Imitation Learning by Behavioral Cloning

Jan Peters
Gerhard Neumann
Takayuki Osa

Purpose of this Lecture

- Learn what imitation is all about?

- Study it first from a biological point of view, then from a technical point of view.

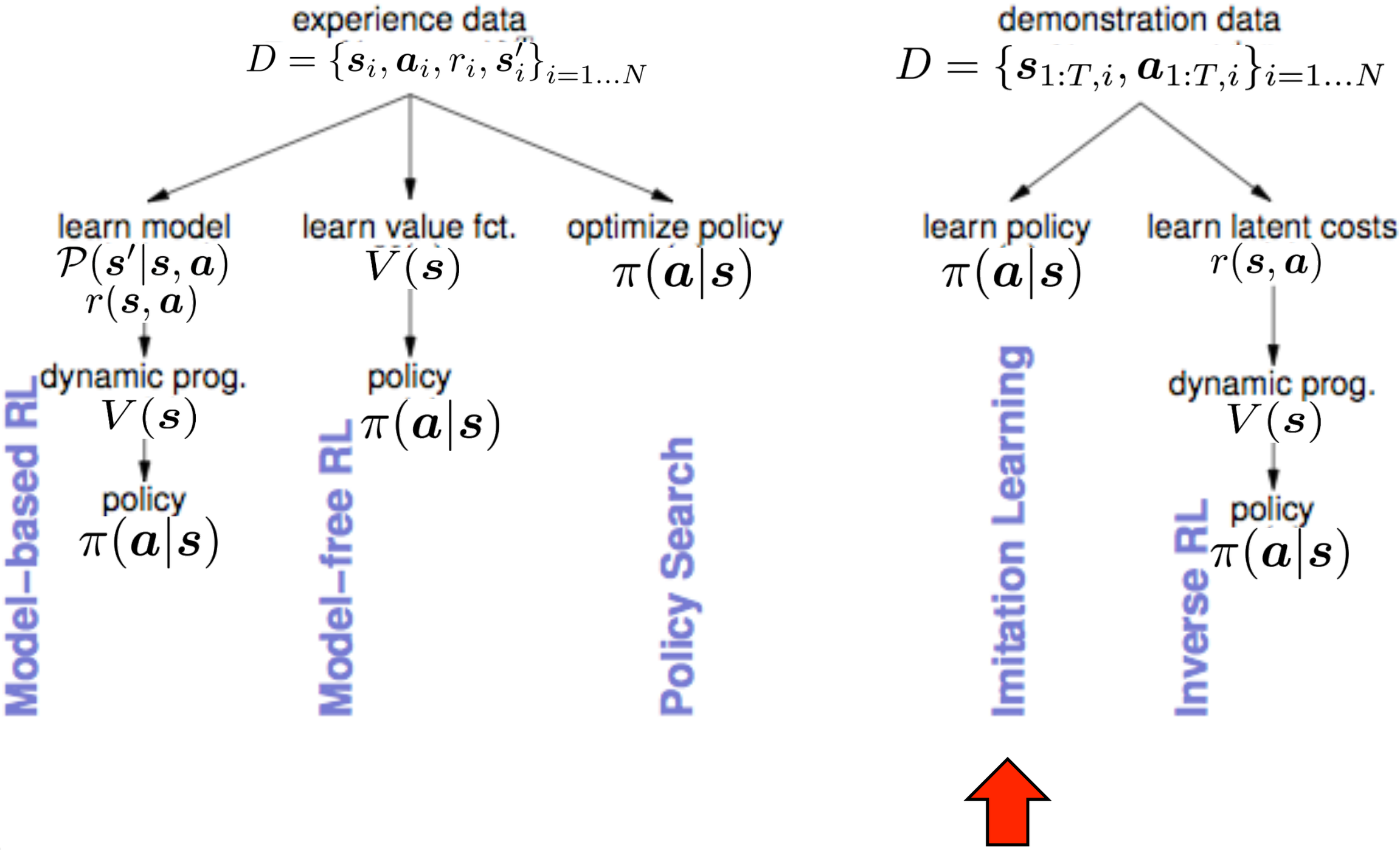
- This lecture only focusses on imitation by behavioral cloning.

Bigger Picture

- You have seen:
  - Model Learning

- Now, we are discussing a very different approach:
  - Imitation Learning by Behavioral Cloning
  - Imitation Learning by Inverse Reinforcement Learning.

Bigger Picture

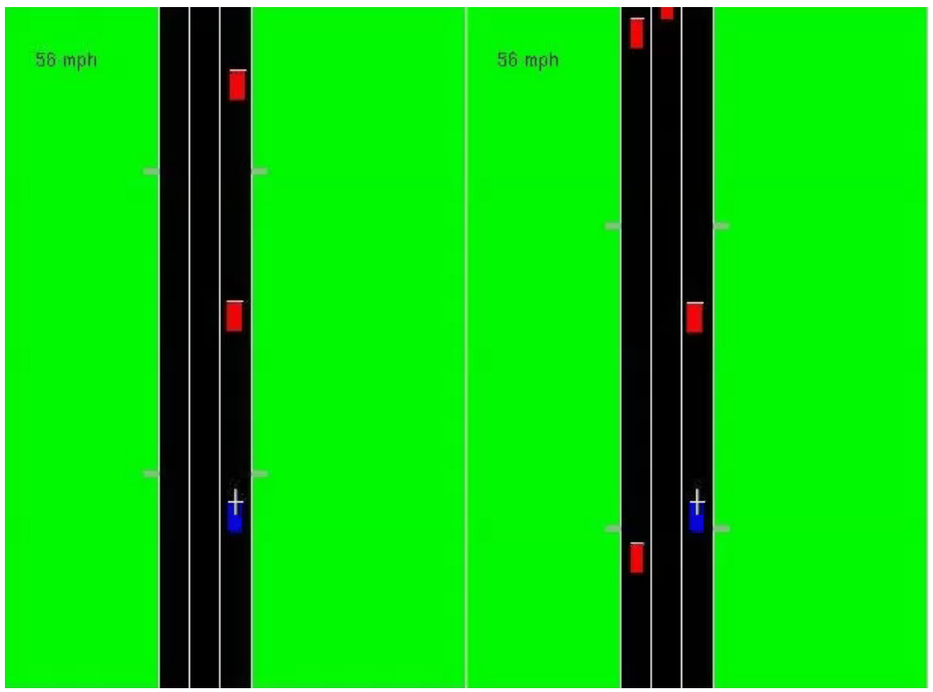


Imitation learning

Pieter Abbeel, et al. 2010

Imitation learning
by behavioral cloning

⇨ Learn motion itself



Pieter Abbeel, et al. 2004

Imitation learning
by inverse reinforcement learning

⇨ Learn the objective of the skill

Outline of the Lecture

1.Introduction

2.Biological Inspiration

3.Foundations

4. A Quick Overview: Well-known Approaches

5. Behavioral Cloning with Dynamical Systems

6. Behavioral Cloning with Forward Models

7. Conclusion

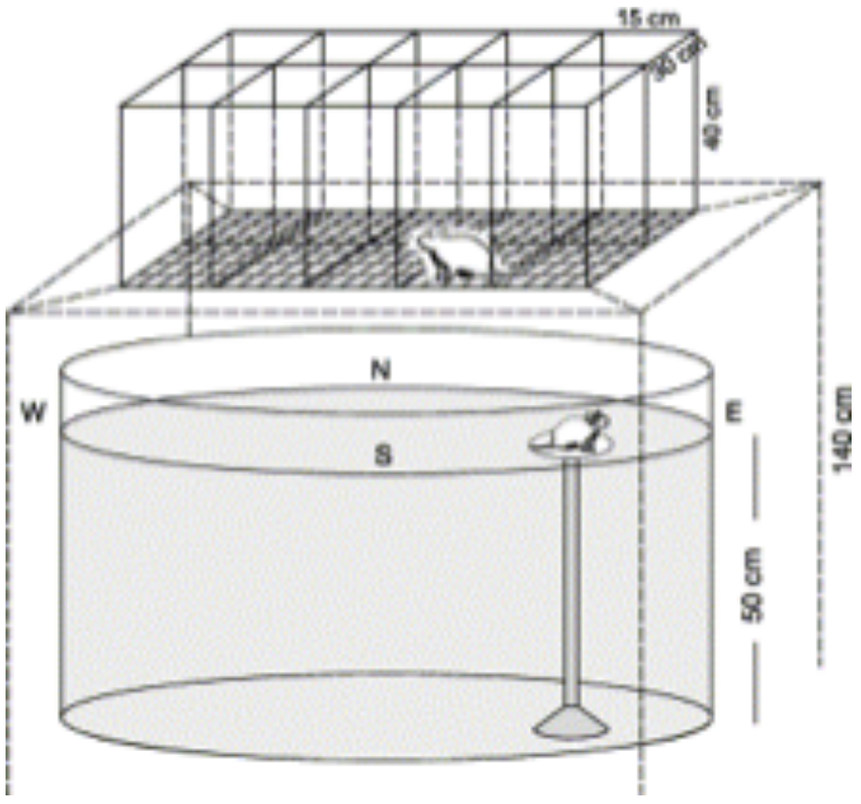
Can chimpanzees learn by Imitation?

The researcher first opens the top door of the box with the tool, manipulates the top hole with the tool and and then uses the front door and the tool to reach the treat hidden inside. Watching the researcher is the chimpanzee Yoyo, a 3-year-old female. Will she imitate the researcher's behaviour?

Can chimpanzees learn by Imitation?

Already rats can imitate!

- Student rats observe companion actor rats performing different spatial tasks differing according to the experimental requirements.



- After the observational training, surgical ablation to block any further learning in the student rat.

- The observer rats displayed exploration abilities that closely matched the previously observed behaviors.

Legio et al, Brain Res. Protocols, 2003
Heyes, Trends in Cog. Sciences, 2001

... and dolphins...

Infants have Imitation build in!

- Infants as young as 42 minutes old copy several facial actions (e.g., Meltzoff & Moore, 1977).

Stefan Schaal's Hypothesis

Imitation Learning is the Route to (intelligent) Humanoid Robots!

(S. Schaal, 1999)

Outline of the Lecture

1.Introduction

2.Biological Inspiration

3.Foundations

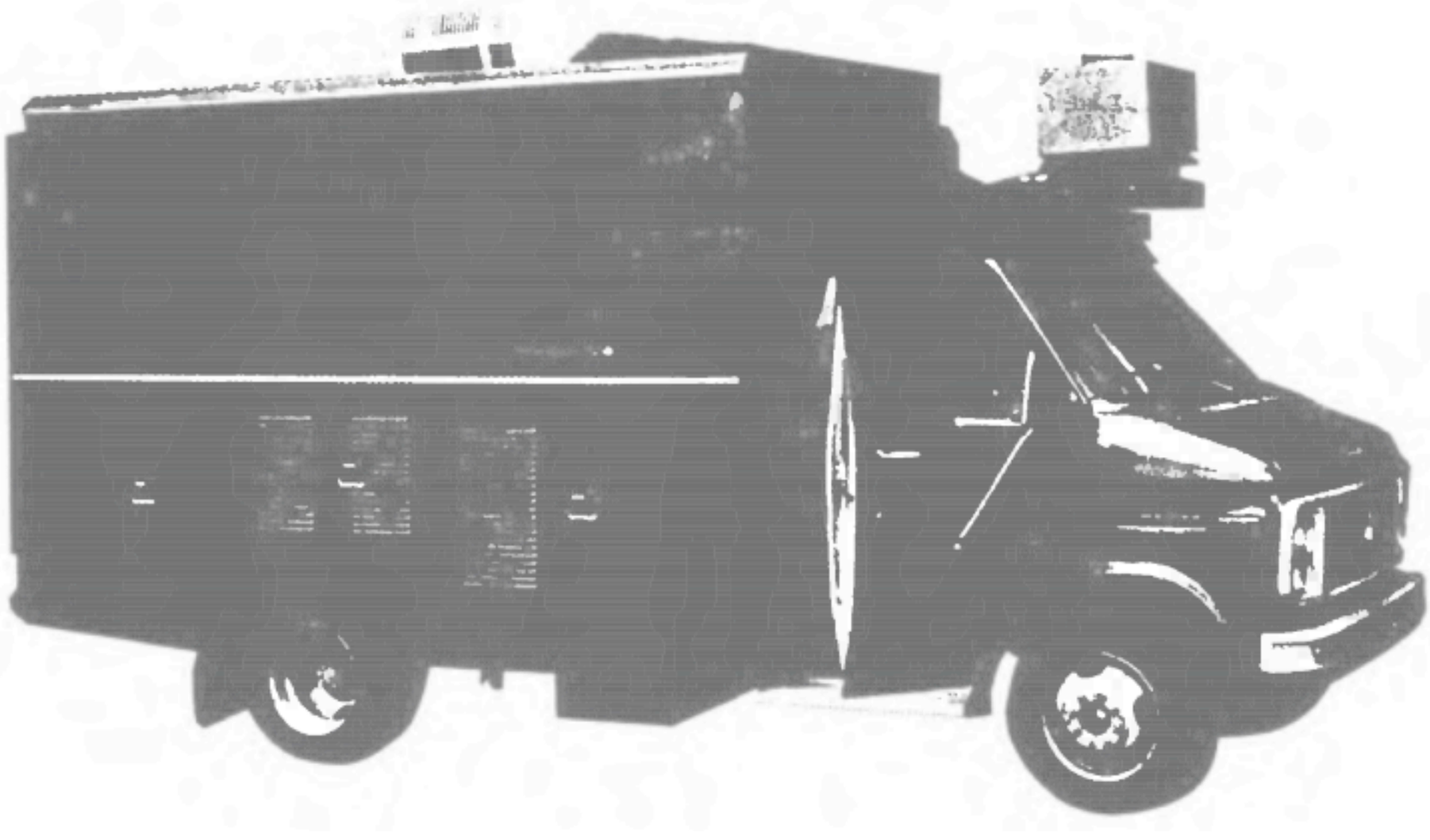
4. A Quick Overview: Well-known Approaches

5. Behavioral Cloning with Dynamical Systems

6. Behavioral Cloning with Forward Models

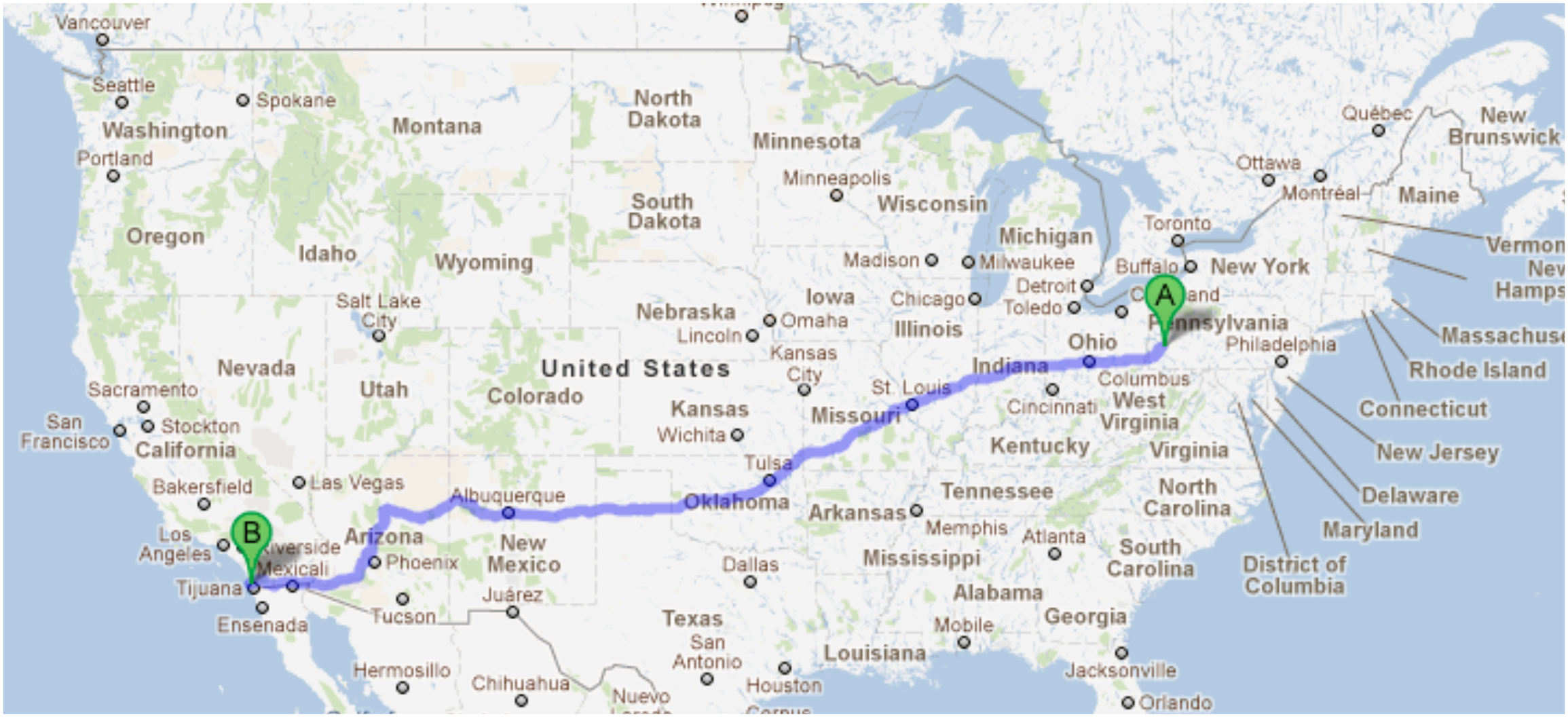
7. Conclusion

ALVINN & Navlab in 1989-1995!



Pomerlau (1989-1995)

No-Hands-Across-America



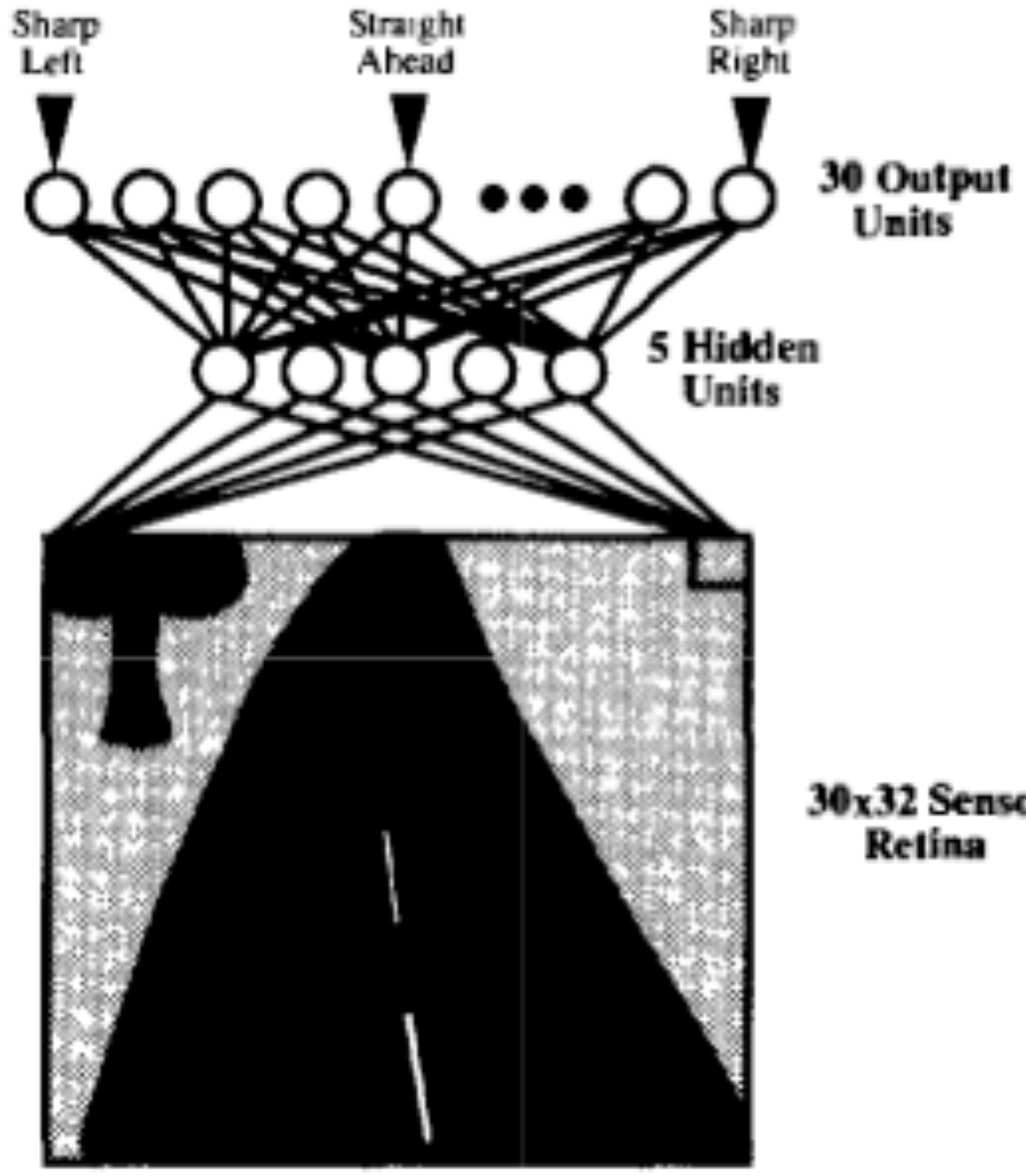
ALVINN allowed the Navlab vehicle of CMU's robotics institute to drive 2796km autonomously as part of their 'No-Hands-Across-America' Tour in 1995.

States and Actions

State:
Camera
Image

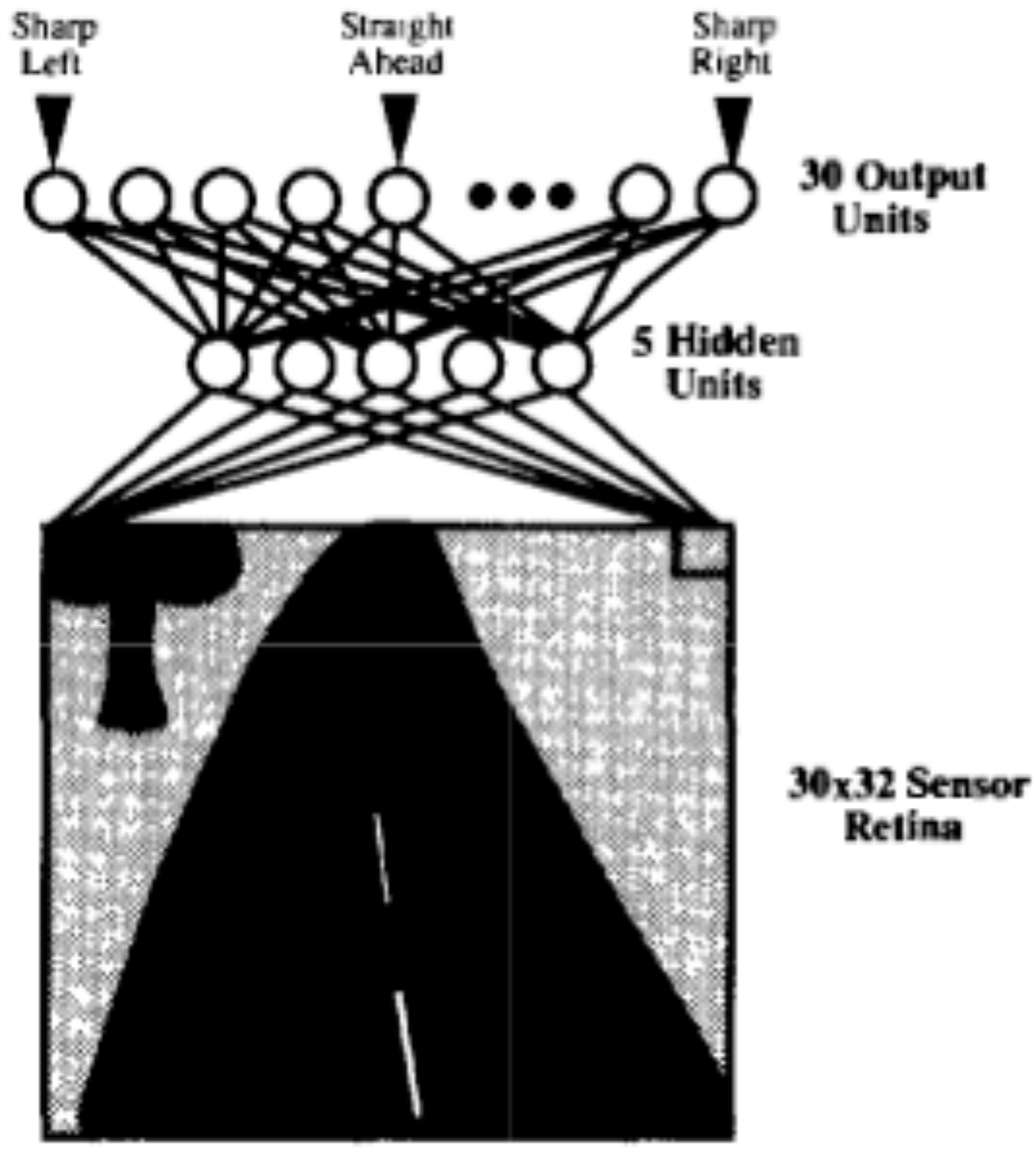
Action:
Steering Wheel,
Brakes, Gas

Function Approximator:
A Two-Layered Neural Network



Function Approximator:
A Two-Layered Neural Network

JUST (nonlinear) REGRESSION!



Video from ALVIN

How to Demonstrate?

- **Teleoperation**: Use a joystick to train an RC car, a mouse for training a Quake III player, the steering wheel of the Navlab, data gloves, etc.

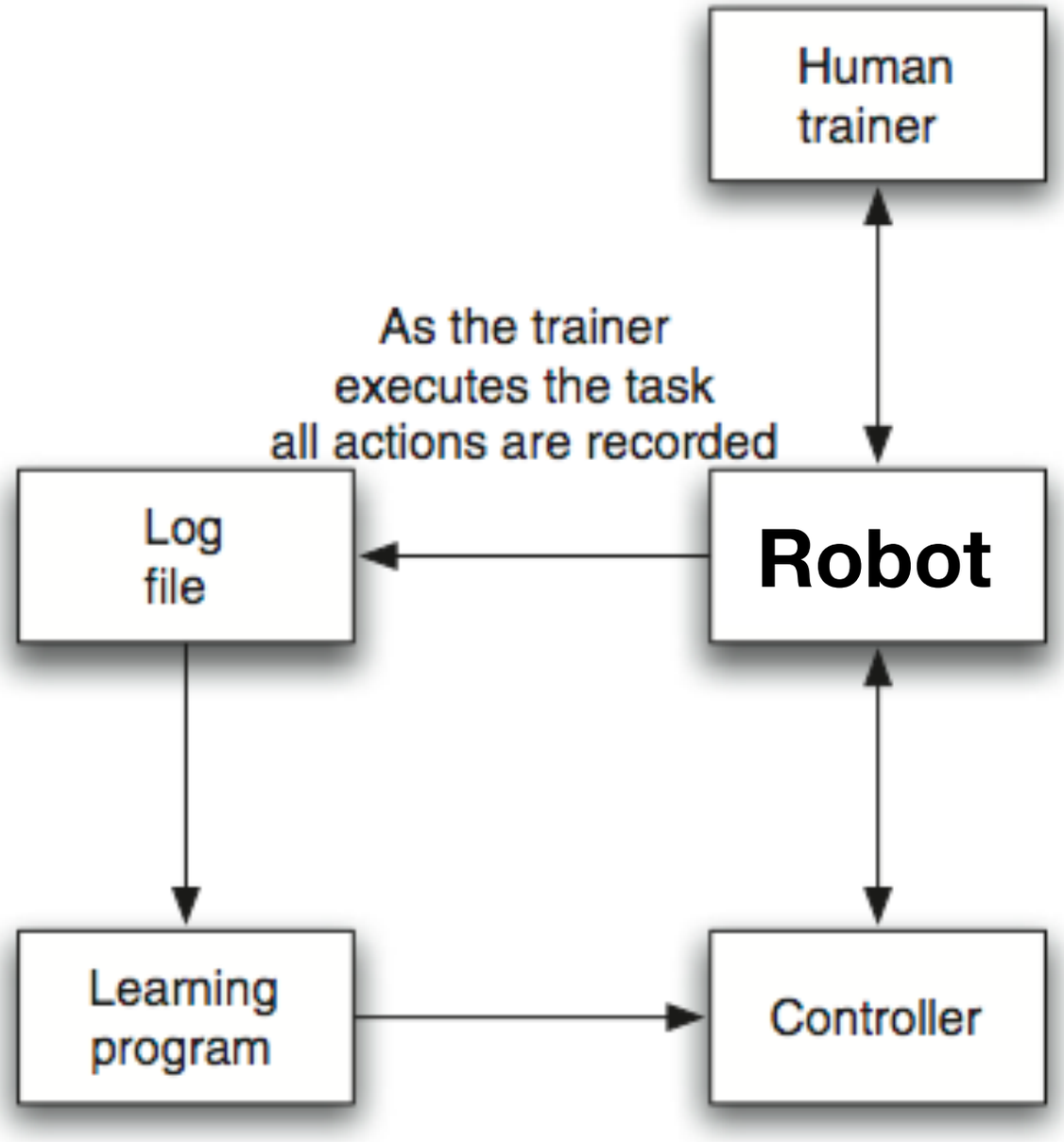
- **Kinesthetic Teach-In**: Take the robot by the hand like a tennis teacher teaches a tennis student.

- **Vision**: Video-based tracking of human beings.

- **Marker-based Tracking**: With markers and a basic skeleton, very precise human data can be obtained.

- **Sensuits**: Suits with encoders and accelerometers attached to human beings.

How do we do imitation?



An learning program uses the logged data to build a controller

Basic Idea of Behavioral Cloning

- Behavioral Cloning is the simplest form of learning from demonstration

- An expert is available and supplies data traces:

$$\mathbf{x}_1 \rightarrow \mathbf{u}_1 \rightarrow \mathbf{x}_2 \rightarrow \mathbf{u}_2 \rightarrow \mathbf{x}_3 \rightarrow \mathbf{u}_3 \rightarrow$$

- The student infers a policy from these data traces, i.e.,

$$\mathbf{u} = \pi(\mathbf{x})$$

- In principle, this can be treated as a supervised learning problem.

