Machine Learning Challenges for Truly Autonomous Robots

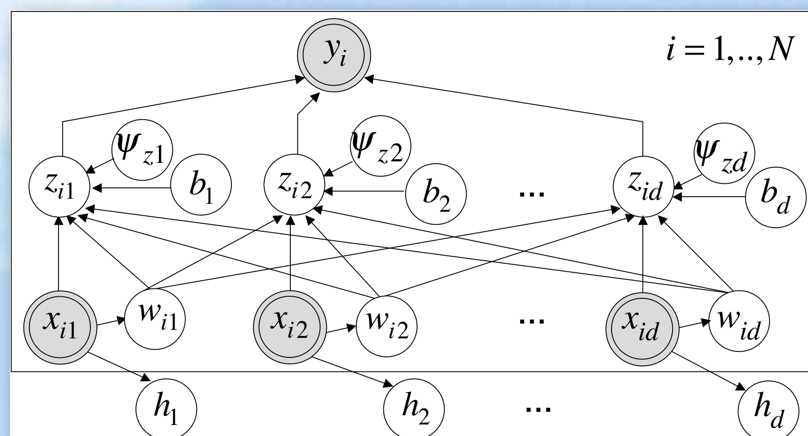
Stefan Schaal

Computer Science, Neuroscience, & Biomedical Engineering
University of Southern California, Los Angeles
&
ATR Computational Neuroscience Laboratory
Kyoto, Japan

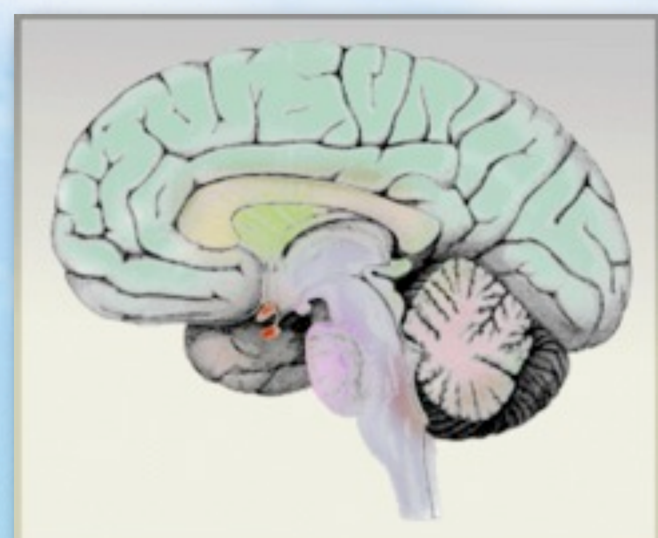
sschaal@usc.edu
http://www-clmc.usc.edu

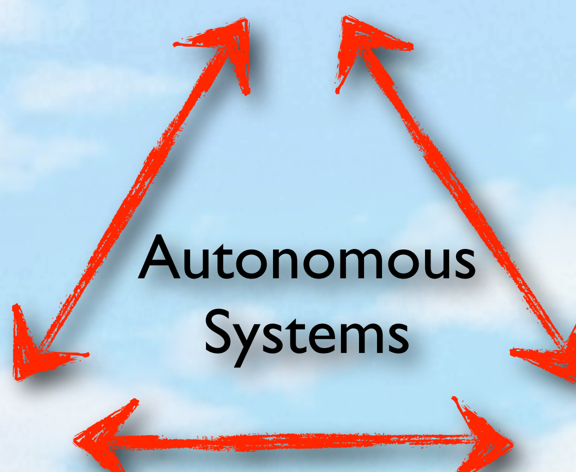
Some Grand Challenges for the Next Century:
Brains, Autonomous Robots, and Information Technology

What are the fundamental principles of autonomous learning, self-organization, self-assembly, planning?
Applications: Models, predictions, and control of systems from cells and nano-structures to robots to societies

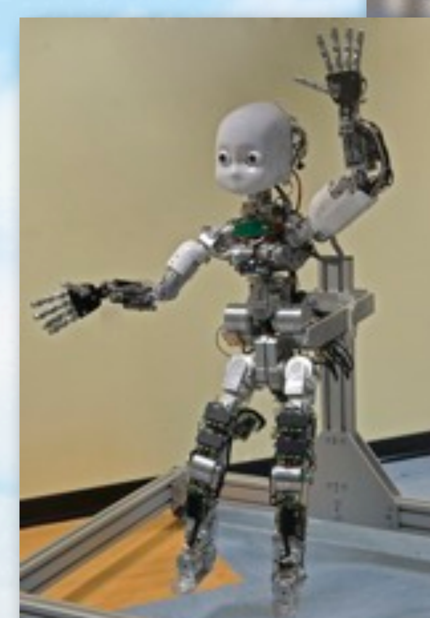


How does the brain learn and control complex motor skills?
Applications: Facilitate and personalize learning, neuro-prosthetics, brain machine interfaces, movement rehabilitation, etc.





Can we create an autonomous robot?
Applications: assistive robotics, hazardous environments, space exploration, etc.



Why Learning At All?

