Movement Primitives 2: Time-Dependent Primitives

Gerhard Neumann

University of Lincoln, UK

Some geography first…

- **Lincoln?**

What we have seen so far...

Learning state-based policies:

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

$$\pi(\boldsymbol{u}|\boldsymbol{s};\boldsymbol{\theta})$$

Examples: Neural Networks, RBF Networks, Gaussian Processes, Locally Weighted Regression Models

---

Trajectory-based policies:

- Policy also depends on time
- For the same time step, the robot is often in similar states
- Simple local models (e.g. linear) are often sufficient!

$$\pi(\boldsymbol{u}|\boldsymbol{s},t;\boldsymbol{\theta})$$

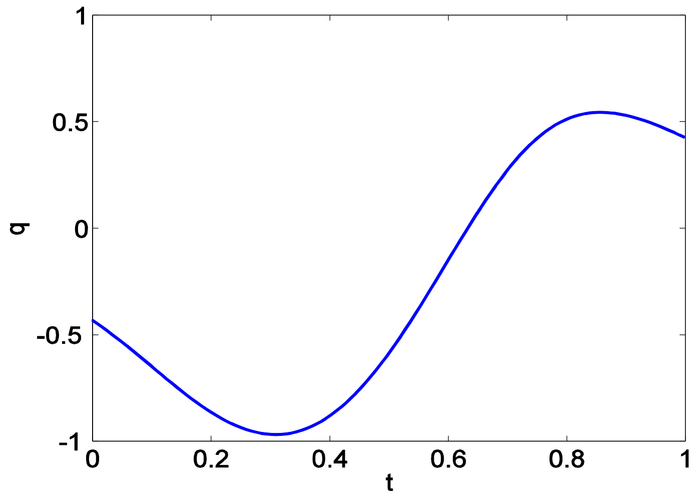
Examples: Variable stiffness controllers, Movement Primitives

Trajectory-Based Movement Primitives

Parametrized trajectories:

$$\boldsymbol{\tau}^* = \boldsymbol{q}_{1:T} = f(\boldsymbol{\theta})$$

- Mean movement
- Followed by trajectory tracking controllers



Properties:

- Compact parametrization
- Easy to learn
- Adaptable
- Learn the desired long term behavior!

Examples:

- Dynamic Movement Primitives (DMPs) [Ijspeert 2002]
- Probabilistic Movement Primitives (ProMPs) [Paraschos 2013]

Outline

Dynamic Movement Primitives

Probabilistic Movement Primitives

- Introduction
- Learning ProMPs
- Case Study 1: Robot Table Tennis
- Case Study 2: Interaction Primitives
- Case Study 3: Prioritization of Primitives

Dynamical Systems as Trajectory Generators

Dynamical systems can be used to represent trajectories

- Integrating the dynamical system results in a trajectory

$$\dot{y} = f(y)$$

- Mimics physical systems
- Build-in Smoothness

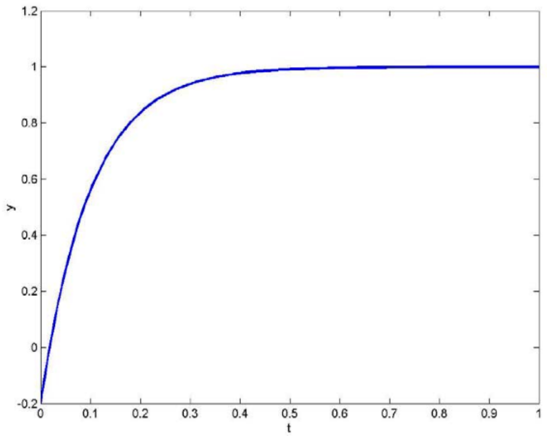
Linear differential equations:

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

What movements can we encode?

First order linear dynamical system:

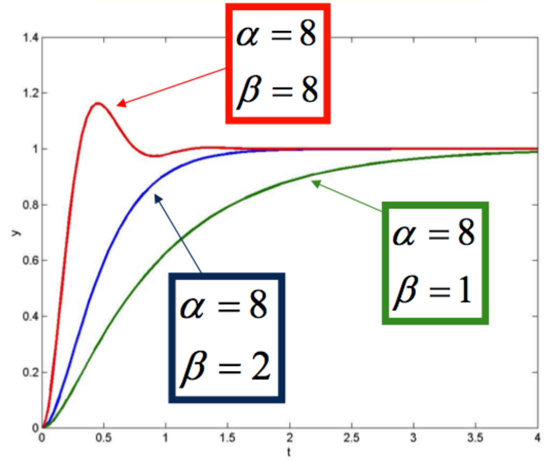
$$\dot{y} = \alpha(g - y)$$



Second order linear dynamical system:

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y})$$

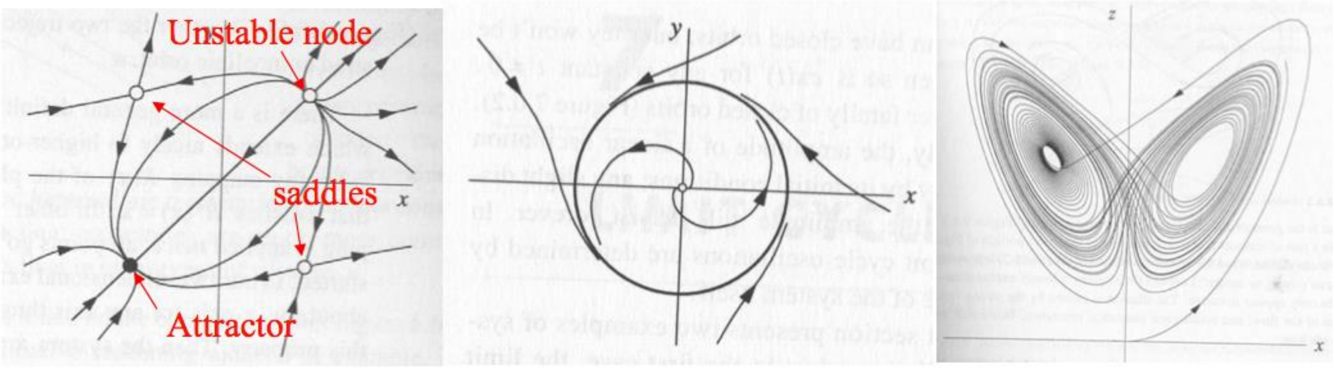
- g ... goal attractor
- $\alpha\beta$ ... spring constant
- α ... damping



How can we make it more representative?

Use **non-linear dynamical systems**?

