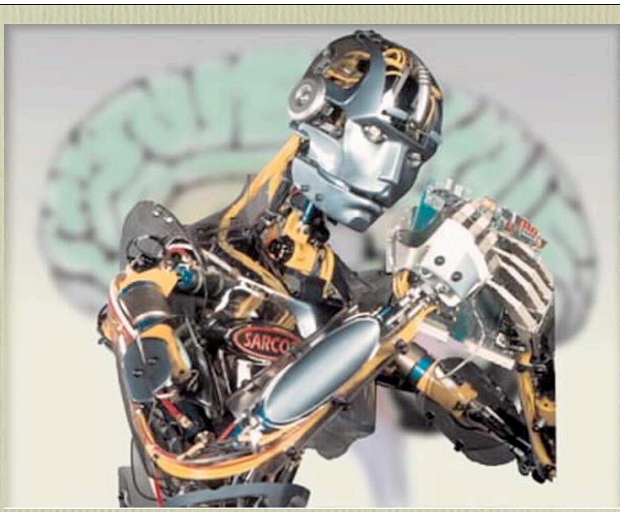
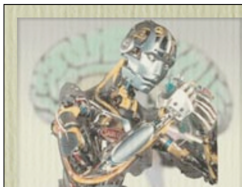


Stefan Schaal
Computer Science & Neuroscience
University of Southern California, Los Angeles
&
ATR Computational Neuroscience Laboratory
Kyoto, Japan

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

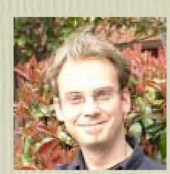
Scaling Reinforcement Learning to Complex Motor Systems

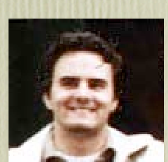
A Lecture Series at the Okinawa Computational Neuroscience Course, June 2005

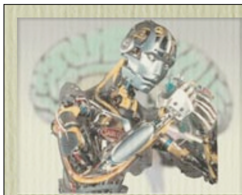


Joint Work With:

- *Auke Ijspeert*
- *Aaron D'Souza*
- *Jun Nakanishi*
- *Jan Peters*
- *Sethu Vijayakumar*
- *Dimitris Pongas*

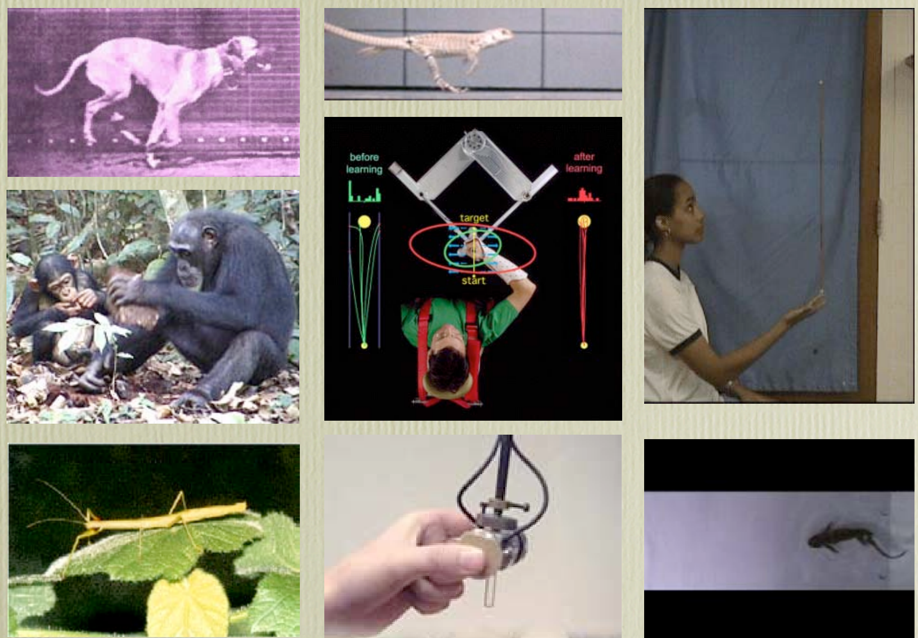


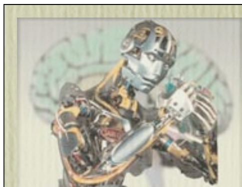





How are Motor Skills Generated?

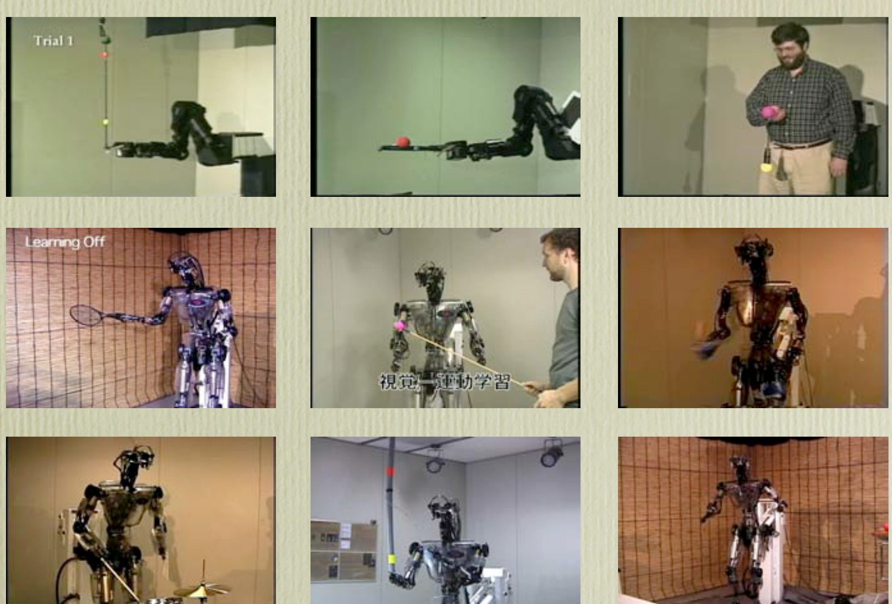
A Question Shared by Biological and Robotics Research



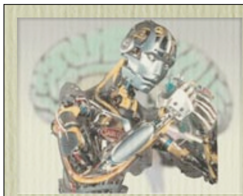


How are Motor Skills Generated?

A Question Shared by Biological and Robotics Research



Movies from collaborations with C. Akteson, S. Kotosaka, S. Vijayakumar

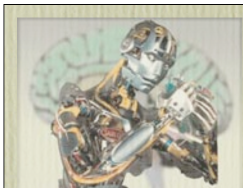


How are Motor Skills Generated?

A Question Shared by Biological and Robotics Research

Unfortunately, each of these skills required manual generation of representations control policies, and learning mechanisms

Movies from collaborations with C. Akteson, S. Kotosaka, S. Vijayakumar

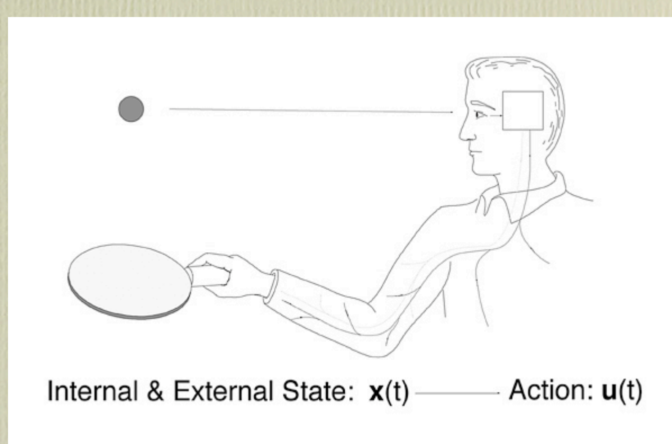


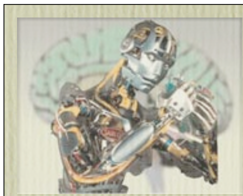
A Computational Approach to Motor Control: Control Policies

- *The General Goal of Motor Learning:*

  Control Policies

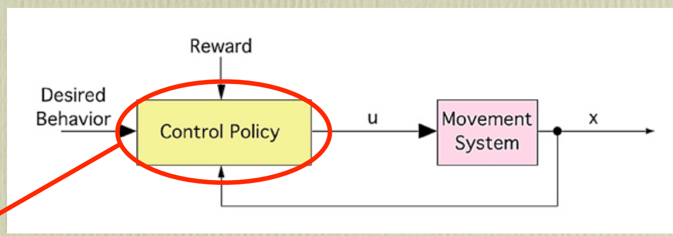
  $\mathbf{u}(t)=p(\mathbf{x}(t),t,\alpha)$





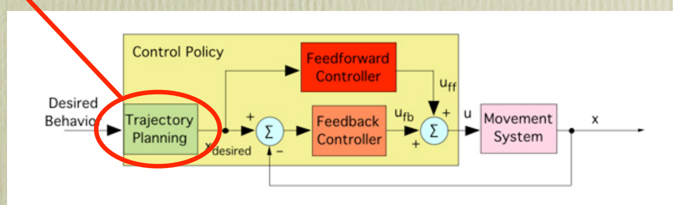
Where Do Control Policies Come From?

Direct Control (Model Free)



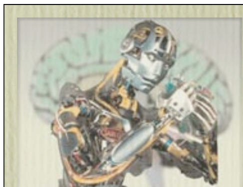
Unstructured Learning of Policies is infeasible

Indirect Control (Model-Based)



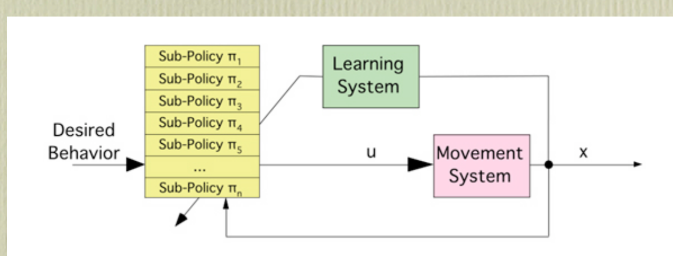
Possible learning methods include:

- Reinforcement Learning
- Dynamic Programming
- Adaptive Control
- Supervised Learning
- Evolutionary Methods



Making Motor Learning Feasible

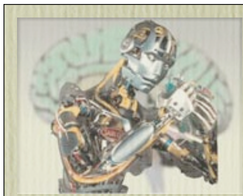
- *Modular Motor Control*



- *Parameterization*

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

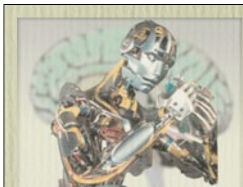
What is needed is research on
Motor Primitives



Motor Primitives

The Search for a Common Building Block in Motor Control

- *Previous Suggestions Included:*
  - Organizational Principles
    - *2/3 Power Law*
    - *Piecewise Planarity*
    - *Speed-Accuracy Tradeoff*
    - *Optimization of Energy, Jerk, Torque Change, Motor Command Change, Task Variance, Stochastic Feedback Control, Effort, etc.*
  - Equilibrium Point/Trajectory Hypotheses
  - Force Fields
  - Pattern Generators and Dynamics System Theory
    - *Focusing mostly on coupling phenomena (e.g.. inter-limb, perception-action, intra-limb) and the necessary interaction of control and musculoskeletal dynamics*
  - Contraction Theory
    - *A version of control theory for modular control*
  - ... and many more