Direct Behavioral Cloning

Standard ML techniques can simply be applied to the data set

$$D = \left\{ \mathbf{x}_i, \mathbf{u}_i \right\}$$

to extract a **policy**

$$\mathbf{u} = \pi(\mathbf{x}) = \phi(\mathbf{x})^T \theta$$

… the problem frequently boils down to a **regression problem**.

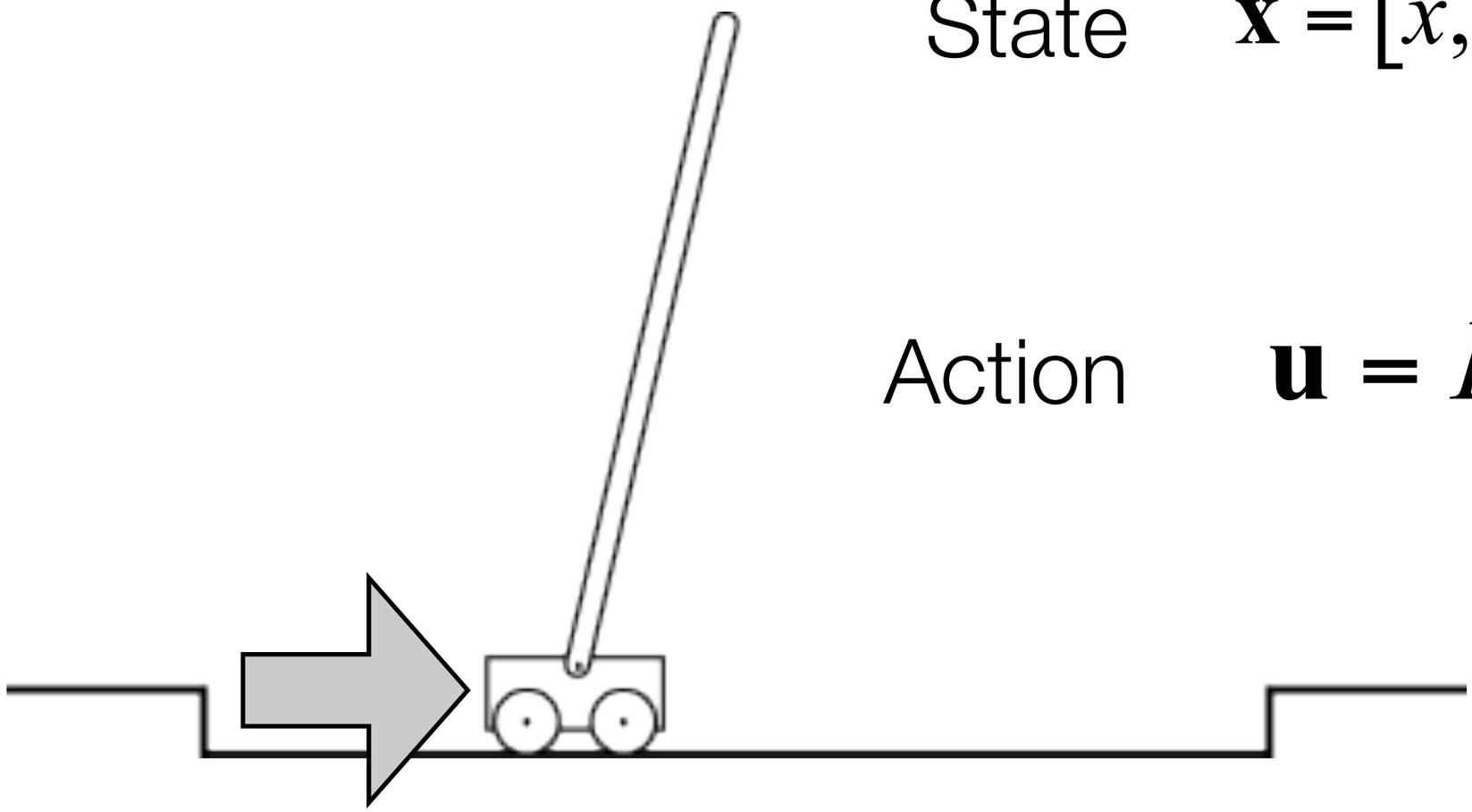
⇨ **The clean-up effect**: due to regularization, the noise in the demonstration is no longer exhibited by the reproduction and, hence, the clone often surpasses the quality of the expert.

Pole Balancing

Widrow and Smith (1964) used supervised learning to acquire a pole balancing policy.

State $\mathbf{x} = \left[x, \dot{x}, \alpha, \dot{\alpha}\right]$

Action $\mathbf{u} = F$



Pole Balancing

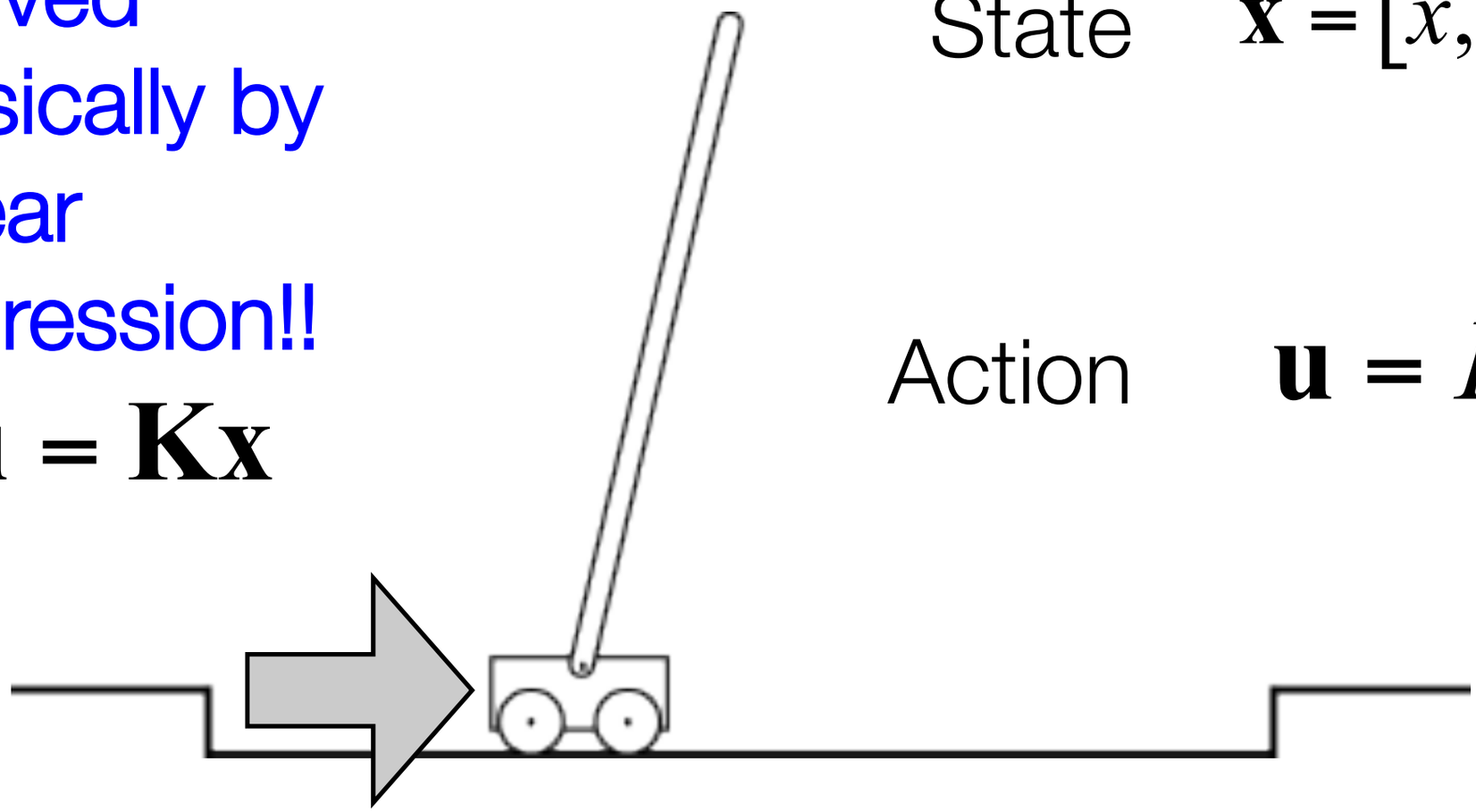
Widrow and Smith (1964) used supervised learning to acquire a pole balancing policy.

Solved basically by linear regression!!

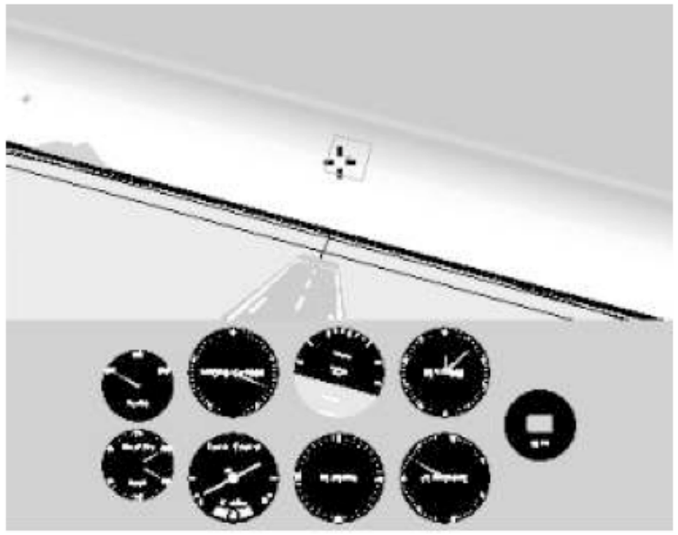
$$\mathbf{u} = \mathbf{K}\mathbf{x}$$

State $\mathbf{x} = \left[x, \dot{x}, \alpha, \dot{\alpha}\right]$

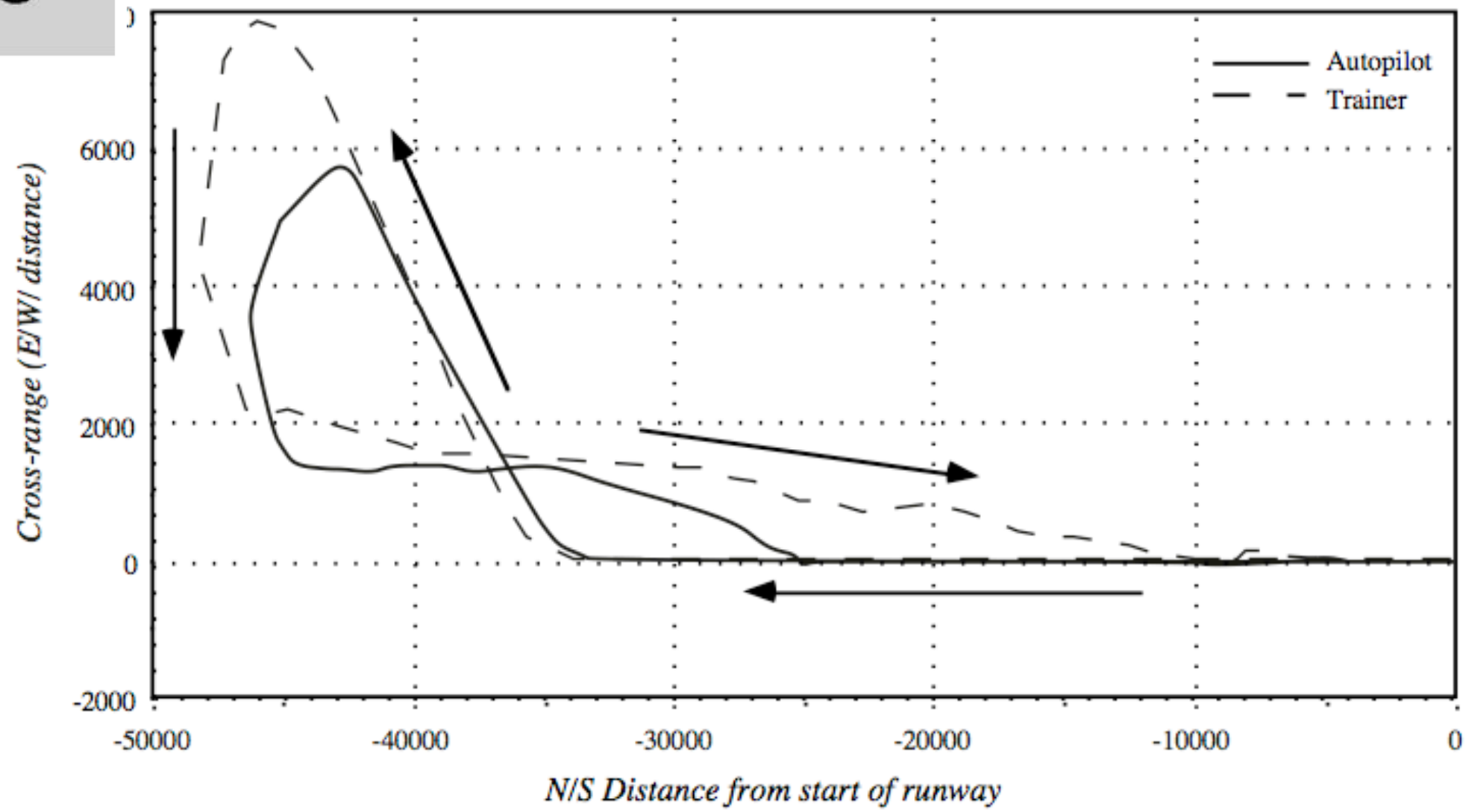
Action $\mathbf{u} = F$



Sammut's Cessna Pilot



(Sammut et al., 1992)



Non-linear state space representations

Radial Basis Function (RBF) networks:

$$f_{\boldsymbol{w}}(\boldsymbol{s}) = \phi^T(\boldsymbol{s})\boldsymbol{\beta}, \quad \phi_i(\boldsymbol{s}) = \exp\left(-0.5\sum_{j=1}^{D}(s_j - \mu_{ij})^2/h_{ij}\right)$$

$$\boldsymbol{w} = \{\boldsymbol{\beta}, \boldsymbol{\mu}_{1:K}, \boldsymbol{h}_{1:K}\}$$

Normalized RBF:

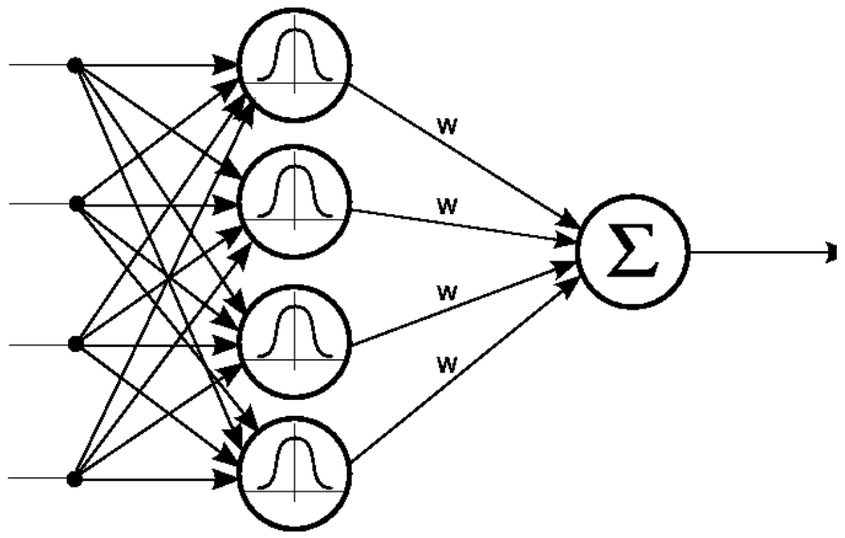
$$f_{\boldsymbol{w}}(\boldsymbol{s}) = \frac{\sum_{i=1}^{K}\phi_i(\boldsymbol{s})\beta_i}{\sum_{i=1}^{K}\phi_i(\boldsymbol{s})}$$

[-] A high number of parameters

[-] Non-convex optimization

[-] Hard to scale ⇨ curse of dimensionality

[+] Automatic feature construction



Alternatives: Gaussian Mixture Models (GMM), Neural Networks

Doubts on Direct Behavioral Cloning

- It becomes brittle for **larger state-spaces** unless you have a task-appropriate representation.

- Frequently leads to **catastrophic failures** if the controller has not been trained in this area of the state-action space (Sammut, 2010) or if there have been small changes in the system (Camacho & Michie, 1995).

- Reproduction of **single human teachers always works best** (Camacho & Michie, 1995).

- There is no **guarantee that the reproduction is meaningful**, nor an interpretation of behavior.

Core Open Questions in Imitation Learning

- **What to Imitate?** The data traces will contain outliers, redundant data, data that is irrelevant to the task. How can the system extract the relevant components?

- **How to Imitate?** The body of the teacher and the student may not have the exact same abilities or even kinematics. Finding the degrees of freedom actually corresponding to the teacher is a severe problem for the student called the „Correspondence Problem“.

- **When to Imitate?** Not all behavior in a data stream may be suited for imitation. Untackled questions

- **Whom to Imitate?** If a scene with several actors ist observed, the correct one needs to be extracted. Untackled questions

(Nehaniv & Dautenhan, 2001)

Outline of the Lecture

1.Introduction

2.Biological Inspiration

3.Foundations

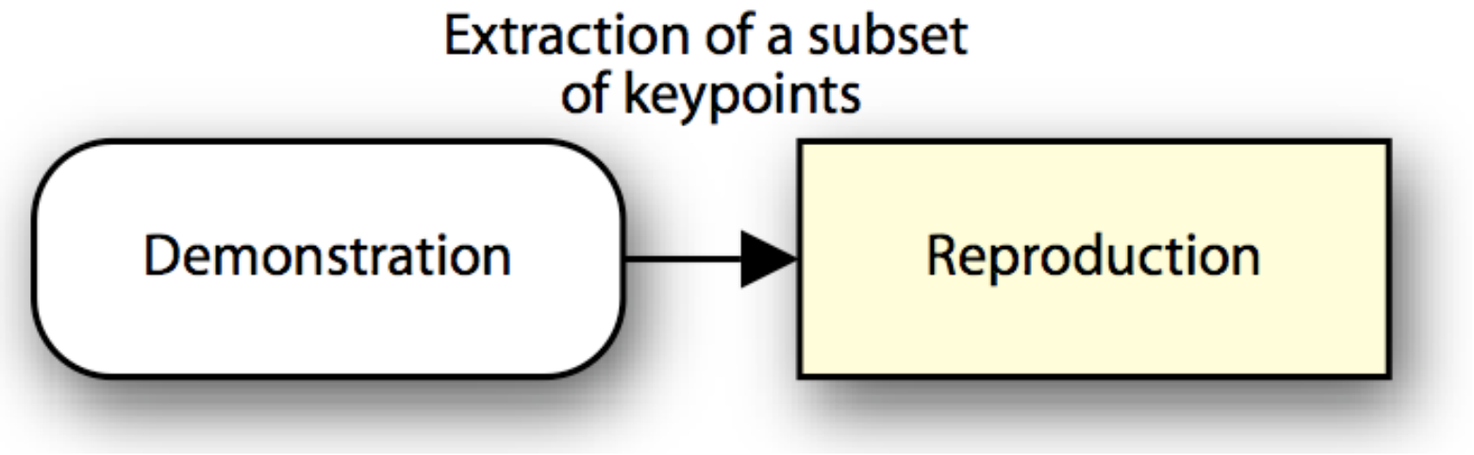
4. A Quick Overview: Well-known Approaches

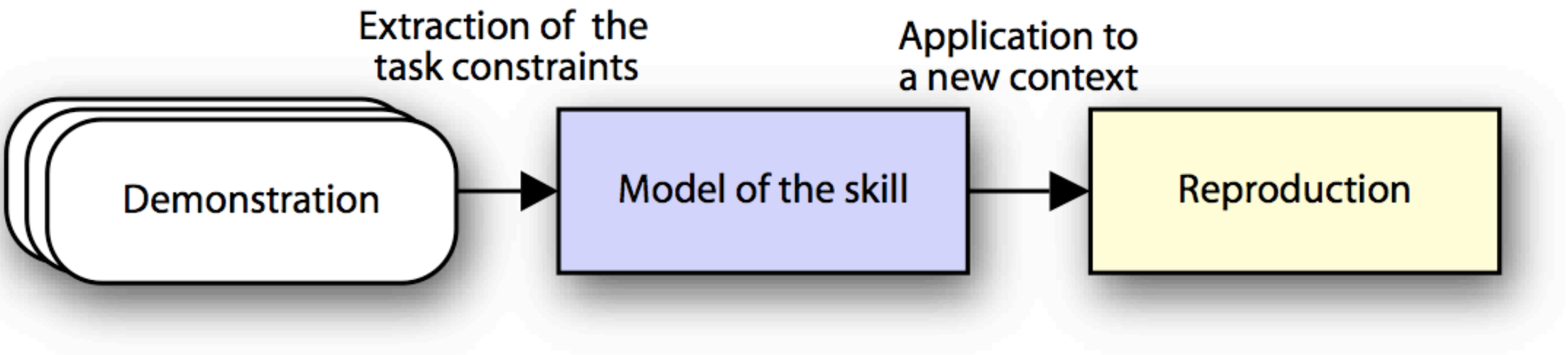
5. Behavioral Cloning with Dynamical Systems

6. Behavioral Cloning with Forward Models

7. Conclusion

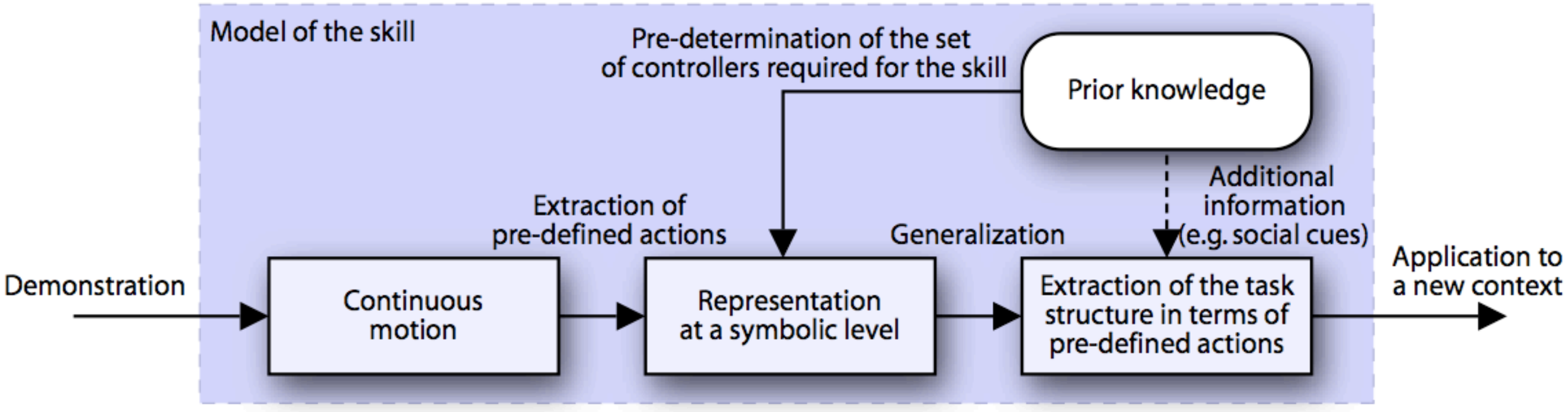
What is Imitation Learning? Two views!



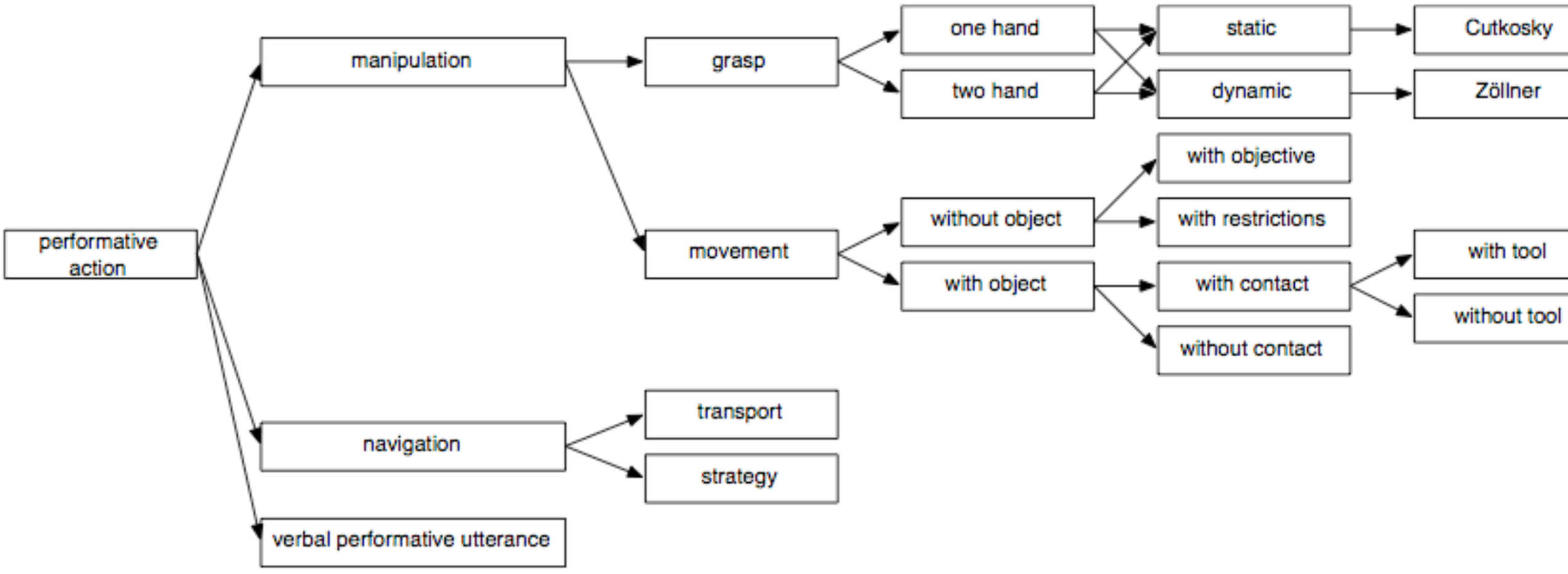


Symbolic Level Imitation

Generalization at a symbolic level:

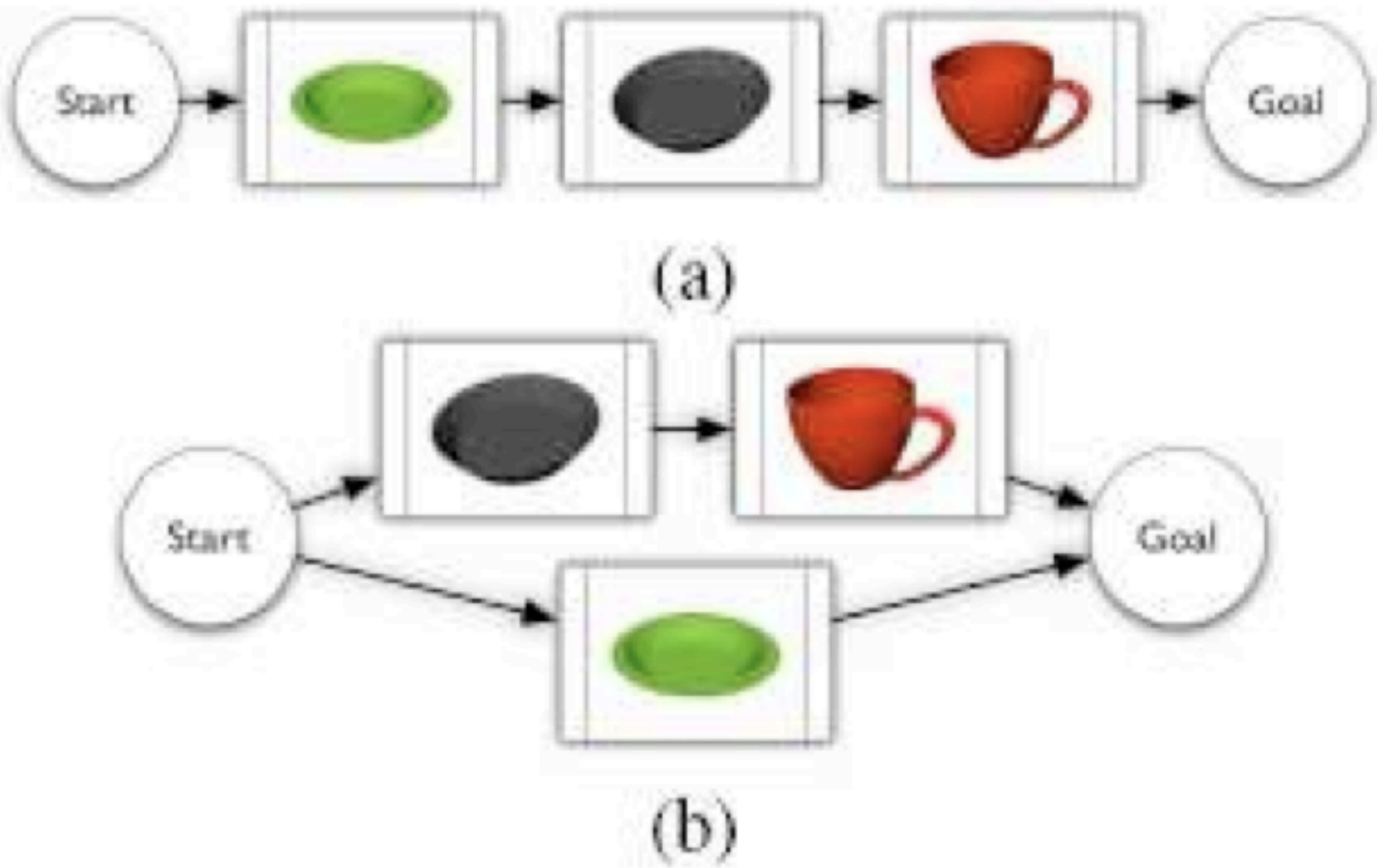


Pre-Coded Symbolic Actions



(Dillmann, *robotics and autonomous systems*, 2004)