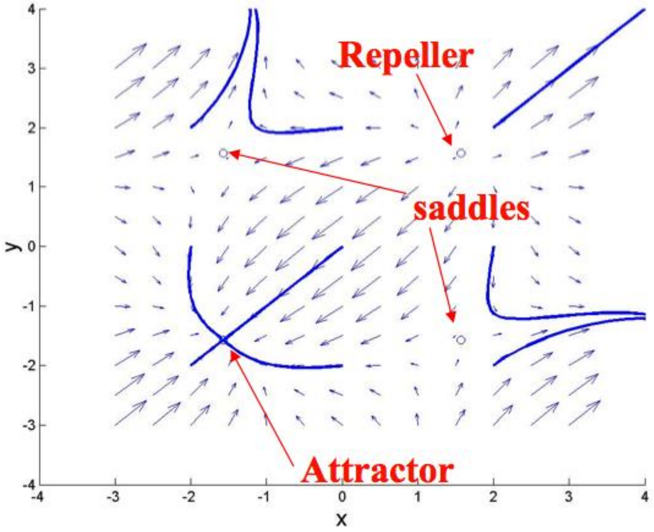
- Different behavior might emerge…



Attractors   Limit cycles   Chaos

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

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

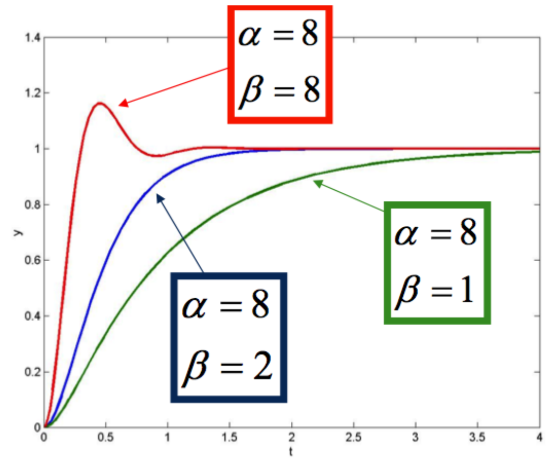


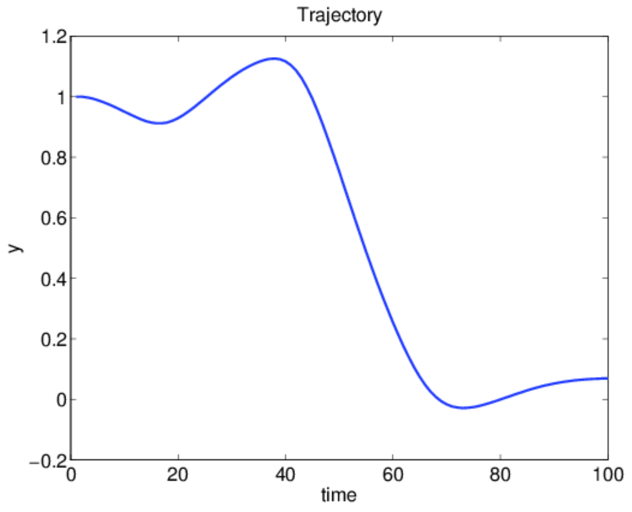
Dynamical movement primitives

- Use linear dynamical systems (2nd order)
- Introduce moving attractor

$$\ddot{y} = \alpha(\beta(g - y) - \dot{y}) + f_{\boldsymbol{w}}(t)$$
$$= \alpha(\beta(\underbrace{g + f_{\boldsymbol{w}}(t)/(\alpha\beta)}_{\text{Moving Attractor}} - y) - \dot{y})$$

- The forcing function $f_{\boldsymbol{w}}(t)$ encodes the desired additional acceleration profile
- $f_{\boldsymbol{w}}(t)$ … learnable function





[Ijspeert et. al., Dynamical movement primitives: learning attractor models for motor behaviors, Neurocomputing, 2013]
[Ijspeert et. al., Learning Attractor Landscapes for Learning Motor Primitives, NIPS, 2003]

Temporal scaling

Modulate the speed of the movement:

- **Introduce phase variable**

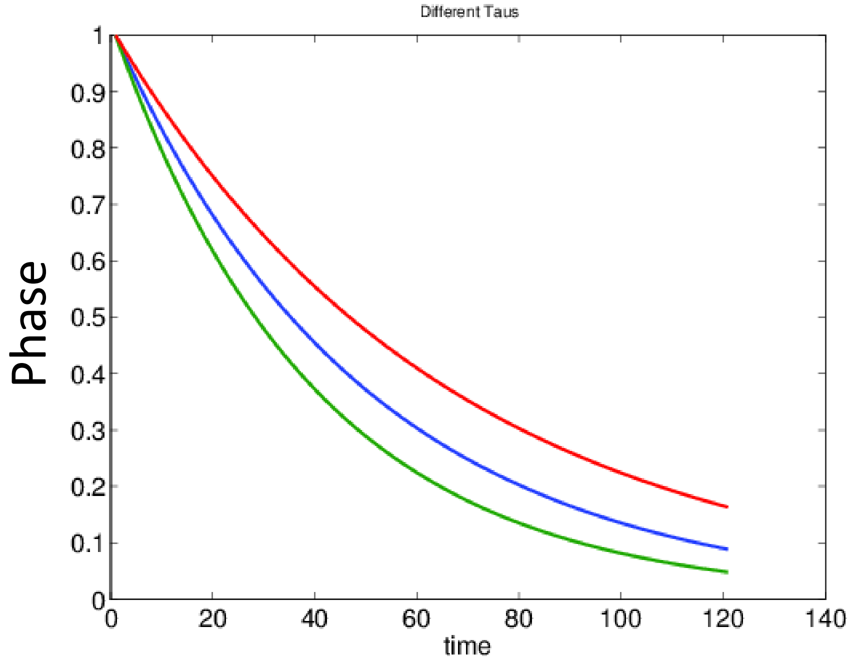
$$\dot{z} = -\tau \alpha_z z$$

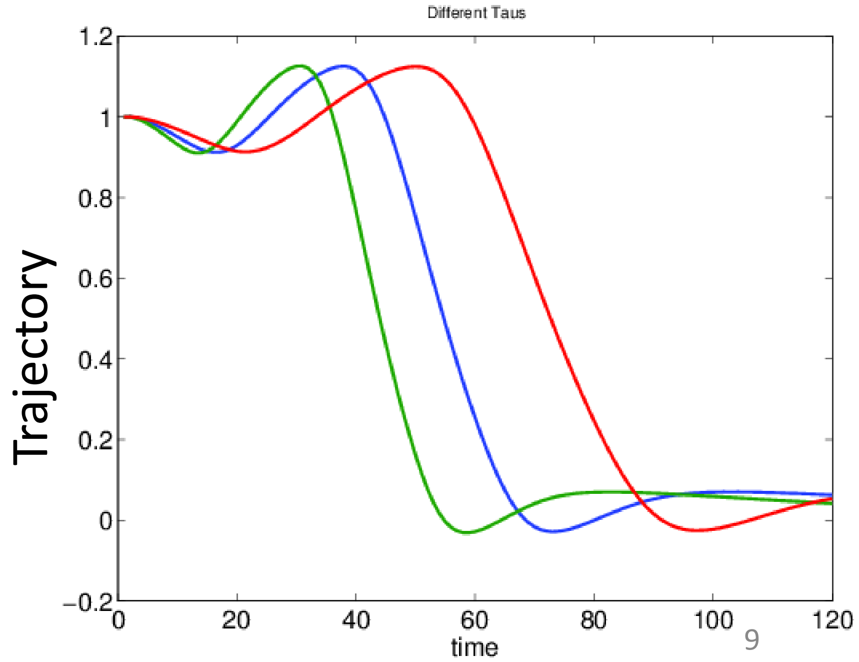
  - Simple first-order system
  - $\tau \dots$ temporal scaling coefficient

- **Replace time with phase**

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

- **Every DoF shares the same phase variable**





Forcing function

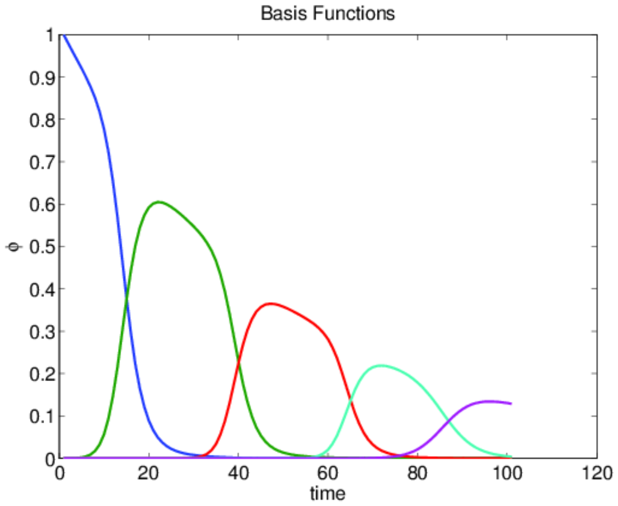
- **Linear model:** $f_{\boldsymbol{w}}(z) = \boldsymbol{\psi}^T(z)\boldsymbol{w}$
- **Normalized RBF basis functions:**

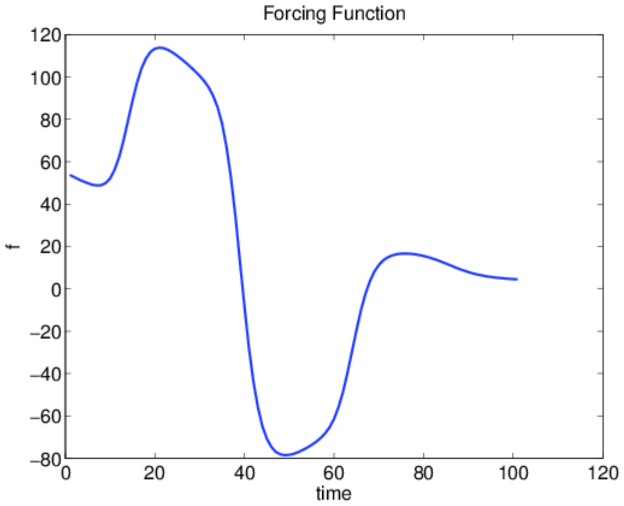
$$\psi_i(z) = \frac{\phi_i(z)z}{\sum_{j=1}^{K}\phi_j(z)}$$