- Couldn't we obtain models of
  - kinematics (from CAD)
  - dynamics (from CAD and system identification)
  - the environment (3D vision, range finders, …)
  - objects (3D models)
  - etc.

  and just perform planning based on these models?
- But ...
  - kinematics and dynamics can change over time (wear and tear) and often we don't have accurate models to begin with (errors, unknown nonlinearities)
  - the environment is dynamic, stochastic, incompletely perceivable
  - new (un-modeled) situations may be encountered
  - the environment is hard to model (friction, contacts, surface properties, complex unknown dynamics)
  - the search spaces for planning become too high dimensional

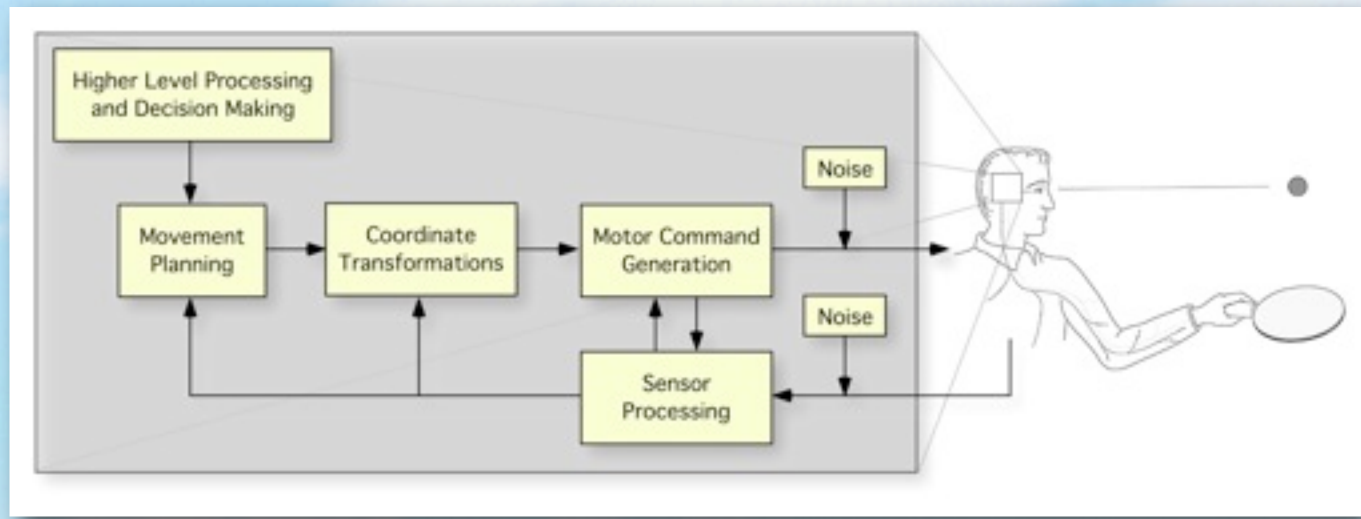
  such that learning seems to become mandatory to operate outside of laboratory environments

What Should be Learned?

- A Library of Robust Perceptuo-Motor Skills for Appropriate Environments/Objects (Affordances)
  – A motor skill is a series of movements that combine to produce a goal directed, efficient action.
    - Can be formalized as learning control policies

$$\mathbf{u}(t) = \pi_i(\mathbf{x}, t, \alpha)$$

    - Thus, at the highest level, we need to learn
      - *the policy π for every motor skill*
      - *the context when to apply it and when to abort (switch) it*
    - If the control policy is structured, subproblems may be learned in isolation, e.g.,
      - *internal models*
      - *planning modules*
      - *state estimators*
      - *etc.*



Different Classes of Tasks Require Different Methods to Compute Policies

- Tracking Tasks
  - e.g., tracing a figure-8 on a piece of paper
- Regulator Tasks
  - e.g., balance control (pole balancing, biped balancing, helicopter hover)
- Discrete Tasks
  - e.g., reach for a cup, tennis forehand, basket ball shot
- Periodic Tasks
  - e.g., legged locomotion, swimming, dancing
- Complex sequences and superposition of the above
  - e.g., assembly tasks, “empty the dishwasher”, playing tennis, almost every daily life behavior

Level of Difficulty

Different Learning Methods are Suitable for Different Tasks

- Supervised Learning
  - direct inverse model learning, forward model learning (prediction)
  - “distal teacher”
  - feedback error learning, adaptive learning controllers
- Reinforcement Learning
  - value-function based approaches
  - direct policy learning (e.g., policy gradients)
- Learning Modularizations
  - primitives, schemas, basis behaviors, units of actions, macros, options
  - parameterized policies
- Imitation Learning
  - learning a policy from observation
  - learning the task/goal intent from observation (inverse RL)
  - learning an initial strategy for subsequent self-improvement
- Dimensionality Reduction, Feature Extraction
  - task relevant variables (in contrast to pure data compression)

Past to Present

Machine Learning is going to be the dominant way to "program" robots

What Can We Already Do Well (?) With Machine Learning?

- Learning internal models
  - dynamics models, kinematics models
  - rapid learning with locally linear models
  - Gaussian Processes
- Imitation learning
  - learning movement primitives
  - learning cost functions
- Learning task controllers
  - learning with task models
  - learning operational space controllers
- Reinforcement Learning and Optimal Control
  - value function-based methods
  - trajectory-based methods start scaling into very high dimensional systems
  - policy gradients
  - probabilistic reinforcement learning (reward-weighted regression, path integrals, KL-divergence)
- State Estimation
  - SLAM
  - “probabilistic robotics”
- Planning
  - Learning with Markov Decision Processes
  - Search techniques (e.g., DP, A*, RRT, PRMs, etc.)