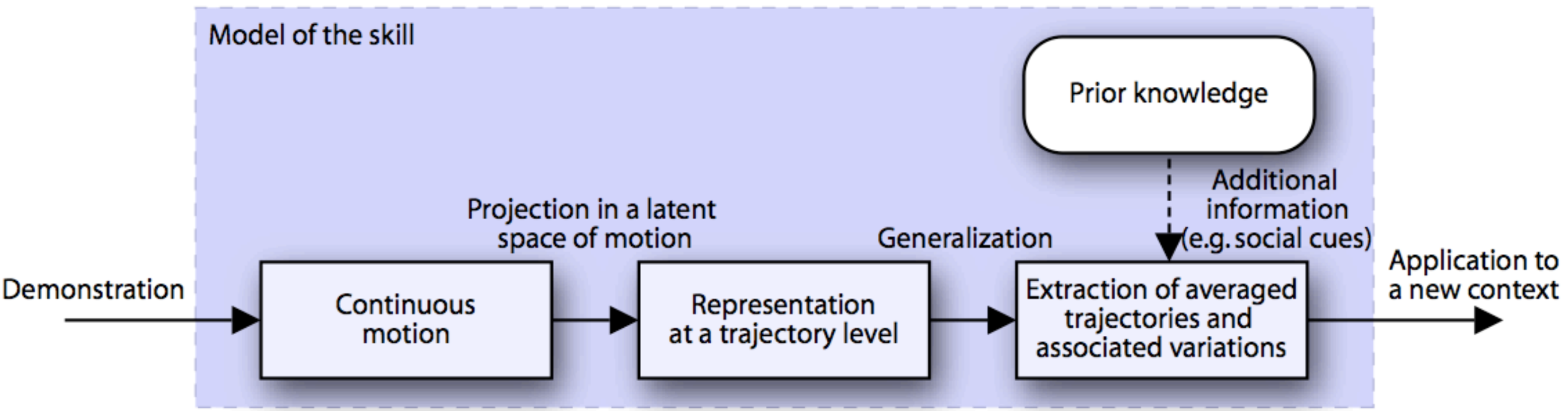
Precedence Graph: Example



(Dillmann, *robotics and autonomous systems*, 2004)

Trajectory-based Imitation Learning

Generalization at a trajectory level:



Learning how to control a helicopter

Pieter Abbeel, “Autonomous Helicopter Aerobatics through Apprenticeship Learning”, IJRR, 2010

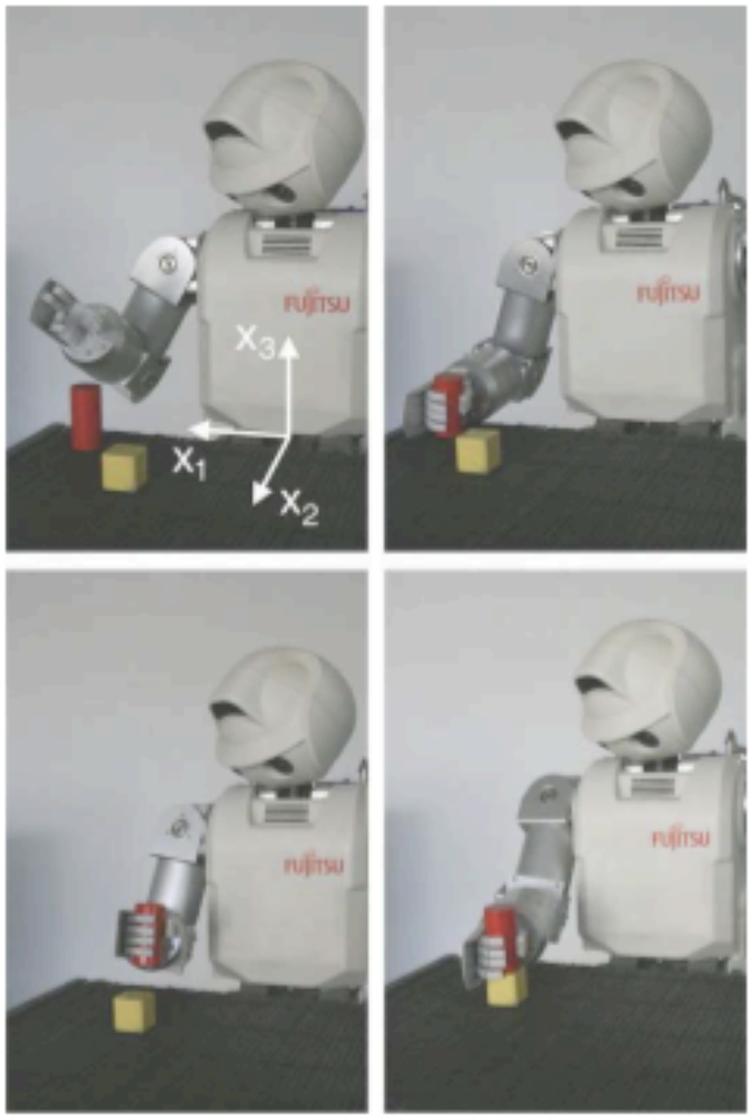
- Learn trajectories by learning from demonstration

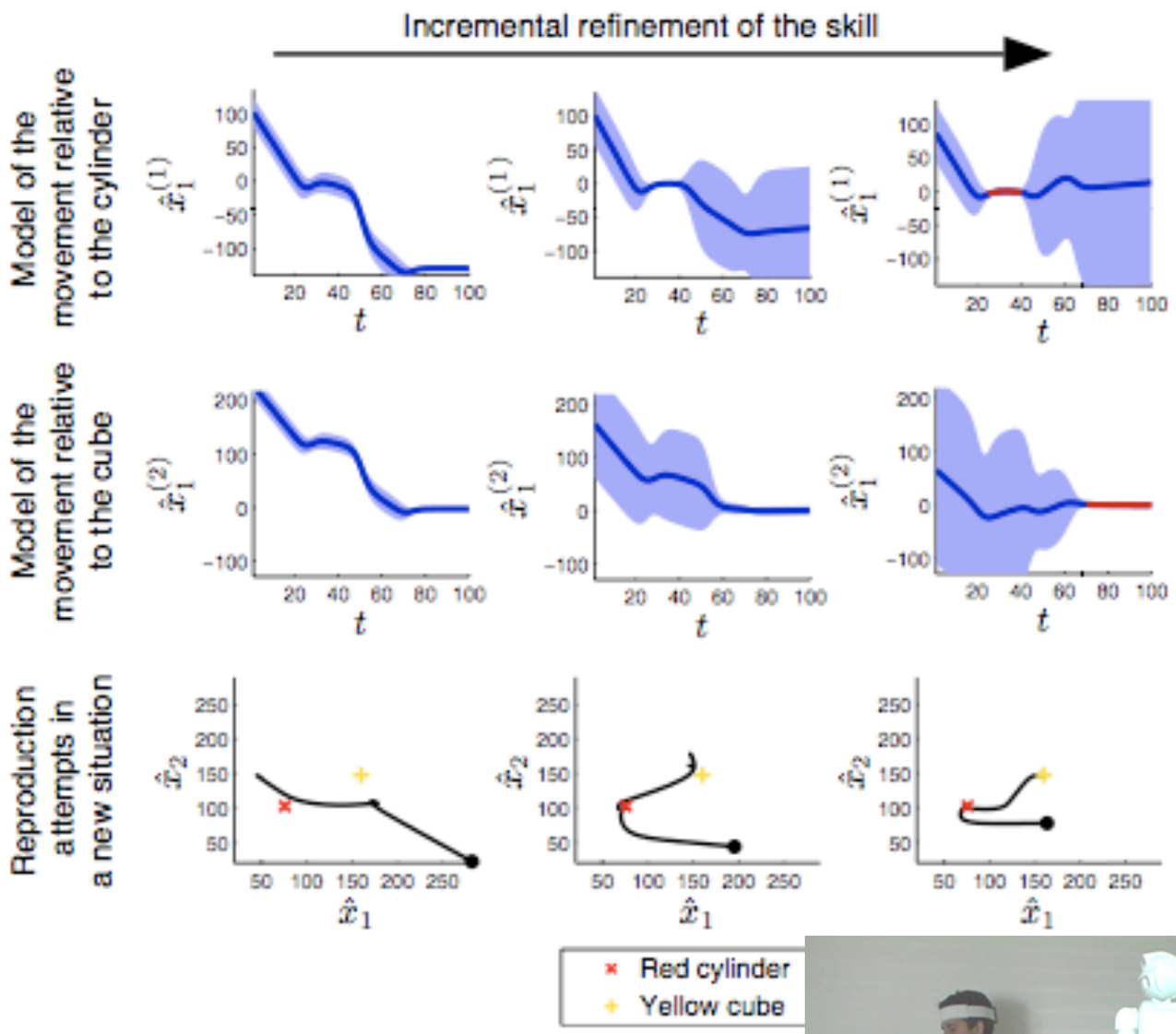
- Align the demonstration with dynamic time warping

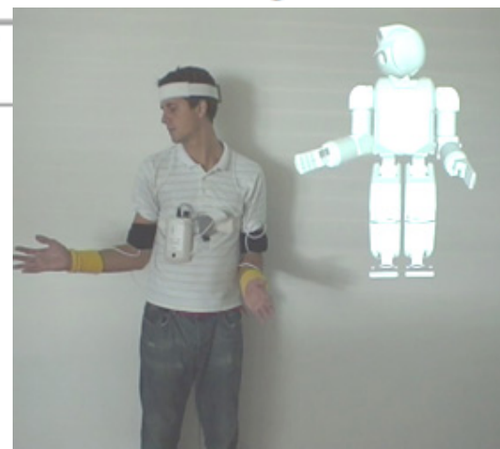
- Estimate the optimal trajectory with the Kalman smoother

- Optimal control

Density Estimation in the State-Action Space



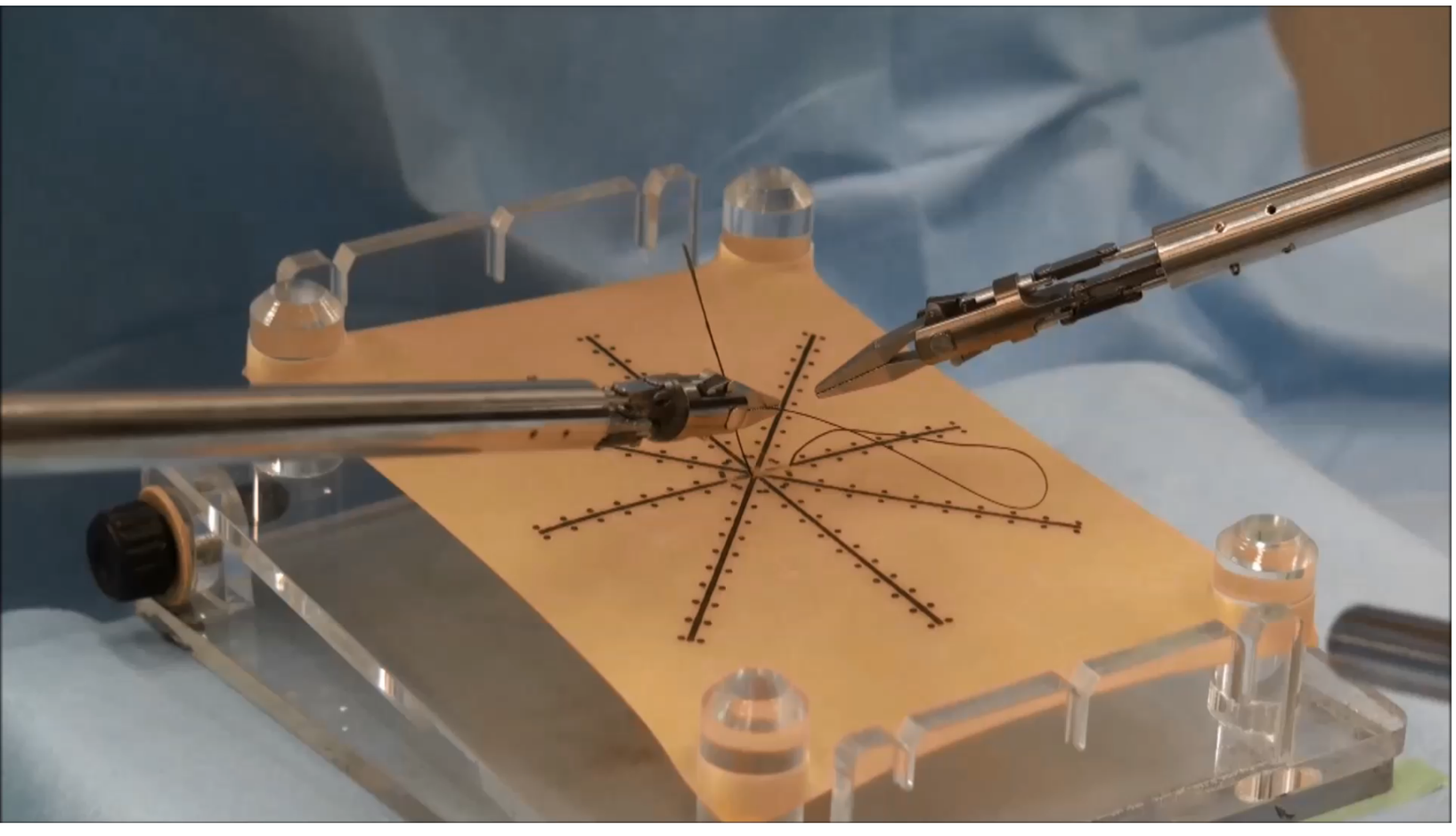




S. Calinon, “On Learning, Representing, and Generalizing a Task in a Humanoid Robot”, Trans. SMC, 2007

Modeling trajectory with Gaussian Process

T. Osa, “Online trajectory planning for surgical task automation under dynamic environment”, R:SS, 2014



Model the distribution of the trajectory as a function of context with Gaussian process

$$P\left(\xi_s(t)\middle|\xi_c\right)$$

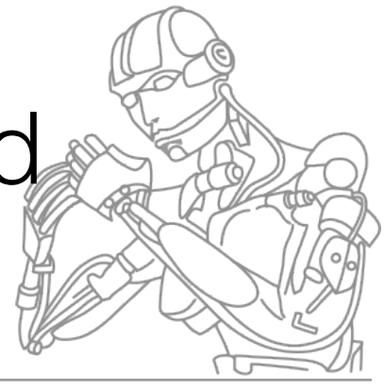
Update the trajectory in real time using the measurement from sensors

$$\hat{\xi}_s^{test}(t) = \arg\max P\left(\xi_s(t)\middle|\xi_c^{test}\right)$$

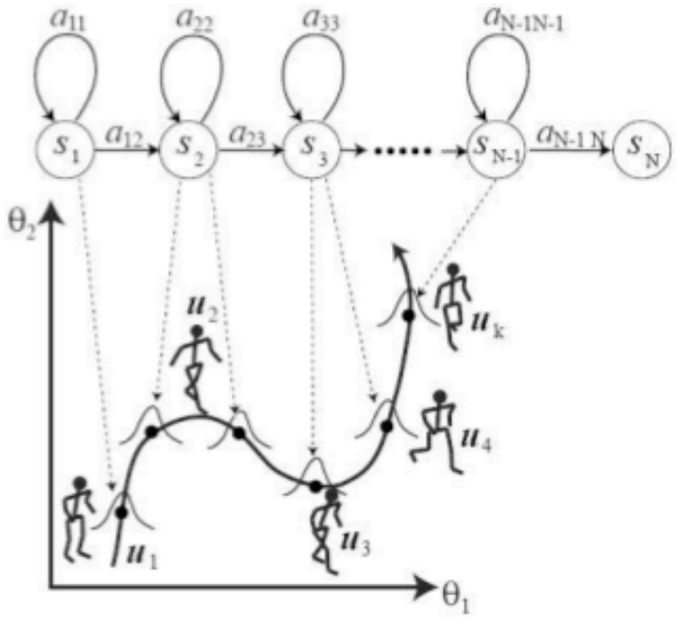
Trajectory vs Symbolic

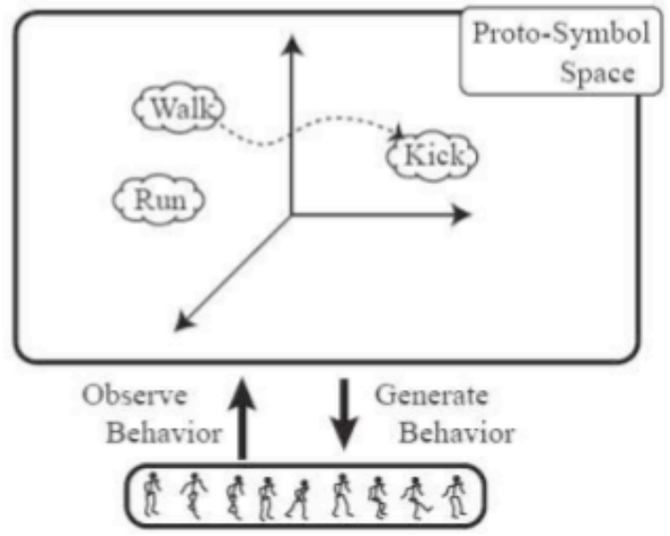
	Span of the generalization process	Advantages	Drawbacks
Symbolic level	Sequential organization of pre-defined motion elements	Allows to learn hierarchy, rules and loops	Requires to pre-define a set of basic controllers for reproduction
Trajectory level	Generalization of movements	Generic representation of motion which allows encoding of very different types of signals/gestures	Does not allow to reproduce complicated high-level skills

Combination of symbolic and trajectory-based representations



Inamura, “Embodied Symbol Emergence Based on Mimesis Theory,” IJRR, 2004

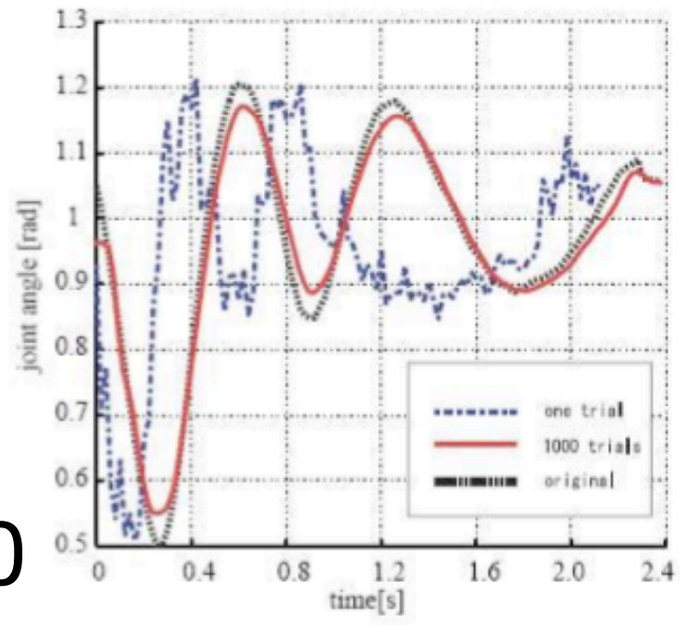


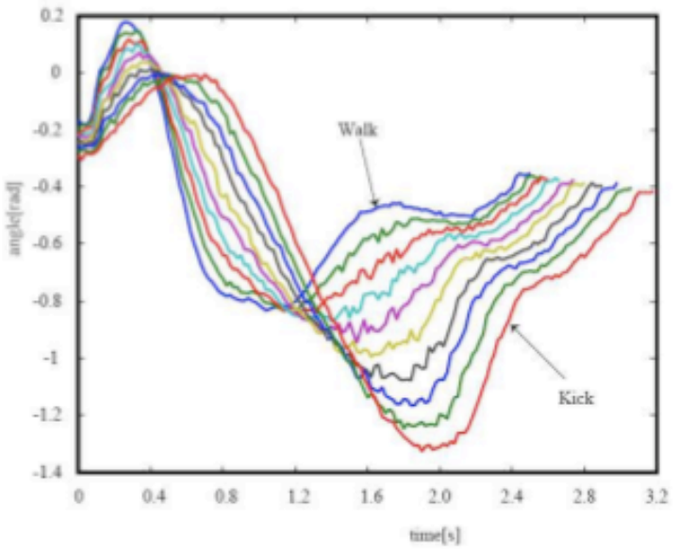


HMM is used to recognize the motions of a human

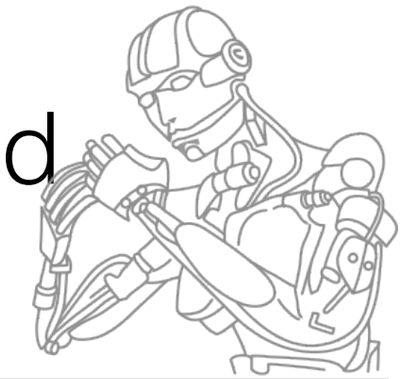
⇩

Generate trajectories for recognized motion type





Combination of symbolic and trajectory-based representations



Nakaoka, “Learning from Observation Paradigm: Leg Task Models for Enabling a Biped Humanoid Robot to Imitate Human Dances,” IJRR, 2007

Learn the dance as a combination of pre-defined motions

Find the correspondence between a humanoid and a human teacher based on a dynamics of the robot

Outline of the Lecture

1.Introduction

2.Biological Inspiration

3.Foundations

4. A Quick Overview: Well-known Approaches

5. Behavioral Cloning with Dynamical Systems

6. Behavioral Cloning with Forward Models

7. Conclusion

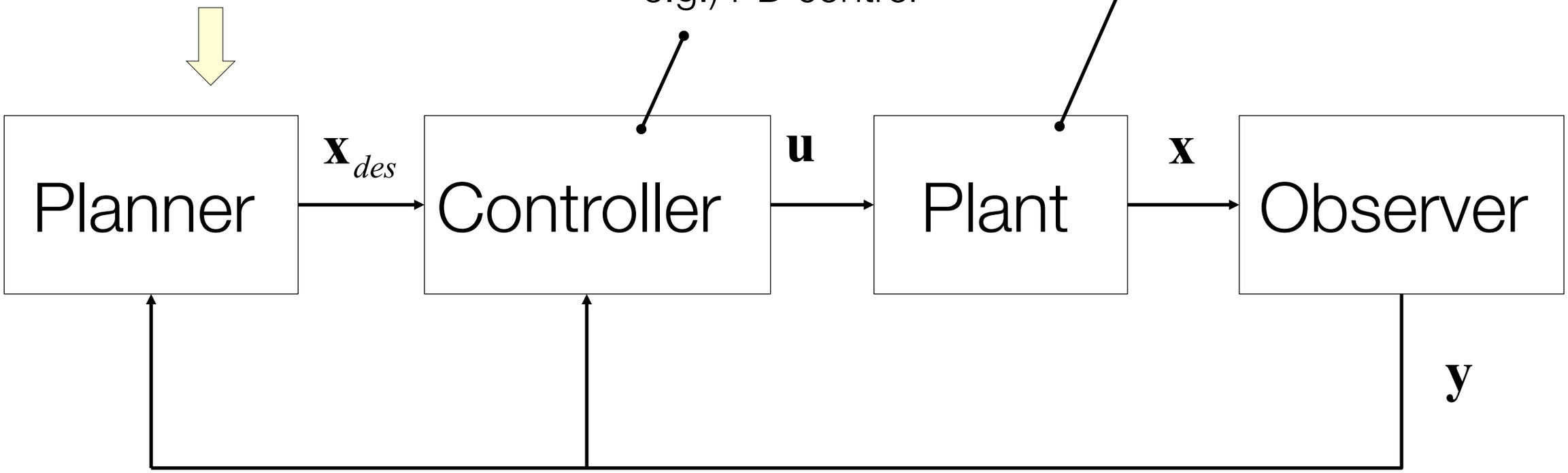
Reminder: control system in robots

We focus on this from now

e.g.) robot dynamics

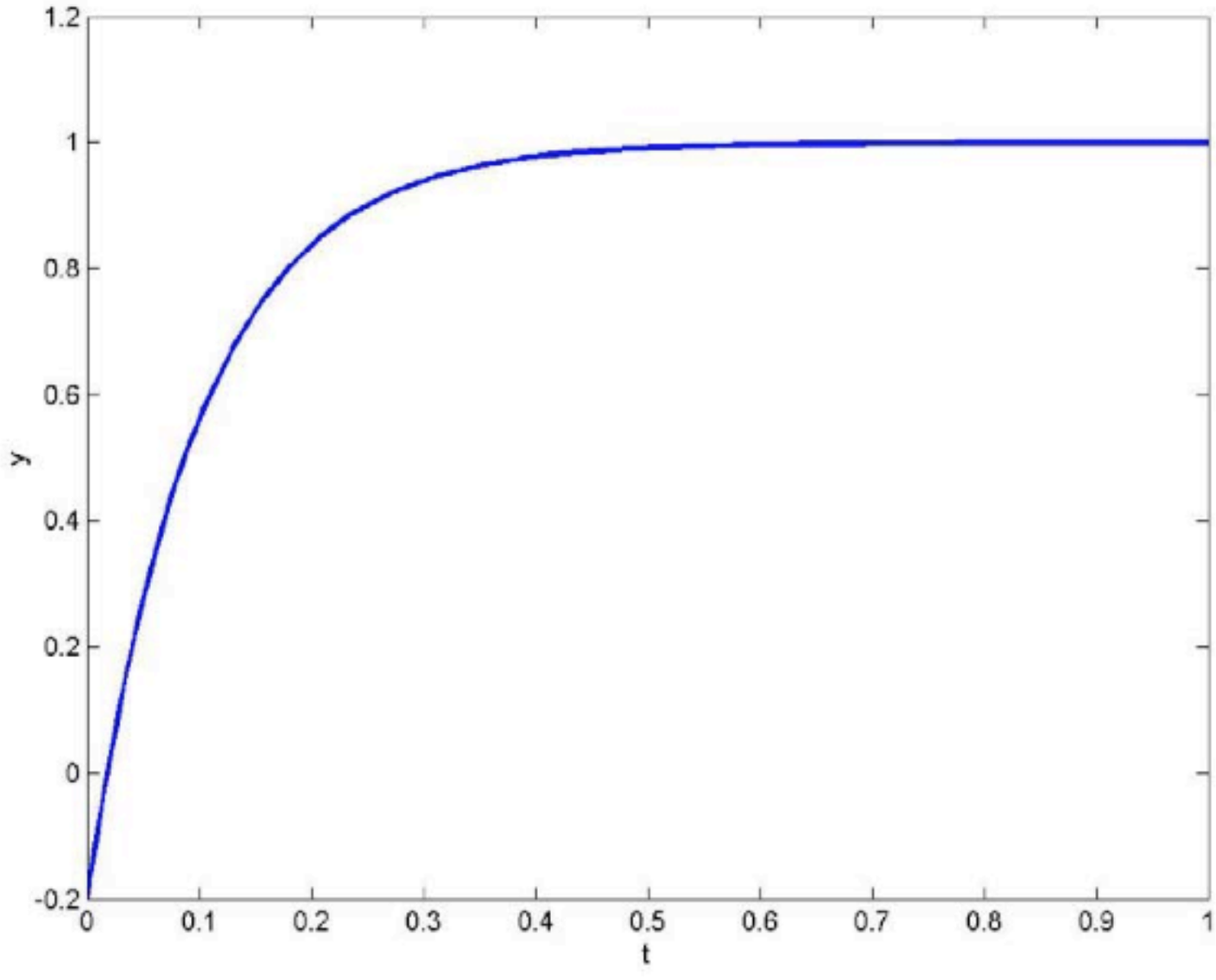
$$\mathbf{M}\ddot{\mathbf{x}} + C(\mathbf{x}, \dot{\mathbf{x}}) + g(\mathbf{x}) = \mathbf{u} + \mathbf{F}_{ext}$$

e.g.) PD control



What movement can a differential equation encode?

$$\dot{y} = \alpha(c - y)$$



What movements can a differential equation encode?