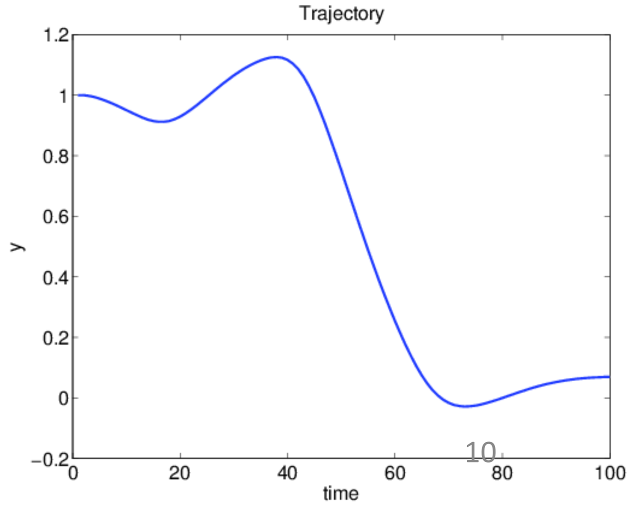
$$\phi_i(z) = \exp(-0.5(z - c_i)^2/h_i)$$

Asymptotic stability by construction:

- Forcing function vanishes for $t \to \infty$
- Then it is just a standard PD controller







Imitation Learning for DMPs

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

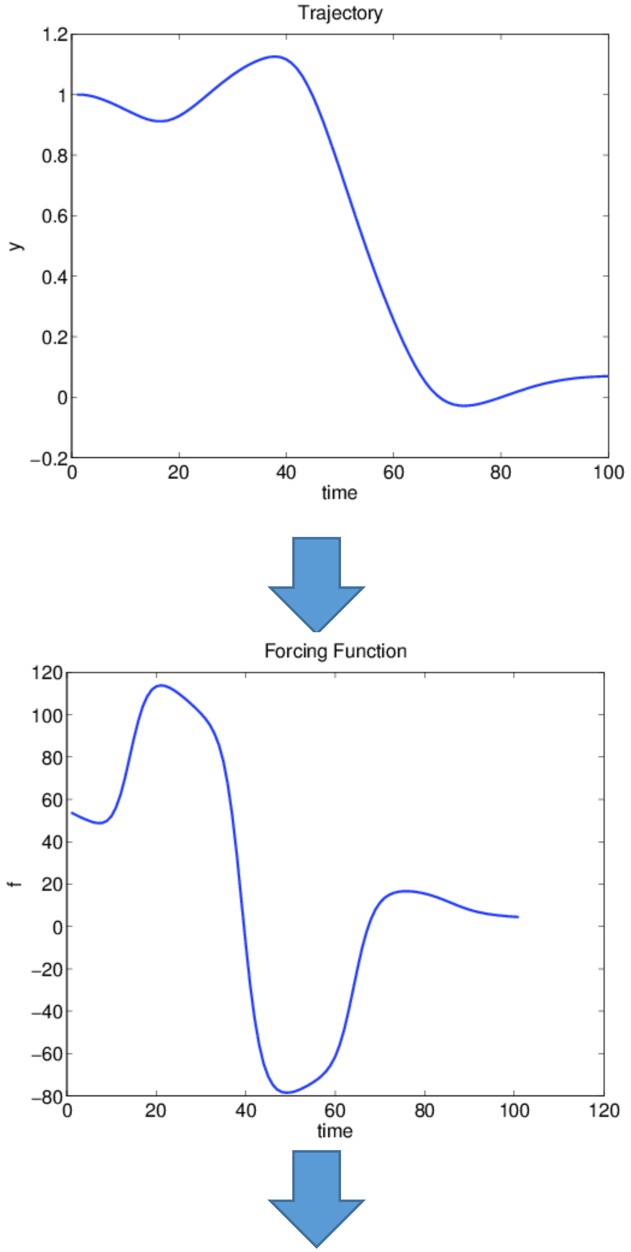
The weights w can be learned by linear regression:

- 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

$$\boldsymbol{w} = (\boldsymbol{\Psi}^T\boldsymbol{\Psi} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{\Psi}^T\boldsymbol{f}$$

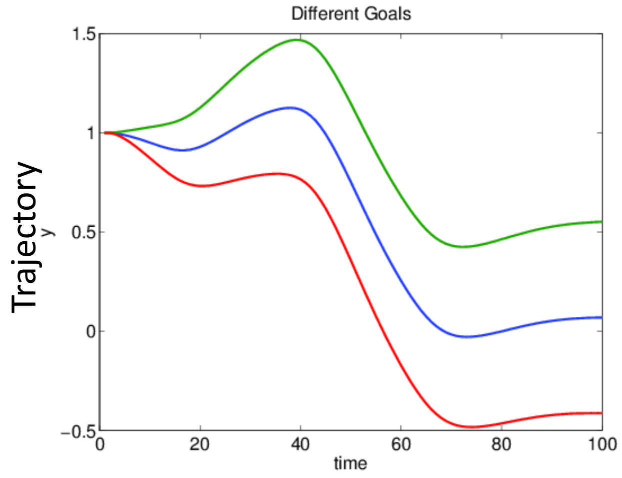


$\boldsymbol{w}$

Adapting the meta-parameters...

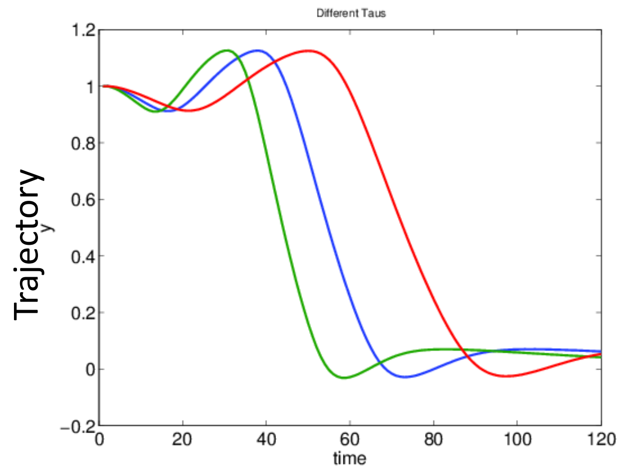
Adapt goal attractor g

- Can change end-point of the movement
- Shape of the movement is changes heuristically

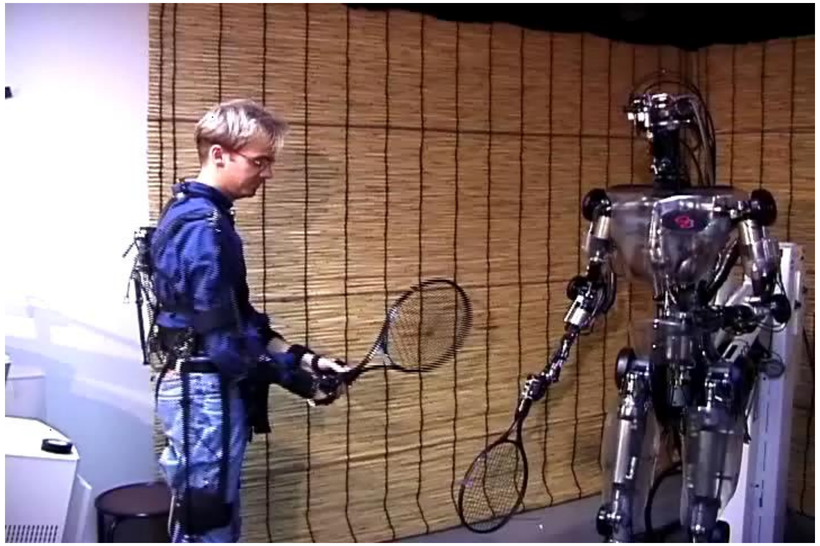


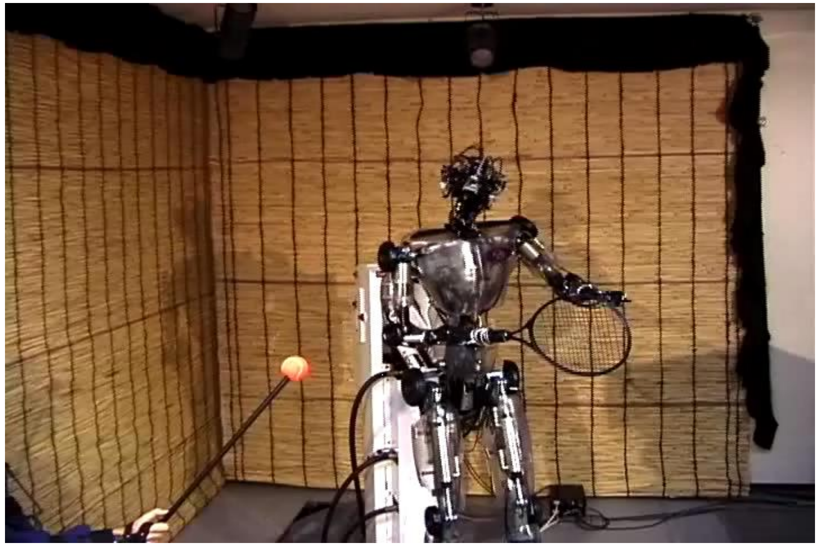
Adapting the temporal scaling τ

- Larger tau result in faster movements
- Can also be modulated online



Example: A Tennis Backhand





[Ijspeert et. al., Learning Attractor Landscapes for Learning Motor Primitives, NIPS, 2003]