What Can We Already Do? Learning Internal Models

- Characteristics
  - Incremental Learning
    - large amounts of data
    - continual learning
    - to be approximated functions of growing and unknown complexity
  - Fast Learning
    - data efficient
    - computationally efficient
    - real-time
  - Robust Learning
    - minimal interference
    - hundreds of inputs
    - redundant inputs
    - irrelevant inputs

- Potential Approaches
  - Classical Neural Networks
    - too slow, too much manual tweaking
  - Mixture Models
    - easy to work with
    - too many local minima
    - tough to select the correct number of models
  - Locally Weighted Learning
    - very computationally efficient in real-time
    - problem of how to select kernel size/shape not solved yet properly
  - Kernel Methods (SVM, GP)
    - excellent out-of-the box performance
    - computationally very expensive and hard to scale to many data points (and incremental learning)

Learning Internal Models: Why local linear models may still be useful

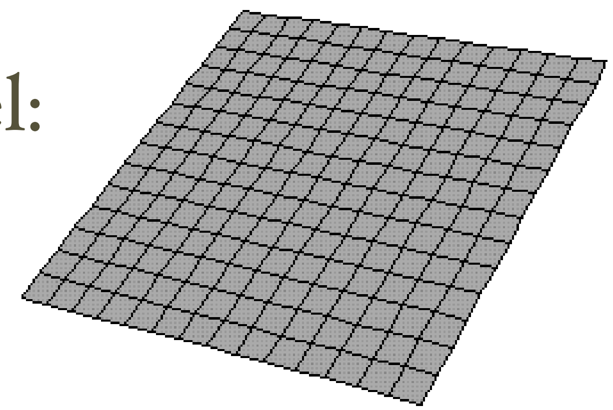
- Expl
  dyna
  – E.g
    - le
    - au
    - N
      i.e
  – E.g
    - ca

$_q)^T \mathbf{D}(\mathbf{q}-\mathbf{c}_q))$

$\rightarrow \dot{\mathbf{x}}$

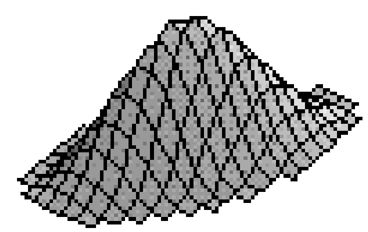
$(\mathbf{q}) = \boldsymbol{\tau}$

Linear Model:



$$\mathbf{y} = \beta_x{}^T\mathbf{x} + \beta_0 = \beta^T\tilde{\mathbf{x}} \quad \text{where} \quad \tilde{\mathbf{x}} = \left[\mathbf{x}^T \ 1\right]^T$$

Weighting Kernel:



$$w = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c})^T \mathbf{D}(\mathbf{x}-\mathbf{c})\right) \quad \text{where} \quad \mathbf{D} = \mathbf{M}^T\mathbf{M}$$

- again, allowing us to exploit the structure of equations for better generalization

Learning Task Controllers: The Bigger Picture: Learning Procedure

- First, learn differential forward kinematics in a piecewise linear way

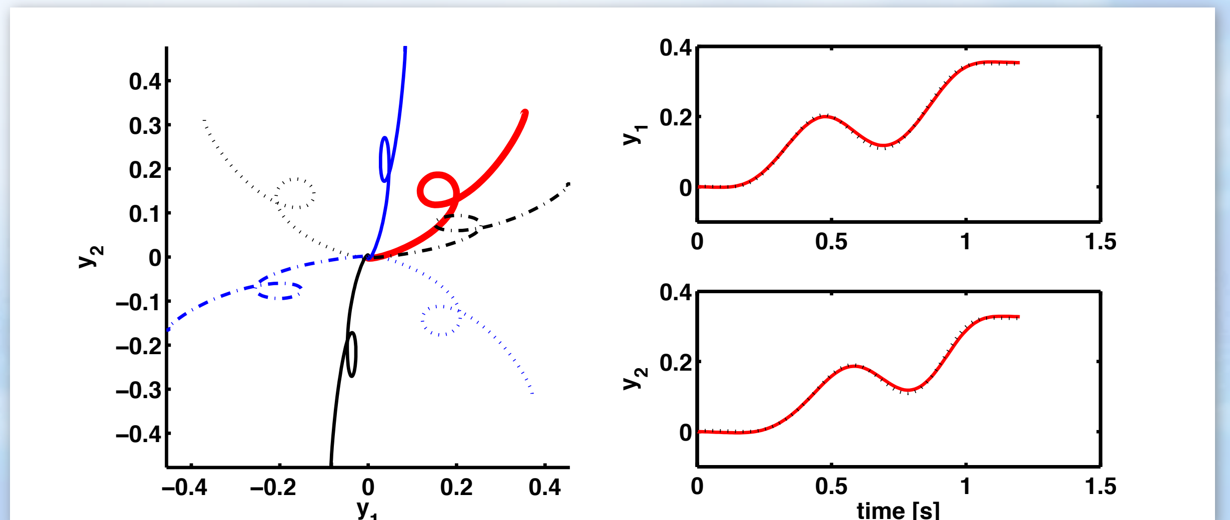
$$\ddot{\mathbf{x}} = \mathbf{A}\left[\begin{array}{c}\dot{\mathbf{q}}\\ \ddot{\mathbf{q}}\end{array}\right]\Bigg|_{\mathbf{q}_0}$$