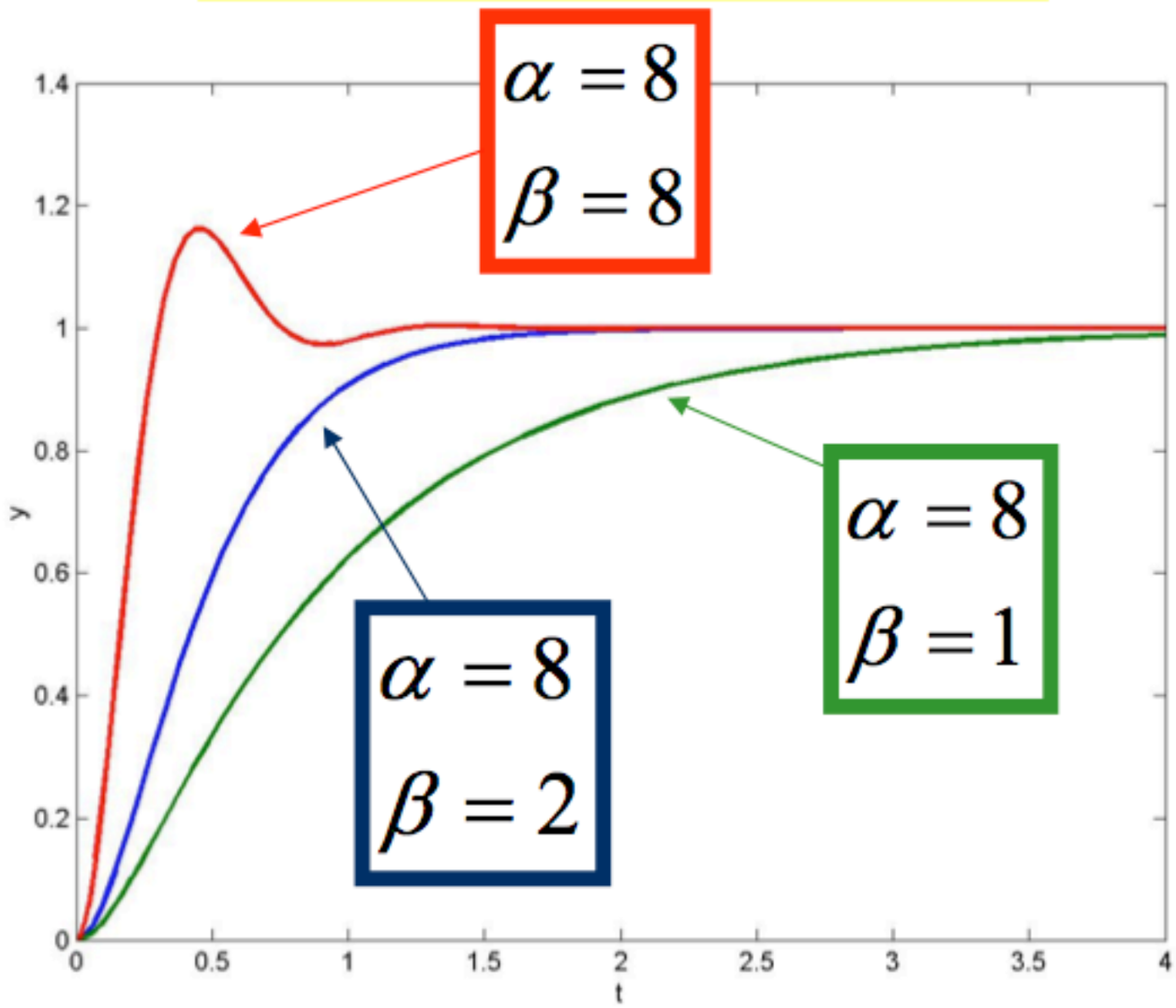
Second order linear differential equation:

$$\dot{y} = \alpha(\beta(c - x) - y)$$

$$\dot{x} = y$$

Linear differential equations:

- well-defined behavior
- But: limited class of movements



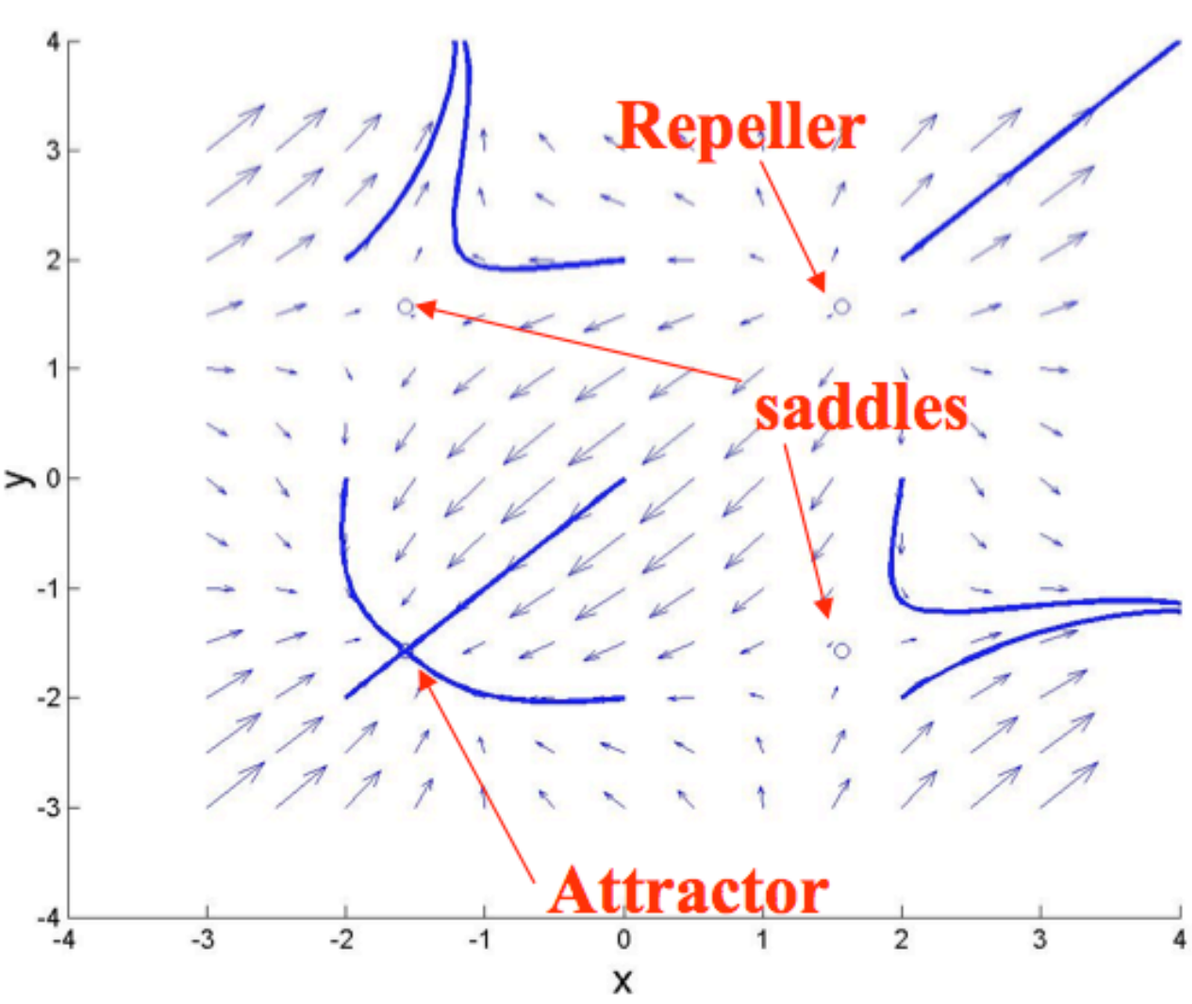
How can we make it more representative?

$$\dot{y} = -2\cos x - \cos y$$

$$\dot{x} = -2\cos y - \cos x$$

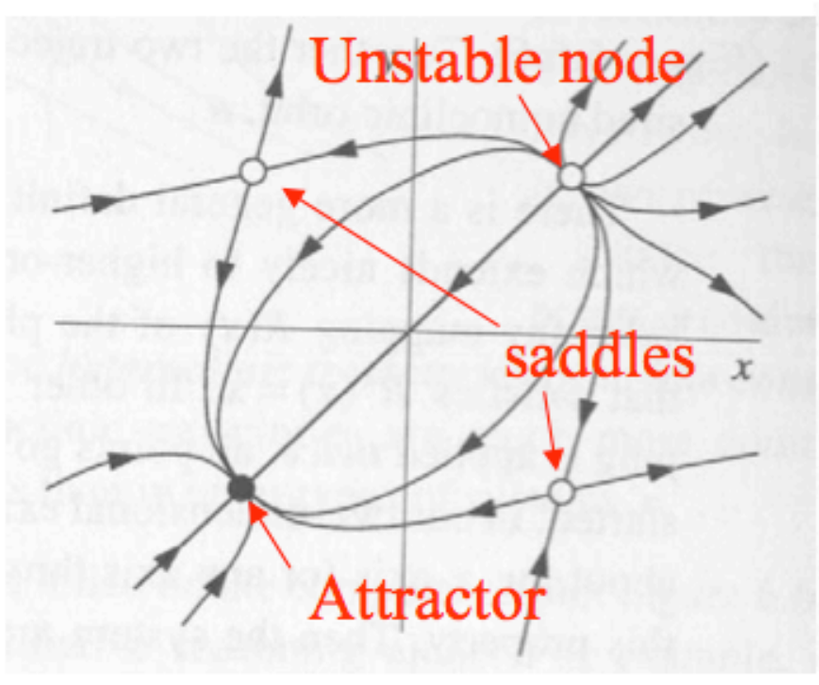
Use **non-linear differential equations** ?

- Can represent more complex behavior
- Can also get unstable! ☹

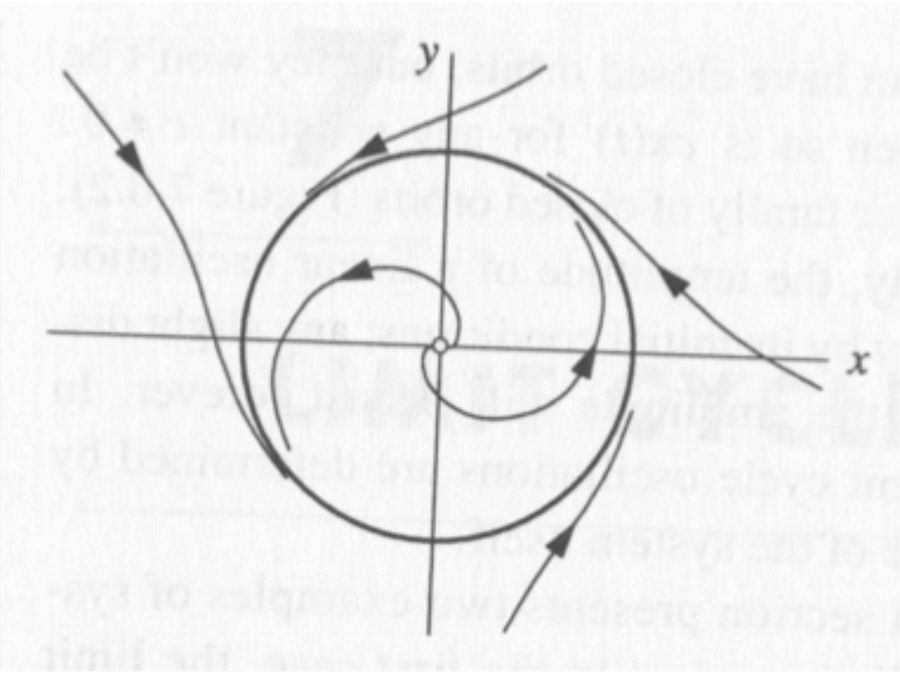


Non-linear dynamical systems

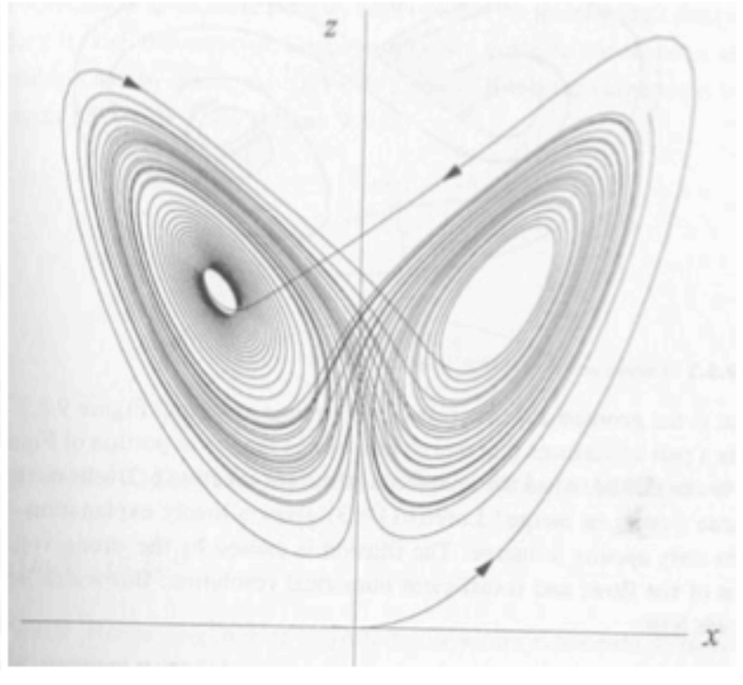
Different behaviors might emerge…



Attractors



Limit cycles

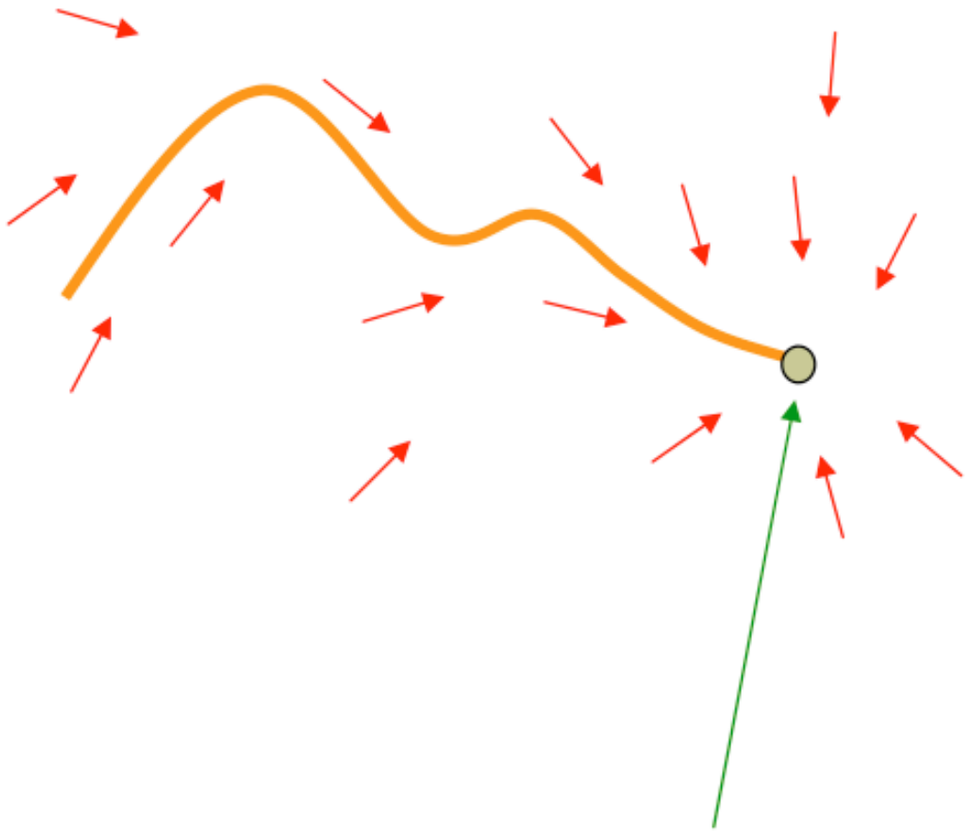


Chaos

Movements as dynamical systems

Discrete movements

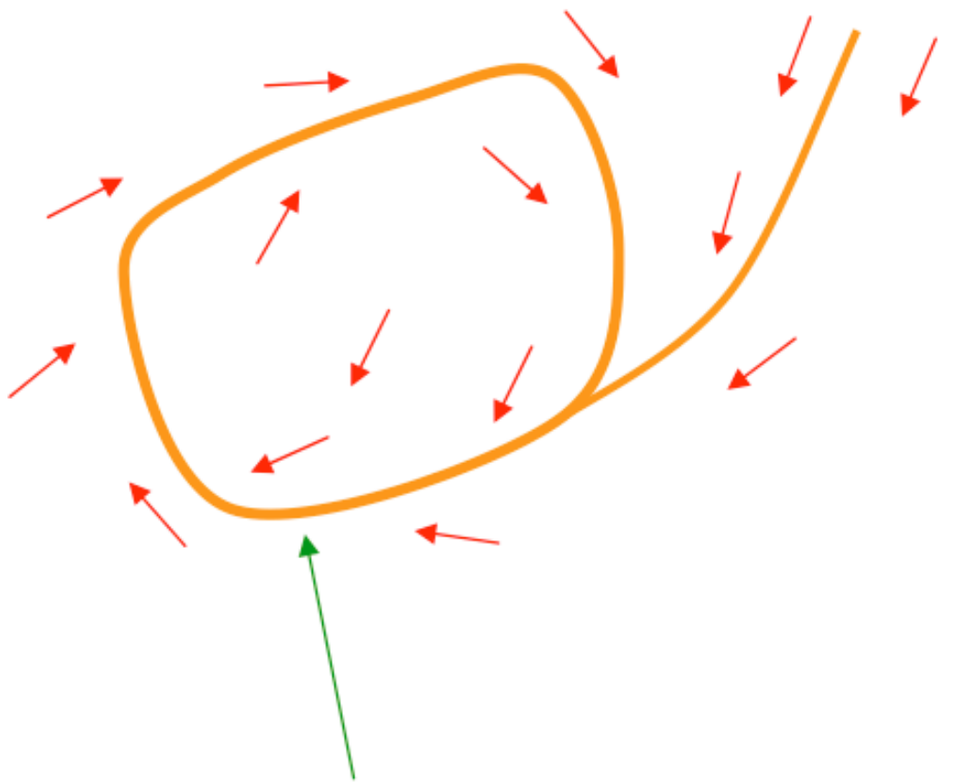
dy/dt



Single point attractor

Rhythmic movements

dy/dt



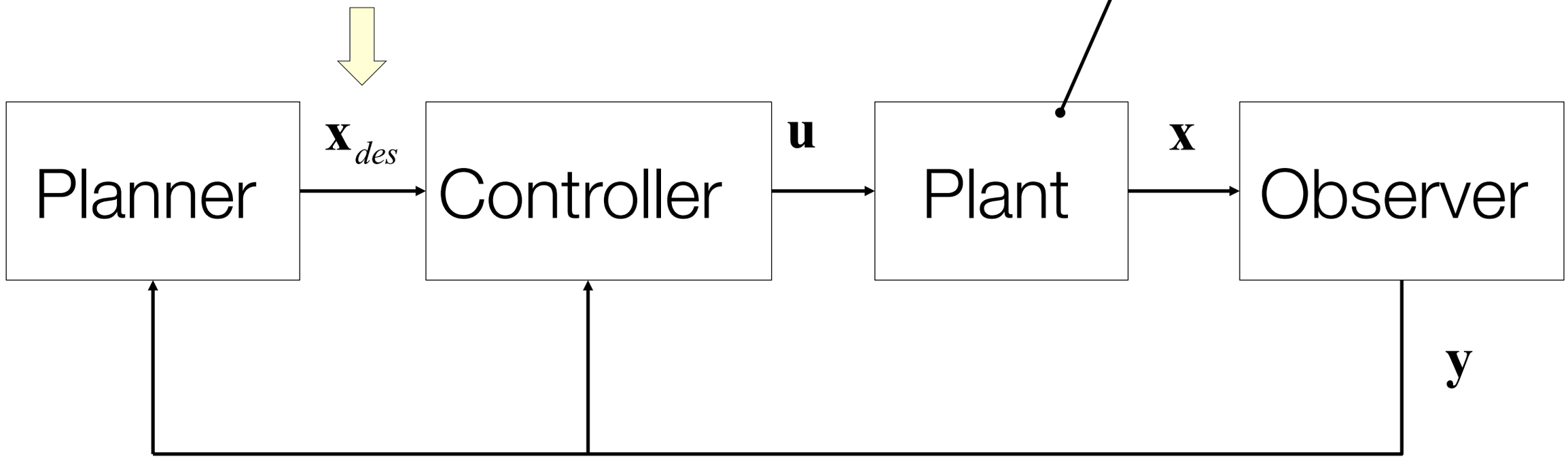
Limit cycle attractor

Reminder: control system in robots

$\mathbf{x}_{des}$ should be a reasonable choice for the system dynamics

e.g.) robot dynamics

$$\mathbf{M}\ddot{\mathbf{x}} + C(\mathbf{x}, \dot{\mathbf{x}}) + g(\mathbf{x}) = \mathbf{u} + \mathbf{F}_{ext}$$



A trajectory described as stable dynamical can be a natural movement for a robotic system

⇨ Can we describe the desired trajectory as a dynamical systems?

Motor Primitives as Dynamic Systems?

The essence of **tasks** in **control policies**

$$\mathbf{u} = \pi(\mathbf{x}, \theta)$$

can be represented through **dynamic systems**

$$\tau\dot{\mathbf{x}} = f(\mathbf{x}, \theta)$$

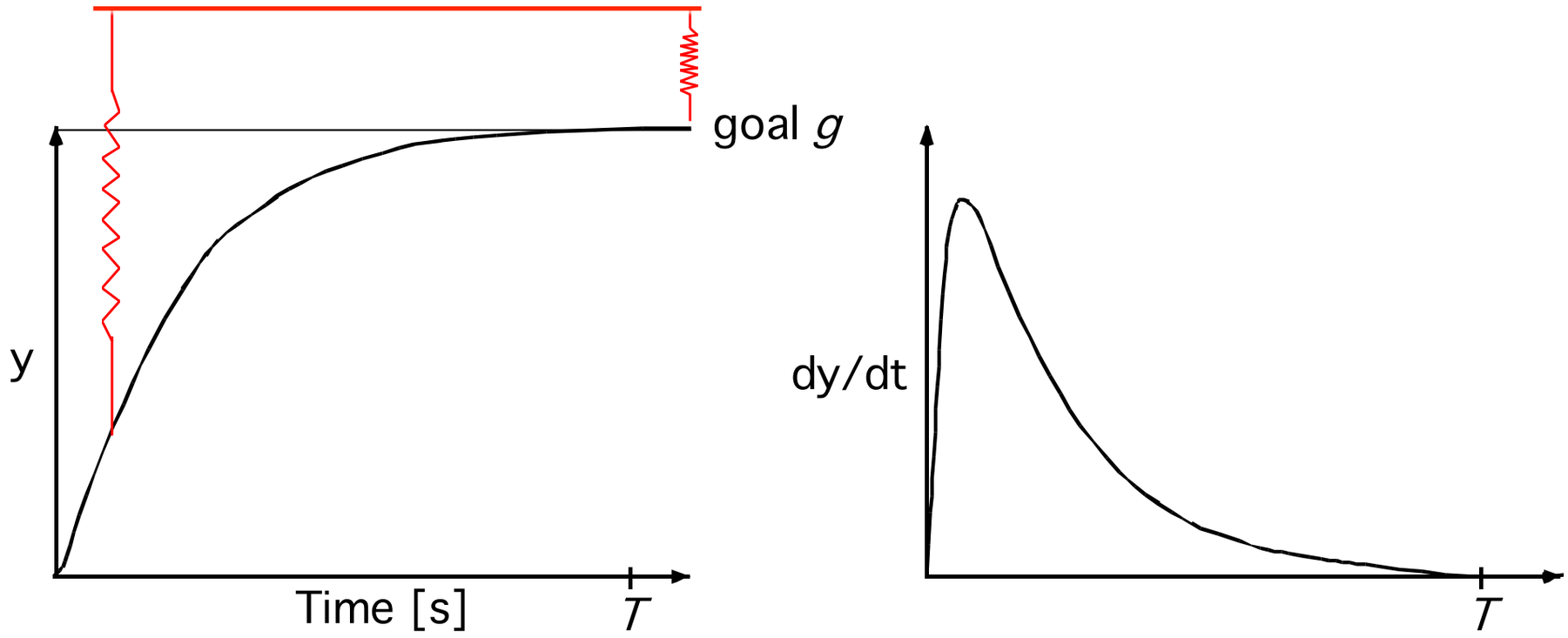
with desirable properties such as

- stability,
- scaling invariance,
- perturbation robustness,
- periodic and point-to-point behaviors.

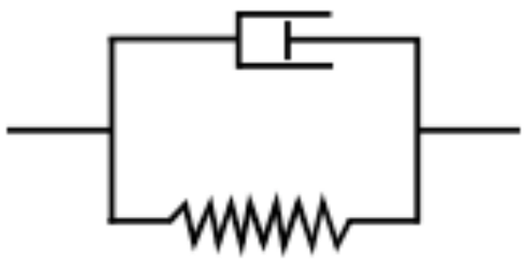
⇨ Dynamic Movement Primitives

(Ijspeert, Nakanishi & Schaal, NIPS 2003; Schaal, Peters, Nakanishi, Ijspeert, ISRR 2003)

Point-to-Point Movements as Dynamical Systems



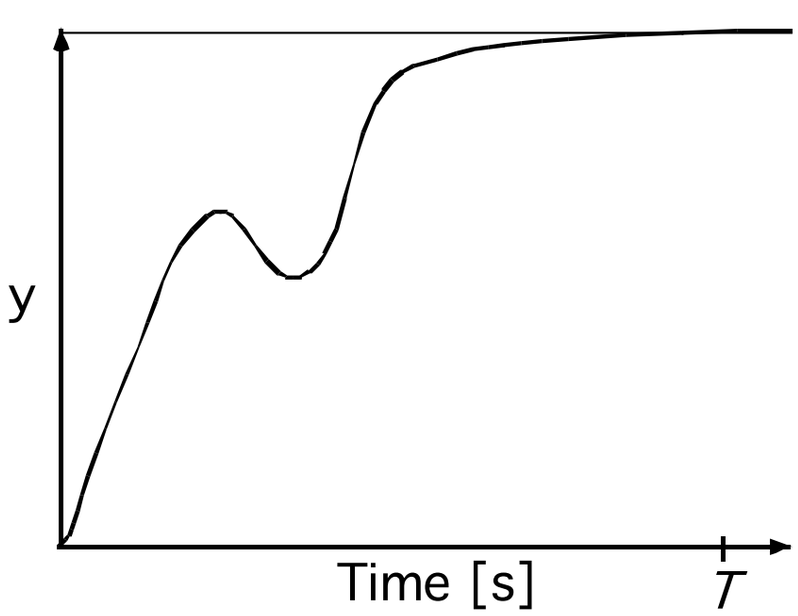
E.g., for a one degree-of-freedom movement, start with a simple damped spring model

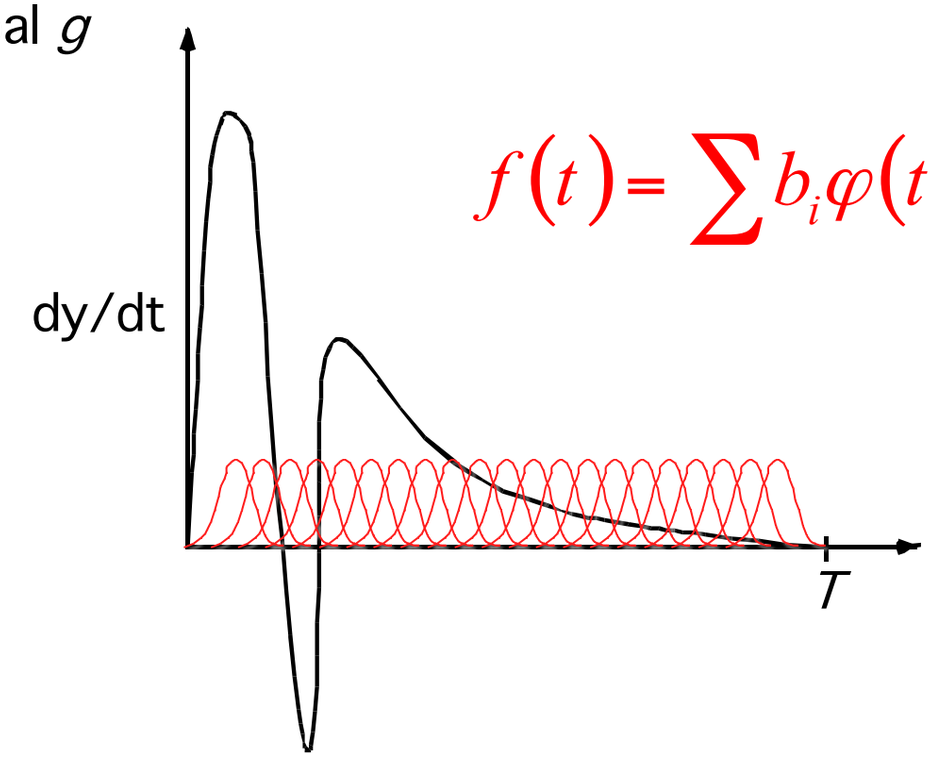


$$\dot{z} = \alpha_z\left(\beta_z(g - y) - z\right)$$
$$\dot{y} = z$$

(Ijspeert, Nakanishi & Schaal, NIPS 2003; Schaal, Peters, Nakanishi, Ijspeert, ISRR 2003)

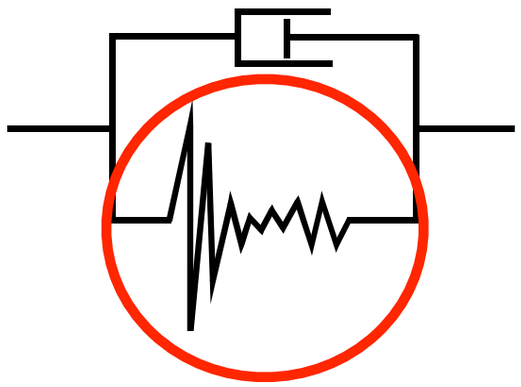
Point-to-Point Movements as Dynamical Systems





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

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



$$\dot{z} = \alpha_z\left(\beta_z(g - y) - z\right)$$

$$\dot{y} = \alpha_y\left(f(?) + z\right)$$

(Iispeert, Nakanishi & Schaal, NIPS 2003; Schaal, Peters, Nakanishi, Iispeert, ISRR 2003)

Dynamic Movement Primitives
- Representation of the forcing function -

How to represent f?

- Normalized RBF basis functions

$$\phi_i(z) = \exp(-0.5(z - c_i)^2/h_i)$$

$$f_{\boldsymbol{w}}(z) = \frac{\sum_{i=1}^{K} \phi_i(z) w_i z}{\sum_{j=1}^{K} \phi_j(z)}$$

- Matrix Form:

Learnable weights

$$f_{\boldsymbol{w}}(z) = \boldsymbol{\psi}^T(z)\boldsymbol{w}, \text{with } \psi_i(z) = \frac{\phi_i(z) z}{\sum_{j=1}^{K} \phi_j(z)}$$

- For $t \to \infty, f_{\boldsymbol{w}}(z) \to 0$ as $z \to 0$

A DMP is stable per construction as the forcing function vanishes ➡ it is just a standard PD for $t \to \infty$

Dynamic Movement Primitives
- Temporal scaling -

How can we encode a temporal scaling?