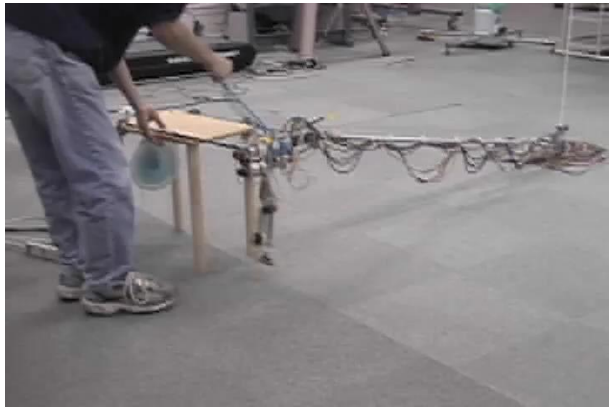
Summary: DMPs

- Dynamical systems define smooth trajectory
- Learn acceleration profile
- Stable per construction
- Easy to modulate execution speed
- Adapt final positions

Extensions:

- Adapt final velocities [Kober et al., ICRA 2009]
- Perceptual coupling [Kober et al., IROS 2008]
- Obstacle avoidance [Pastor et al., ICRA 2009]
- Force profiles [Denisa et al, IEEE Transaction on Mechatronics 2016]
- Rhythmic Movements

[Kober et al., ICRA 2010]



[Nakanishi et al., 2012]

Outline

Dynamic Movement Primitives

Probabilistic Movement Primitives

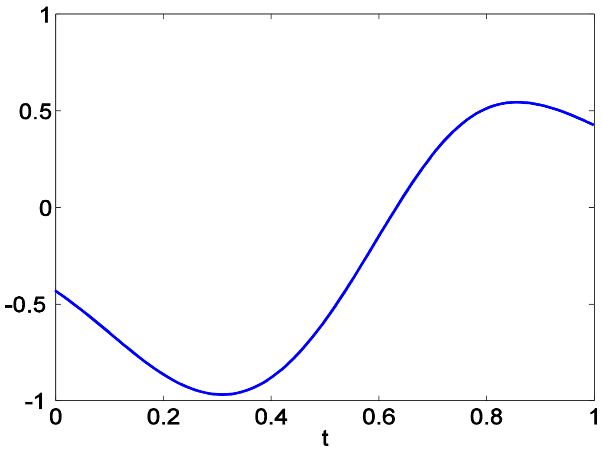
- Introduction
- Learning ProMPs
- Case Study 1: Robot Table Tennis
- Case Study 2: Interaction Primitives

Probabilistic Movement Primitives

Parametrized trajectories:

$$\boldsymbol{\tau}^* = \boldsymbol{y}_{1:T} = f(\boldsymbol{\theta})$$

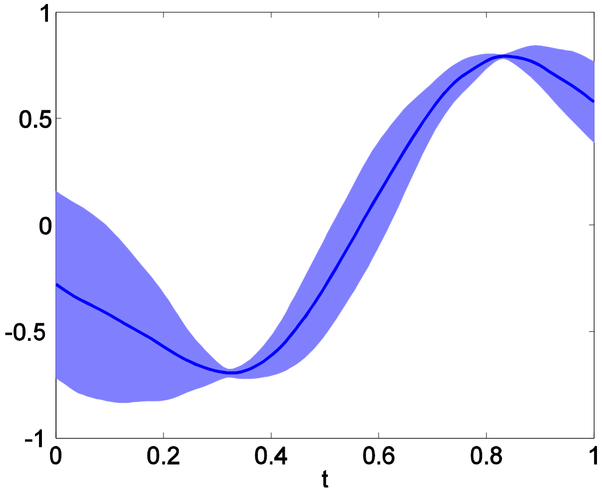
- Mean movement
- Followed by trajectory tracking controllers
- Example: Dynamic Movement Primitives (DMPs) [Ijspeert 2002]



Parametrized trajectory distributions:

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

- Family of movements
- **Gaussian:** Mean and Variance
- **Probabilistic Movement Primitives (ProMPs)** [Paraschos 2013]



A. Paraschos, …, **G. Neumann,** *Probabilistic Movement Primitives*, NIPS, 2013

Trajectory Representation

Representation of a single trajectory

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

- Approximation in position instead of acceleration space

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

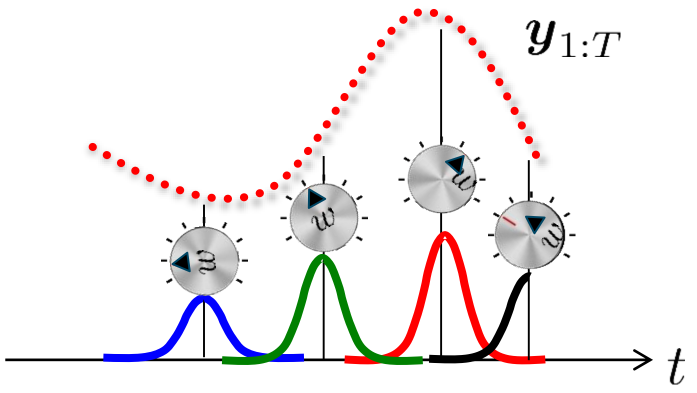
- For example, normalized Gaussian basis functions

$$\psi_i(z) = \frac{\phi_i(z)}{\sum_{j=1}^{K} \phi_j(z)}, \ \phi_i(z) = \exp(-0.5(z - c_i)^2 / h_i)$$

Probabilistic model for a single trajectory:

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

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



$$\boldsymbol{w} = [w_1 \ w_2 \ w_3 \ ... \ w_K]^T$$

Trajectory Distributions

Trajectory distribution:

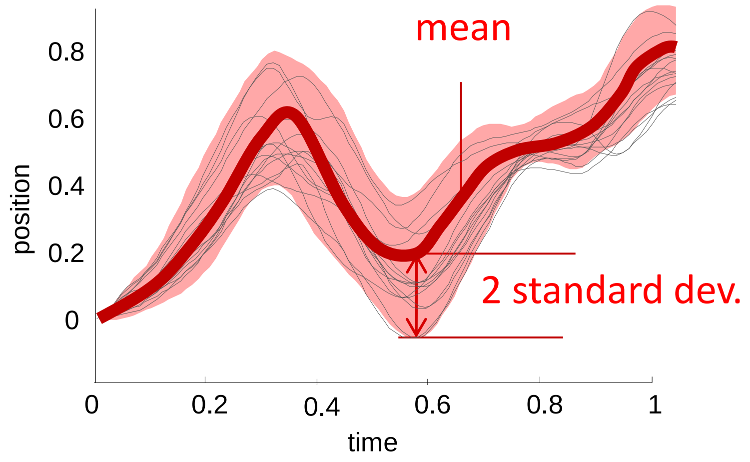
- Treat $\boldsymbol{w}$ as latent variable with distribution

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}})$$

- Integrate it out

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

Establishes correlation between time points



Multiple DoFs

How can we encode a distribution over multiple DoFs?