(At Least) Two Different Approaches to Motor Primitives

The Self-organizing View

- *Regularities of motor coordination are the "emergent" results of (well-tuned) self-organizing dynamic systems*

Examples:

- *Interlimb coordination*
- *Locomotion*
- *Perception-Action coupling*
- *etc.*

The Optimizing View

- *Regularities of motor coordination are the results of explicit or implicit learning/optimization processes based on inherent organizational criteria*

Examples:

- *Visually guided reaching*
- *Eye movement*
- *Motor learning*
- *etc.*

Optimization
vs.
Self-organization

The Self-organizing View

Pros:
- *Independent of initial conditions (generalization)*
- *Inherent stability due to attractor dynamics*
- *Coupling with external signals is relatively straightforward*
- *etc.*

Cons:
- *Hard to analyze*
- *Hard to design in general*
- *Hard to apply learning*
- *etc.*

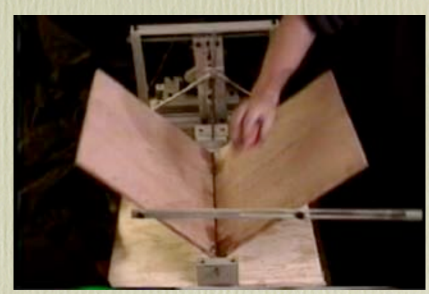
The Optimizing View

Pros:
- *Learning is relatively easy*
- *Potential existence of general optimization criteria*
- *Established numerical tools to perform optimization*

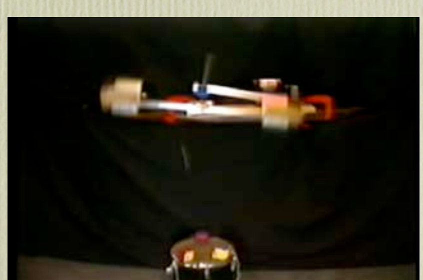
Cons:
- *Optimization is time consuming and often complex*
- *Dependence of initial conditions*
- *Often explicit time dependence*
- *Problems with generalization*
- *How to express complex tasks*
- *etc.*

11

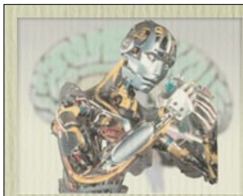
Investigating Principles of Self-organization



Start with simple systems ...

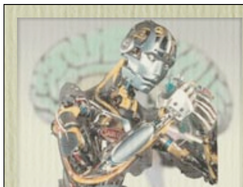


12

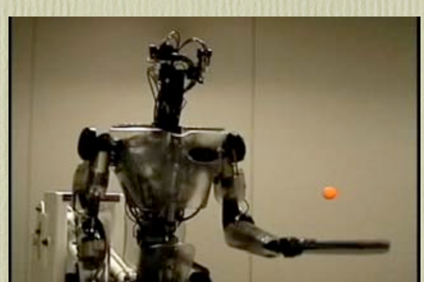


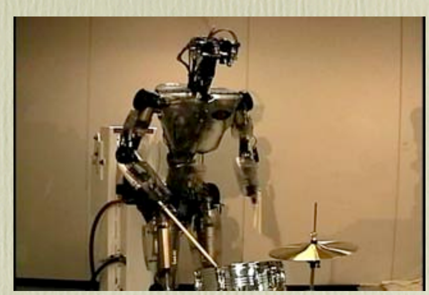
Investigating Principles of Self-organization

…move on to more complex systems …

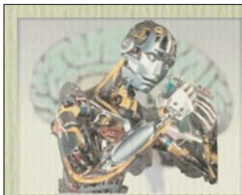


Investigating Principles of Self-organization



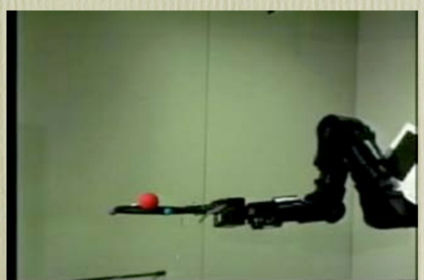


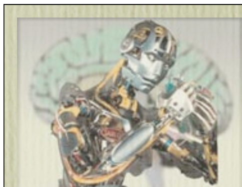
…even more complex systems …



Investigating Principles of Self-organization

...and add additional feedback control.

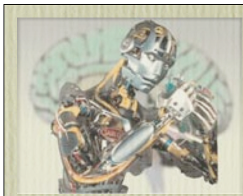




Investigating Principles of Self-organization

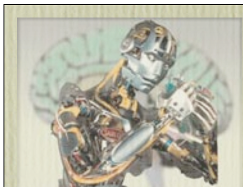
- However, the amount of human insight and manual tuning remained very significant in all these examples.

- General principles of designing and adjusting dynamic systems were missing.



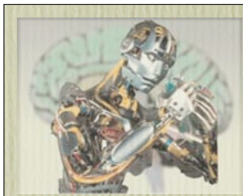
Goals of this Lecture Series

- *Introduce Dynamic Motor Primitives (DMPs)*
- *Discuss Some Evidence of DMPs in Behavioral Science and Neuroscience*
- *Introduce the Formal Framework of DMPs*
- *Imitation Learning with DMPs*
- *Movement Recognition with DMPs*
- *Reinforcement Learning with DMPs*



Outline

- *Part I: Dynamic Movement Primitives as a Computational Model for Human Movement?*
  - Some behavioral and fMRI data ...
- *Part II: The Formal Framework of Dynamic Motor Primitives*
  - Algorithms, imitation learning, and movement recognition
- *Part III: Reinforcement Learning with DMPs*
  - Optimization, skill learning, and other applications

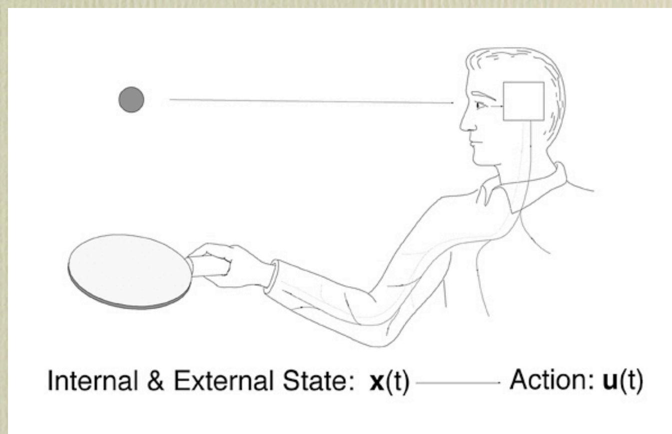


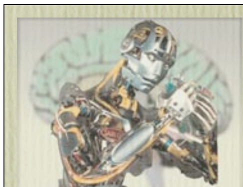
Reminder: Control Policies

- *The General Goal of Motor Learning:*

  Control Policies

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





Parameterized Dynamic Movement Primitives

- *Note the similarity between a generic control policy*

$$\mathbf{u}(t) = p\left(\mathbf{x}(t), t, \alpha\right)$$

and nonlinear differential equations

$$\mathbf{u}(t) = \dot{\mathbf{x}}_{desired}(t) = p\left(\mathbf{x}_{desired}(t), goal, \alpha\right)$$

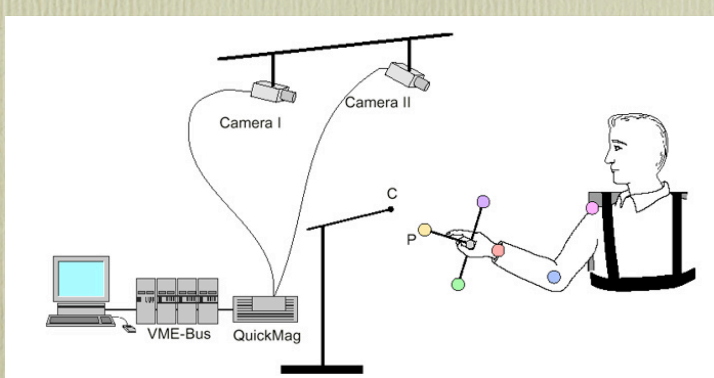
From the "Self-Organizing View", this creates a natural distinction between two major movement classes:

- **Rhythmic Movement**
- **Discrete Movement**

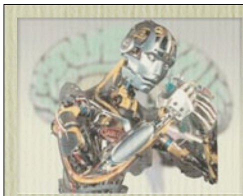
Are Rhythmic and Discrete Movement Different Primitives?

- *Two major hypotheses in the behavioral and neurophysiological literature:*
  - All Movements are Discrete (i.e., stroke-based)
    - *Morasso, Lacquaniti, Terzuolo, Soechting, Viviani, Equilibrium-Point supporters, Optimization-supporters (via-points), and many others (1981-today)*
    - *Derived from monkey and human experiments*
  - All Movement are Oscillatory Dynamical Systems (i.e., complete and incomplete limit cycles)
    - *Dynamic System's Approach to Motor Control (Turvey, Kelso, Kugler, Schöner, et.c, 1978-today)*
    - *CPG Research (Grillner, Selverston, Cohen, etc., 1970-today)*
    - *Derived from invertebrate, lower vertebrate, and human (infant) experiments*

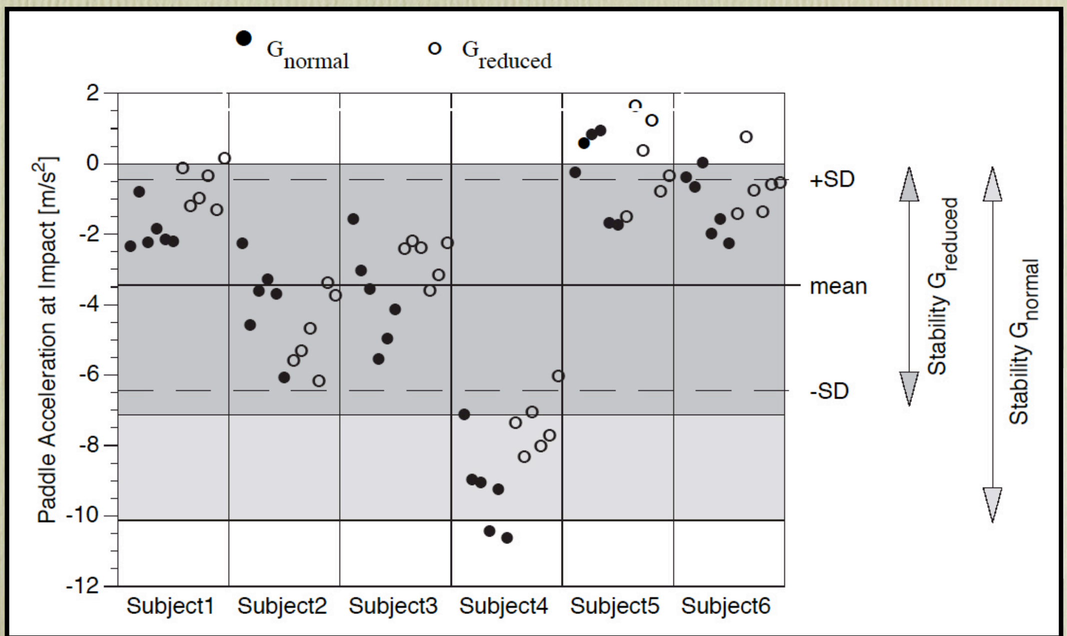
Behavioral Experiments with Humans and Robots

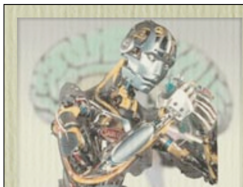


Are all arm movements stroke-based, i.e., discrete?