Introduce a phase variable z_t to replace time

$$\ddot{y} = \tau^2 \alpha(\beta(g - y) - \dot{y}/\tau) + \tau^2 f_{\boldsymbol{w}}(z)$$

$$\dot{z} = -\tau \alpha_z z$$

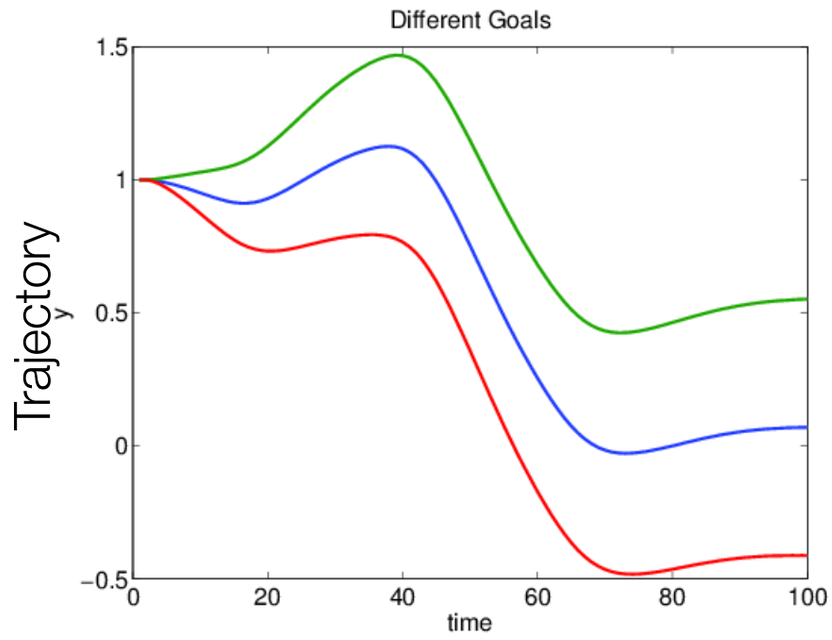
Also uses dynamical system to model phase z

τ ... temporal scaling variable

(Ijspeert, Nakanishi & Schaal, NIPS 2003; Schaal, Peters, Nakanishi, Ijspeert, ISRR 2003)

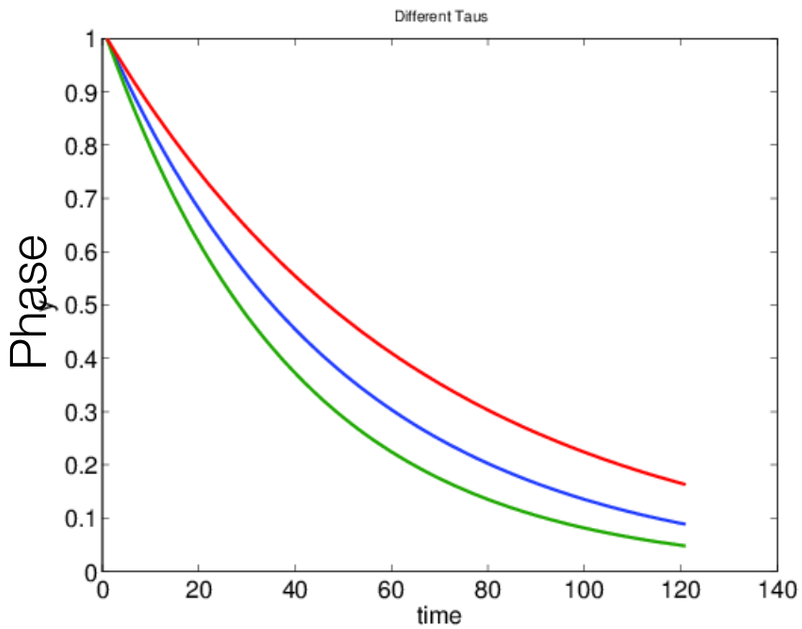
Dynamic Movement Primitives
- Adapting the meta-parameters -

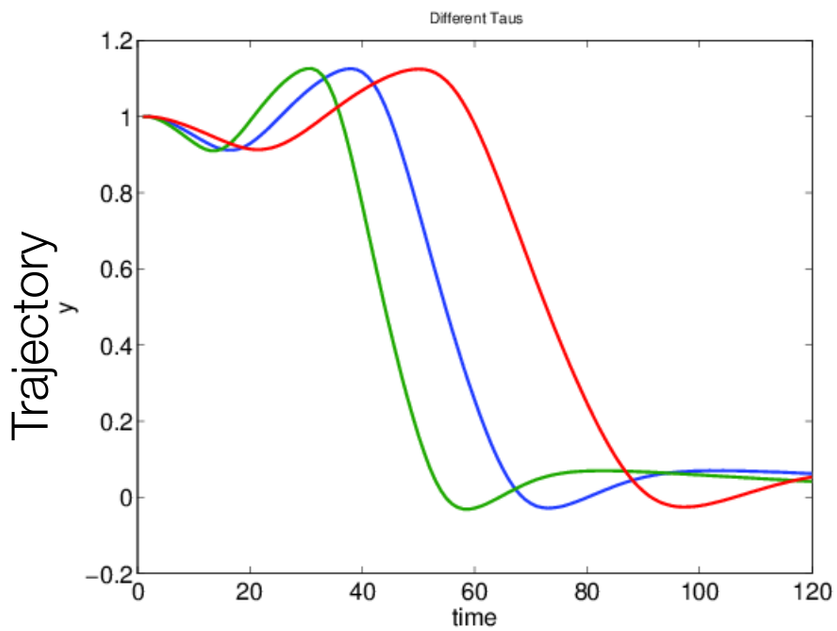
Adapting the goal attractor g



⇨ Changes final position

Adapting the temporal scaling τ





Imitation Learning with DMPs

Learn:

$$\ddot{y} = \tau^2 \alpha(\beta(g - y) - \dot{y}/\tau) + \tau^2 f_{\boldsymbol{w}}(z)$$

$$\dot{z} = -\tau \alpha_z z$$

Given:

- A desired trajectory and its derivatives $\boldsymbol{q}_{1:T}, \dot{\boldsymbol{q}}_{1:T}, \ddot{\boldsymbol{q}}_{1:T}$
- A goal attractor g (e.g. final position of trajectory)
- Parameters: α, β, α_z (typically fixed)
- Temporal Scaling τ: Adjusted to movement duration

Algorithm:

- Compute target values for each time step
  $$f_t = \ddot{q}_t/\tau^2 - \alpha(\beta(g - q_t) - \dot{q}/\tau)$$
- Compute shape parameters $\boldsymbol{w}$ by linear (ridge) regression
  $$\boldsymbol{w} = (\boldsymbol{\Psi}^T \boldsymbol{\Psi} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{\Psi}^T \boldsymbol{f}$$

(Ijspeert, Nakanishi & Schaal, NIPS 2003; Schaal, Peters, Nakanishi, Ijspeert, ISRR 2003)

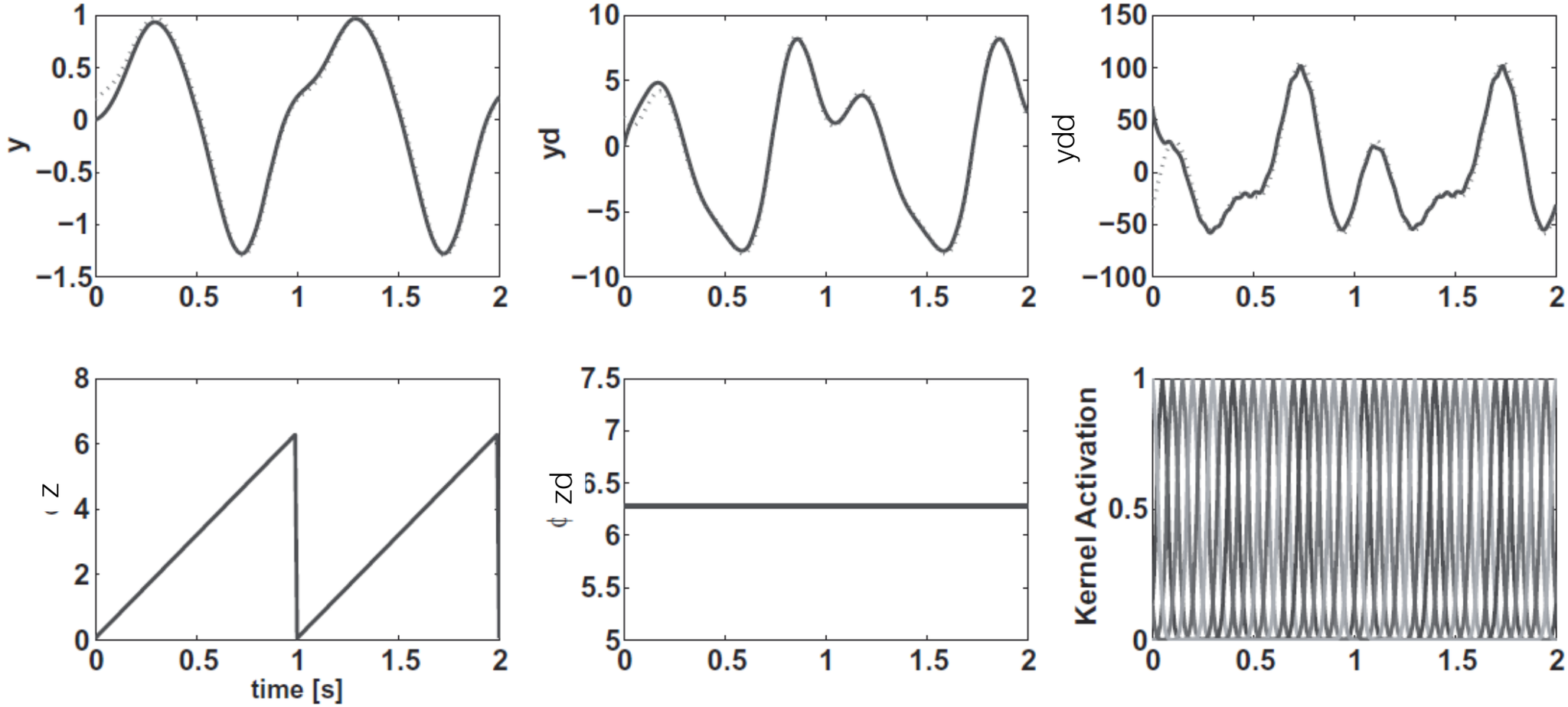
Dynamic Movement Primitives

For multi-DoF robots, we use an individual DMP per DoF

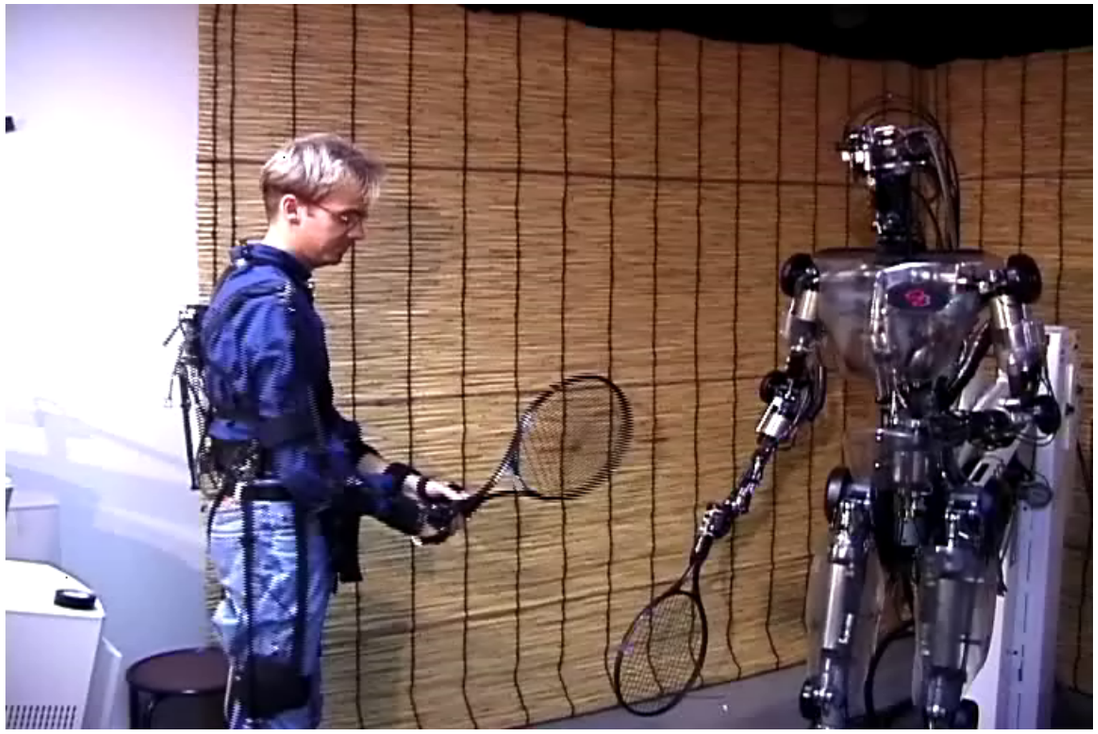
Phase variable z is shared

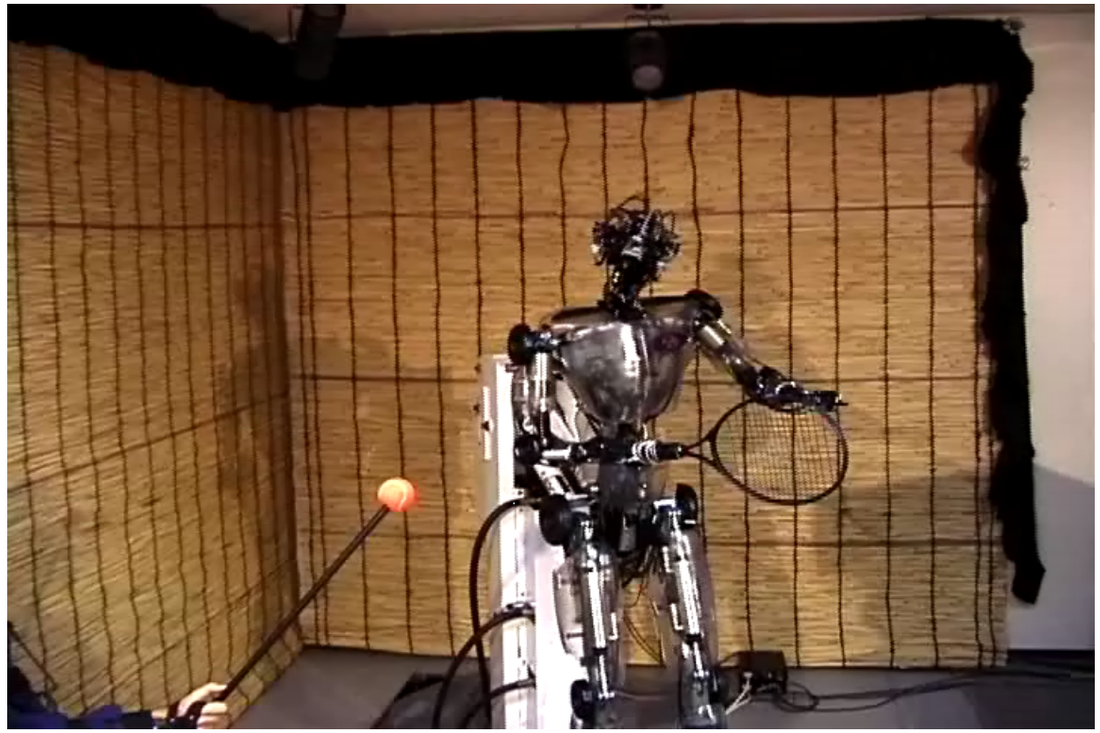
Coupling between joints due to the shared phase

For periodic movements, we can use periodic phase variables



Example: A Tennis Backhand

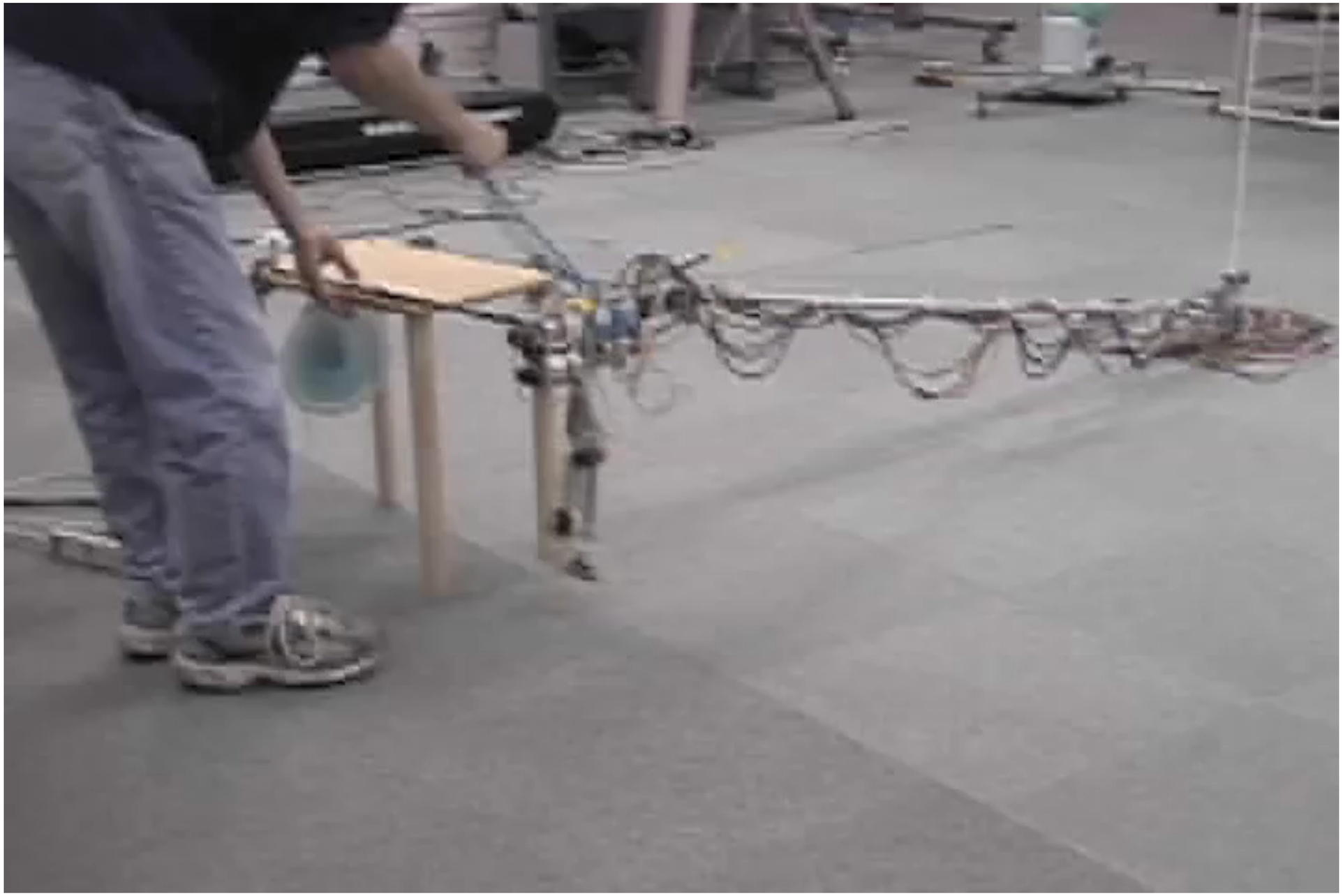




Rhythmic Motor Primitives

(Kober & Peters, ICRA 2009)

Fast Coupling between System and Gait

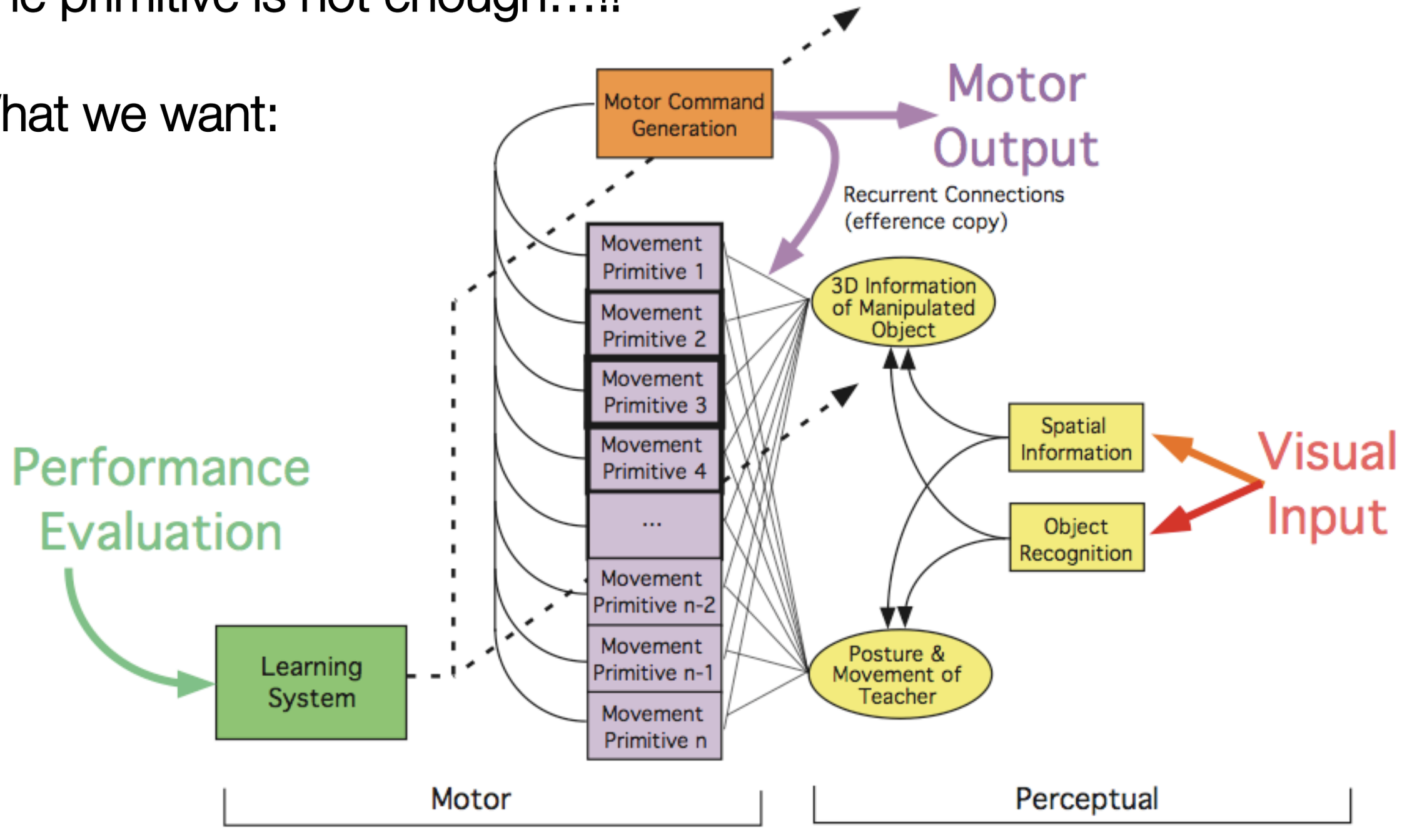


Is one primitive enough?

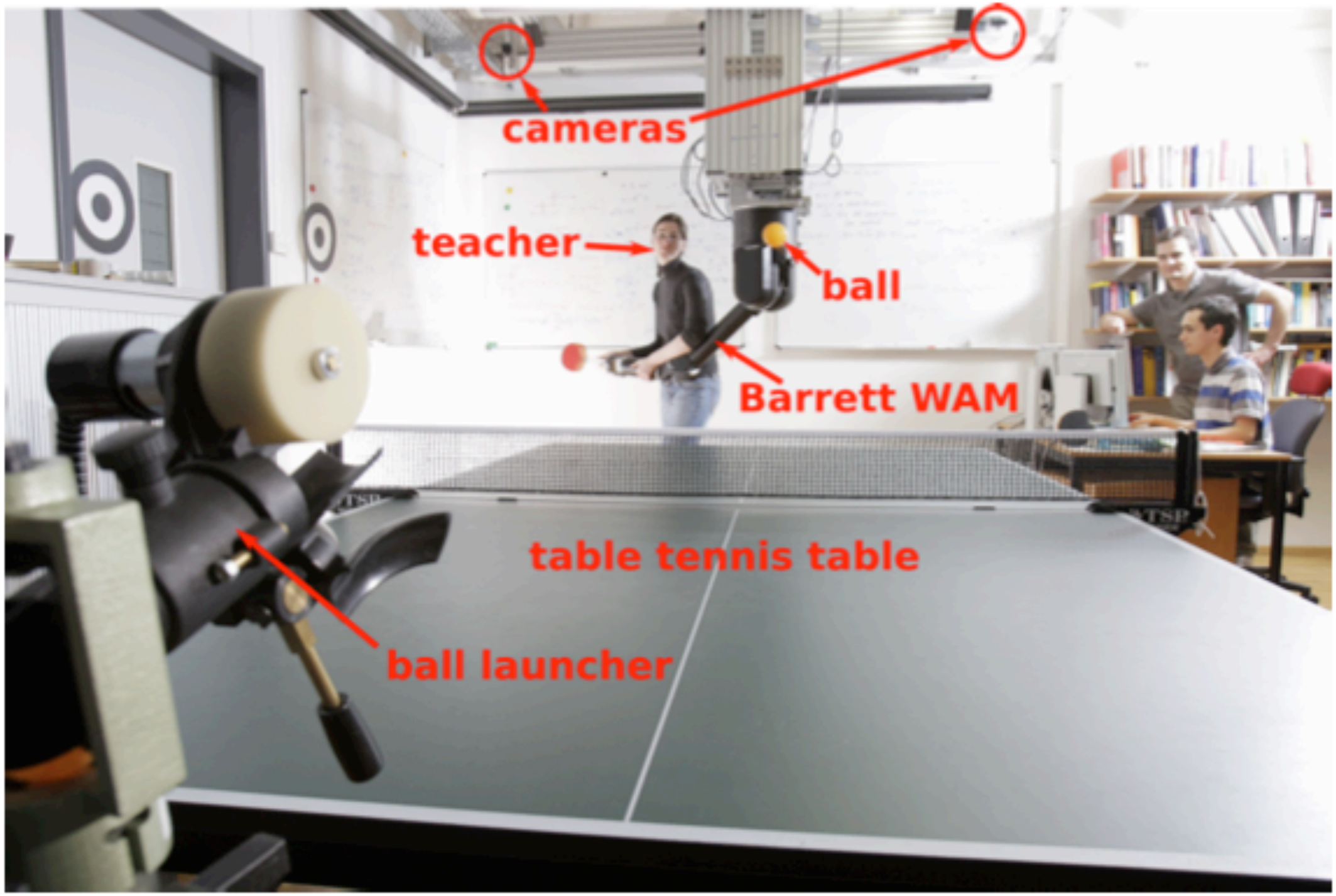
Libraries of Primitives

One primitive is not enough…!!

What we want:



Imagine the following situation...



What about many primitives?

Forehand 1

Forehand 2

Smash

Backhand 1

Incoming Ball?

Own position?

Opponent's Movement?

Prior Opponent Game Play?

Gating Network

+

Desired Behavior

What you can do with it...

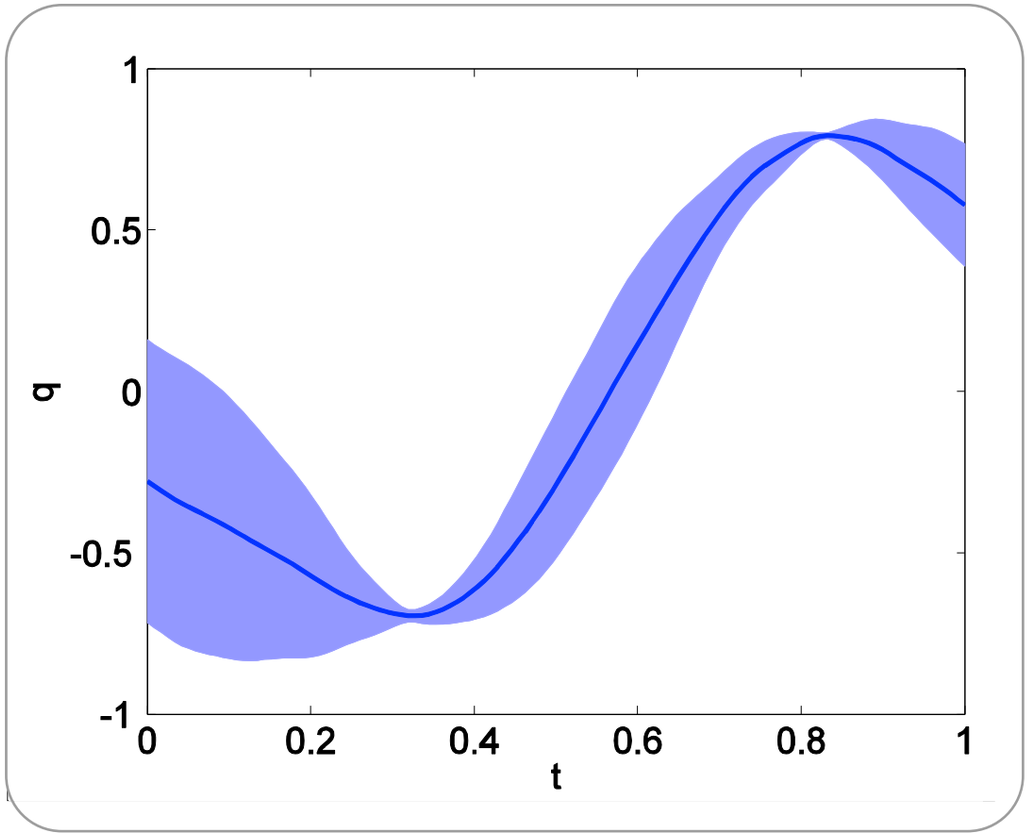
Probabilistic Movement Primitives

Stochastic representation of trajectories:

Use $\boldsymbol{w}$ to represent a single trajectory

$$y_t = \boldsymbol{\psi}_t^T \boldsymbol{w} + \epsilon_y \qquad \epsilon_y \sim \mathcal{N}(0, \sigma^2)$$

Learn a distribution $P(\omega)$



Why is this useful?