  - Importantly, the learning algorithms determines a local region (modeled by a kernel) where the linearization is valid
- Second, use the kernels from the forward kinematics to learn a local inverse controller with reward weighted regression
  - This is just straight-forward weighted linear regression
- NOTE: After the forward model is known, controllers can be learned VERY fast for all new control situations, e.g., joint-space inv.dyn, inv. kinematics, stochastic inv. control

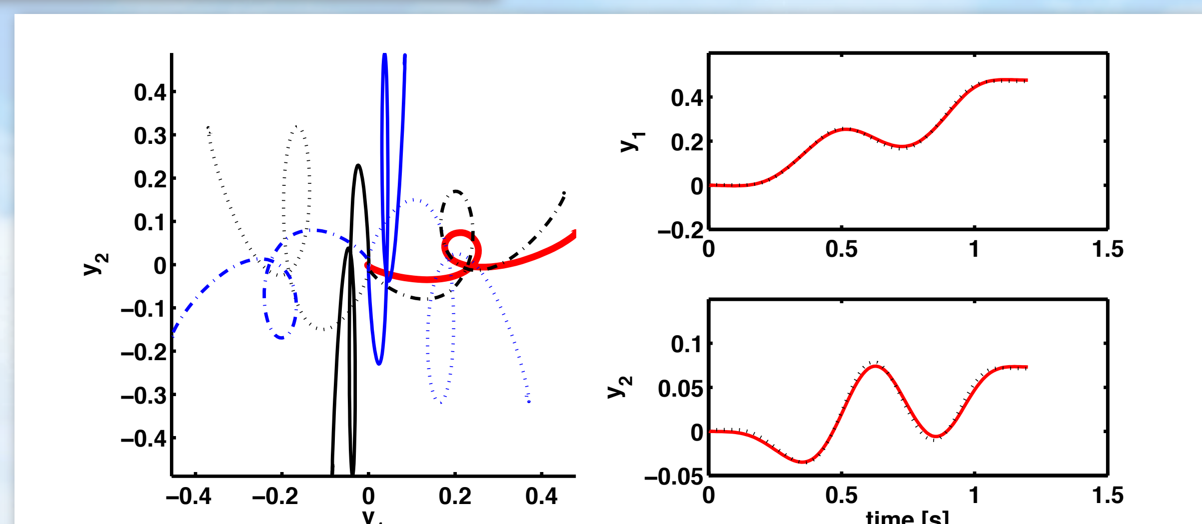
What Can We Already Do? Imitation Learning

- State-of-the-Art
  - Many approaches exist to exploit imitation
    - motor primitive-based methods
    - some work in search/planning which exploits distributions from demonstrations
- Key Open Issues
  - generalization
  - on-line modulation
  - libraries of re-usable primitives
  - perception based on primitives
  - ... otherwise, what is gained over, e.g., spline methods?

Imitation Learning: Generalization depends on the choice of Coordinates

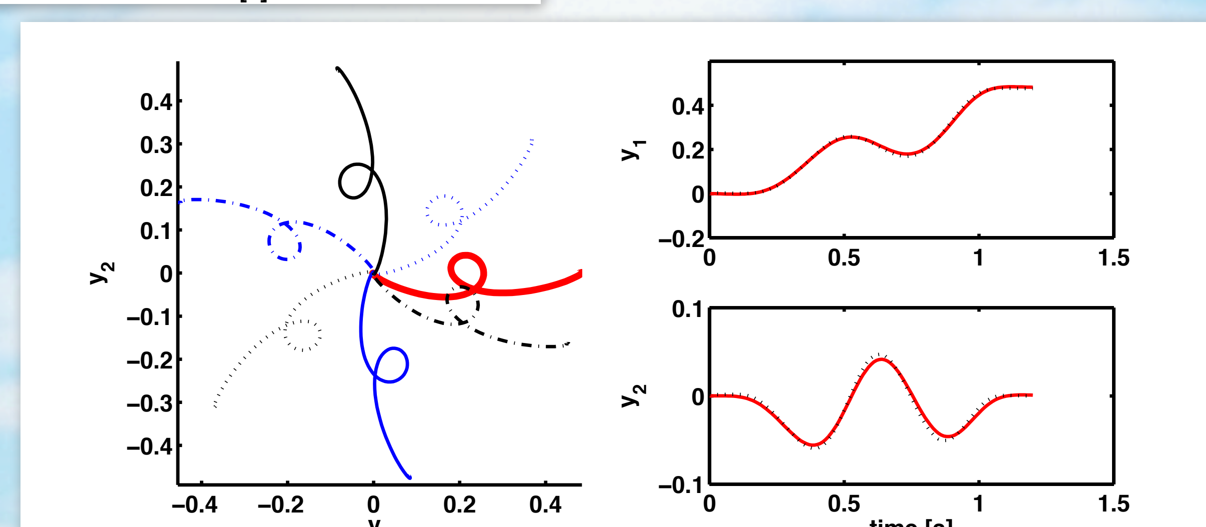


Demonstration facilitates generalization



Demonstration causes “strange” generalization

Cylindrical coordinates avoid the problem



What Can We Already Do? Reinforcement Learning

- State-of-the-Art
  - Many approaches exist to exploit imitation
    - motor primitive-based methods
    - some work in search/planning which exploits distributions from demonstrations
- Key Open Issues
  - generalization
  - on-line modulation
  - libraries of re-usable primitives
  - perception based on primitives
  - ... otherwise, what is gained over, e.g., spline methods?