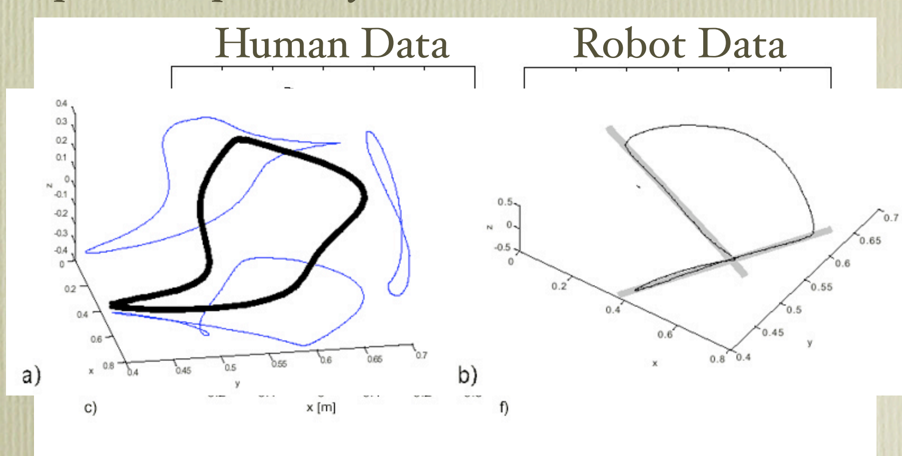
Ball-Bouncing Seems to Be a Coupled Oscillator System



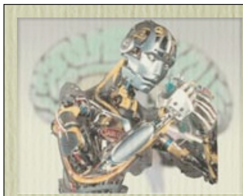


The Phenomenon of Piecewise Planarity is an Artifact

- *Human hand movements in rhythmic movement exhibit piecewise planarity*



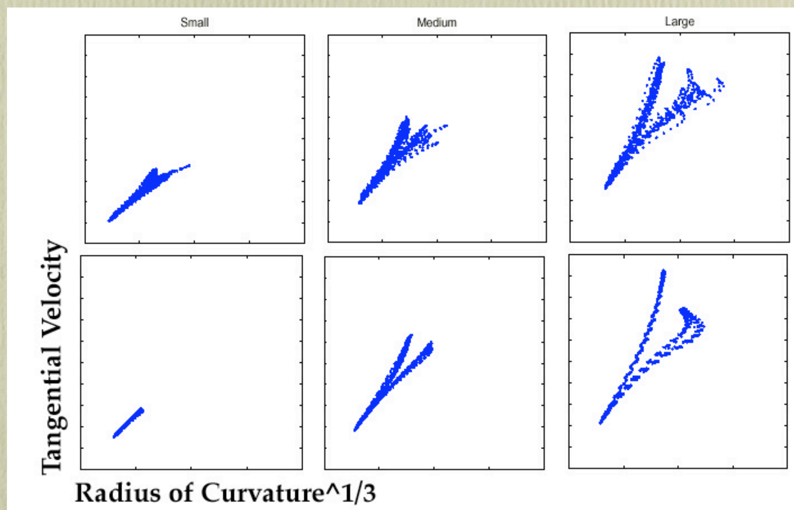
Piecewise planarity can be explained by joint space rhythmic pattern generators



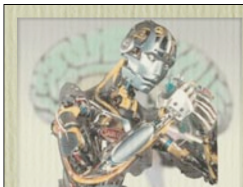
Segmentation Based on the 2/3 Power Law is an Artifact

Human Data

Robot Data



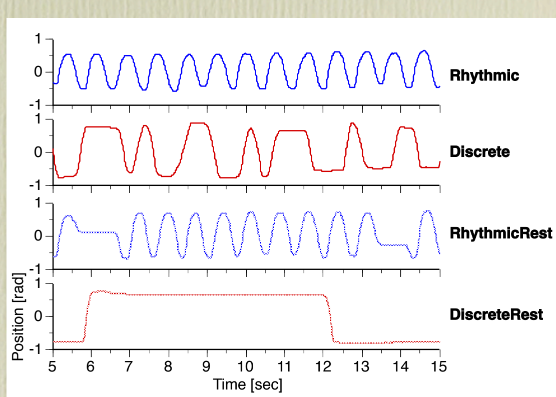
Movement segmentation based on the 2/3 Power Law can be explain by joint space rhythmic pattern generators

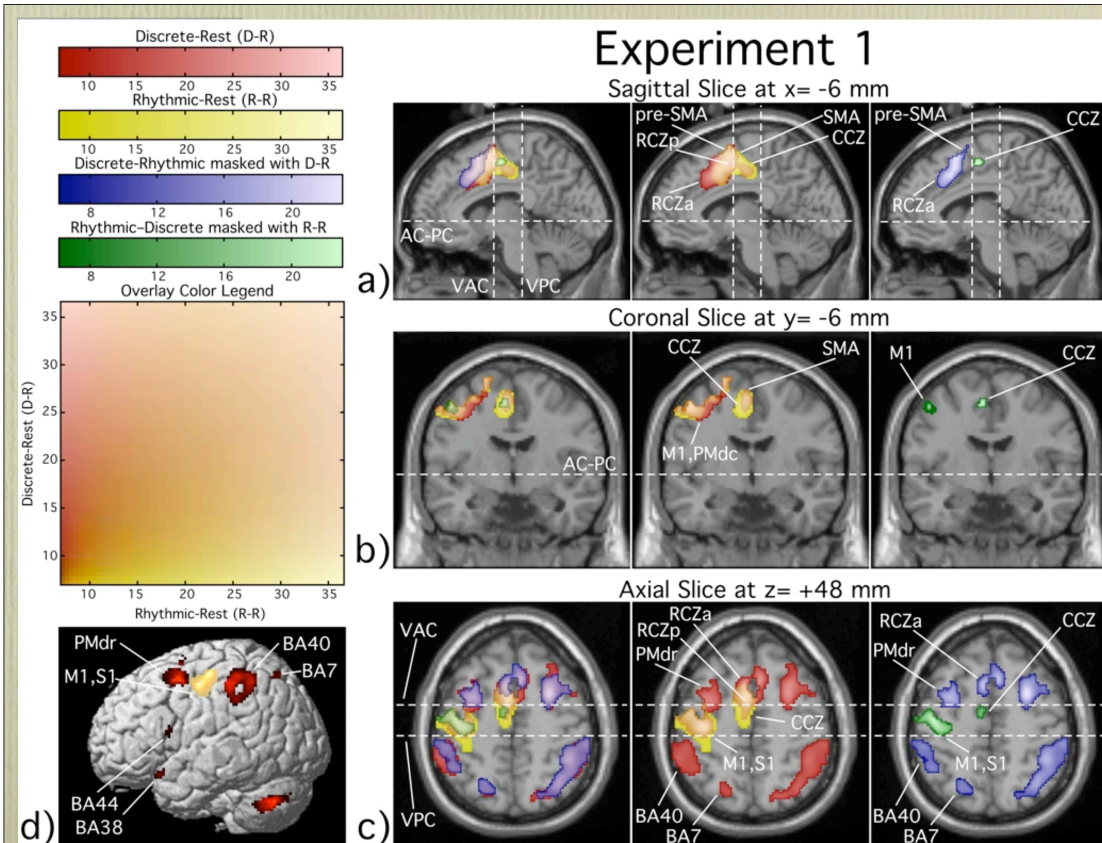


Rhythmic and Discrete Movement in fMRI

Subject in fMRI Scanner

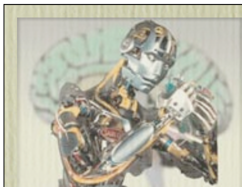
Typical Wrist Trajectories

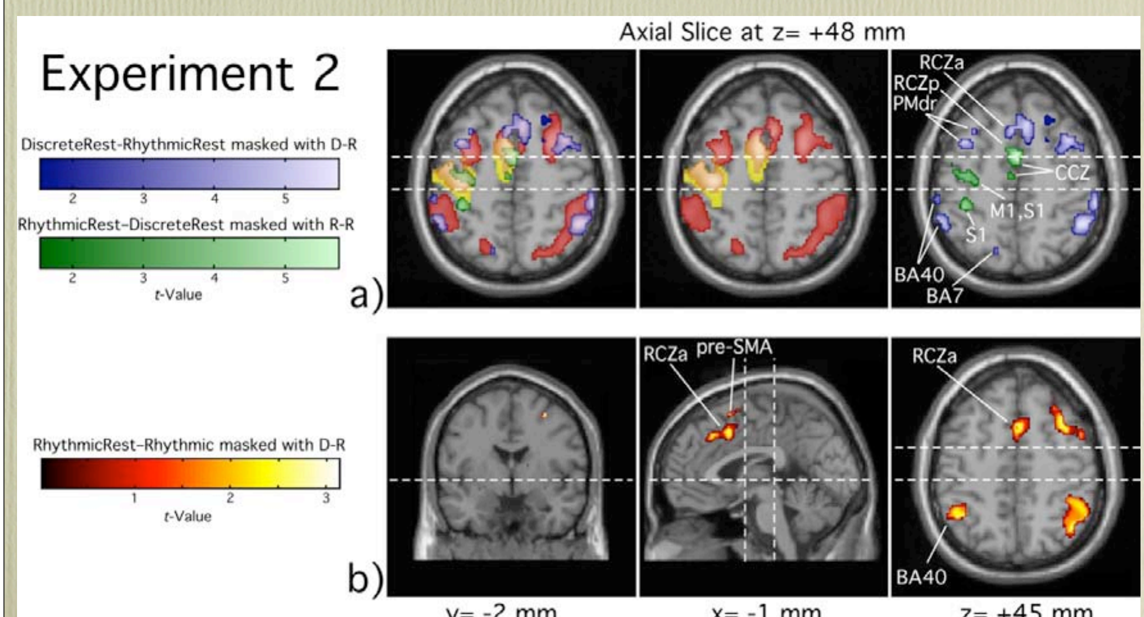




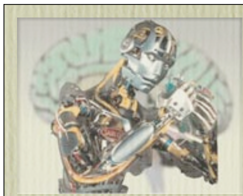
27

Rhythmic Movement is not Discrete!