- We can also represent uncertainty
- Uncertainty gives us information on importance of time points
- We can apply probabilistic operations

(Parachos, NIPS 2013)

Outline of the Lecture

1.Introduction

2.Biological Inspiration

3.Foundations

4. A Quick Overview: Well-known Approaches

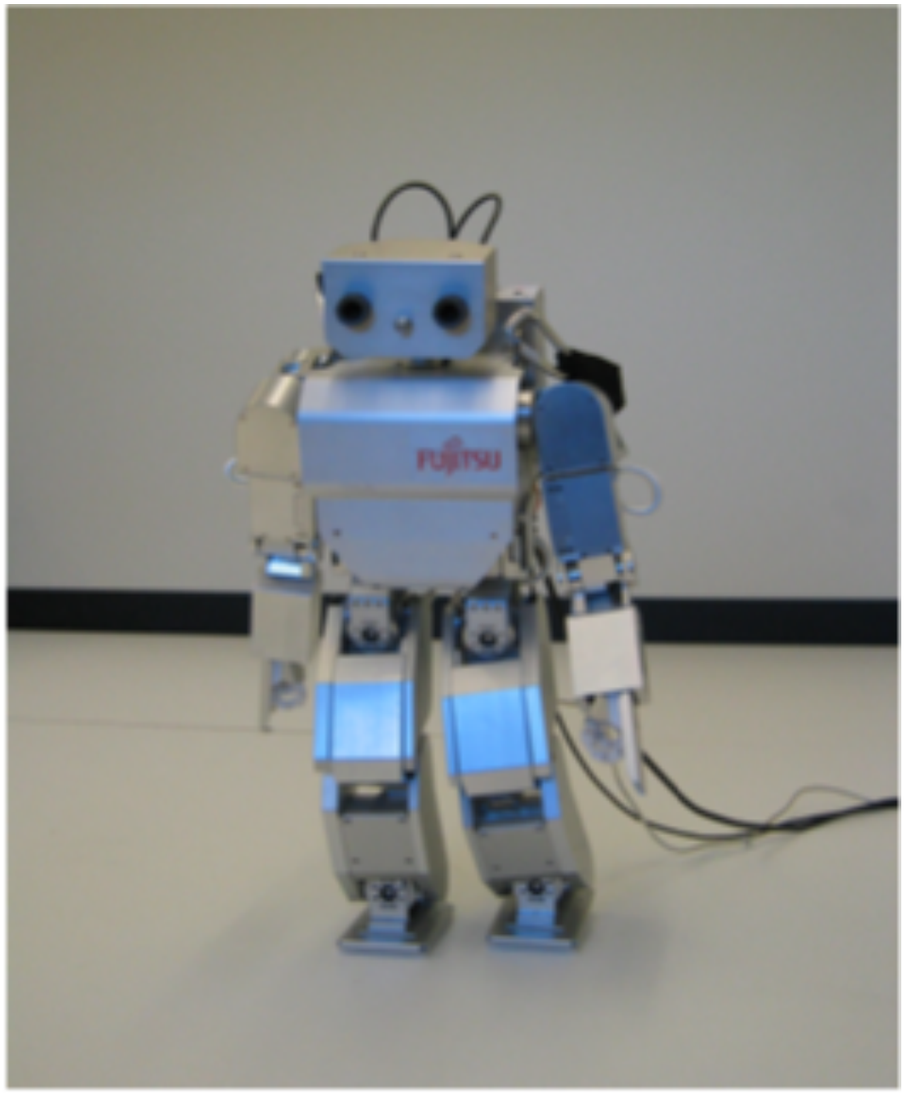
5. Behavioral Cloning with Dynamical Systems

6. Behavioral Cloning with Forward Models

7. Conclusion

Learning motion with a Forward Model

David B. Grimes and Rajesh P.N. Rao, “Learning Actions through Imitation and Exploration: Towards Humanoid Robots That Learn from Humans”,2009

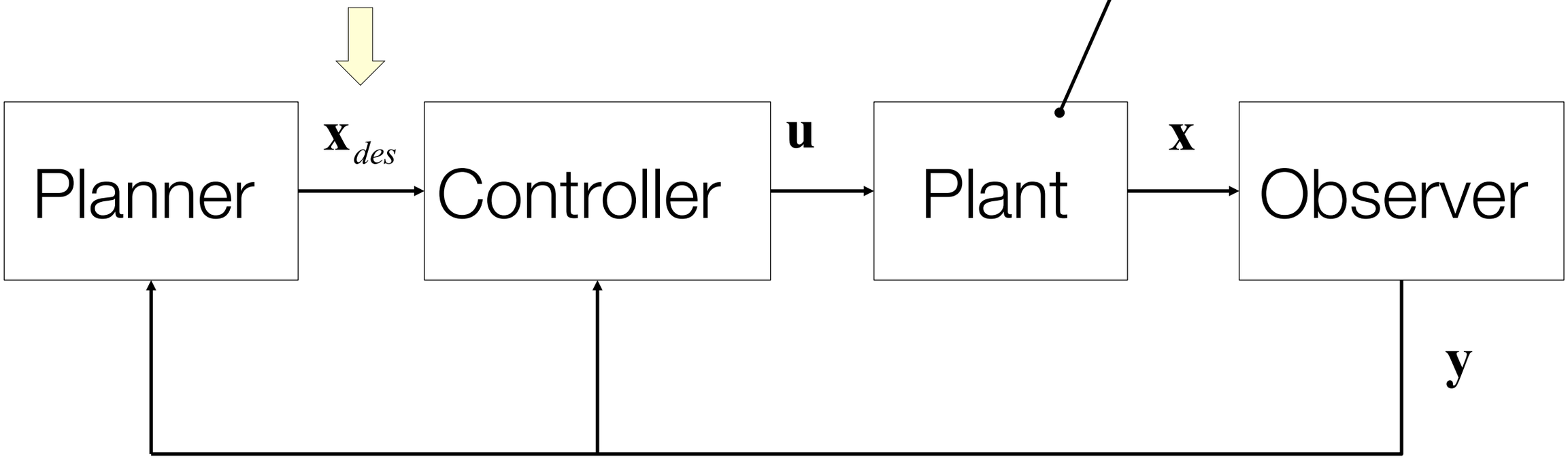


Reminder: control system in robots

$\mathbf{x}_{des}$ should be a reasonable choice for the system dynamics

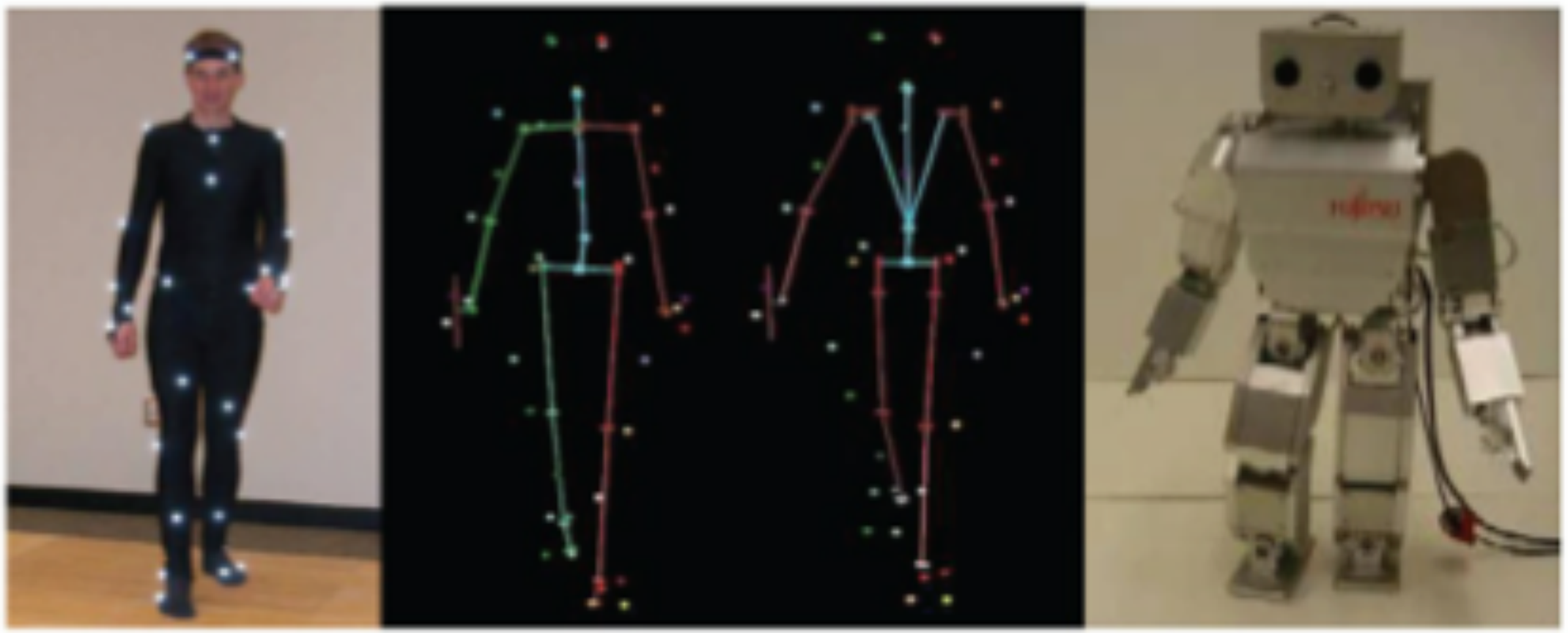
e.g.) robot dynamics

$$\mathbf{M}\ddot{\mathbf{x}} + C(\mathbf{x}, \dot{\mathbf{x}}) + g(\mathbf{x}) = \mathbf{u} + \mathbf{F}_{ext}$$



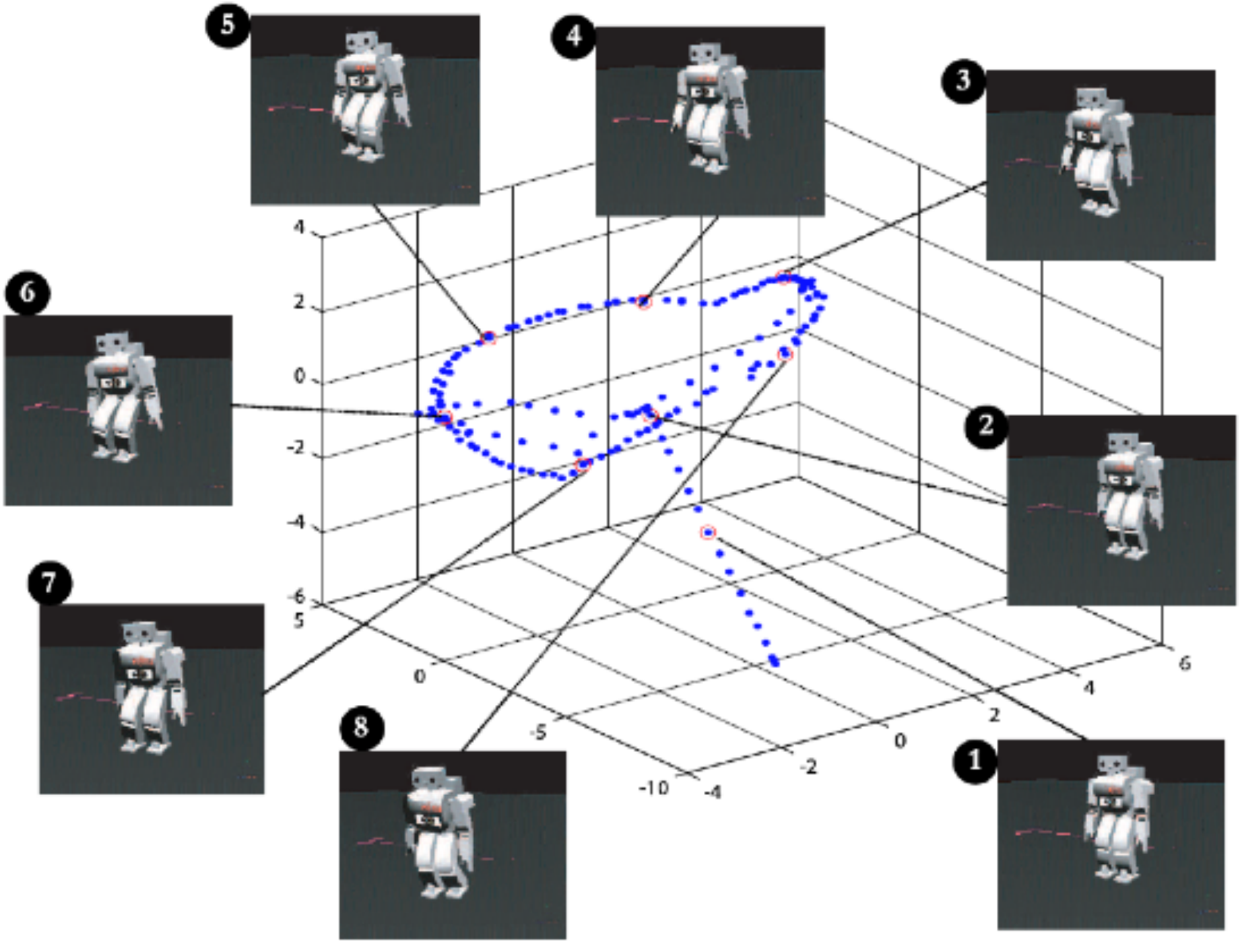
Demonstrated trajectory is suitable for the teacher, but it is not always suitable for a student robot…

1. Find a Rough Kinematic Mapping

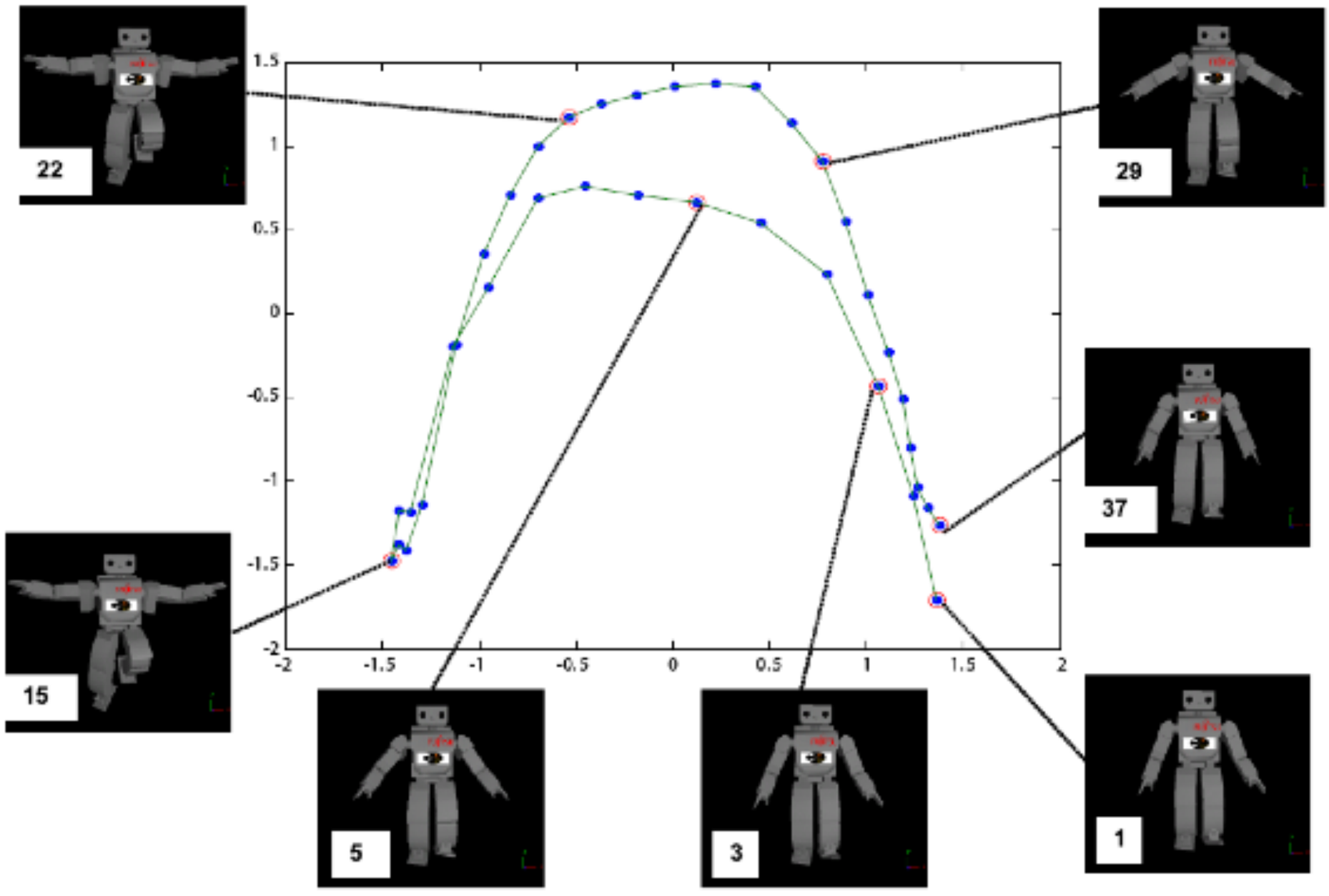


- Need to solve the “correspondence problem” Solved by assuming markers are on scaled version of robot body Standard inverse kinematics recovers joint angles for motion

2. Step: Eigenposture for Walking



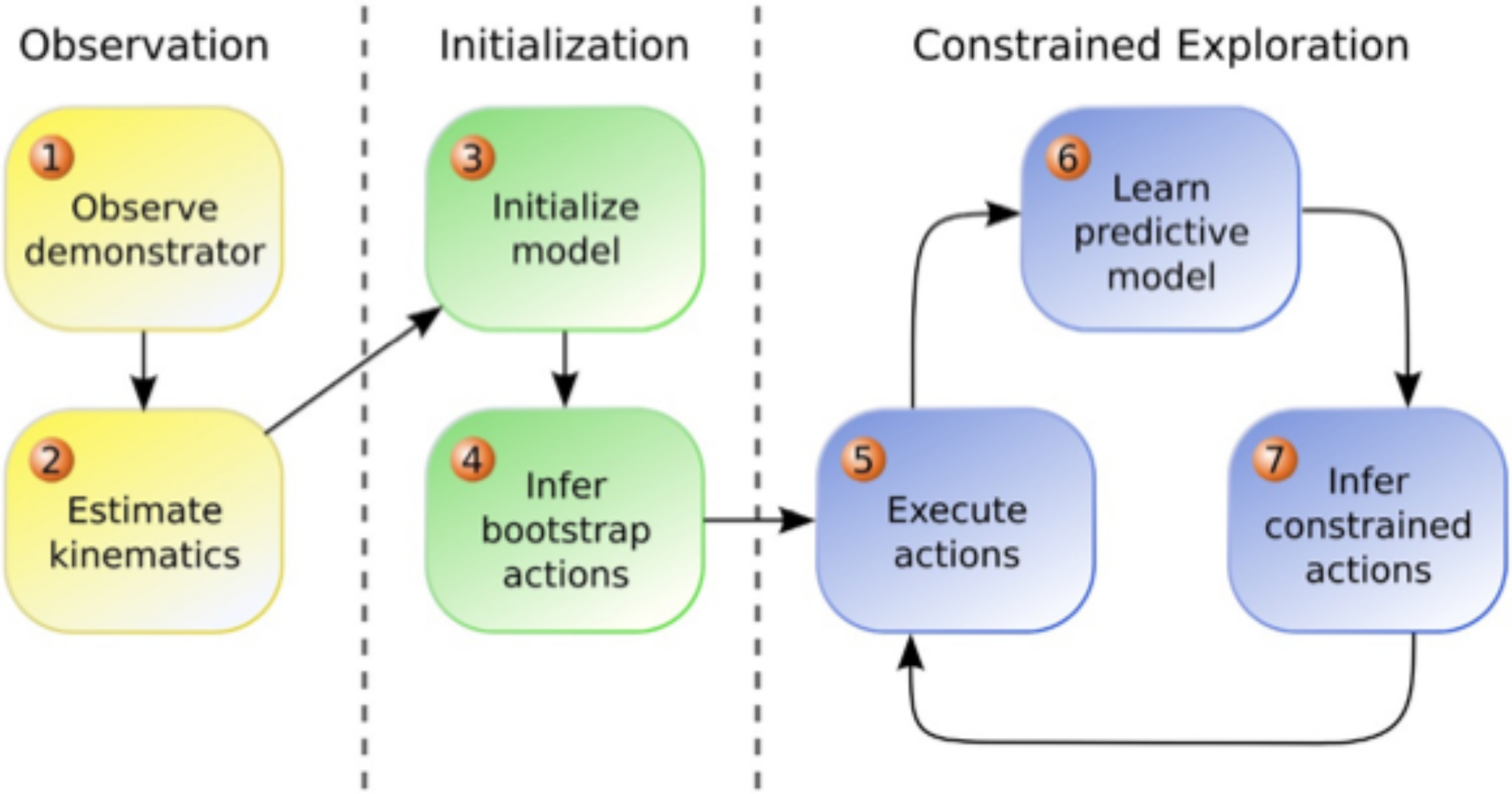
2. Step: Eigenpostures for Arm-Waving



3. Improve Behavior with a Forward Model

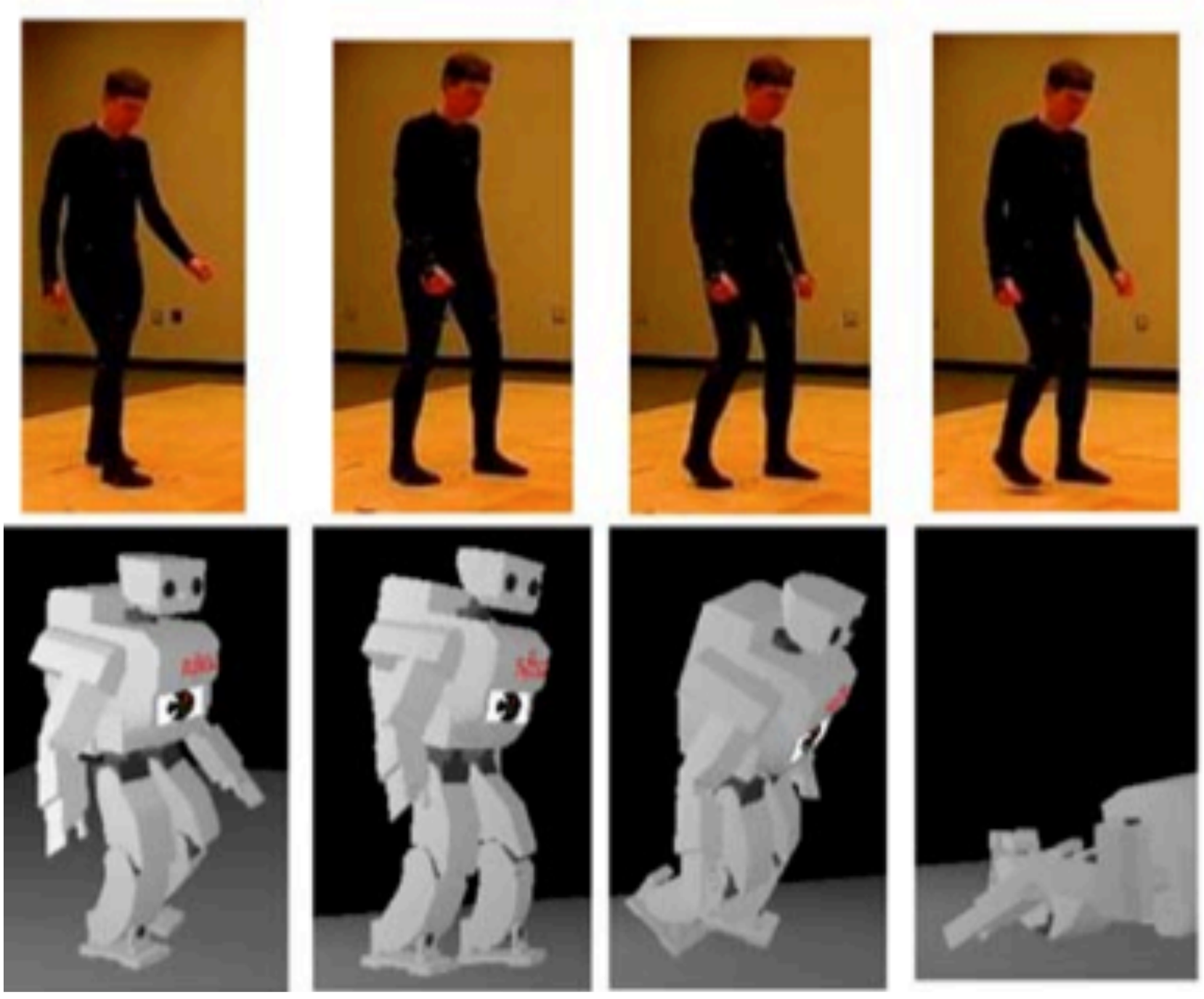
- 1. Learn forward model in the neighborhood of teacher demonstration
  - Use function approximation techniques to map actions to observed sensory consequences
- 2. Use the learned model to infer stable trajectory for imitation
- 3. Iterate between 1 and 2 for higher accuracy

Resulting Setup



Results for Walking

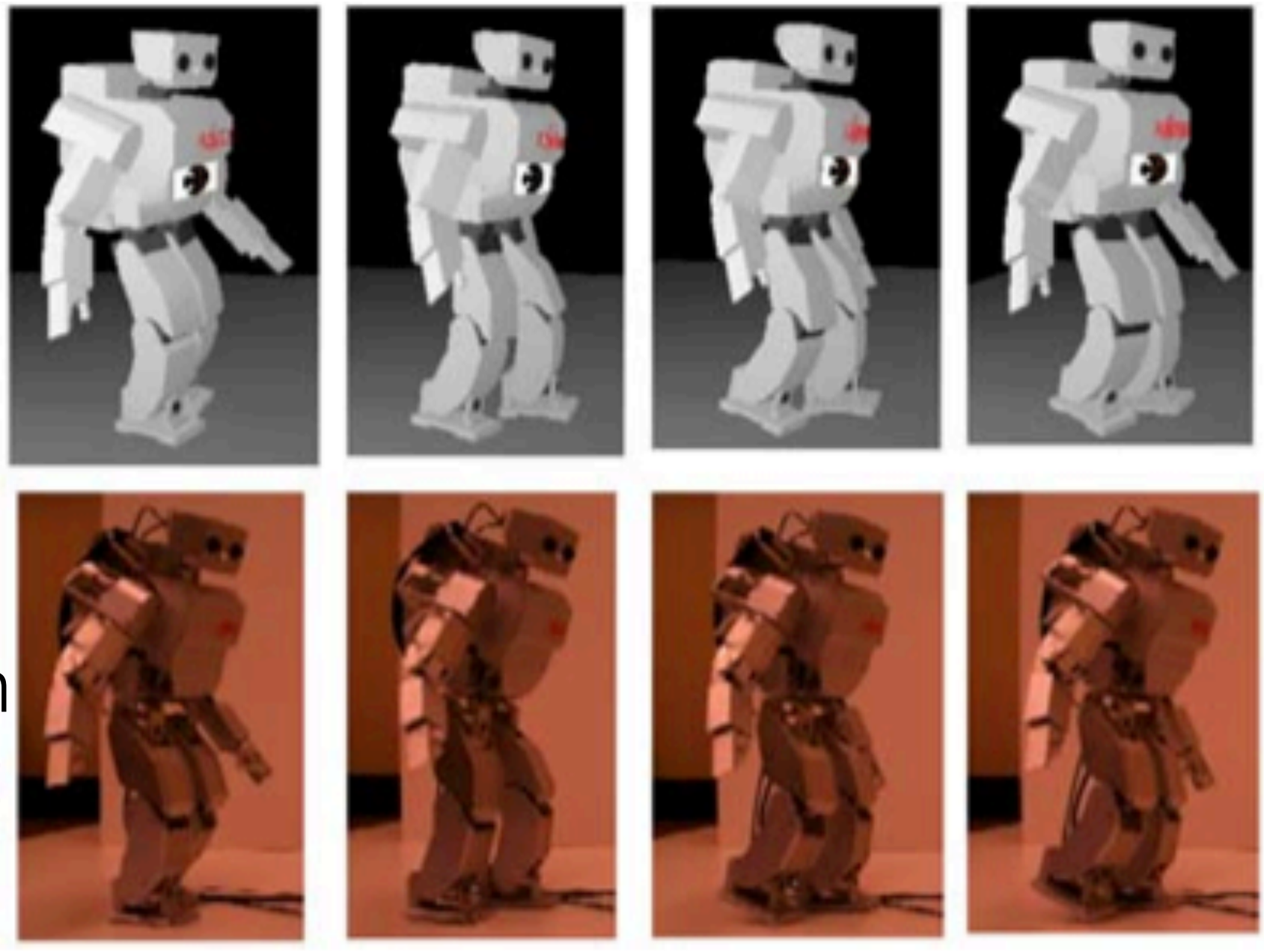
Demonstration



Imitation without a Forward Model

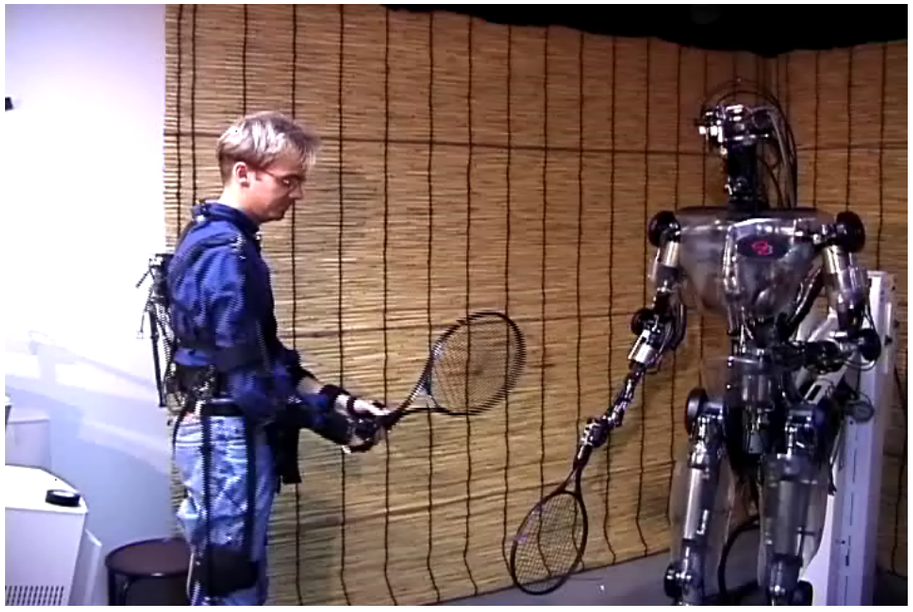
Results for Walking

Imitation with a Forward Model

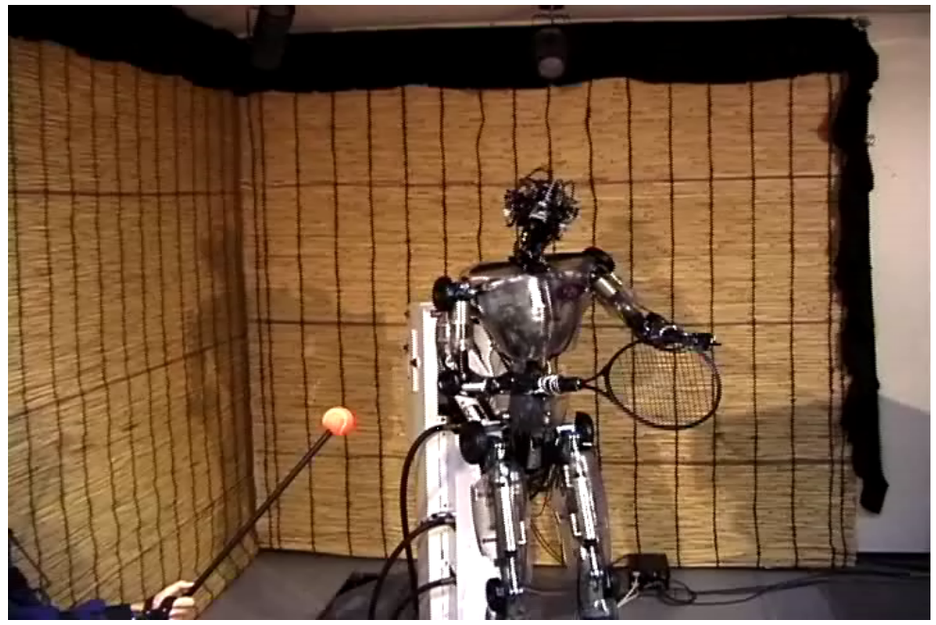


Reproduction on the robot

Imitation Learning



Imitation



Problems of Imitation Learning

- Correspondence Problem ➜ requires reinforcement learning
- Imperfect demonstrations ➜ require reinforcement learning
- Intent identification ➜ requires inverse reinforcement learning