What Can We Already Do?
Reinforcement Learning from Trajectories

- State-of-the-art of Reinforcement Learning from Trajectories:
  - Given the cost per trajectory τ : $$J = E_\tau \left\{ \sum_{i=0}^{T} r_i \right\}$$
  - The motor primitives with parameters θ: $$\tau \dot{\mathbf{y}} = f(\mathbf{y}, goal, \theta)$$
  - RL with Natural Gradients $$\theta^{new} = \theta^{old} + \alpha \frac{\partial J_{NAC}}{\partial \theta}$$
  - Probabilistic RL with Reward-Weighted Regression $$\theta^{new} \propto \sum_T R_\tau \theta_\tau \,/\, \sum_T R_\tau$$
  - Trajectory-based Q-learning (fitted Q-iteration)
    - an actor-critic based method based on an action-value function over trajectories
  - RL with path-integrals (a probabilistic, model-based/model-free approach derived from stochastic optimal control)

Reinforcement Learning Based on Path Integrals

- For dynamic motor primitives, a beautifully simple “black-box” algorithm results:

1) Create K trajectories of the motor primitive for a given task with noise.

2) We can write the cost to go from every time step t of the trajectory as:

$$R_t = q_T + \sum_{i=t}^{T} r_i$$

3) The probability of a trajectory becomes

$$P\left(\xi_t^k\right) = \frac{\exp\left(-\frac{1}{\lambda} R_t^k\right)}{\sum_{j=1}^{K} \exp\left(-\frac{1}{\lambda} R_t^j\right)}$$

4) Update the parameter θ of the motor primitive as

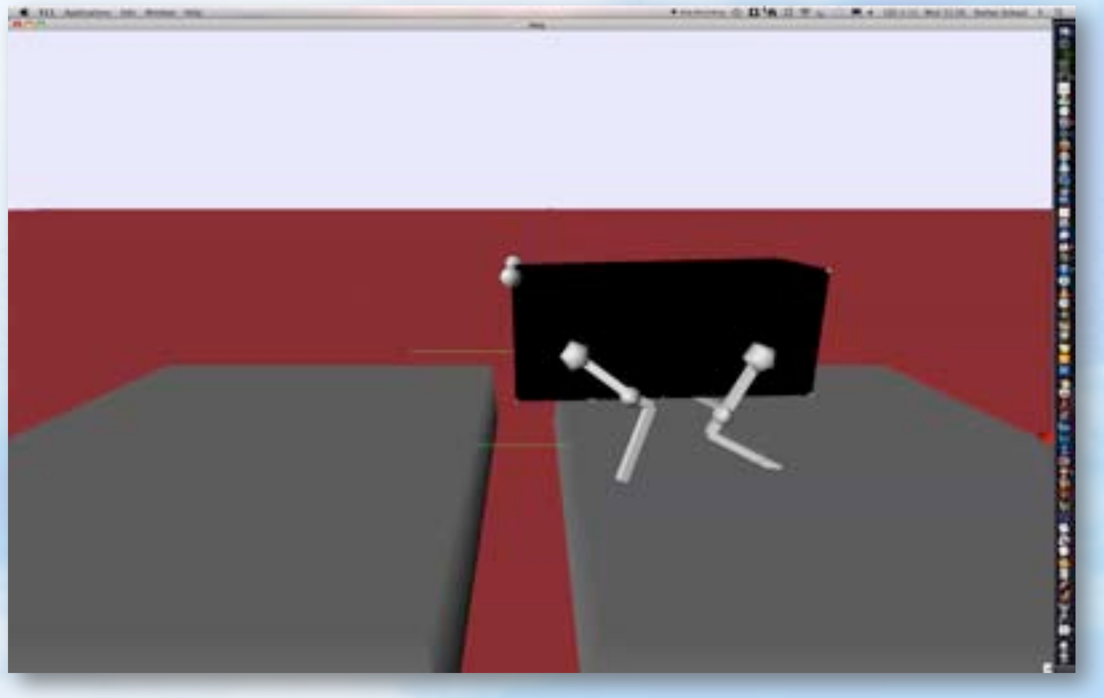
$$\Delta\theta_t = \sum_{k=1}^{K} P\left(\xi_t^k\right) \frac{\mathbf{R}^{-1}\mathbf{g}^k(\mathbf{x}_t)\mathbf{g}^k(\mathbf{x}_t)^T}{\mathbf{g}^k(\mathbf{x}_t)^T \mathbf{R}^{-1}\mathbf{g}^k(\mathbf{x}_t)} \varepsilon_t^k$$