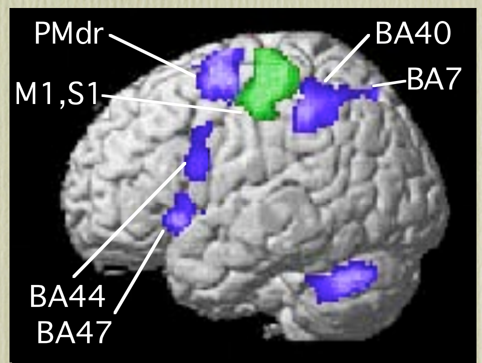




28



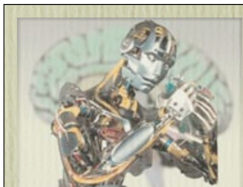
Rhythmic Movement is not Discrete!



Summary Results

DISCRETE-RHYTHMIC
RHYTHMIC-DISCRETE

29

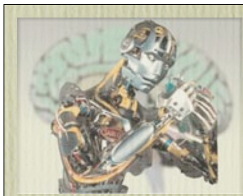


Summary of Behavioral Results

- *Rhythmic Movement is not generated from discrete strokes*
- *Rhythmic and Discrete Movements seem to be two different motor primitives in human motor control*
- *The hypothesis of Dynamic Movement Primitives may be viable*

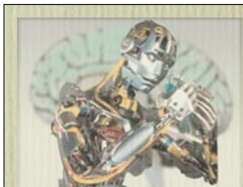
...but how would one model motor control with dynamic systems?

30



Outline

- *Part I: Dynamic Movement Primitives as a Computational Model for Human Movement?*
  - Some behavioral and fMRI data ...
- *Part II: The Formal Framework of Dynamic Motor Primitives*
  - Algorithms, imitation learning, and movement recognition
- *Part III: Reinforcement Learning with DMPs*
  - Optimization, skill learning, and other applications



How to Model Motor Primitives?

Some Inspiration from Hodkin-Huxley Models

Total current flow :

$$I_m = C_m \dot{V} + I_{Na} + I_K + I_{leak}$$

where

$$I_{Na} = (V - V_{Na})\overline{g}_{Na} m^3 h$$

$$I_K = (V - V_k)\overline{g}_K n^4$$

$$I_{leak} = (V - V_{leak}) g_{leak}$$

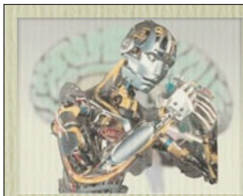
Channel Dynamics :

$$\dot{m} = \alpha_m (1 - m) - \beta_m m$$

$$\dot{h} = \alpha_h (1 - h) - \beta_h h$$

$$\dot{n} = \alpha_n (1 - n) - \beta_n n$$

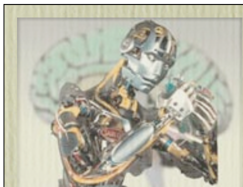
Can one build (or learn) motor primitives from similar equations?



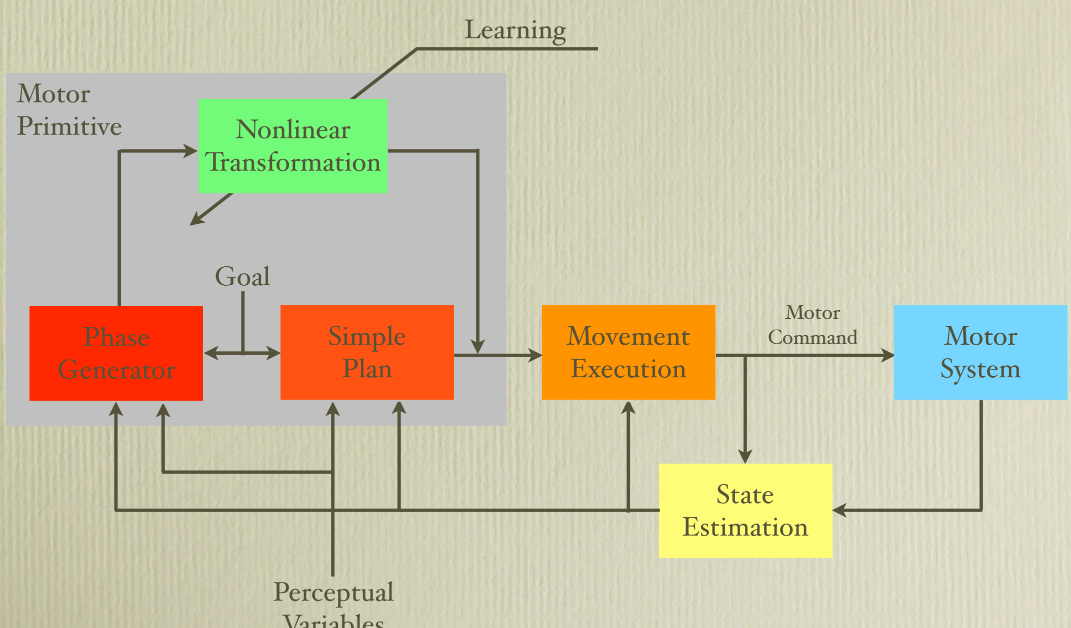
Computational Goals

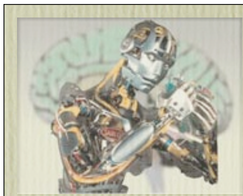
$$\dot{\mathbf{x}} = f\left(\mathbf{x}, goal\right)$$

- *A Class of Dynamic Systems that Can Code:*
  - Point-to-point and periodic behavior as their attractor
  - Multi-dimensional systems that required phase locking
  - Attractors that have rather complex shape (e.g., complex phase relationships, movement reversals)
  - Learning and optimization
  - Coupling phenomena
  - Timing (without requiring explicit time)
  - Generalization (structural equivalence for parameter changes)
  - Robustness to disturbances and interactions with the environment
  - Stability guarantees

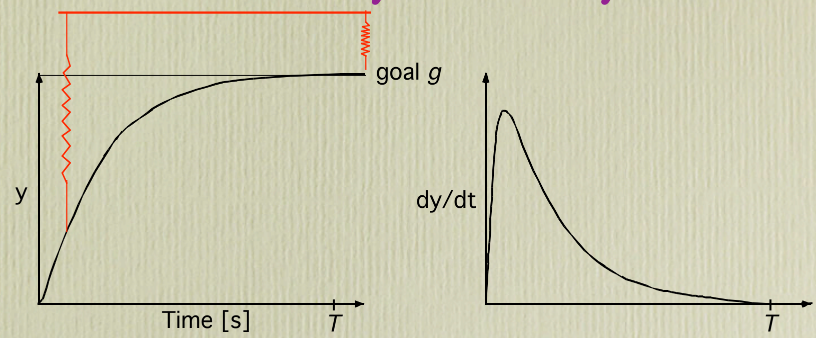


A Sketch of a Control Diagram Using Motor Primitives

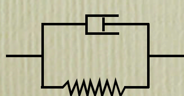




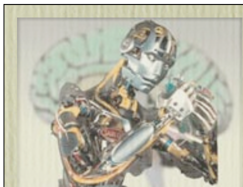
Point-to-Point Movements as Dynamic Systems



E.g., for a one degree-of-freedom movement, start with a simple damped spring model



$$\equiv \quad \ddot{y} + b\dot{y} + k(g-y) = 0 \quad \equiv \quad \begin{aligned} \dot{z} &= \alpha_z(\beta_z(g-y)-z) \\ \dot{y} &= z \end{aligned}$$

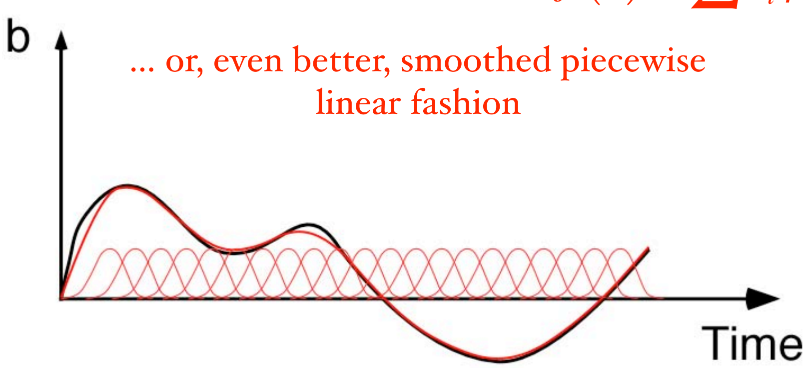


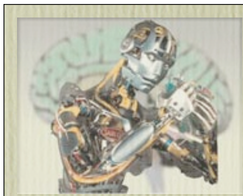
Point-to-Point Movements as Dynamic Systems

How to Create a Very Nonlinear Spring?

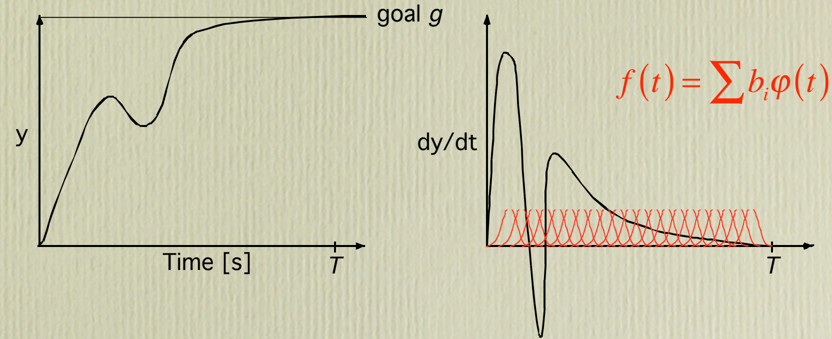
For instance, create a time varying damping term:

$$f(?) = \sum b_i \varphi(?)$$



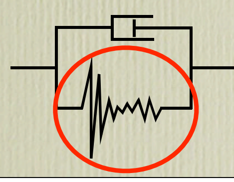


Point-to-Point Movements as Dynamic Systems



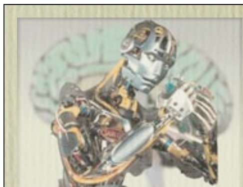
$f(t) = \sum b_i \varphi(t)$

Can one create more complex dynamics with a very nonlinear spring?