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

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

- 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:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}_t|\boldsymbol{\Phi}_t\boldsymbol{\mu}_{\boldsymbol{w}}, \sigma^2\boldsymbol{I} + \underbrace{\boldsymbol{\Phi}_t\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Phi}_t^T}_{\text{Establishes correlation between joints}})$$

Trajectory distributions for control

From ProMP:

- Compute derivative of mean and covariance using ProMP

$$\dot{\boldsymbol{\mu}}_t = \dot{\boldsymbol{\Phi}}_t \boldsymbol{\mu}_{\boldsymbol{w}}, \quad \dot{\boldsymbol{\Sigma}}_t = \dot{\boldsymbol{\Phi}}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Phi}_t^T + \boldsymbol{\Phi}_t \boldsymbol{\Sigma}_{\boldsymbol{w}} \dot{\boldsymbol{\Phi}}_t^T$$

From linear(ized) system model:

$$\dot{\boldsymbol{y}}_t = \boldsymbol{A}\boldsymbol{y}_t + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{b}$$

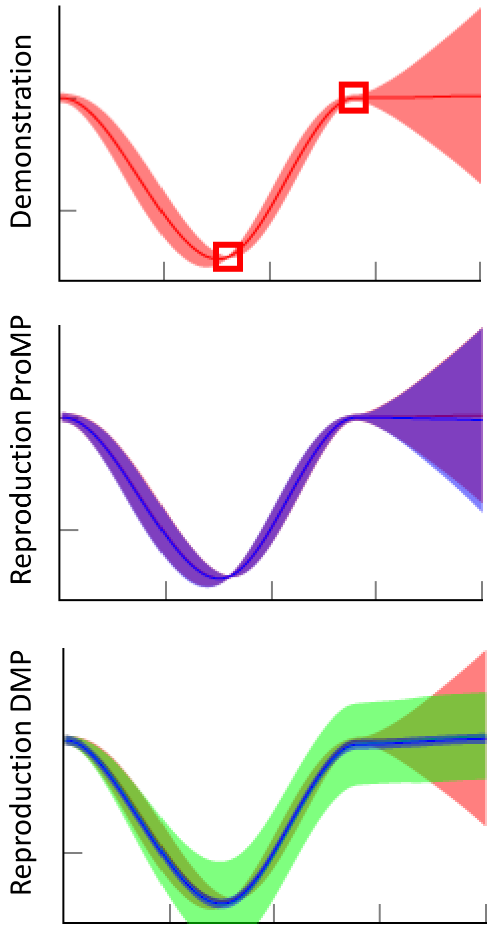
- Assume (stochastic) linear controller with time varying gains

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

- Compute derivative of mean and variance using this linear model

Match derivatives of mean and variance

- $\boldsymbol{K}_t$ and $\boldsymbol{k}_t$ can be obtained in closed form
- Variable stiffness controller



Adaptation of ProMPs

Adapt final/intermediate position/ of the movement

- **Conditioning/Bayes Theorem**

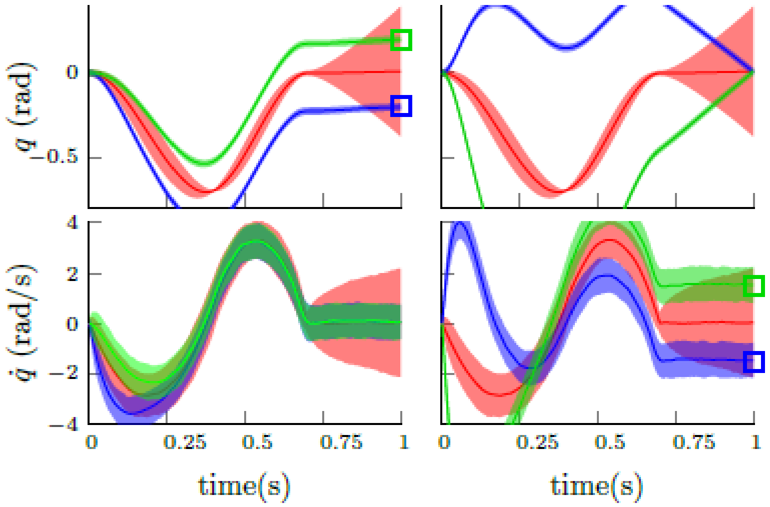
$$p(\boldsymbol{w}|\boldsymbol{y}_t = \boldsymbol{y}_t^*) = \frac{p(\boldsymbol{y}_t^*|\boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}_t^*)}$$

- **For Gaussian Distributions:**

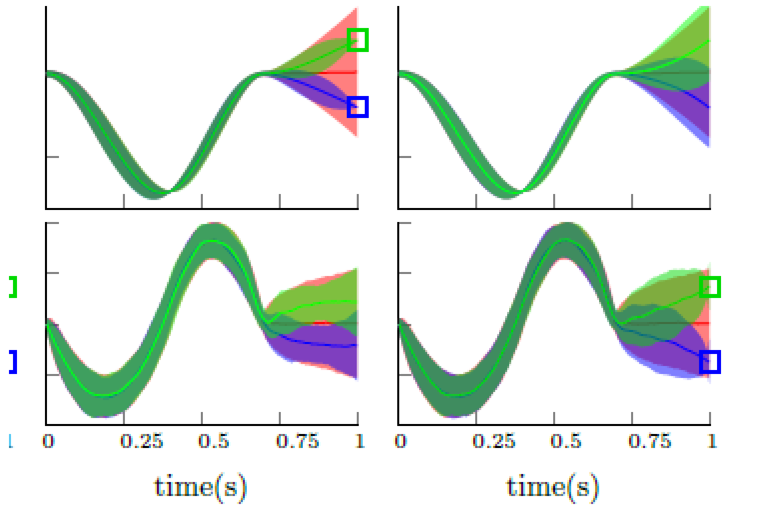
$$\boldsymbol{\mu}_{\boldsymbol{w}}^* = \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{L}\left(\boldsymbol{y}_t^* - \boldsymbol{\Psi}_t^T\boldsymbol{\mu}_{\boldsymbol{w}}\right) \qquad \boldsymbol{L} = \boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}_t\left(\boldsymbol{\Sigma}_{\boldsymbol{y}} + \boldsymbol{\Psi}_t^T\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}_t\right)$$

$$\boldsymbol{\Sigma}_{\boldsymbol{w}}^* = \boldsymbol{\Sigma}_{\boldsymbol{w}} - \boldsymbol{L}\boldsymbol{\Psi}_t^T\boldsymbol{\Sigma}_{\boldsymbol{w}}$$

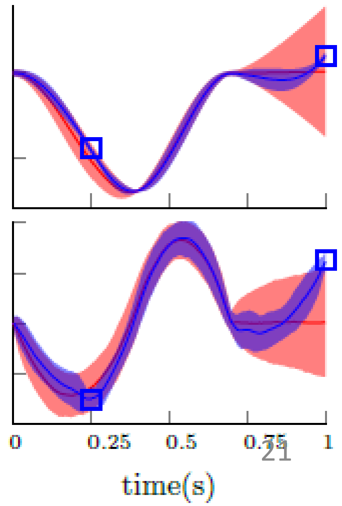
DMPs Position DMPs Velocities



ProMPs Position ProMPs Velocities



ProMPs Joint Conditioning



Outline

Dynamic Movement Primitives

Probabilistic Movement Primitives

- Introduction
- Learning ProMPs
- Case Study 1: Robot Table Tennis
- Case Study 2: Interaction Primitives
- Case Study 3: Prioritization of Primitives

Learning ProMPs

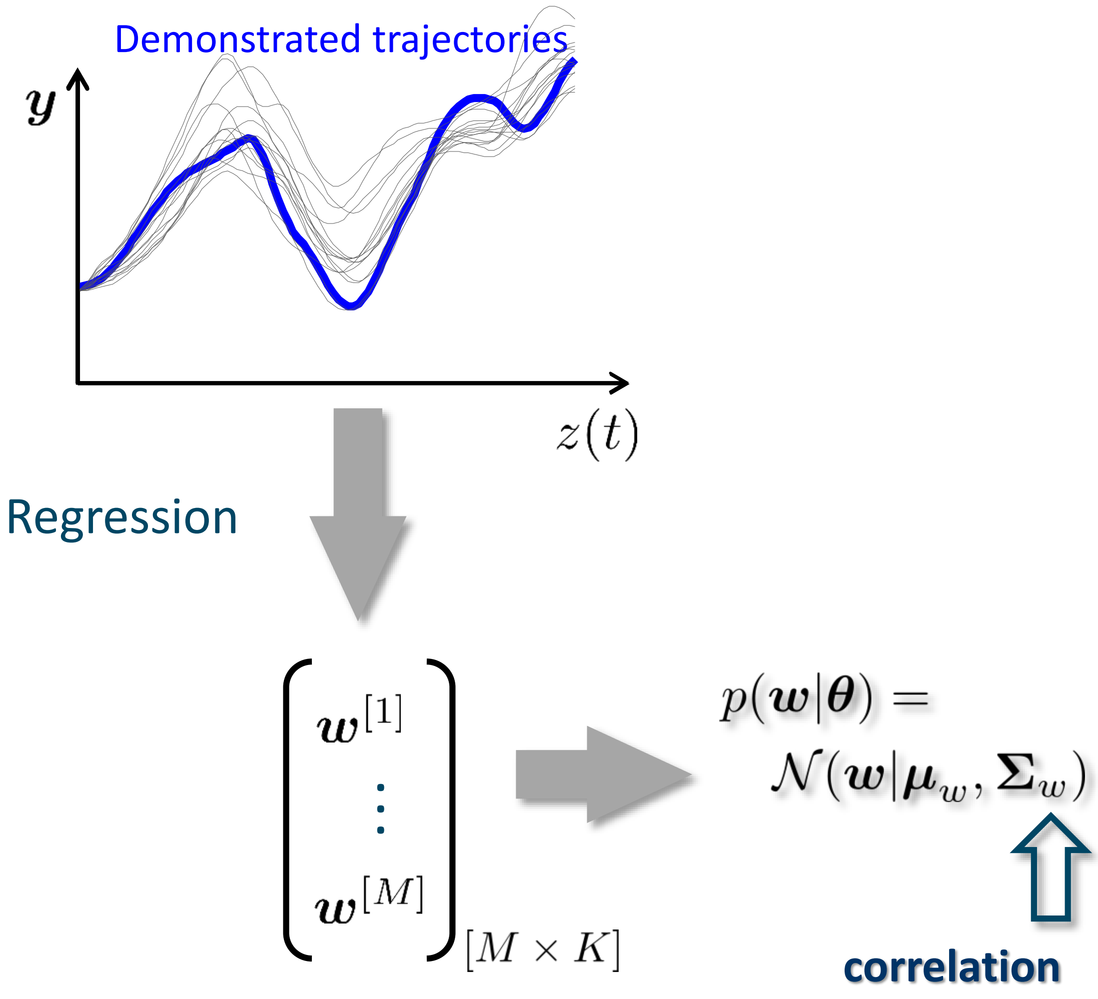
For each trajectory $\boldsymbol{\tau}_i$, obtain $\boldsymbol{w}_i$

