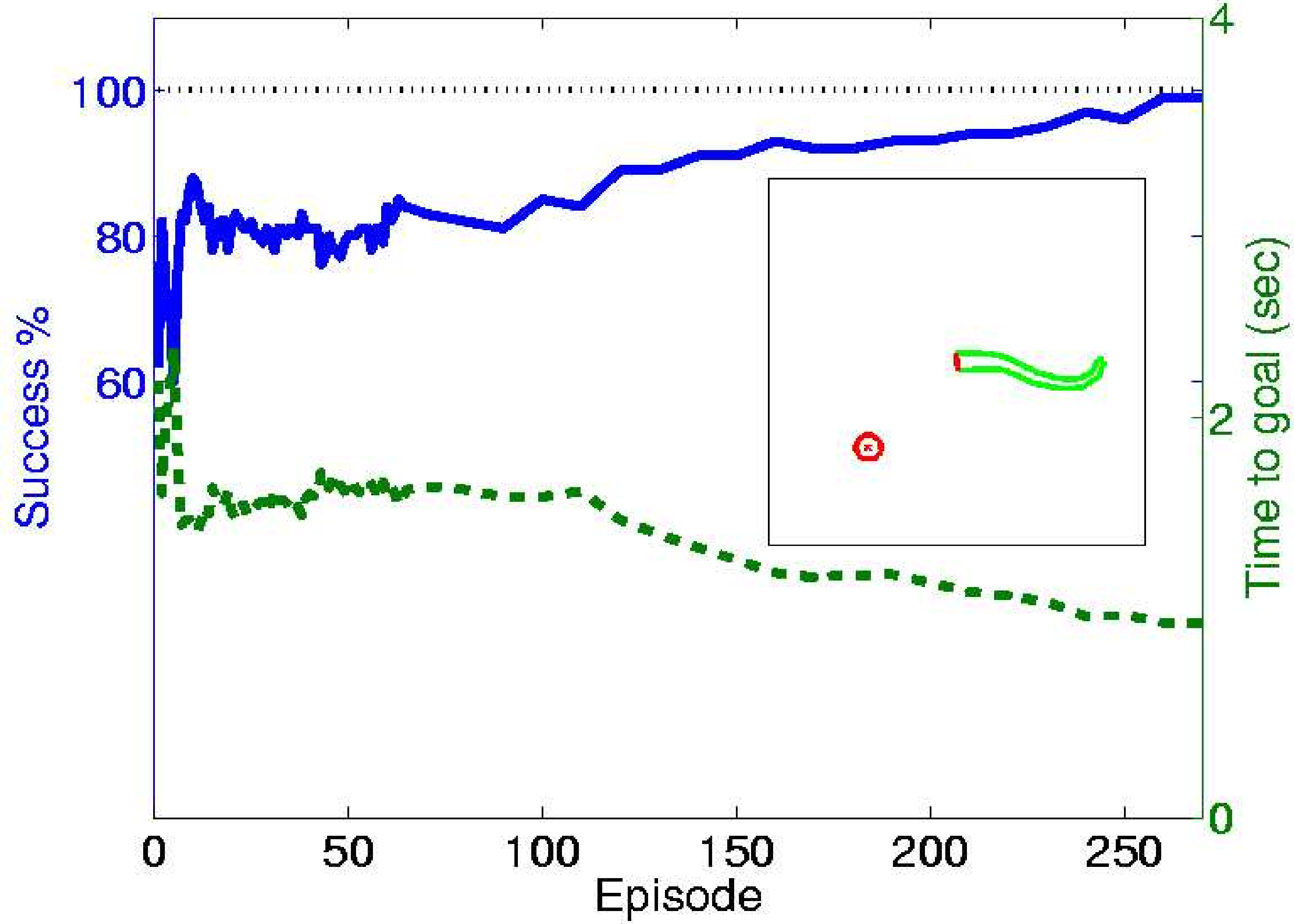
Fixed Base Task I



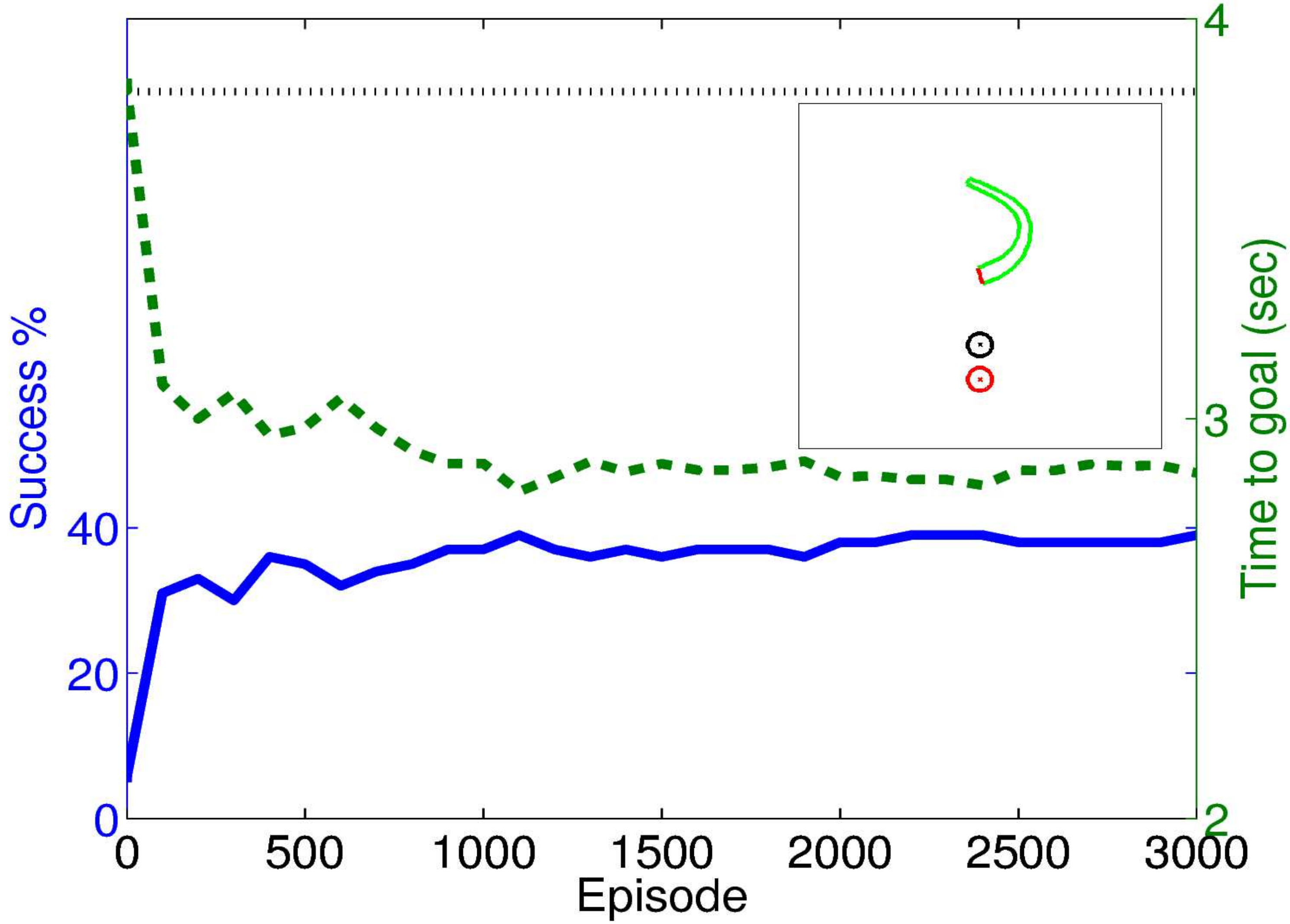
Fixed Base Task II



Rotating Base Task I



Rotating Base Task II



To Wrap Up

- There's more to GPs than regression and classification
- Online sparsification works

Challenges

- How to use value uncertainty?
- What's a disciplined way to select actions?
- What's the best noise covariance?
- More complicated tasks