$$\dot{z} = \alpha_z(\beta_z(g-y)-z)$$
$$\dot{y} = f(?) + z$$

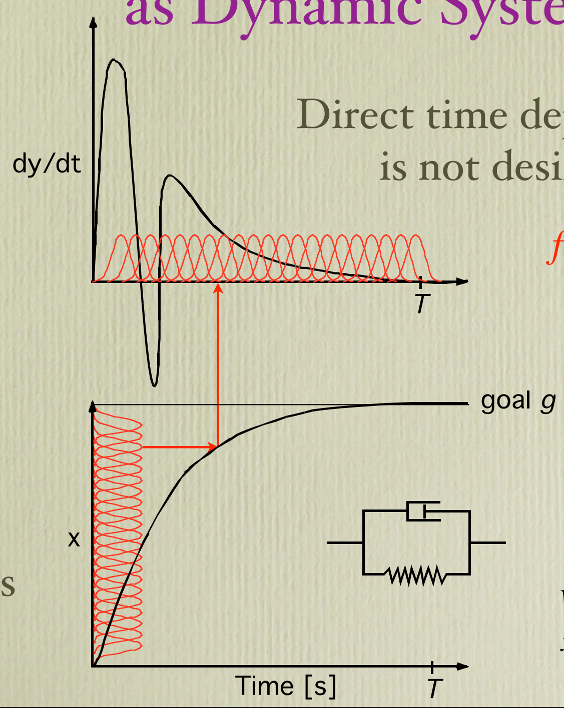
37



Point-to-Point Movements as Dynamic Systems

Direct time dependence is not desirable

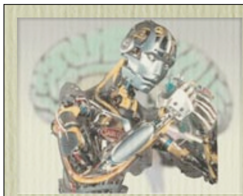
$f = f(x,v)$



Introduce a behavioral phase dynamics

$$\dot{v} = \alpha_v(\beta_v(g-x)-v)$$
$$\dot{x} = \alpha_x v$$

38



A Dynamic Systems Model for Discrete Movement

- *A learnable nonlinear point attractor with guaranteed stability properties*

Behavioral Phase

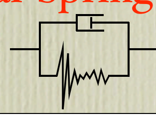
$$\dot{v} = \alpha_v \left(\beta_v \left(g - x\right) - v\right)$$
$$\dot{x} = \alpha_x v$$

Nonlinear Function

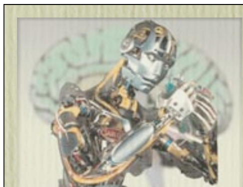
$$f\left(x, v\right)$$

$$\dot{z} = \alpha_z \left(\beta_z \left(g - y\right) - z\right)$$
$$\dot{y} = \alpha_y \left(f\left(x, v\right) + z\right)$$

Nonlinear Spring-Damper



39



A Dynamic Systems Model for Discrete Movement

- *A learnable nonlinear point attractor with guaranteed stability properties*

Trajectory Plan Dynamics

$$\dot{z} = \alpha_z \left(\beta_z \left(g - y\right) - z\right)$$
$$\dot{y} = \alpha_y \left(f\left(x, v\right) + z\right)$$

where

Canonical Dynamics

$$\dot{v} = \alpha_v \left(\beta_v \left(g - x\right) - v\right)$$
$$\dot{x} = \alpha_x v$$

“Phase Velocity” for Nonlinear Amplification

“Phase” for Localization

Local Linear Model Approx.

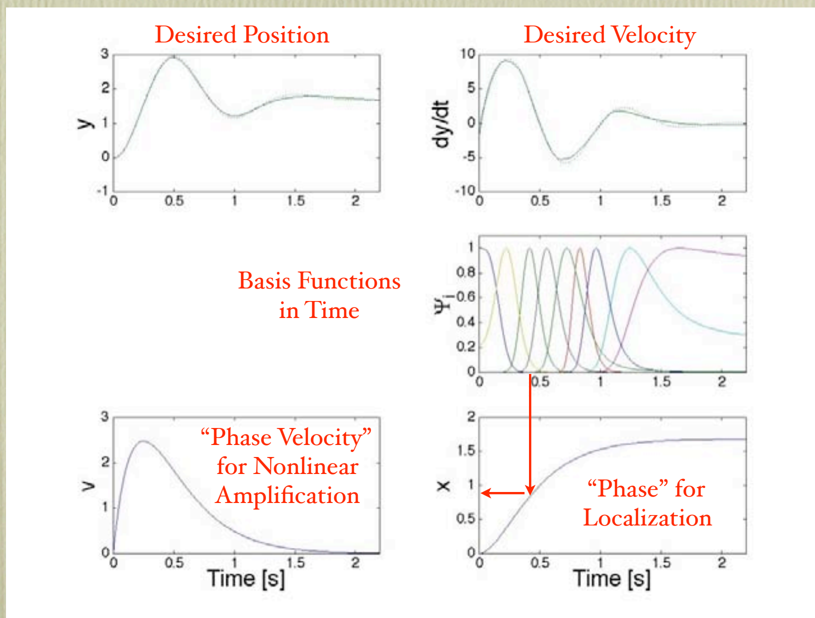
$$f\left(x, v\right) = \frac{\sum_{i=1}^{k} w_i b_i v}{\sum_{i=1}^{k} w_i}$$

Learnable Weights

$$w_i = \exp\left(-\frac{1}{2} d_i \left(\bar{x} - c_i\right)^2\right) \text{ and } \bar{x} = \frac{x - x_0}{g - x_0}$$

40

An Example



Extension to Periodic Systems

- *A learnable nonlinear limit cycle attractor with guaranteed stability properties*

Behavioral Phase

$$\dot{r} = \alpha_r (A - r)$$
$$\dot{\varphi} = \omega$$

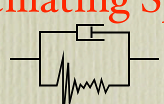
Phase Oscillator
with amplitude A

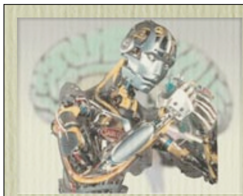
Nonlinear Function

$$f(r, \varphi)$$

$$\dot{z} = \alpha_z (\beta_z (g - y) - z)$$
$$\dot{y} = \alpha_y (f(r, \varphi) + z)$$

Nonlinear Oscillating Spring-Damper





Extension to Periodic Systems

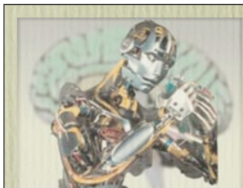
- *Use van Mises Basis Function for Local Linear Models*

Trajectory Plan Dyanmics
$$\begin{cases} \dot{z} = \alpha_z\left(\beta_z\left(g - y_m\right) - z\right) \\ \dot{y} = \alpha_y\left(f\left(r,\varphi\right) + z\right) \end{cases}$$

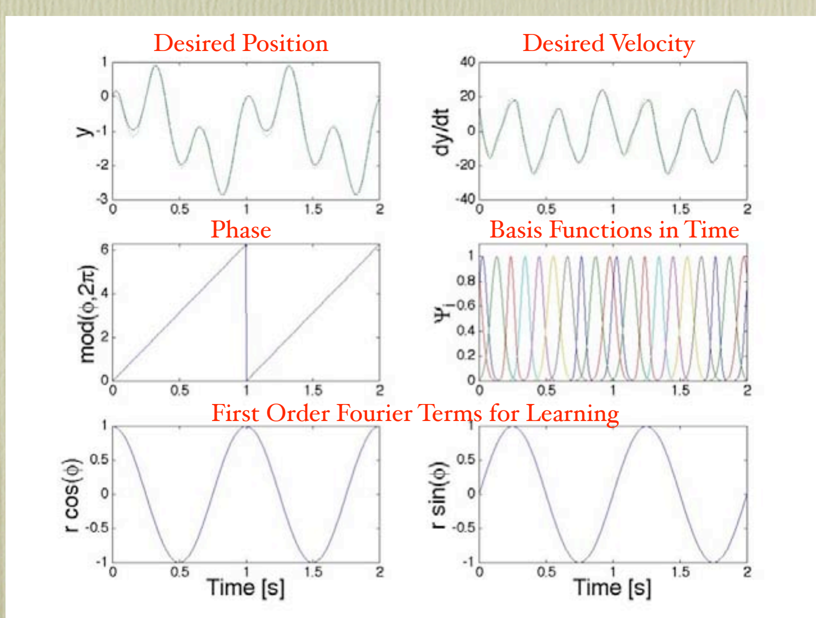
where

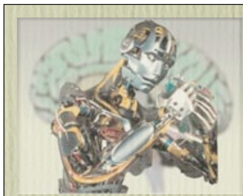
Canonical System
$$\begin{cases} \dot{r} = \alpha_r\left(A - r\right) \\ \dot{\varphi} = \omega \end{cases}$$

Local Linear Models using van Mises bases
$$\begin{cases} f\left(x,v\right) = \dfrac{\sum_{i=1}^{k} w_i \mathbf{b}_i^T \mathbf{x}}{\sum_{i=1}^{k} w_i} \quad \text{where } \mathbf{x} = \begin{bmatrix} r\cos\varphi \\ r\sin\varphi \end{bmatrix} \\ w_i = \exp\left(d_i\left(\cos\left(\varphi - c_i\right) - 1\right)\right) \end{cases}$$



An Example

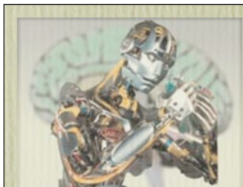




Real Demos

Pointer Attractor Demo

Limit Cycle Demo



How Did These Demos Work?

Trajectory Plan Dynamics

$$\begin{cases} \dot z = \alpha_z(\beta_z(g-y)-z) \\ \dot y = \alpha_y(f(x,v)+z) \end{cases}$$

where

Canonical Dynamics

$$\begin{cases} \dot v = \alpha_v(\beta_v(g-x)-v) \\ \dot x = \alpha_x v \end{cases}$$

Local Linear Model Approx.

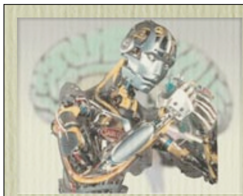
$$\begin{cases} f(x,v) = \dfrac{\sum_{i=1}^{k} w_i b_i v}{\sum_{i=1}^{k} w_i} \\ w_i = \exp\left(-\frac{1}{2} d_i(\bar{x}-c_i)^2\right) \text{ and } \bar{x} = \dfrac{x-x_0}{g-x_0} \end{cases}$$

Note: This is a one-shot learning problem, i.e., no iterations!