$$\boldsymbol{w}_i = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{\tau}_i$$

Compute mean and variance

$$\boldsymbol{\mu}_{\boldsymbol{w}} = \frac{1}{N}\sum_i \boldsymbol{w}_i$$

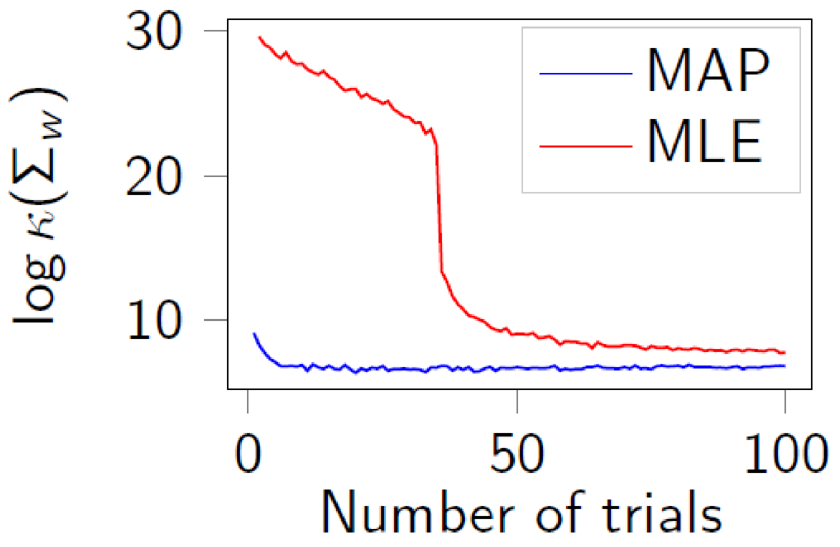
$$\boldsymbol{\Sigma}_{\boldsymbol{w}} = \frac{\sum_i (\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})(\boldsymbol{w}_i - \boldsymbol{\mu}_{\boldsymbol{w}})^T}{N-1}$$



Bayesian Learning of ProMPs

- **Maximum likelihood solution:**
  - Overfitting (large number of parameters)
  - Needs a lot of data
  - Numerical issues
- **Stability can be improved by:**
  - × Artificial noise (add inaccuracies)
  - × Reduce complexity (model joints as independent)
  - ✓ Bayesian regularization

Log Condition Number of Σ_w



- Number of Trajectories approx. number of weights for MLE
- Otherwise conditioning infeasable

Maximum A-Posteriori

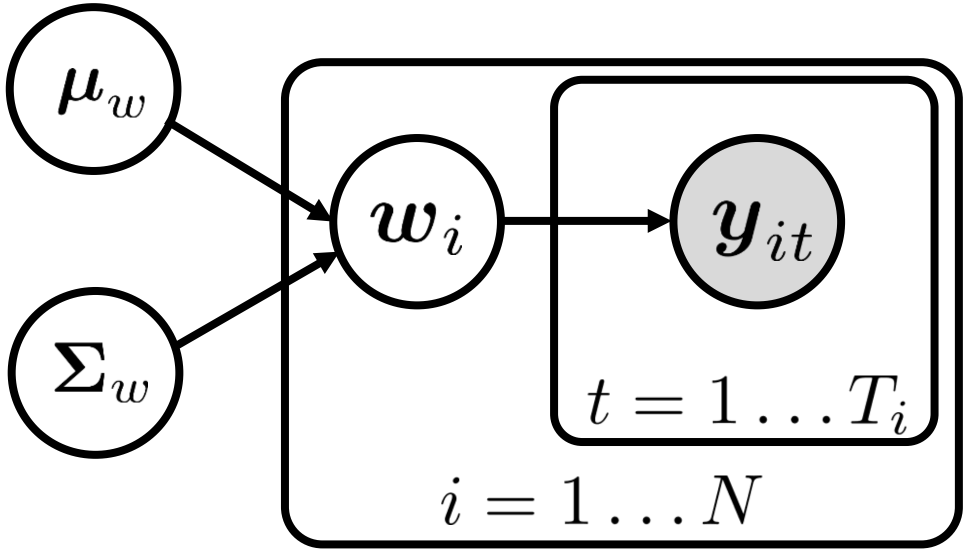
- **Prior distribution over ProMP parameters**

$$p(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) = \mathrm{NIW}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w | k_0, \boldsymbol{m}_o, v_o \boldsymbol{S}_0)$$
$$= \mathcal{N}\left(\boldsymbol{\mu}_w | \boldsymbol{m}_0, \frac{1}{k_0}\boldsymbol{\Sigma}_0\right) \mathcal{W}^{-1}(\boldsymbol{\Sigma}_w | v_0, \boldsymbol{S}_0)$$

  - Conjugate prior distribution
  - Prior and posterior have the same form
  - Encodes that DoFs are uncorrelated

- **MAP estimate**

$$\arg\max_{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w} \underbrace{\left(\prod_i p(\boldsymbol{\tau}_i | \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)\right)}_{\text{Likelihood}} \underbrace{p(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)}_{\text{Prior}}$$



Training MAP ProMPs

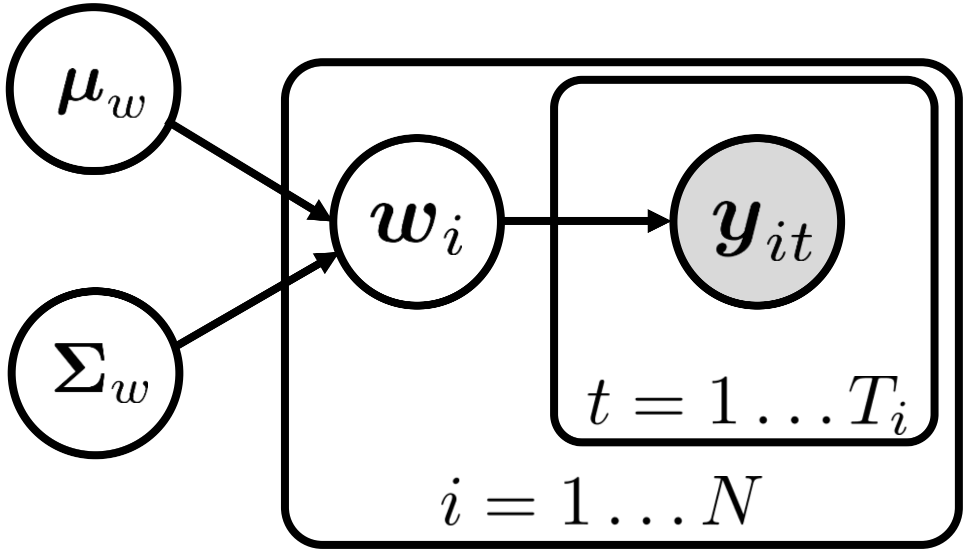
- **Closed form update solutions**

$$\boldsymbol{\mu}_w = \frac{k_0 \boldsymbol{m}_0 + \sum_i^N \boldsymbol{w}_i}{N + k_0}$$