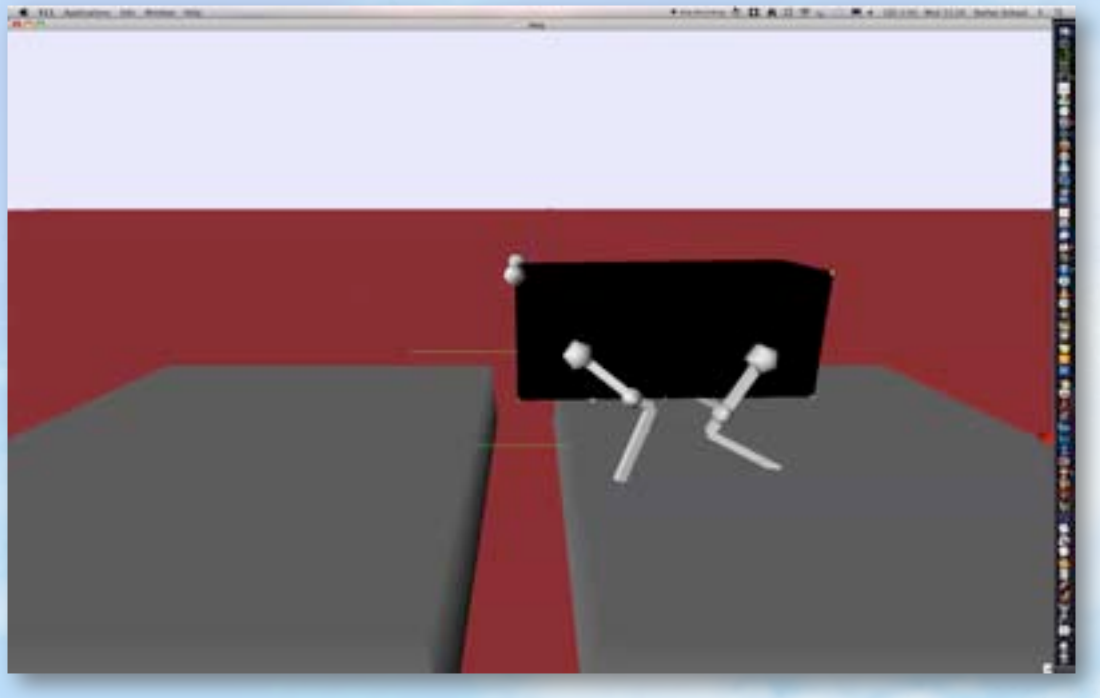
5) Final parameter update

$$\theta^{new} = \theta^{old} + \overline{\Delta\theta_t}$$

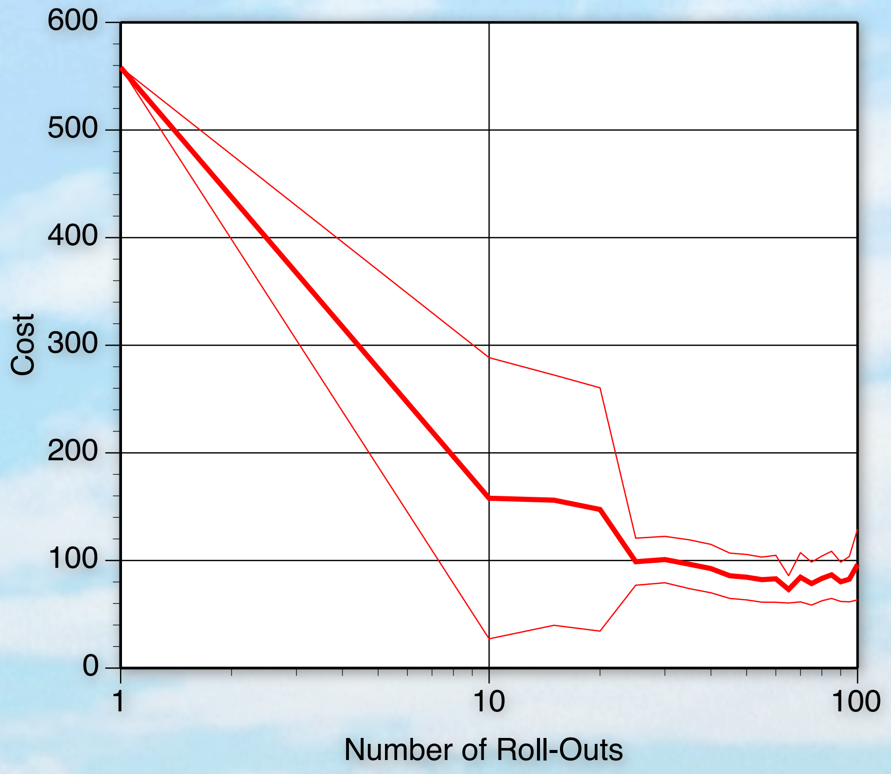
Note that there are NO open tuning parameters except for the exploration noise

Example: Learning to Jumping over a Gap





This is a 12 DOF motor system, using 50 basis functions per primitive. Learning converges after about 20-30 trial! Performance improved by 15cm (0.5 body lengths)



Reinforcement Learning of Toy Manipulation

Kober & Peters, 2008

What Can We Already Do?
Reinforcement Learning from Trajectories

- Surprisingly, reinforcement learning suddenly looks like a topic that has fairly mature and functional algorithms that can work on complex robots!
- Remaining problems:
  – Cost function design (inverse reinforcement learning)
  – Understanding the intend of observed behavior