- *Given*
  - A go
  - A de
- *Algori*
  - Extr
  - Adjust time constants of canonical dynamics to movement duration
  - Using Locally Weighted Learning to solve nonlinear function approximation problem

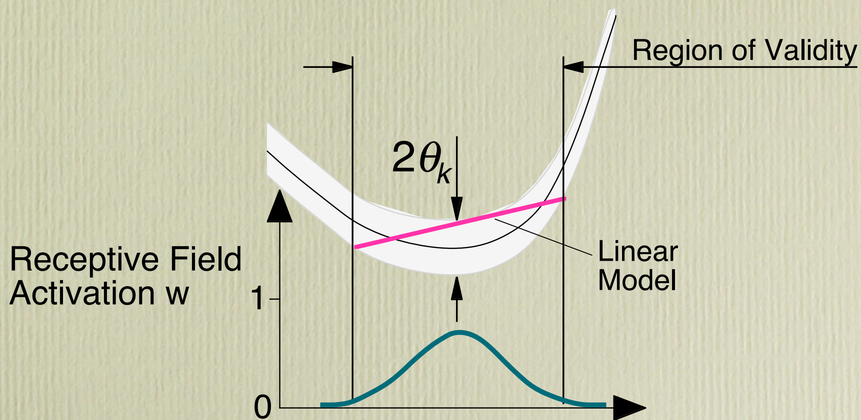
$$\dot{y}_{\text{target}} = \frac{\dot{y}_{demo}}{\alpha_y} - z = f(x,v)$$

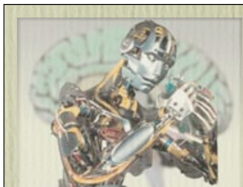
  - where z can be calculated from desired trajectory



Locally Weighted Learning

with locally linear models

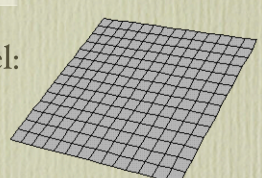




Locally Weighted Learning

with locally linear models

Linear Model:



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

learned with

Recursive weighted least squares:

$$\beta_k^{n+1} = \beta_k^n + w\mathbf{P}_k^{n+1}\mathbf{x}\left(\mathbf{y} - \tilde{\mathbf{x}}^T\beta_k^n\right)^T$$

$$\mathbf{P}_k^{n+1} = \frac{1}{\lambda}\left(\mathbf{P}_k^n - \frac{\mathbf{P}_k^n\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\mathbf{P}_k^n}{\frac{\lambda}{w} + \tilde{\mathbf{x}}^T\mathbf{P}_k^n\tilde{\mathbf{x}}}\right)$$

Weighting Kernel:

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

learned with

Gradient descent in penalized leave-one-out local cross-validation (PRESS) cost function:

$$\mathbf{M}_k^{n+1} = \mathbf{M}_k^n - \alpha\frac{\partial J}{\partial \mathbf{M}}$$

$$J = \frac{1}{\sum_{i=1}^{N} w_{k,i}}\sum_{i=1}^{N} w_{k,i}\left\|\mathbf{y}_i - \hat{\mathbf{y}}_{k,i,-i}\right\|^2 + \gamma\sum_{i=1,j=1}^{n} D_{k,ij}^2$$

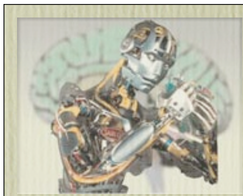
Combined Prediciton:

$$\mathbf{y} = \frac{\sum_{i=1}^{K} w_i\mathbf{y}_k}{\sum_{i=1}^{K} w_i}$$

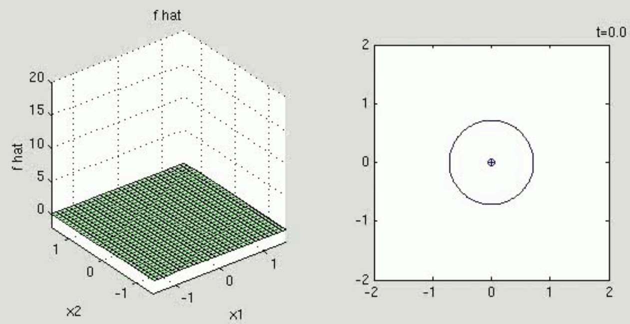
add model when

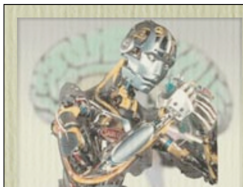
$$\textit{if} \ \min_k\left(w_k\right) < w_{gen}$$

$$\textit{create new RF at } \mathbf{c}_{K+1} = \mathbf{x}$$



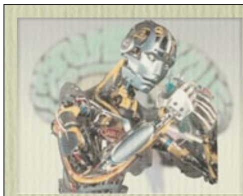
Visualization of On-line Function Approximation





Additional Issues to Be Addressed

- *Coordinating multiple Degrees-of-Freedom*
- *Movement Recognition*
- *Dealing with Perturbations*



Coordinating Multiple Degrees-of-Freedom

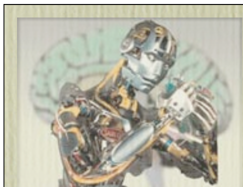
Behavioral Phase

$$\dot{r} = \alpha_r (A - r)$$
$$\dot{\varphi} = \omega$$

Nonlinear Function

$$f(r,\varphi)$$

$$\dot{z} = \alpha_z \left(\beta_z (g - y) - z\right)$$
$$\dot{y} = \alpha_y \left(f(r,\varphi) + z\right)$$



Coordinating Multiple Degrees-of-Freedom

Behavioral Phase

$$\dot{r} = \alpha_r (A - r)$$
$$\dot{\varphi} = \omega$$

Nonlinear Function

$$\dot{z}_1 = \alpha_z \left(\beta_z (g_1 - y_1) - z_1\right)$$
$$\dot{y}_1 = \alpha_y \left(f(r,\varphi) + z_1\right)$$

$f_1(r,\varphi)$ DOF-1

$$\dot{z}_2 = \alpha_z \left(\beta_z (g_2 - y_2) - z_2\right)$$
$$\dot{y}_2 = \alpha_y \left(f(r,\varphi) + z_2\right)$$

$f_2(r,\varphi)$ DOF-2

$$\dot{z}_3 = \alpha_z \left(\beta_z (g_3 - y_3) - z_3\right)$$
$$\dot{y}_3 = \alpha_y \left(f(r,\varphi) + z_3\right)$$

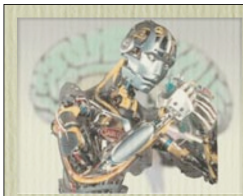
$f_3(r,\varphi)$ DOF-3

$$\vdots$$

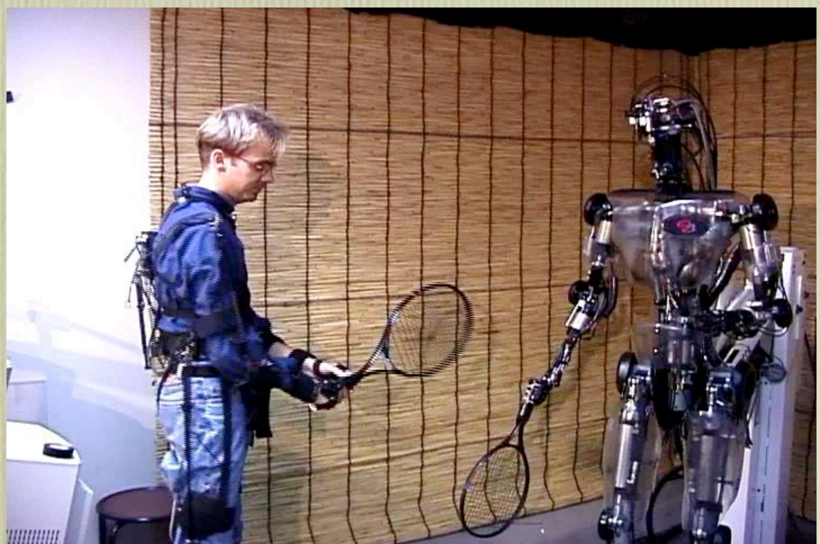
$$\dot{z}_N = \alpha_z \left(\beta_z (g_N - y_N) - z_N\right)$$
$$\dot{y}_N = \alpha_y \left(f(r,\varphi) + z_N\right)$$

$f_N(r,\varphi)$ DOF-N

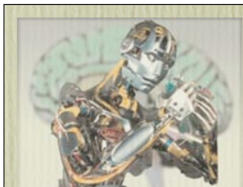
A Central Patten Generator?



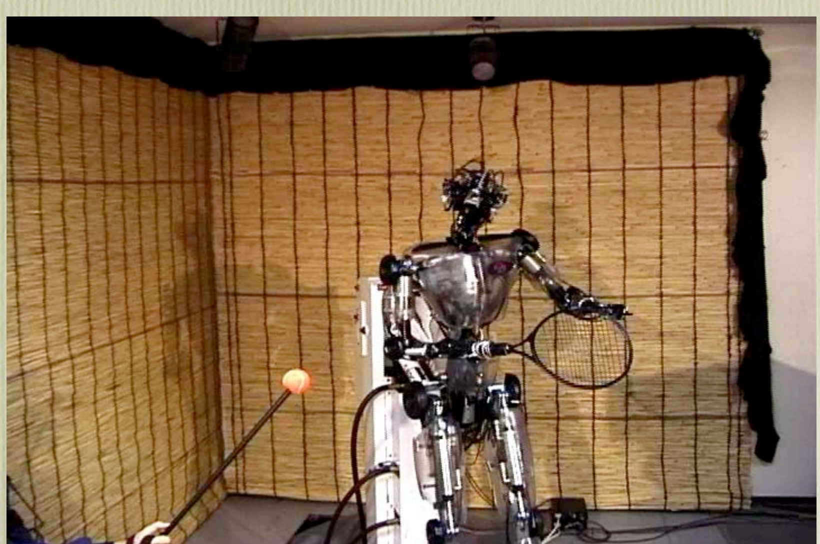
Example: A Tennis Backhand as a Dynamic System



53



Example: A Tennis Backhand as a Dynamic System



54

Example: Various Periodic Movement Patterns

Dealing with Perturbations: Adding On-line Modification

- *Among the most interesting properties of the DMP approach is the on-line modification with coupling terms*

Trajectory Plan Dynamics

$$\begin{cases} \dot{z} = \alpha_z\left(\beta_z(g-y)-z\right) \\ \dot{y} = \alpha_y\left(f(x,v)+z\right) + \alpha_p\left(y_{real}-y\right) \end{cases}$$

where

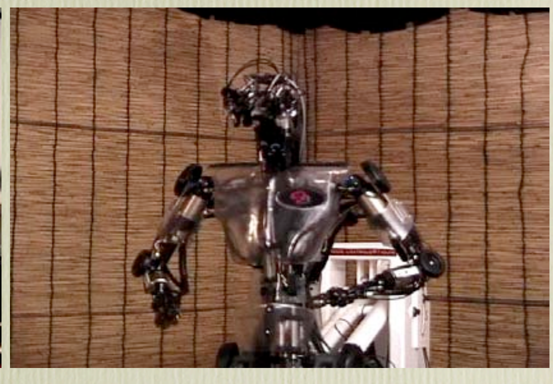
Canonical Dynamics

$$\begin{cases} \dot{v} = \alpha_v\left(\beta_v(g-x)-v\right) \\ \dot{x} = \alpha_x v\left(1+\alpha_p\left(y_{real}-y\right)^2\right)^{-1} \end{cases}$$

Local Linear Model Approx.