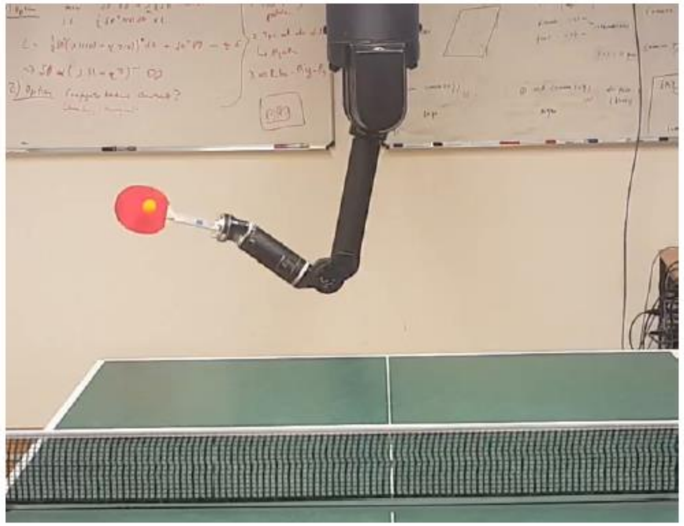
$$\boldsymbol{\Sigma}_w = \frac{v_0 \boldsymbol{S}_0 + \sum_i^N (\boldsymbol{w}_i - \boldsymbol{\mu}_w)(\boldsymbol{w}_i - \boldsymbol{\mu}_w)^T}{N + v_0 + 1}$$

We ignored that there is also uncertainty on w

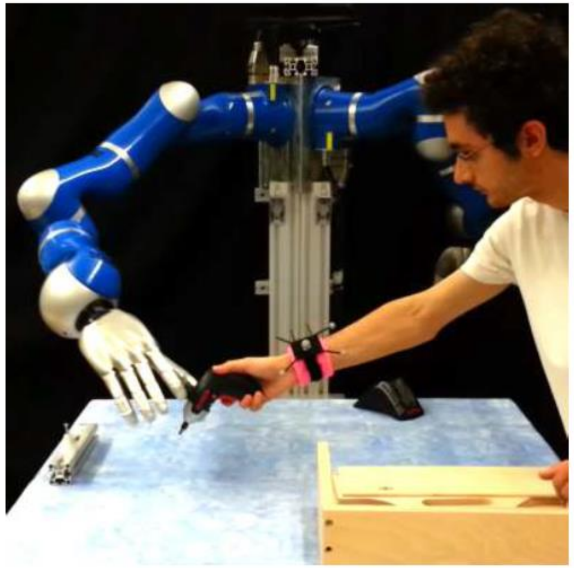
- In particular if we have partial trajectories
- Uncertainty depends on $p(\boldsymbol{w})$
- Training with EM (leads to better solutions)



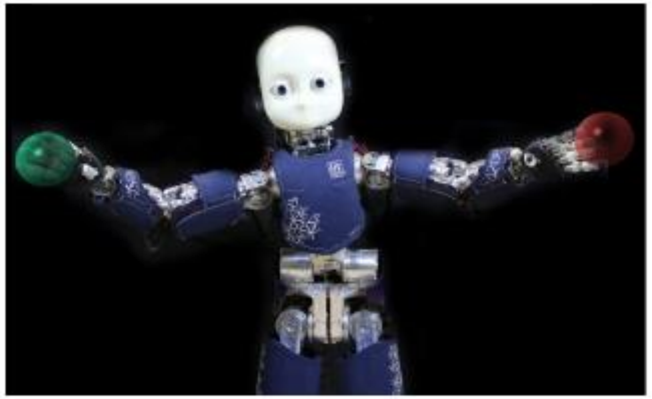
3 case studies



Robot Table Tennis



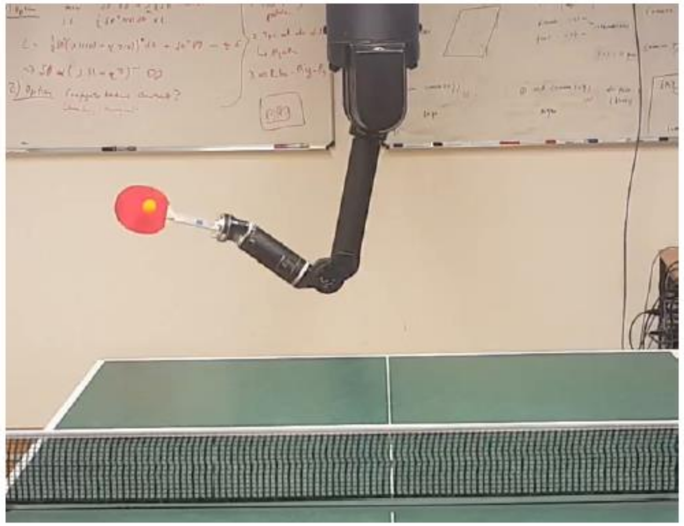
Learning Interaction Primitives



Prioritization of Movement Primitives

Robot Table Tennis

- **Learning:**
  - Demonstrate different table tennis strikes (6 demonstrations)
  - Learn joint correlation using MAP estimate
- **Testing:**
  - Predict incoming ball position (Kalman filtering)
  - Condition on racket position (i.e. Inverse Kinematics)
  - No need to specify orientation (learned from data)
  - Test learning joint correlation vs. no correlation



Conditioning in Task Space

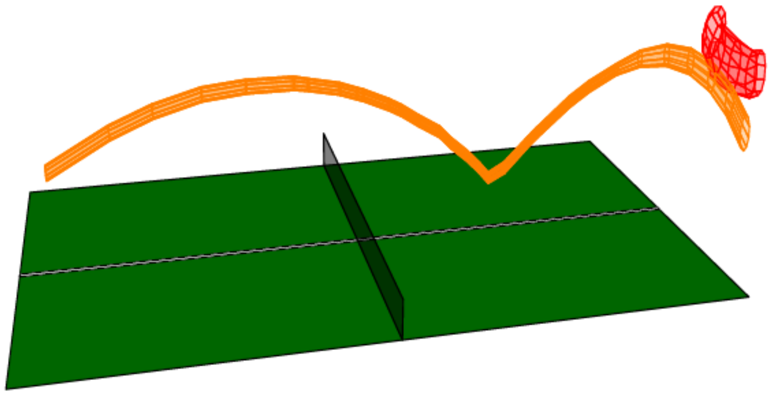
- Desired end-effector position: $p(\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$
- Forward kinematics: $p(\boldsymbol{x}_t | \boldsymbol{y}_t) = \delta\left(\boldsymbol{x}_t - f(\boldsymbol{y}_t)\right)$
- Posterior:

$$
\begin{aligned}
p(\boldsymbol{y}_t | \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) &= \int p(\boldsymbol{y}_t | \boldsymbol{x}_t) p(\boldsymbol{x}_t | \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) d\boldsymbol{x}_t \\
&\propto p(\boldsymbol{y}_t) \int p(\boldsymbol{x}_t | \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) p(\boldsymbol{x}_t | \boldsymbol{y}_t) d\boldsymbol{x}_t \\
&\propto p(\boldsymbol{y}_t) \mathcal{N}(f(\boldsymbol{y}_t) | \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)
\end{aligned}
$$