What Can We Already Do?
State Estimation

- State-of-the-Art
  - SLAM, “Probabilistic Robotics”, have matured to very successful and well-working algorithms

What Can We Already Do? Planning

- State-of-the-Art
  - Impressive results from RRT, PRMs (see James Kuffner's talk later)
  - Optimal control and reinforcement learning algorithms have created another set of well working tools for planning

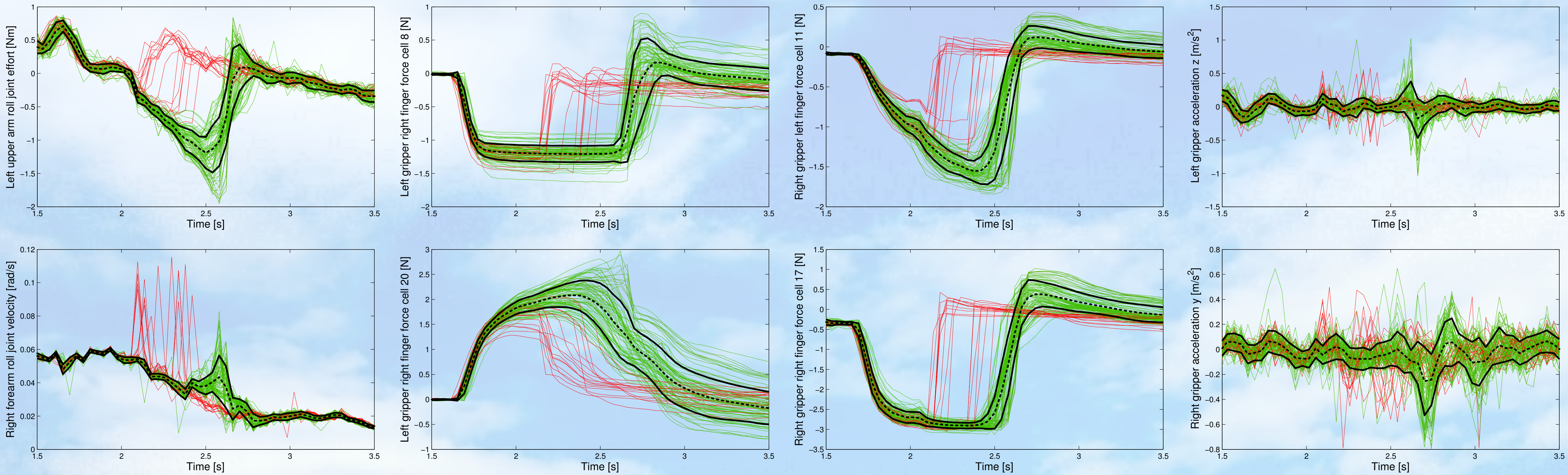
Topics Which Deserve Much More Research Attention

- Learning complex motor skills from sequencing and superimposing primitives
- Theoretically sound real-time and life-long learning
- Automatic feature extraction for task-level control
- Automatic learning of useful modularization
- Learning fine manipulation (touch, grasp)
- Learning reactive policies for stochastic and dynamic environments
- Sensor data mining for prediction and recovery
- ...
- **Learning to create complete, truly autonomous learning and control systems**