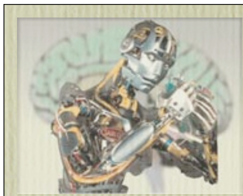
$$\begin{cases} f(x,v) = \dfrac{\sum_{i=1}^{k} w_i b_i v}{\sum_{i=1}^{k} w_i} \\ w_i = \exp\left(-\frac{1}{2} d_i\left(\bar{x}-c_i\right)^2\right) \text{ and } \bar{x} = \dfrac{x-x_0}{g-x_0} \end{cases}$$

Example: A Periodic Movement with Perturbation

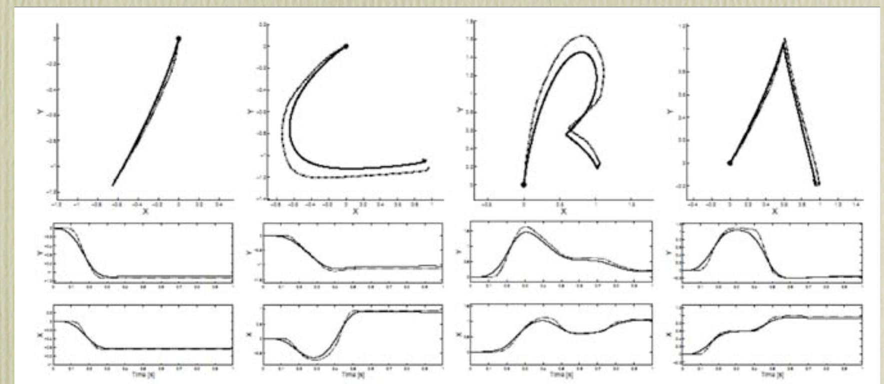


Movement Recognition

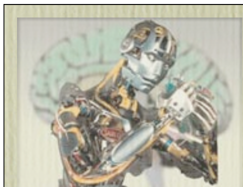
- *An Important Property:*
  - By design, the dynamic systems are structurally equivalent under scaling the distance to the goal for point attractor systems, and the amplitude for limit cycles
  - Structural equivalence also holds for a uniform scaling of the time constants
  - Thus, the parameters of the nonlinear function are invariant under spatial and temporal scaling of a movement and can be used to classify a movement pattern



Example: Recognizing Graffiti Characters



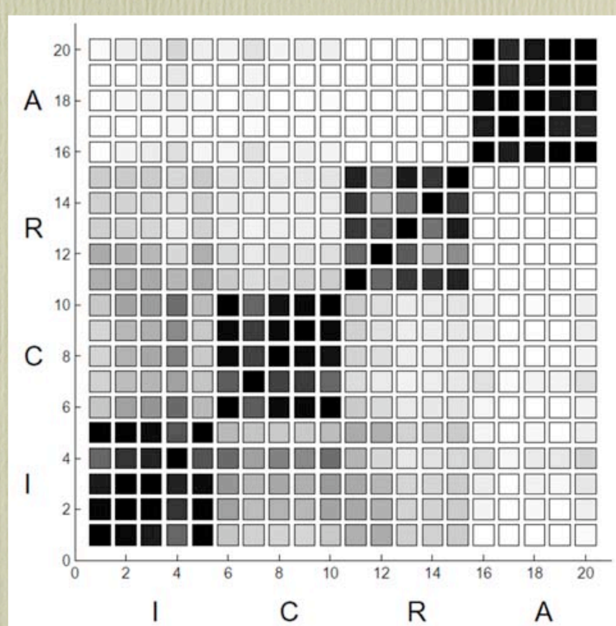
Characters are fit with 2DOF dynamic point attractors

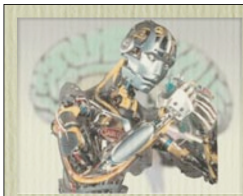


Example: Recognizing Graffiti Characters

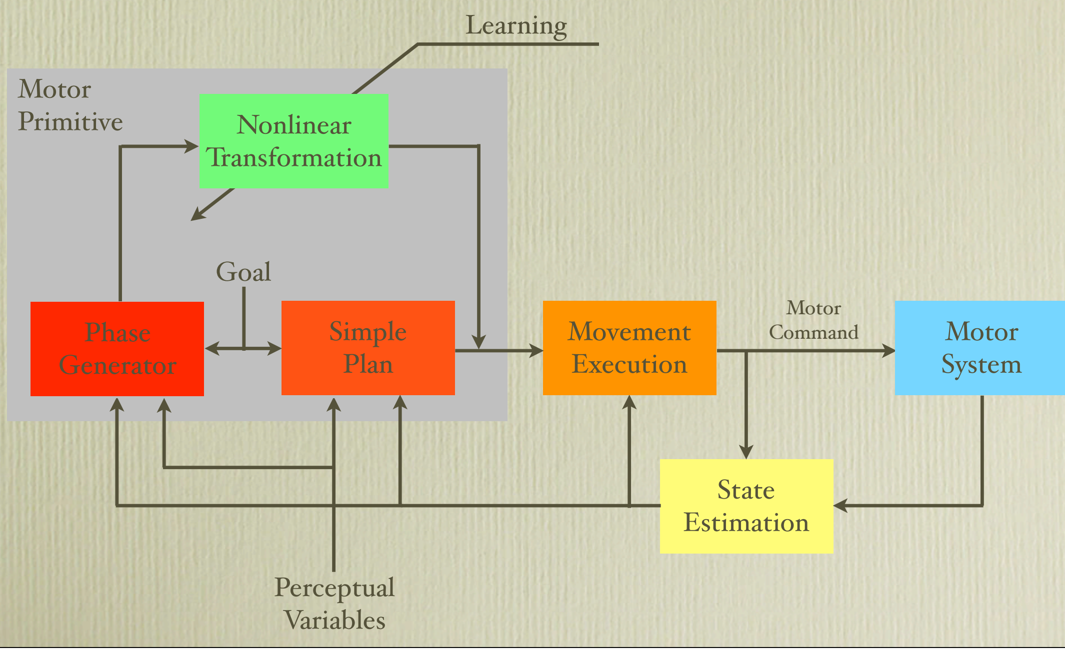
About 87% correct classification using correlation coefficient

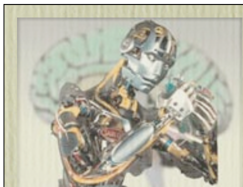
$$Score = \frac{\mathbf{b}_{Template}^{T}\mathbf{b}_{Observed}}{\left\|\mathbf{b}_{Template}\right\|\left\|\mathbf{b}_{Observed}\right\|}$$





A Sketch of a Control Diagram Using Motor Primitives

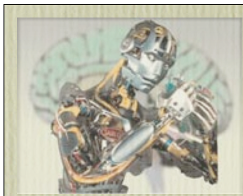




Computational Goals: What Did We Accomplish?

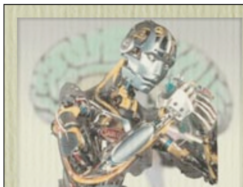
$$\dot{\mathbf{x}} = f\left(\mathbf{x}, goal\right)$$

- *A Class of Dynamic Systems that Can Code:*
  - ✔ Point-to-point and periodic behavior as their attractor
  - ✔ Multi-dimensional systems that required phase locking
  - ✔ Attractors that have rather complex shape (e.g., complex phase relationships, movement reversals)
  - ✔ Learning and optimization
  - ✔ Coupling phenomena
  - ✔ Timing (without requiring explicit time)
  - ✔ Generalization (structural equivalence for parameter changes)
  - ✔ Robustness to disturbances and interactions with the environment
  - ✔ Stability guarantees



Outline

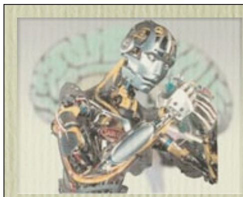
- *Part I: Dynamic Movement Primitives as a Computational Model for Human Movement?*
  - Some behavioral and fMRI data ...
- *Part II: The Formal Framework of Dynamic Motor Primitives*
  - Algorithms, imitation learning, and movement recognition
- *Part III: Reinforcement Learning with DMPs*
  - Optimization, skill learning, and other applications



Reinforcement Learning with Movement Primitives

- *Given:*
  - An acceleration-based motor primitives with parameters θ
  - $$\frac{1}{\tau}\dot{z} = \alpha_z\left(\beta_z\left(s-y\right)-z\right)+f \qquad \frac{1}{\tau}\dot{v} = \alpha_v\left(\beta_v\left(g-x\right)-v\right)$$
    $$\frac{1}{\tau}\dot{y} = z \qquad \frac{1}{\tau}\dot{x} = v$$
    $$\frac{1}{\tau}\dot{s} = \alpha_g\left(g-s\right)$$
    such that
    $y, \dot{y}, \ddot{y} = \dot{z}$ is the desired trajectory

$$f\left(x,v,g\right) = \frac{\sum_{i=1}^{N}\psi_i b_i v}{\sum_{i=1}^{N}\psi_i}, \text{ where } \psi_i = \exp\left(-h_i\left(\frac{x-x_0}{g-x_0}-c_i\right)^2\right)$$



Reinforcement Learning with Movement Primitives

- *Given (cont'd):*
  - A stochastic realization of the motor primitive

$$u = \ddot{y} + \varepsilon$$

$$\pi(u \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} \left(u - \ddot{y}(\mathbf{x})\right)^2 \right)$$

$$\mathbf{x} = [x, v, z, y, s]^T$$

  - Several n-step roll-outs (a.k.a. trajectory, sample path, etc.)

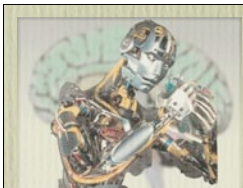
$$\xi = \left\{ \mathbf{x}_1, u_1, \mathbf{x}_2, u_2, \ldots, \mathbf{x}_n, u_n, \mathbf{x}_{n+1} \right\}$$

  - An immediate reward

$$r_i = r\left(\mathbf{x}_i, u_i\right)$$

  - A discounted long term reward of a roll-out

$$R(\xi) = \sum_{i=1}^{n} \gamma^{i-1} r_i$$



Reinforcement Learning with Movement Primitives

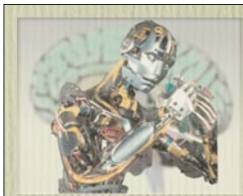
- *Given (cont'd):*
  - A cost criterion to be optimized

$$J(\theta) = E\left\{ R(\xi) \right\}_{\xi}$$

- *Approach*
  - Gradient descent in the primitive parameter using stochastic policy gradient methods

$$\theta = \left[\mathbf{b}^T \quad \sigma^2 \right]^T$$

$$\theta^{n+1} = \theta^n + \alpha \frac{\partial J}{\partial \theta}$$

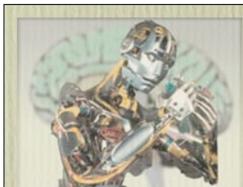


Computing the Policy Gradient

$$\begin{aligned}
\nabla_\theta J(\theta) = & \\
& = \nabla_\theta E\left\{R(\xi)\right\} \\
& = \nabla_\theta \int_\Xi p_\theta(\xi) R(\xi) d\xi \\
& = \int_\Xi \nabla_\theta p_\theta(\xi) R(\xi) d\xi \\
& = \int_\Xi p_\theta(\xi) \nabla_\theta \log p_\theta(\xi) R(\xi) d\xi \\
& = \left\langle \nabla_\theta \log p_\theta(\xi) R(\xi) \right\rangle_\xi
\end{aligned}$$

$$p_\theta(\xi) = p(\mathbf{x}_1) \prod_{i=1}^{n} \pi_\theta(u_i \mid \mathbf{x}_i) p(\mathbf{x}_{i+1} \mid \mathbf{x}_i, u_i)$$

$$\nabla_\theta \log p_\theta(\xi) = \sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(u_i \mid \mathbf{x}_i)$$