Outline of the Lecture

1.Introduction

2. Foundations

3. Biological Inspiration

4. A Quick Overview: Well-known Approaches

5. Behavioral Cloning with Dynamical Systems

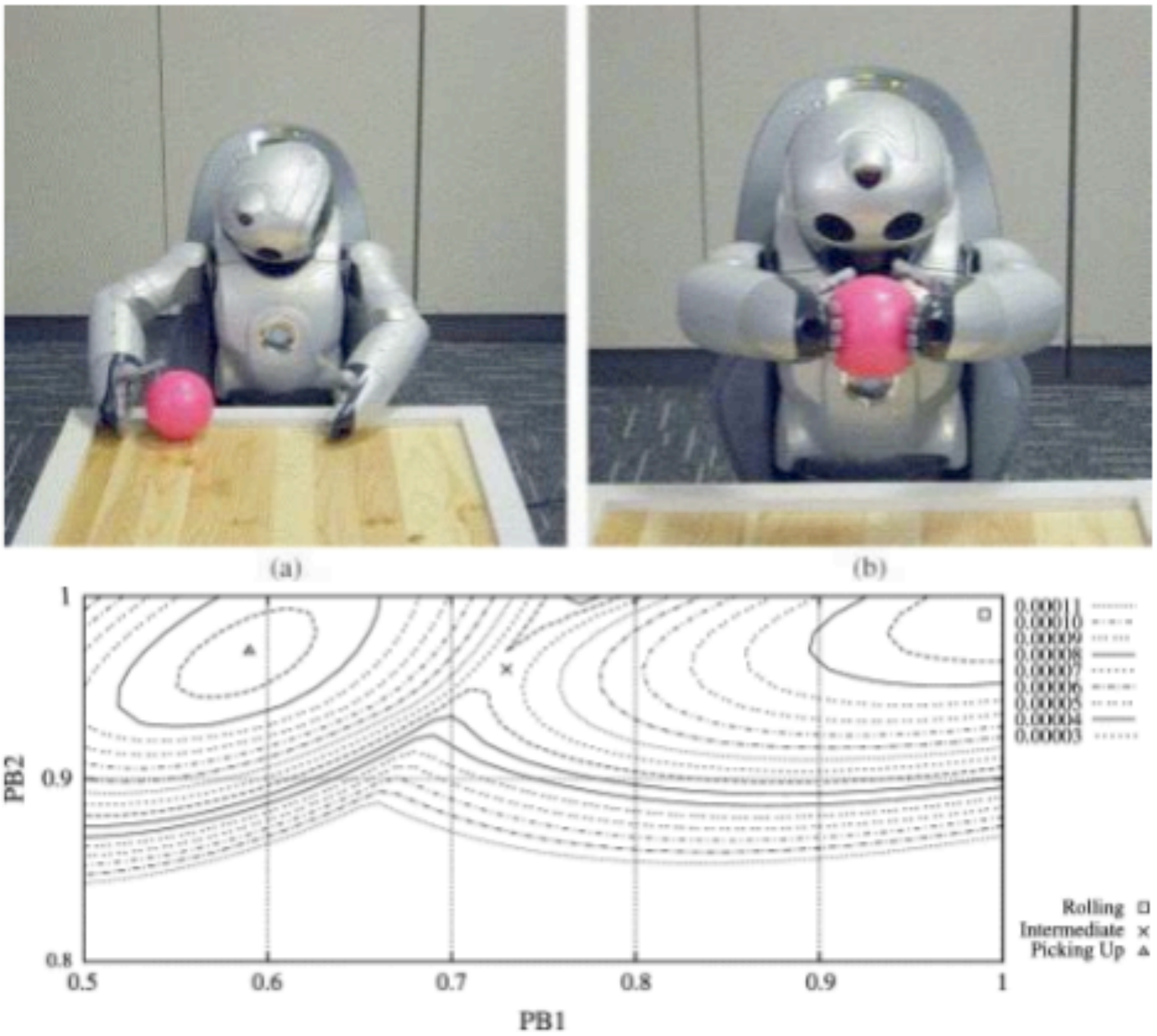
6. Behavioral Cloning with Forward Models

7. Conclusion

When to use Imitation Learning?

- **Advantages: When is Imitation useful?**
  - It is a powerful means of transferring skills
  - It speeds up the learning by experience process by showing possible solutions or conversely by showing bad solutions
- **Disadvantages: When is Imitation not useful?**
  - Not appropriate: When a good solution for the teacher is not a possible solution for the learner
  - Disadvantageous: When it induces you in error - bad teacher

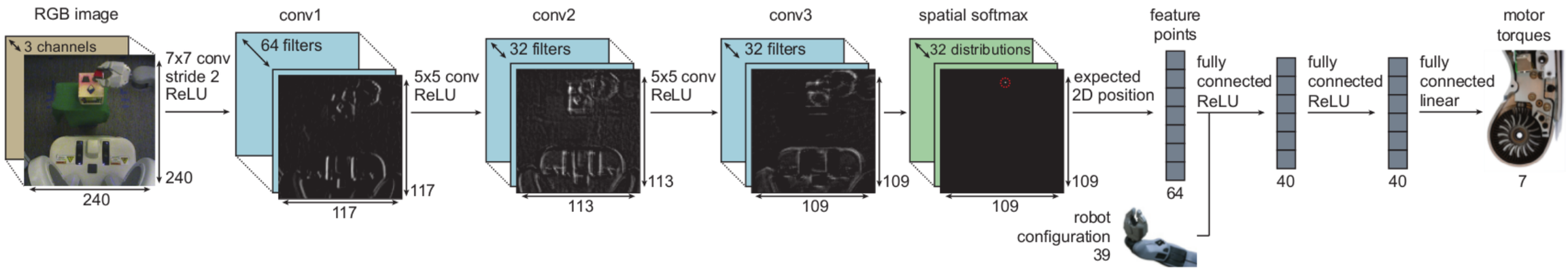
Fitting a recurrent neural network...



(Tani, 2006)

Regression with Neural Network using image

S. Levine, “End-to-End Training of Deep Visuomotor Policies”, arXiv, 2015



Initialization with imitation learning
+ improvement with reinforcement learning
+ generalization as regression

Deep Convolutional neural network with 92,000 parameters and 7 layers for **extracting features and determining control input**

Regression with Neural Network using image

Parameterized Dynamic Systems as Movement Primitives

- *Note the similarity between a generic control policy*

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

and nonlinear differential equations

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

Outline:

1. **Learning Policies from Demonstrations by Supervised Learning**
2. **Policy Representations**
   - State-space representations
   - Trajectory-based representations
3. **Imitation Learning with Movement Primitives**
   - Dynamic Movement Primitives
   - Probabilistic Movement Primitives
   - Beyond a single primitive

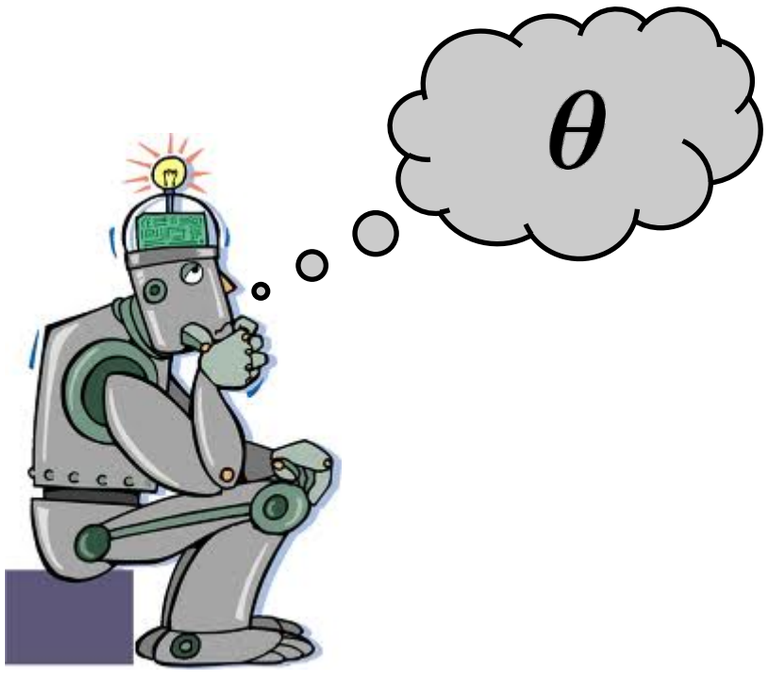
Why do we use parametric policies?

A parametric policy is a conditional probability distribution $\pi(u|s;\theta)$ that chooses the actions u depending on the state s of the robot

- Parametric policy naturally incorporates continuous actions
- **Estimate from demonstration / imitation learning**

  ⇨ Generalize to unseen situations
- **Search for improved parameters / reinforcement learning**

  ⇨ Autonomous self improvement!



What are desirable properties?

- **Compactness:** Low number of parameters
- **Learn-ability**: Easy to learn from demonstration and by reinforcement learning
- **Stochasticity:** Can encode exploration and variability
- **Optimality:** Can encode optimal behavior?
- **Scalability:** Can be used for a high number of DoFs?
- **Modularity:**

  Adaptability: Reusable for new situations?

  Co-activation and Blending of movements

- **Useable** for **stroke-based** and **rhythmic movements**

Stochastic vs. deterministic policies

Why use a stochastic policy?

Used for exploration in reinforcement learning (later)

Can also capture variability of movements

Exploration models:

No exploration: $\boldsymbol{u} = \pi(\boldsymbol{s}) = f_{\boldsymbol{w}}(\boldsymbol{s}), \quad \boldsymbol{\theta} = \boldsymbol{w}$

Uncorrelated Exploration: $\pi(\boldsymbol{u}|\boldsymbol{s};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{u}|f_{\boldsymbol{w}}(\boldsymbol{s}), \sigma^2\boldsymbol{I}), \quad \boldsymbol{\theta} = \{\boldsymbol{w}, \sigma^2\}$

Correlated Exploration: $\pi(\boldsymbol{u}|\boldsymbol{s};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{u}|f_{\boldsymbol{w}}(\boldsymbol{s}), \boldsymbol{\Sigma}), \quad \boldsymbol{\theta} = \{\boldsymbol{w}, \boldsymbol{\Sigma}\}$

➡ We also have to learn the variances of the linear models

Exploration might also hurt

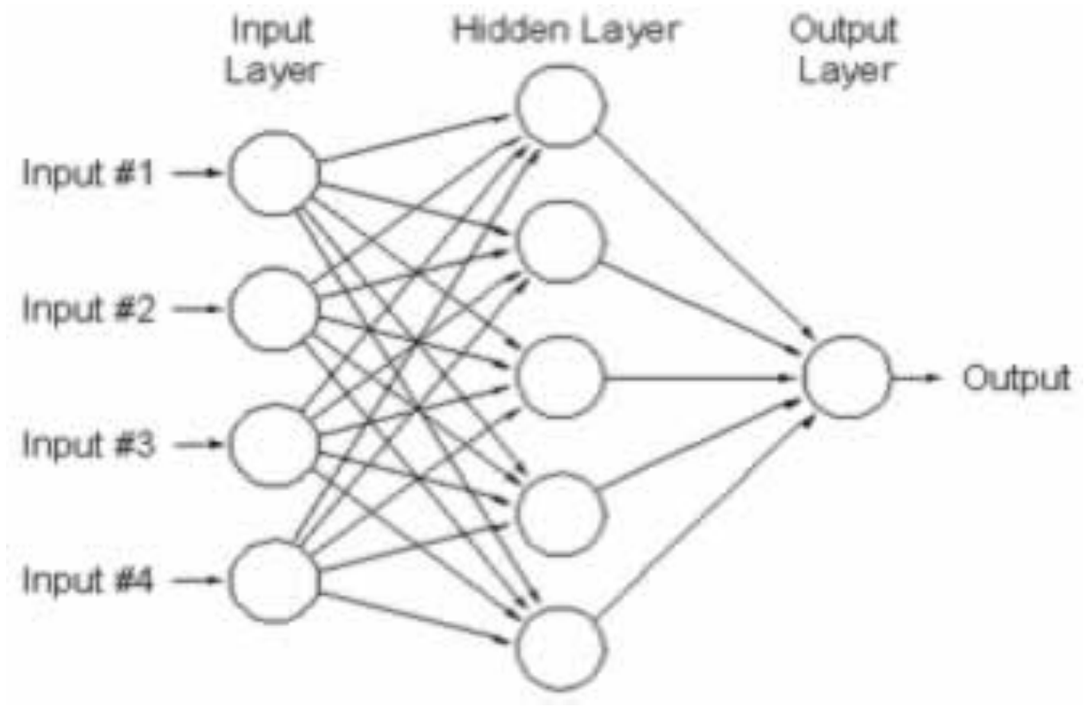
State space vs. trajectory space representations

State space representation: $\pi(\boldsymbol{u}|\boldsymbol{s};\boldsymbol{\theta})$

- Policy depends on the state and on the parameters
- Represents a globally valid policy
- Complex non-linear representations are needed

Examples:

- Neural Networks
- RBF Networks
- Gaussian Processes
- Locally Weighted Regression Models



State space representations

Linear controllers: $f_{\boldsymbol{w}}(\boldsymbol{s}) = \phi^T(\boldsymbol{s})\boldsymbol{w}$

Most simple case: linear PD controller

$$\phi(\boldsymbol{s}) = \begin{bmatrix} 1 \\ \boldsymbol{s} \end{bmatrix}, \quad f_{\boldsymbol{w}}(\boldsymbol{s}) = \boldsymbol{K}\boldsymbol{s} + \boldsymbol{k}$$

[-] Good feature representation needs to be known

[+] Very compact representation (low number of parameters)

[+] Easy to learn (linear regression)

State space vs. trajectory space representations

Time-dependent representation: $\pi(\boldsymbol{u}|\boldsymbol{s}, t; \boldsymbol{\theta})$

Policy also depends on time, e.g., follow a specific trajectory

For the same time step, the robot is often in similar states

Simple local models are often sufficient!

Time-dependend representations

For example: **Time-dependent linear feedback controllers**

$$f_{\boldsymbol{w}}(\boldsymbol{s}, t) = \sum_{i=1}^{K} \phi_i(t)(\boldsymbol{K}_i \boldsymbol{s} + \boldsymbol{k}_i)$$

- Time dependent basis functions, e.g., normalized RBF functions
- Scales quadratically with # DoF D: $KD/2(D+1)$
- Equivalent to PD-trajectory tracking with time-varying controller gains
  - Variable stiffness controllers
- Locally **optimal representation** (why we will see in the next lectures!)

Trajectory-based representations

Trajectory Generators:

Directly learn desired trajectory $\boldsymbol{q}^*(t;\boldsymbol{w})$

Use feedback controller to follow trajectory

$$\pi(\boldsymbol{s},t;\boldsymbol{w}) = \boldsymbol{K}_P(\boldsymbol{q}^*(t;\boldsymbol{w}) - \boldsymbol{q}) + \boldsymbol{K}_D(\dot{\boldsymbol{q}}^*(t;\boldsymbol{w}) - \dot{\boldsymbol{q}})$$

where typically $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ are hand tuned diagonal matrices

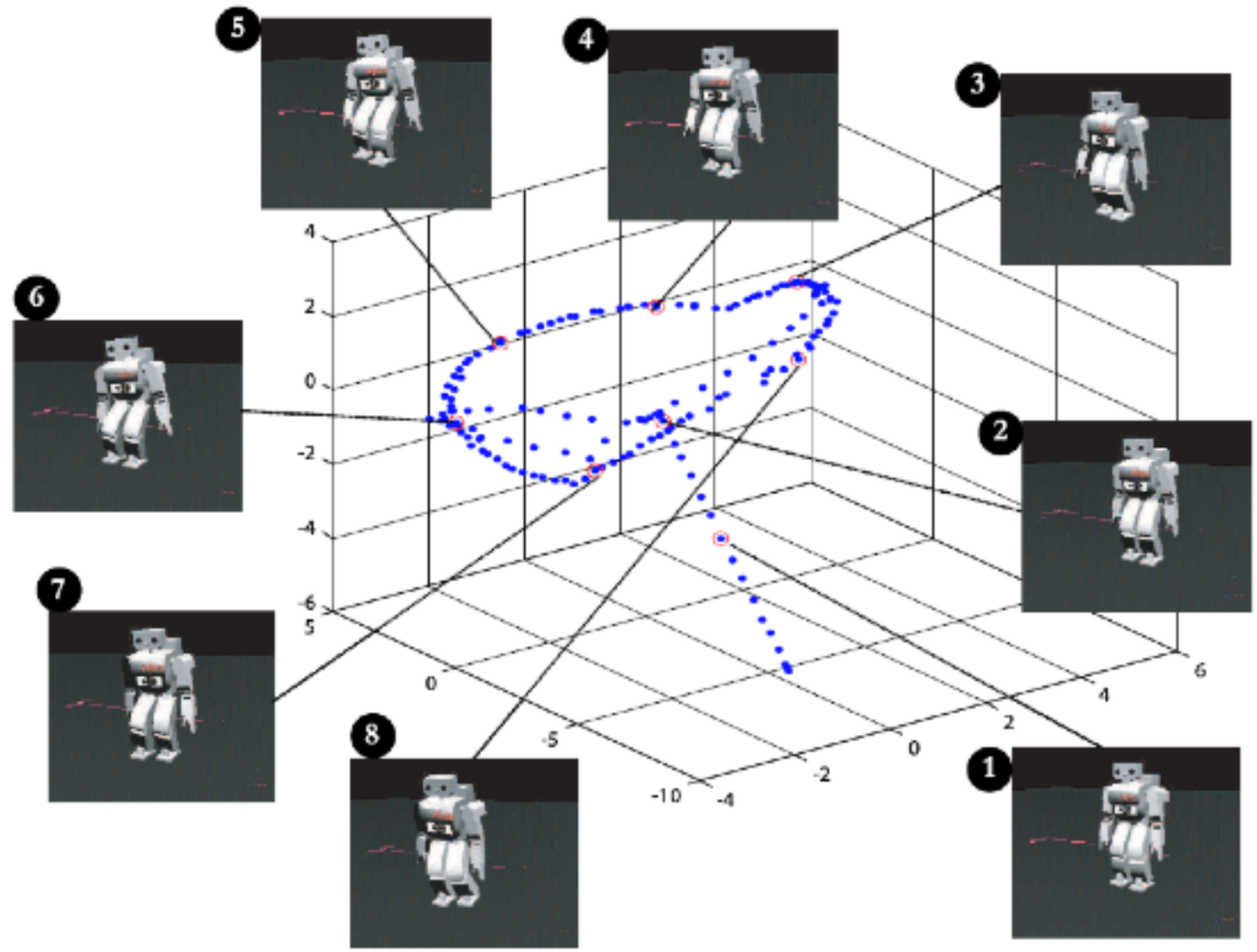
Possible Trajectory Representations:

- Splines $\boldsymbol{q}^*(t;\boldsymbol{w}) = \sum_{i=0}^{5} w_i t^i$
- Linear basis function models (RBFs) $\boldsymbol{q}^*(t;\boldsymbol{w}) = \boldsymbol{\phi}^T(t)\boldsymbol{w}$
- Dynamical Systems

State space representations

Represent controller in a **low-dimensional manifold**

⇨ E.g. Eigenpostures for walking



(Grimes, Rashid, & Rao, NIPS 2007)

Precedence Graphs

Demonstration 1: A → B → E → G

Demonstration 2: A → E → B → F → G

Constraints:

A < B	A < E	A < G	F < G	E < F
A < F	B < F	B < G	E < G	

Possible Execution Sequence: A → E → B → G

(Dillmann, *robotics and autonomous systems*, 2004)

Outline:

1. **Learning Policies from Demonstrations by Supervised Learning**
2. **Policy Representations**
   - State-space representations
   - Trajectory-based representations
3. **Imitation Learning with Movement Primitives**
   - Dynamic Movement Primitives
   - Probabilistic Movement Primitives
   - Beyond a single primitive

Movement Primitives

What are movement primitives?

- Movement primitives are a compact representation of a movement
- Often represented as parametrized trajectory generator

$$\boldsymbol{\tau} = f(\boldsymbol{w}), \quad \boldsymbol{w} \ldots \text{ parameters of the primitive}$$

Imitation Learning with trajectory generators

- By learning the desired trajectory, we also learn the desired long term behavior!
- However, we still have to learn how to follow this trajectory
- If we do not have good trajectory tracking controllers, it does not work

Dynamical systems as Trajectory Generators

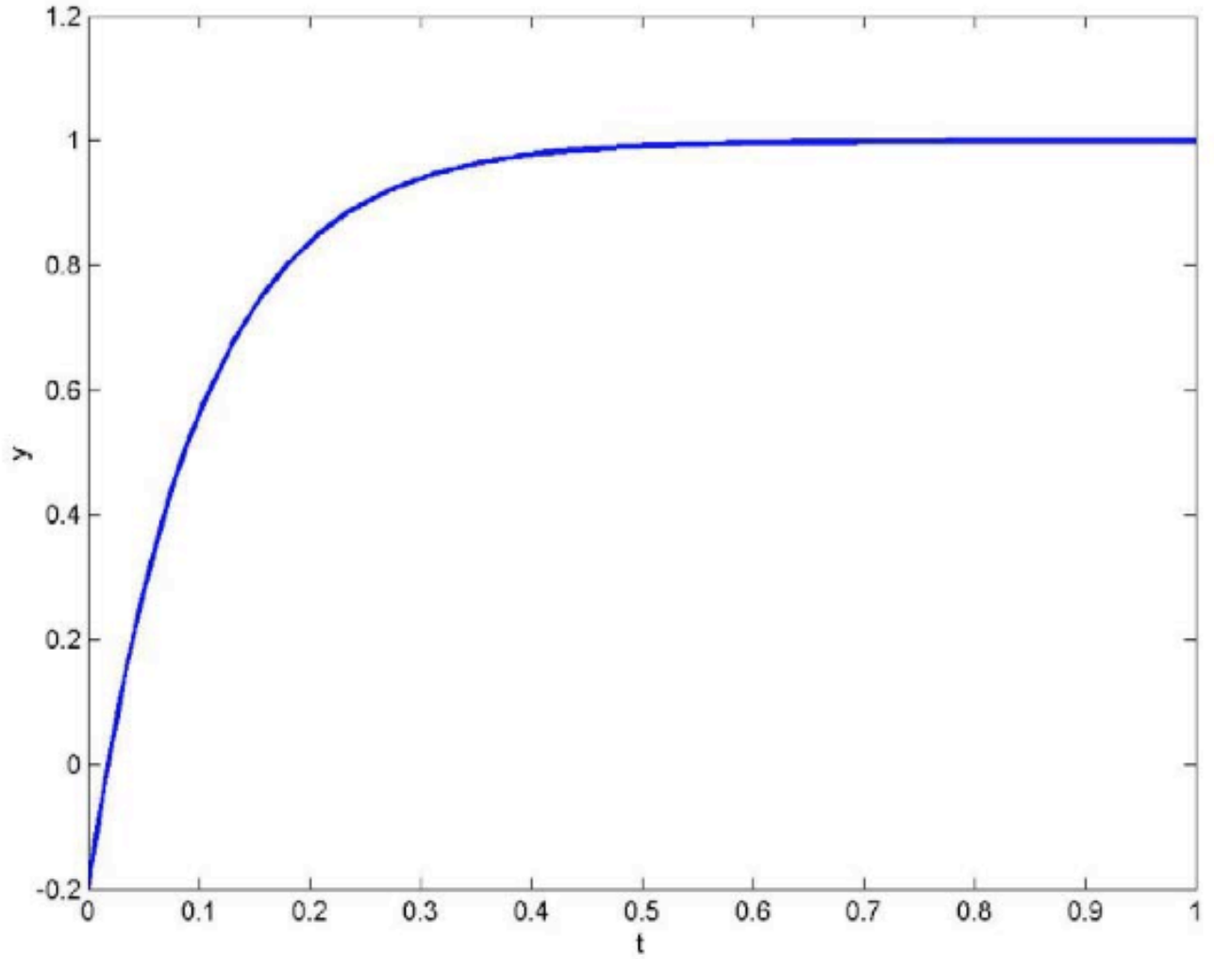
Dynamical systems can be used to represent trajectories

➡ Integrating the dynamical system results in a trajectory

$$\dot{y} = \alpha(c - y)$$

- What movement can a differential equation encode?

- **Example:** First order linear dynamical system:



Dynamic Movement Primitives (DMPs)

We can encode desirable properties such as:

- stability
- perturbation robustness
- periodic and point-to-point behaviors
- Attractors that have rather complex shape
- Easy to learn
- Coupling of a high number of DoFs
- Timing, temporal scaling
- Generalization (structural equivalence for parameter changes)

DMPs: [Ijspeert, Nakanishi & Schaal, NIPS 2003, Schaal, Peters, Nakanishi, Ijspeert, ISRR 2003

Dynamic Movement Primitives

How can we encode a desired behavior?

➡ Add a forcing function to obtain a moving attractor

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y}) + f_{\boldsymbol{w}}(t)$$

$$= \alpha(\beta(g + f_{\boldsymbol{w}}(t)/(\alpha\beta) - y) - \dot{y})$$

➡ The forcing function encodes the desired additional acceleration profile

➡ $f_{\boldsymbol{w}}$ … learnable function

(Ijspeert, Nakanishi & Schaal, NIPS 2003; Schaal, Peters, Nakanishi, Ijspeert, ISRR 2003)

Movement Primitives

Important properties of movement primitive representations

- Data-Driven: easily learnable from demonstrations
- Generalization: easily adaptable to a new situation
- Combination: Co-activate multiple primitives to solve a combination of tasks
- Temporal Scaling: Modulate the execution speed of the movement
- Coupling: Represent the coupling between a high number of joints
- Variability: Reproduce the stochasticity in the demonstrations
- Optimality: Can we represent optimal behavior?
- Can be applied for rhythmic and stroke-based movements

Movement Primitives

What we have so far…

- Data-Driven: Yes
- Generalization: Only adapt final positions
- Combination: No idea how…
- Temporal Scaling: Yes
- Coupling: Yes, but only the mean is coupled, no correlations
- Variability: No
- Optimality: Is following a single trajectory really optimal? No
- Can be applied for rhythmic and stroke-based movements: Yes

Probablistic Movement Primitives

- Stochastic representation of trajectories:

$$\boldsymbol{\tau} \sim p(\boldsymbol{\tau})$$

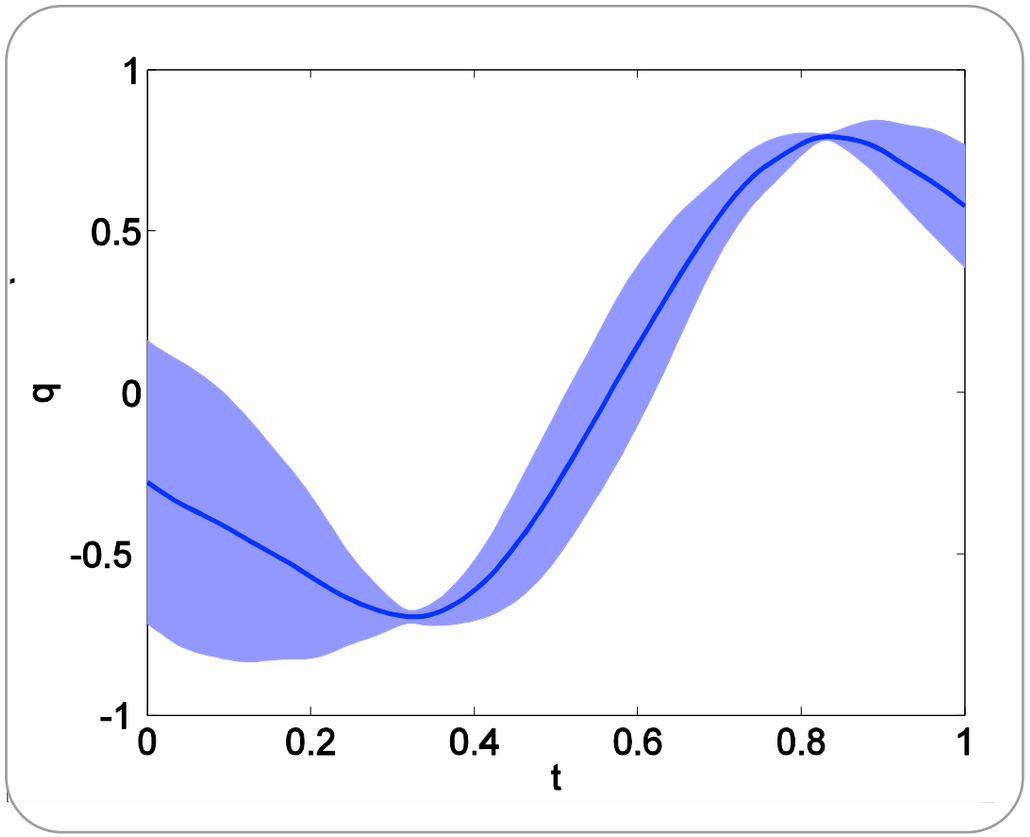
- Use $\boldsymbol{w}$ to represent a single trajectory

$$\boldsymbol{\tau} = f(\boldsymbol{w}) + \boldsymbol{\epsilon}$$

$p(\boldsymbol{w})$ $\boldsymbol{w}$

- Learn a distribution $\boldsymbol{w}$ $p(\boldsymbol{\tau})$ over the vecto

- Integrate out to obtain



Why is this useful?