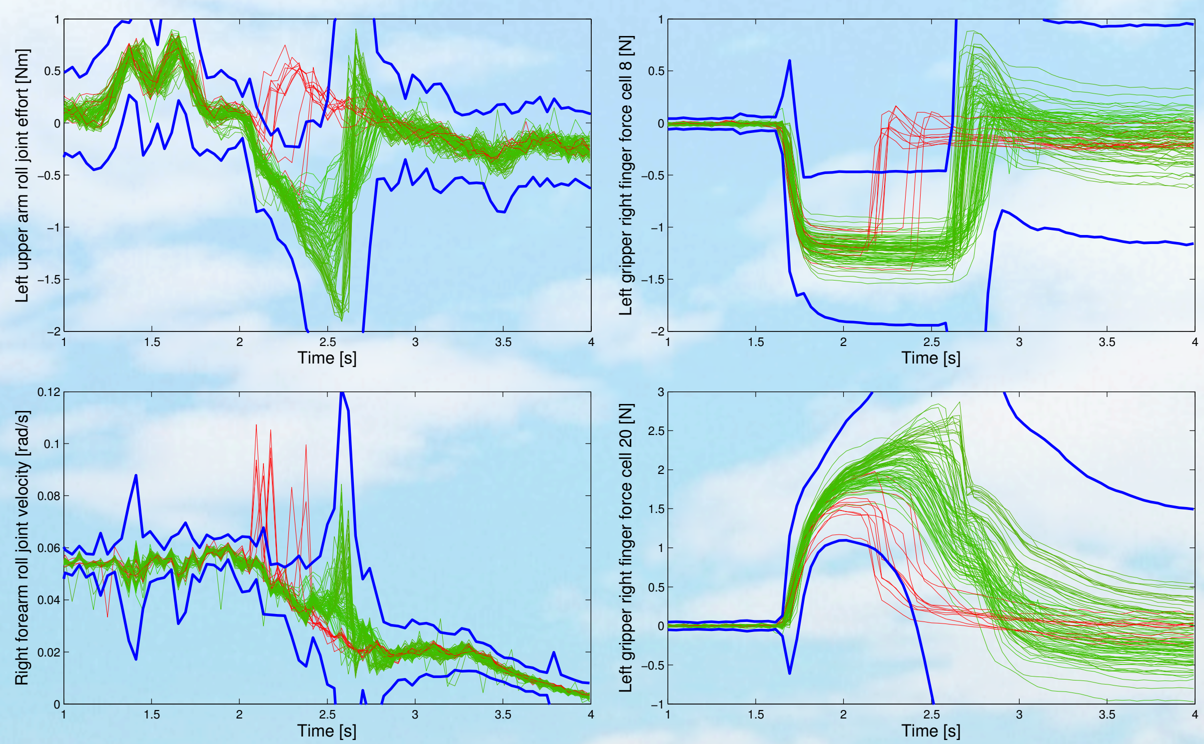
Sensor Data Mining

Peter Pastor   Mrinal Kalakrishnan   Sachin Chitta
Research conducted at Willow Garage

Sensor Data Mining: An Associative Sensor Memory

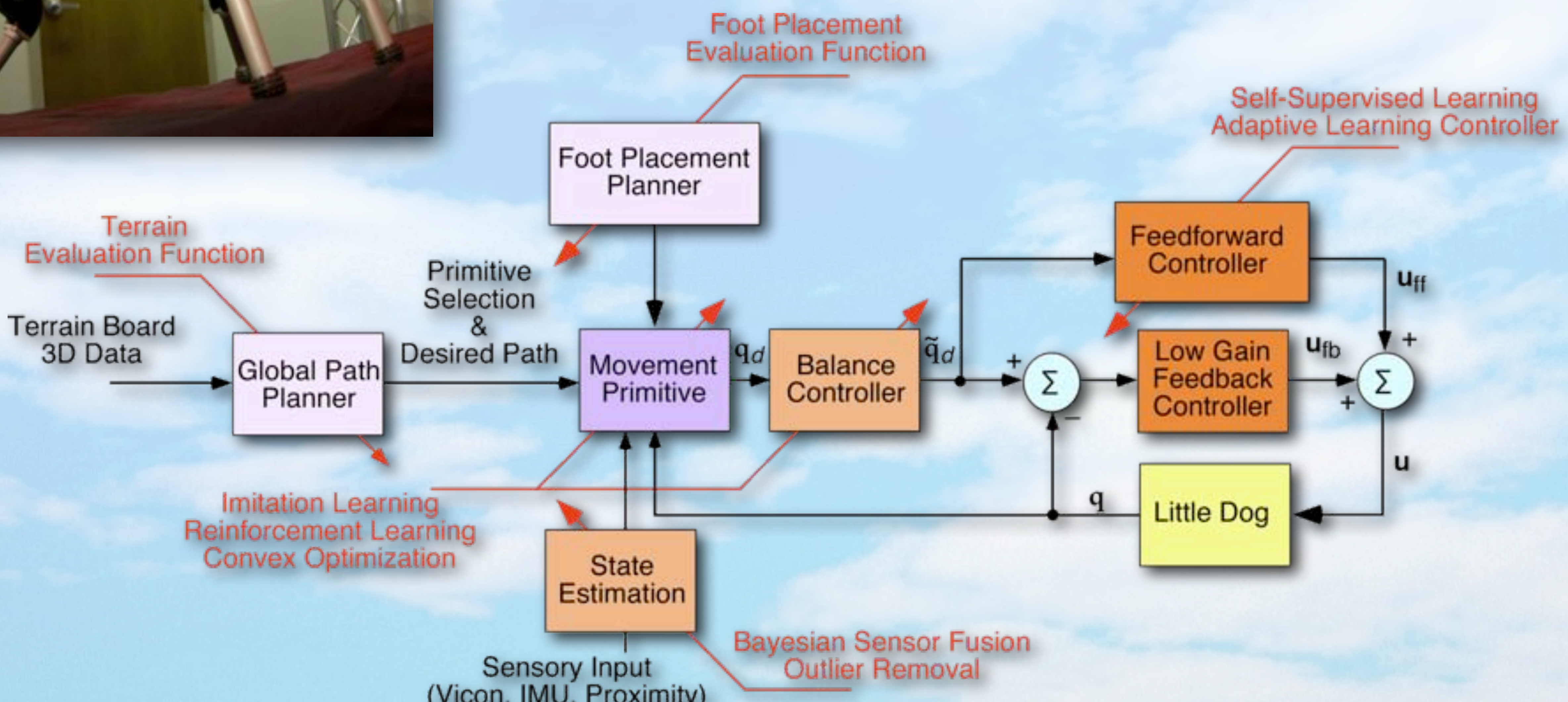


Training Data



Test Data

Example: Learning Locomotion with Little Dog



Learning Locomotion with LittleDog

http://www-clmc.usc.edu

Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal

Note: A similar video can be shown by teams of CMU, IHMC, MIT, Stanford, UPenn

Imagine:

If someone would fund Machine Learning for Robotics with $1 Billion

One Result Could Be ....

Autonomous driver included ...