Computing the Policy Gradient

- *A useful observation*

$$\int_\Xi p_\theta(\xi) d\xi = 1$$

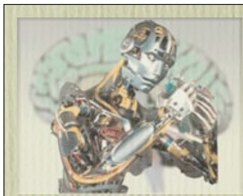
Thus

$$\nabla_\theta \int_\Xi p_\theta(\xi) d\xi = \int_\Xi p_\theta(\xi) \nabla_\theta \log p_\theta(\xi) d\xi = 0$$

and

$$\beta \int_\Xi p_\theta(\xi) \nabla_\theta \log p_\theta(\xi) d\xi = 0$$

for any parameter β



Policy Gradient I

Classic Policy Gradient

- *Thus, the policy gradient with baseline becomes*

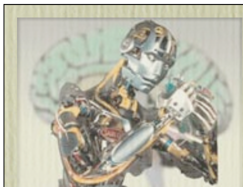
$$\boxed{\nabla_\theta J(\theta) = \left\langle \left(\sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(u_i \mid \mathbf{x}_i) \right) \left(R(\xi) - \beta \right) \right\rangle_\xi}$$

Episodic REINFORCE

- *The baseline that minimized the variance of the policy gradient can be shown to be:*

$$\boxed{\beta_i = \frac{\left\langle \left(\nabla_{\theta_i} \log p_\theta(\xi) \right)^2 R(\xi) \right\rangle_\xi}{\left\langle \left(\nabla_{\theta_i} \log p_\theta(\xi) \right)^2 \right\rangle_\xi}}.$$

William, 1992; Lawrence et al. 2004



Improving the Gradient

- *Inserting the definition of the roll-out reward*

$$\begin{aligned} \nabla_\theta J(\theta) &= \left\langle \left(\sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(u_i \mid \mathbf{x}_i) \right) \left(R(\xi) - \beta \right) \right\rangle_\xi \\ &= \left\langle \left(\sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(u_i \mid \mathbf{x}_i) \right) \left(\sum_{i=1}^{n} \gamma^{i-1} r_i - \beta \right) \right\rangle_\xi \end{aligned}$$

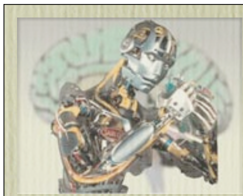
- *Realizing that rewards at time k cannot be affected by actions at time i>k, the gradient can be rewritten*

$$\nabla_\theta J(\theta) = \left\langle \sum_{k=1}^{n} \left(\sum_{j=1}^{k} \nabla_\theta \log \pi_\theta(u_j \mid x_j, u_j) \right) \left(\gamma^{k-1} r(x_k, u_k) - \beta(k) \right) \right\rangle_\xi$$

$$\beta_i(k) = \frac{\left\langle \left(\nabla_{\theta_i} \log p_\theta(\xi_{1:k}) \right)^2 r(x_k, u_k) \right\rangle_{\xi_{1:k}}}{\left\langle \left(\nabla_{\theta_i} \log p_\theta(\xi_{1:k}) \right)^2 \right\rangle_{\xi_{1:k}}}$$

GPOMDP

Baxter et al. 1999



Adding Function Approximation

- *Replace the roll-out reward with a function approximator*

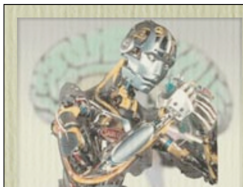
$$\nabla_\theta J(\theta) = \left\langle \left(\sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(u_i \mid \mathbf{x}_i) \right) \left(R(\xi) - \beta \right) \right\rangle_\xi$$

$$R(\xi) - \beta = f(\xi)$$

- *It can be shown (Sutton et al, Konda et al, 2000) that in order to avoid biasing the gradient, the function approximator needs to be of the form*

$$f(\xi) = \begin{bmatrix} \nabla_\theta \log p_\theta(\xi)^T & 1 \end{bmatrix} \mathbf{w}$$

71



The Natural Gradient

- *Inserting the function approximator into the gradient, and canceling all irrelevant terms results in*

$$\nabla_\theta J(\theta) = \left\langle \left(\sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(u_i \mid \mathbf{x}_i) \right) \left(\sum_{i=1}^{n} \nabla_\theta \log \pi_\theta(u_i \mid \mathbf{x}_i) \right)^T \right\rangle_\xi \tilde{\mathbf{w}}$$

$$= \mathbf{F}\tilde{\mathbf{w}}$$

where $\tilde{\mathbf{w}}$ is the $\mathbf{w}$ vector without the constant cofficient ,

and $\mathbf{F}$ is the Fisher Information Matrix

- *Amari (1999) demonstrated that a more efficient gradient in stochastic optimization is*

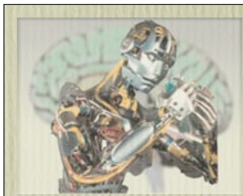
$$\nabla_\theta J(\theta)_{nat} = \mathbf{F}^{-1} \; \nabla_\theta J(\theta)$$

Thus:

$$\nabla_\theta J(\theta)_{nat} = \mathbf{F}^{-1}\mathbf{F}\tilde{\mathbf{w}} = \tilde{\mathbf{w}}$$

Kakade, 2002; Peters, 2004; Bagnell 2004

72



Policy Gradient II

Natural Policy Gradient

$$\nabla_\theta J(\theta)_{nat} = \tilde{\mathbf{w}}$$

where

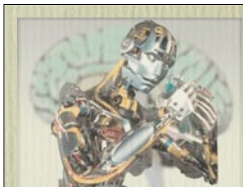
$$\mathbf{w} = \left[\tilde{\mathbf{w}}^T \;\; w_0\right]^T$$

$$\mathbf{w} = \left(\mathbf{X}^T\mathbf{X}\right)\mathbf{X}^T\mathbf{Y}$$

$$\mathbf{X} = \begin{bmatrix} \nabla_\theta \log p_\theta(\xi_1)^T & 1 \\ \nabla_\theta \log p_\theta(\xi_1)^T & 1 \\ \vdots & \\ \nabla_\theta \log p_\theta(\xi_K)^T & 1 \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} R(\xi_1) \\ R(\xi_2) \\ \vdots \\ R(\xi_K) \end{bmatrix}$$

Note: There is a also a “natural” gradient based on GPOMDP

73

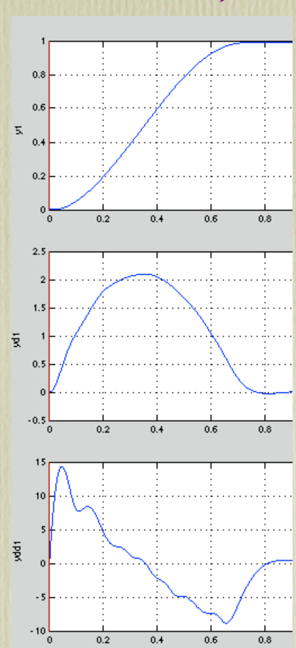


Example: Learning a Minimum Motor Command Trajectory

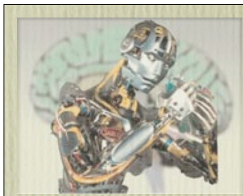
- *Assume that goal and duration of a movement are given, learn the trajectory that minimizes*

  $$r_i = c\,u_i^2$$

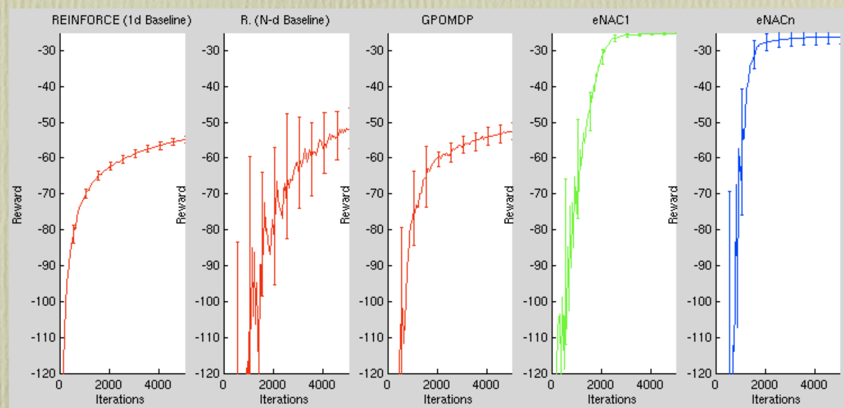
  accumulated over the entire trajectory
- *Compare different stochastic policy algorithms*

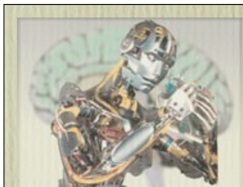


74

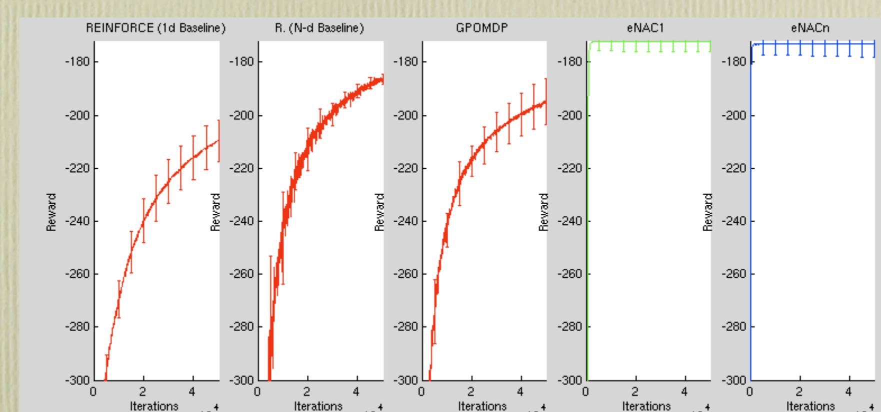


Example: Learning a Minimum Motor Command Trajectory





Example: Learning a Minimum Motor Command Trajectory



These results used a cubic spline representation.

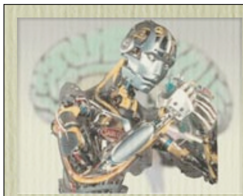
Example: Imitation Learning with Self-Improvement

Goal: Hit ball precisely   Note: about 150 trials are needed.

77

Coupling of External Beat and Control

78



Discussion

- *Evidence from behavioral and fMRI data supports the idea of motor primitives, in particular in a dynamic systems framework*
- *Formulating motor primitives as kinematic dynamic systems for movement planning offers a model of movement generation, which can can address many issues, including:*
  - Optimization
  - Reinforcement learning, supervised learning, imitation learning
  - Perception-Action Coupling
  - Motor Primitives
  - Generalization
- *The suggested approach is more of a design principle rather than fixed formalism*
- *The necessary computations of this approach remain (so far) manageable and potentially biologically plausible, and may thus serve as a tool to model primate motor control phenomena*