- We can also represent uncertainty

- Uncertainty gives us information on

How to represent trajectory distributions?

Representation of a single trajectory

$$y_t = \boldsymbol{\psi}_t^T \boldsymbol{w} + \epsilon_y \qquad \epsilon_y \sim \mathcal{N}(0, \sigma^2)$$

Phase-dependent basis: $\boldsymbol{\psi}_t = \boldsymbol{\psi}(z_t)$

For example, normalized Gaussian basis functions

Probabilistic model:

$$p(\boldsymbol{\tau}|\boldsymbol{w}) = \prod_t \mathcal{N}(y_t|\boldsymbol{\psi}_t^T \boldsymbol{w}, \sigma^2) = \mathcal{N}(\boldsymbol{\tau}|\boldsymbol{\Psi}\boldsymbol{w}, \sigma^2 \boldsymbol{I}),$$

$$\text{with } \boldsymbol{\Psi} = [\boldsymbol{\psi}_1, \ldots, \psi_T]^T$$

Trajectory distribution: distribution over the parameters $p(\boldsymbol{w}|\boldsymbol{\theta})$

$$p(\boldsymbol{\tau}|\boldsymbol{\theta}) = \int_{\boldsymbol{w}} p(\boldsymbol{\tau}|\boldsymbol{w}) p(\boldsymbol{w}|\boldsymbol{\theta}) d\boldsymbol{w}$$

How to represent trajectory distributions?

You can always rely on old friends…

Lets use a **Gaussian**: $p(\boldsymbol{w}|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}})$

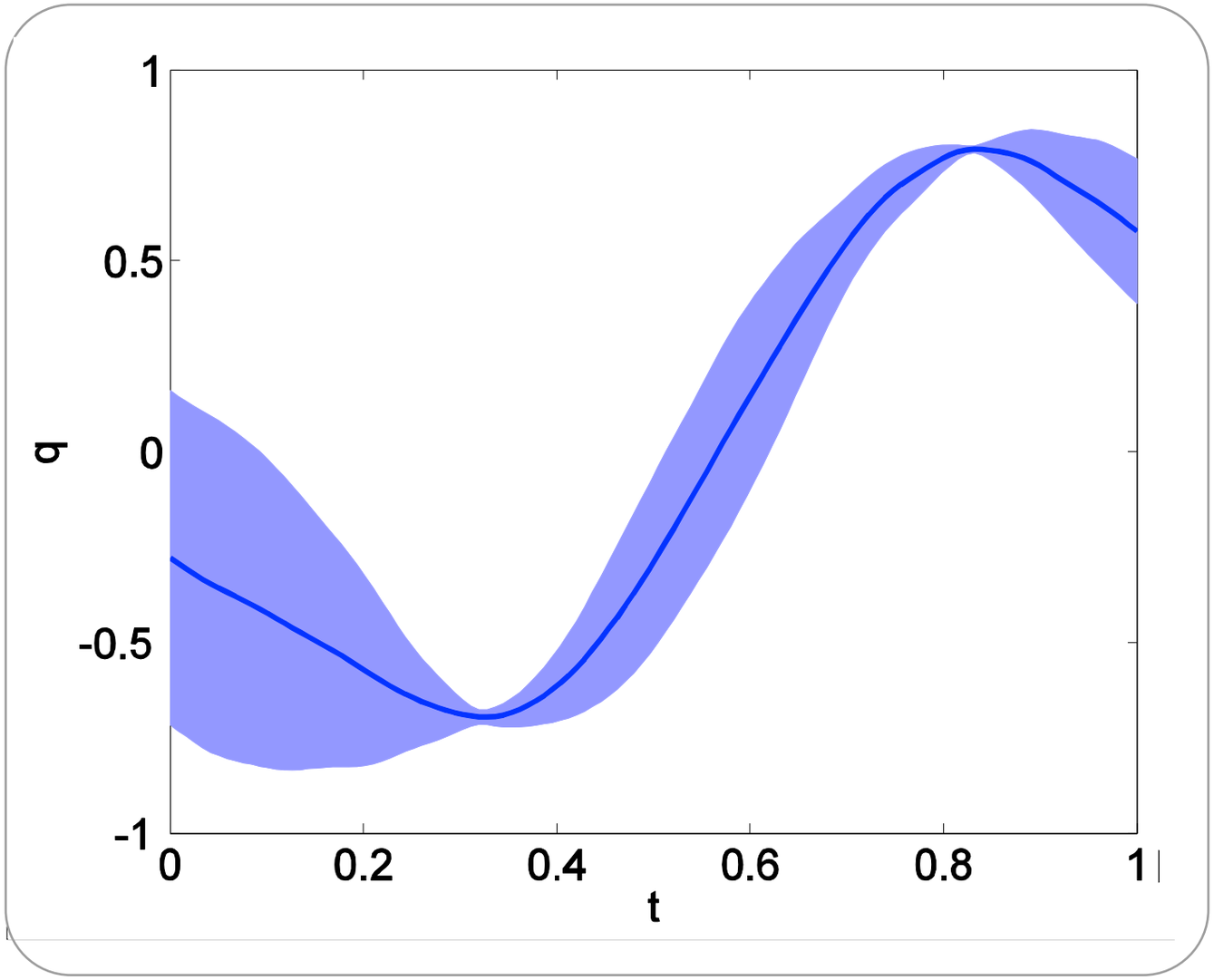
Computing the trajectory distribution is now easy

$$
\begin{aligned}
p(\boldsymbol{\tau}|\boldsymbol{\theta}) &= \int p(\boldsymbol{\tau}|\boldsymbol{w}) p(\boldsymbol{w}|\boldsymbol{\theta}) d\boldsymbol{w} \\
&= \int \mathcal{N}(\boldsymbol{\tau}|\boldsymbol{\Psi}\boldsymbol{w}, \sigma^2 \boldsymbol{I}) \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}) d\boldsymbol{w} \\
&= \mathcal{N}(\boldsymbol{\tau}|\boldsymbol{\Psi}\boldsymbol{\mu}_{\boldsymbol{w}}, \sigma^2 \boldsymbol{I} + \boldsymbol{\Psi}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}^T)
\end{aligned}
$$

Hence, we can easily **evaluate mean and variance** for any time point

How to represent trajectory distributions?

Hence, we can easily **evaluate mean and variance** for any time point



How to represent trajectory distributions?

How can we encode a distribution over multiple DoFs?

- Use a concatenated weight and trajectory vector and block-diagonal basis matrix

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\tau}_1, \\ \vdots \\ \boldsymbol{\tau}_D \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_1, \\ \vdots \\ \boldsymbol{w}_D \end{bmatrix} \quad \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\Psi} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Psi} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{\Psi} \end{bmatrix}$$

- The same linear relation holds: $\boldsymbol{\tau} = \boldsymbol{\Phi}\boldsymbol{w}$
- We use a distribution $p(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}})$ over the parameters of all DoFs

For a single time step: $p(\boldsymbol{y}|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\phi}_t\boldsymbol{\mu}_{\boldsymbol{w}}, \sigma^2\boldsymbol{I} + \boldsymbol{\phi}_t\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\phi}_t^T)$

Covariance matrix encodes correlation between the joints

Trajectory distribution tracking

How do we use a trajectory distribution for robot control?

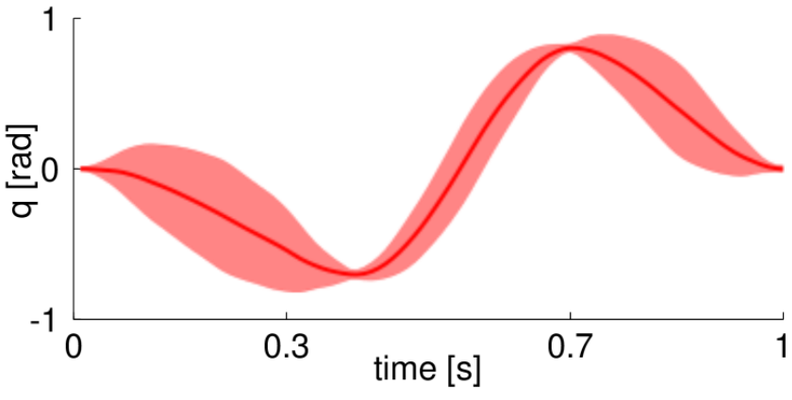
- We can obtain a time-varying stochastic feedback-controller in closed form

$$\boldsymbol{u}_t = \boldsymbol{k}_t + \boldsymbol{K}_t \boldsymbol{y}_t + \boldsymbol{\epsilon_u}, \quad \boldsymbol{\epsilon}_u \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_u)$$

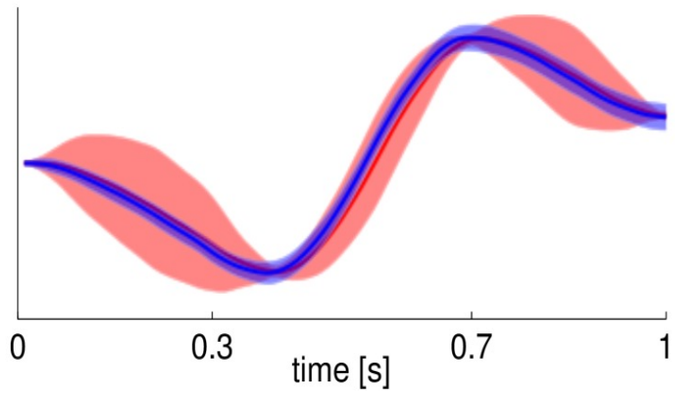
from $\boldsymbol{\tau} \sim p(\boldsymbol{\tau}|\boldsymbol{\theta})$ that exactly reproduces the given trajectory distribution (mean and variances)

- Same structure as optimal controllers for linear(ized) systems

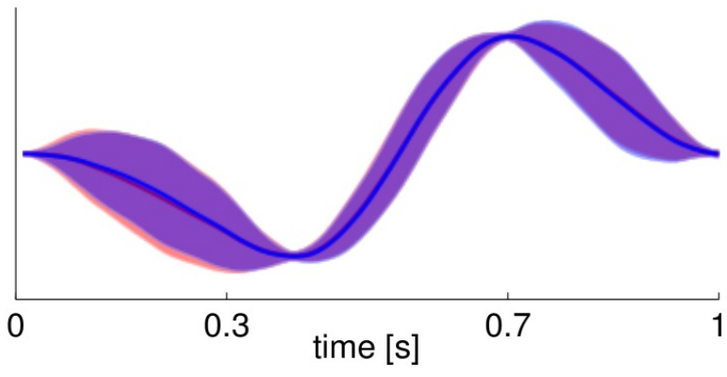
- But it needs an accurate model



Optimal control



DMPs



ProMPs

Generalization via Conditioning

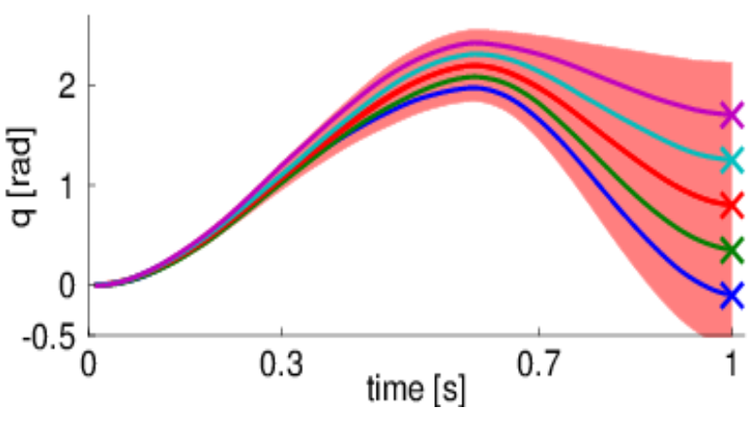
- **Generalization:** Change intermediate or end-point of the movement

- We can condition $p(\boldsymbol{w})$ on reaching position $\boldsymbol{q}_t^*$ at time-step t

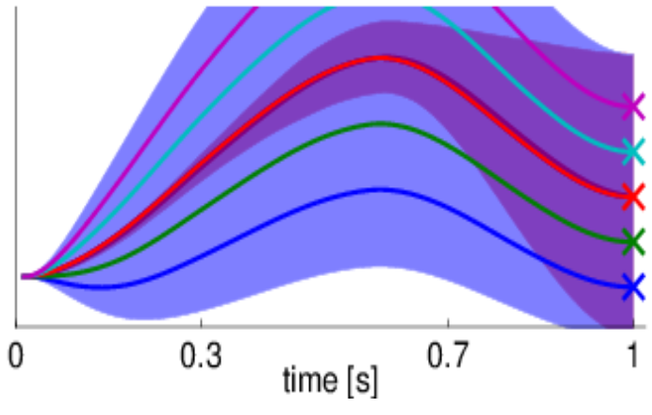
$$p(\boldsymbol{w}|\boldsymbol{q}_t = \boldsymbol{q}_t^*)$$

➡ New trajectory distribution is obtained by Bayes theorem

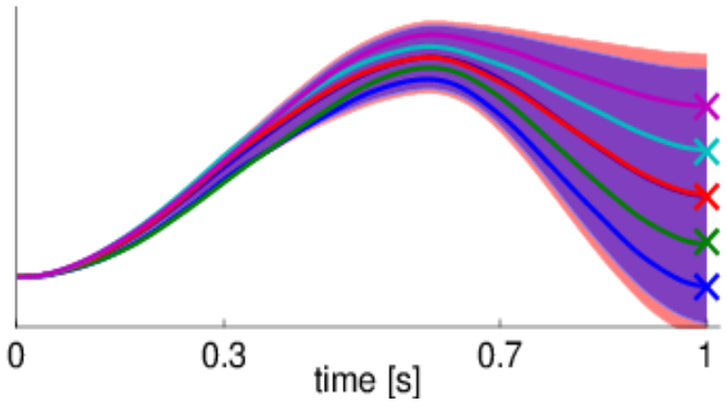
➡ Closed-form solution for Gaussian trajectory distributions



Demonstration



Dynamic MPs



ProMPs

Combination of Movement Primitives

Modularity: Combine primitives to solve a combination of tasks

- Implemented as product of distributions:

➡ **„Intersection“ of trajectory distributions**

➡ Area, in which all distributions have high probability

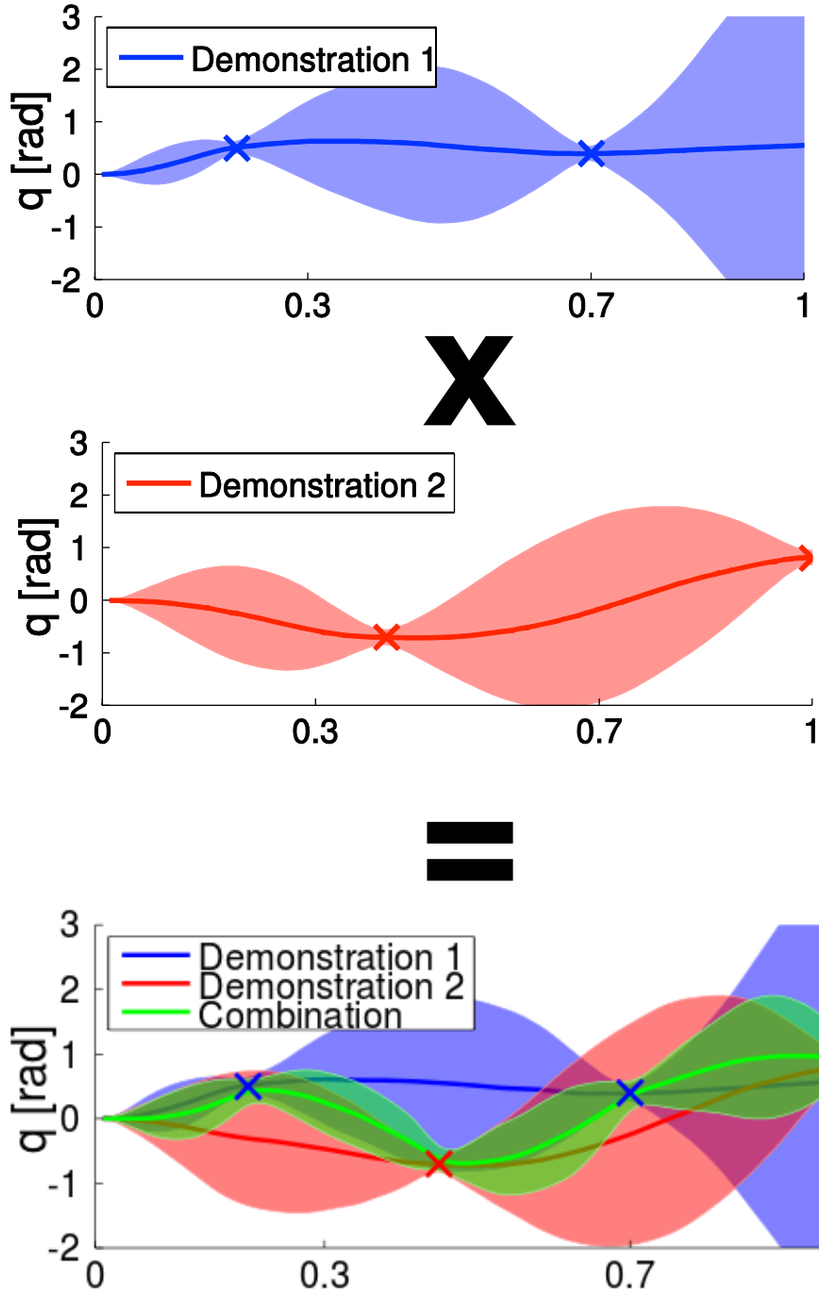
$$p_{\text{co}}(\boldsymbol{q}_t) \propto \prod_{i=1}^{N} p_i(\boldsymbol{q}_t)^{\alpha_i(t)}$$

- i-th movement primitive
- activation factors

$p_i(\boldsymbol{q}_t) \ldots$

$\alpha_i(t) \ldots$

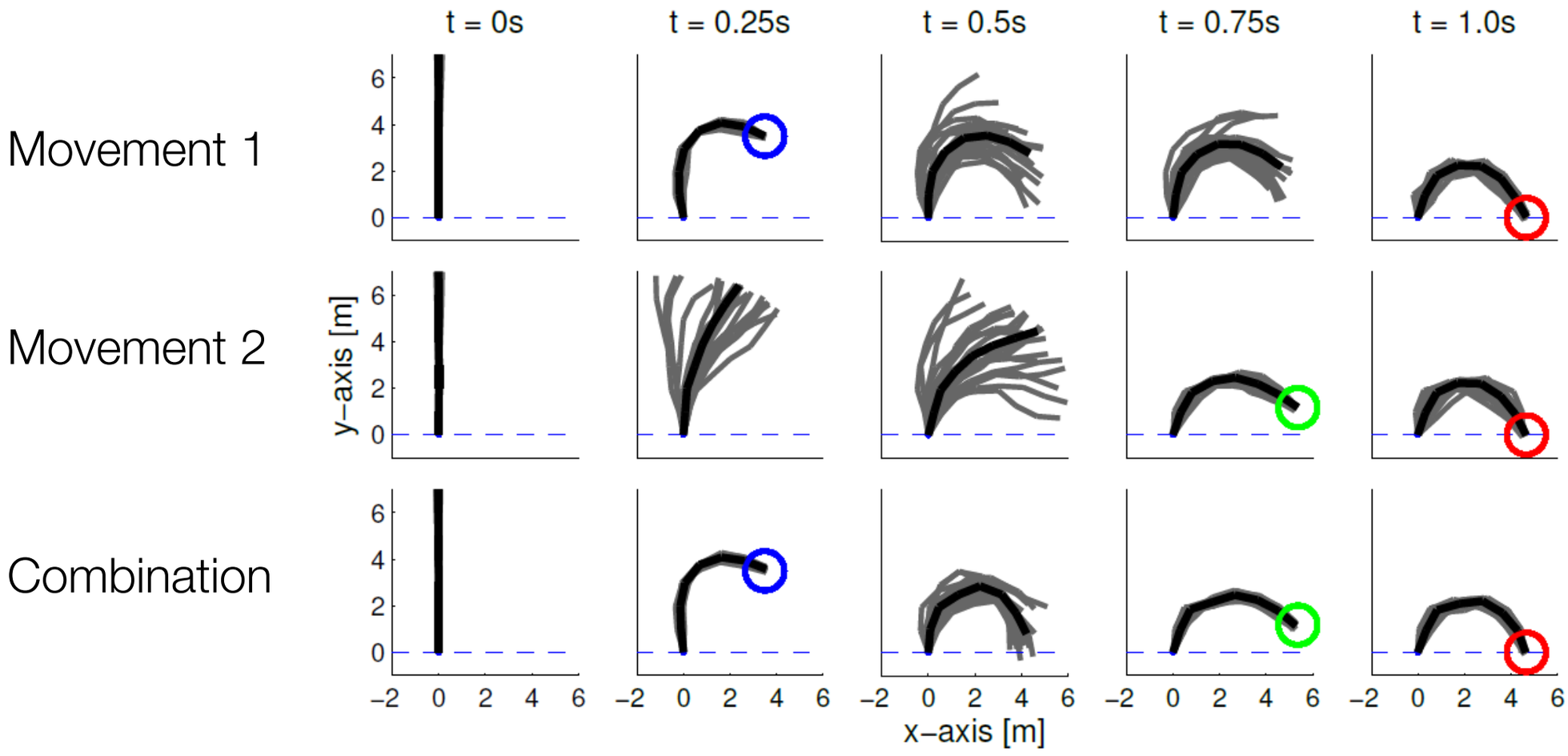
➡ Computed in closed-form for Gaussian distributions



Experiments: Co-Activation

7-link planar robot arm, controlled by inverse dynamics

- Trained 2 movements for reaching different via-points at different time steps
- Combination of the movements reaches all 2 via-points

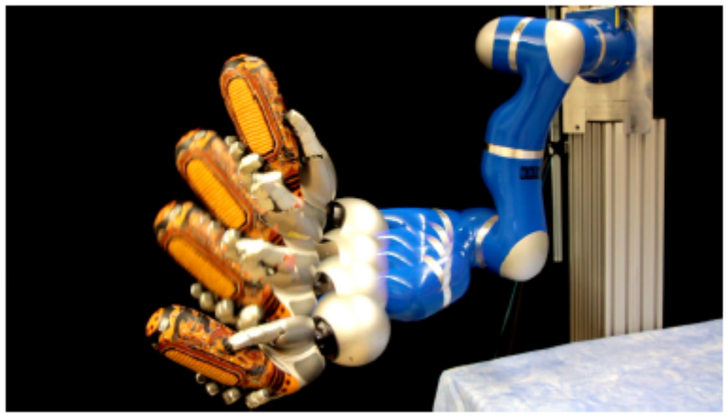


Experiments: Blending and temporal scaling

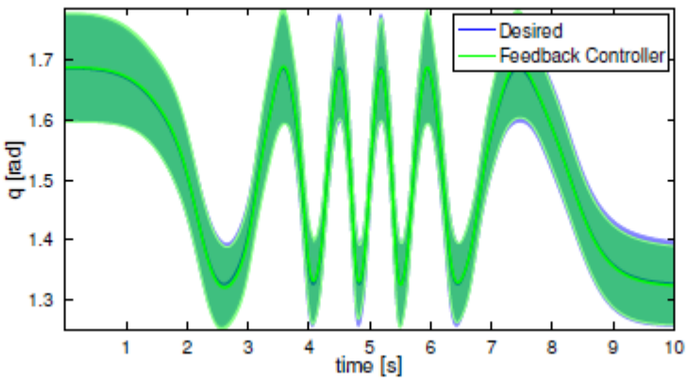
7-link KUKA robot arm, playing maracas

- Record rhythmic movements to produce sounds
- Blend between different rhythmic movements

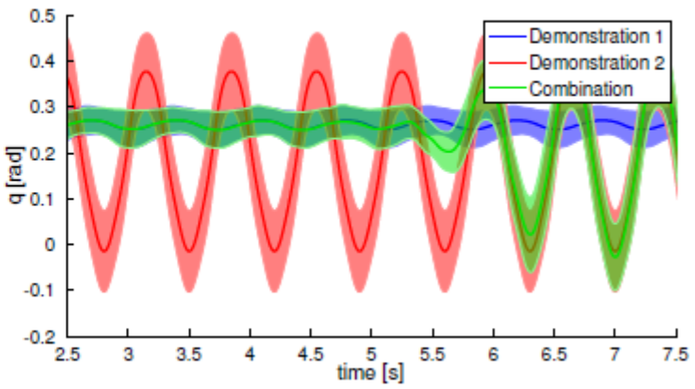
Maracas



Temporal scaling



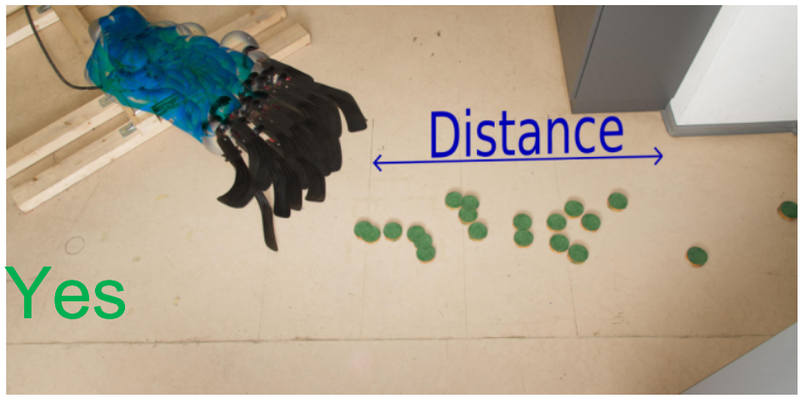
Blending



Case Study: Robot Hockey

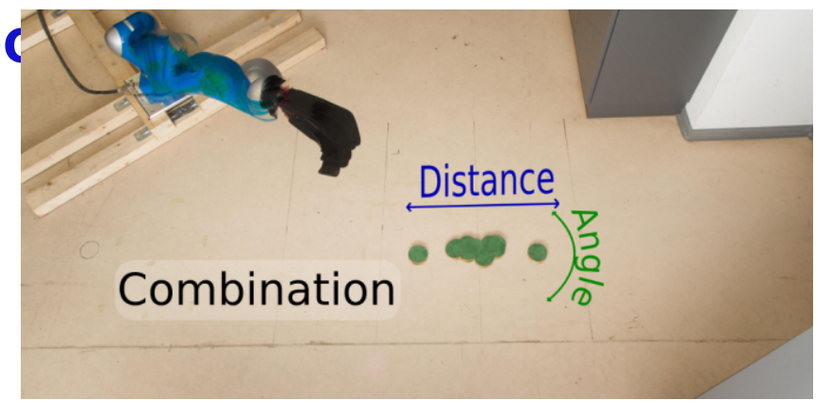
- **7-link KUKA robot arm, playing hockey**
  - Train 2 primitives with high variance in shooting angle or in distance

Demonstration 1

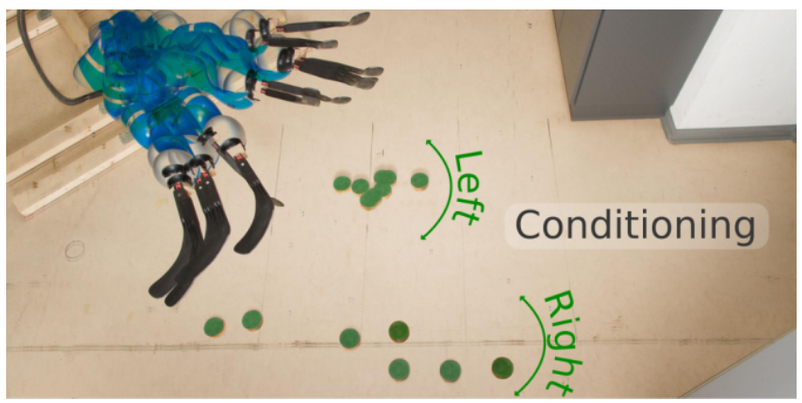


Demonstration 2

  - Product of the primitives:
    Co



  - **Conditioning** to select the shooting angle



Movement Primitives

What we have so far…

- Data-Driven: Yes
- Generalization: Yes
- Combination: Yes
- Temporal Scaling: Yes
- Coupling: Yes
- Variability: Yes
- Optimality: Yes
- Can be applied for rhythmic and stroke-based movements: Yes

Summary…

What you should know…

- State-space representations versus trajectory-based policy representations
- What is imitation learning and when does it fail?
- What are the main ideas of using movement primitives?
- Why do use dynamical systems? Advantages/Disadvantages?
- Why do use a probabilistic representation?