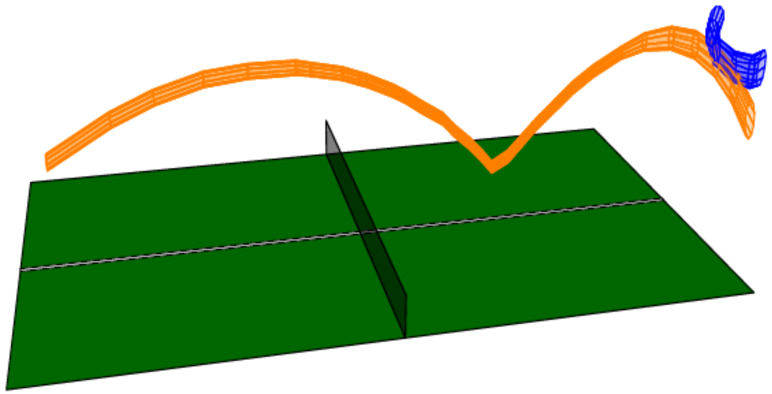
Bayes Theorem

Dirac Delta

Prior Racket Distribution



Posterior Racket Distribution



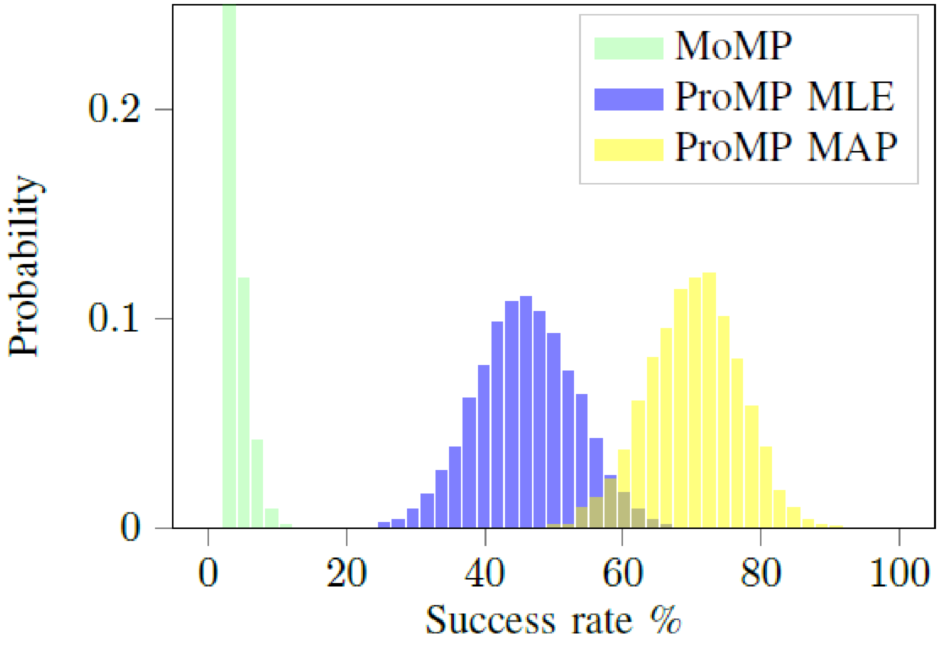
Conditioning in Task Space

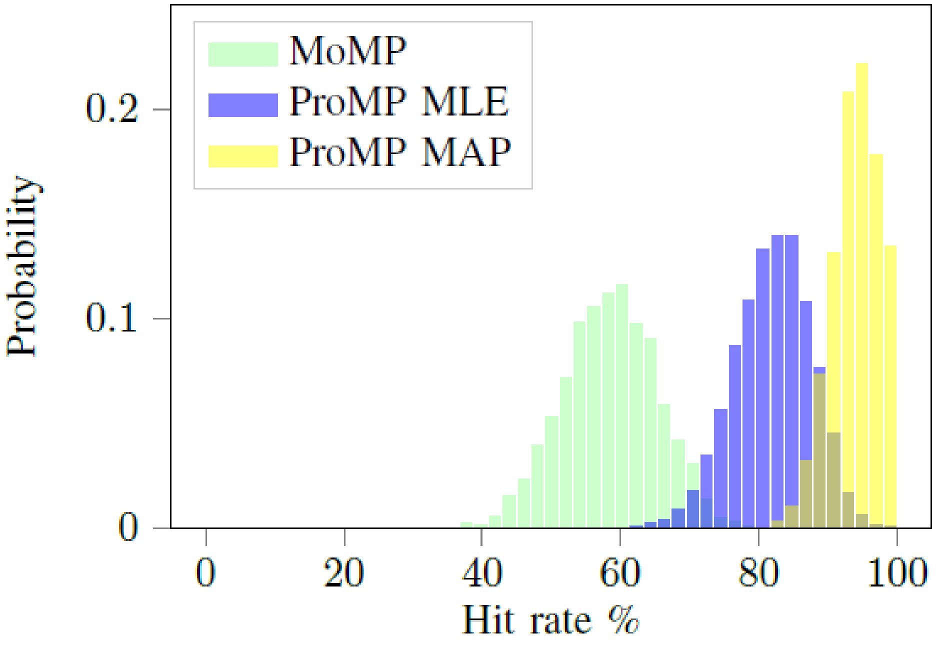
- **Laplace approximation:** $$p(\boldsymbol{y}_t|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \propto p(\boldsymbol{y}_t)\mathcal{N}(f(\boldsymbol{y}_t)|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \approx \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$$
    - Mean: $$\boldsymbol{\mu}_y \leftarrow \arg\max_{\boldsymbol{y}_t} \log p(\boldsymbol{y}_t|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$$
    - Covariance: $$\boldsymbol{\Sigma}_y \leftarrow \nabla_{\boldsymbol{y}_t \boldsymbol{y}_t} \log p(\boldsymbol{y}_t|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)\Big|_{\boldsymbol{y}_t=\boldsymbol{\mu}_y}$$
- Use $\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y$ to condition in joint space

$$\boldsymbol{\mu}_{\boldsymbol{w}}^* = \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{L}\left(\boldsymbol{\mu}_y - \boldsymbol{\Psi}_t^T \boldsymbol{\mu}_{\boldsymbol{w}}\right) \qquad \boldsymbol{L} = \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t \left(\boldsymbol{\Sigma}_{\boldsymbol{y}} + \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_{\boldsymbol{w}} \boldsymbol{\Psi}_t\right)$$

$$\boldsymbol{\Sigma}_{\boldsymbol{w}}^* = \boldsymbol{\Sigma}_{\boldsymbol{w}} - \boldsymbol{L}\boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_{\boldsymbol{w}}$$

Results on Table Tennis



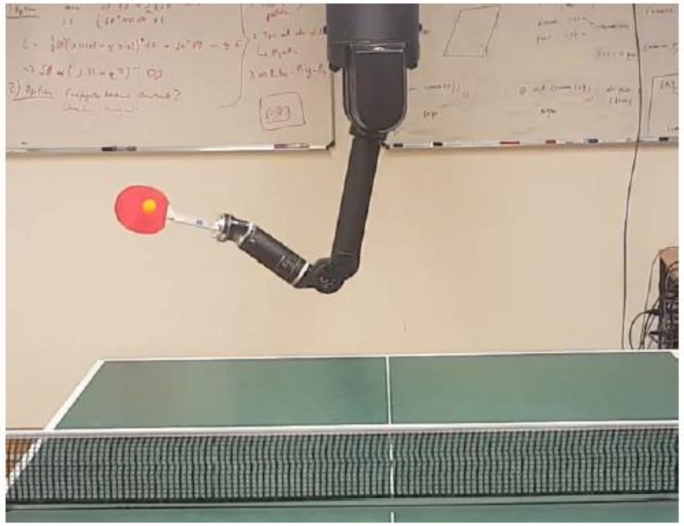


MoMP: State of the art approach for Table Tennis
ProMP MLE: DoFs are modelled independently
ProMP MAP: Correlation between DoFs is learned

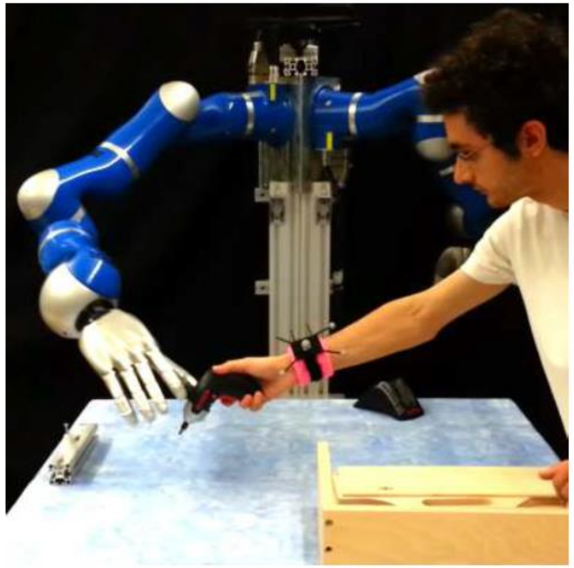
Results on Table Tennis

- Video

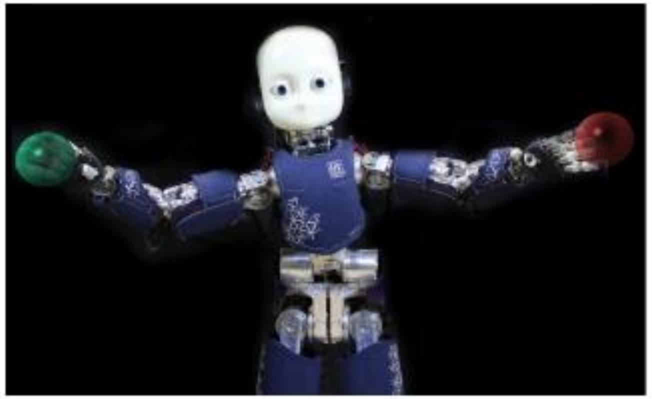
3 case studies



Robot Table Tennis



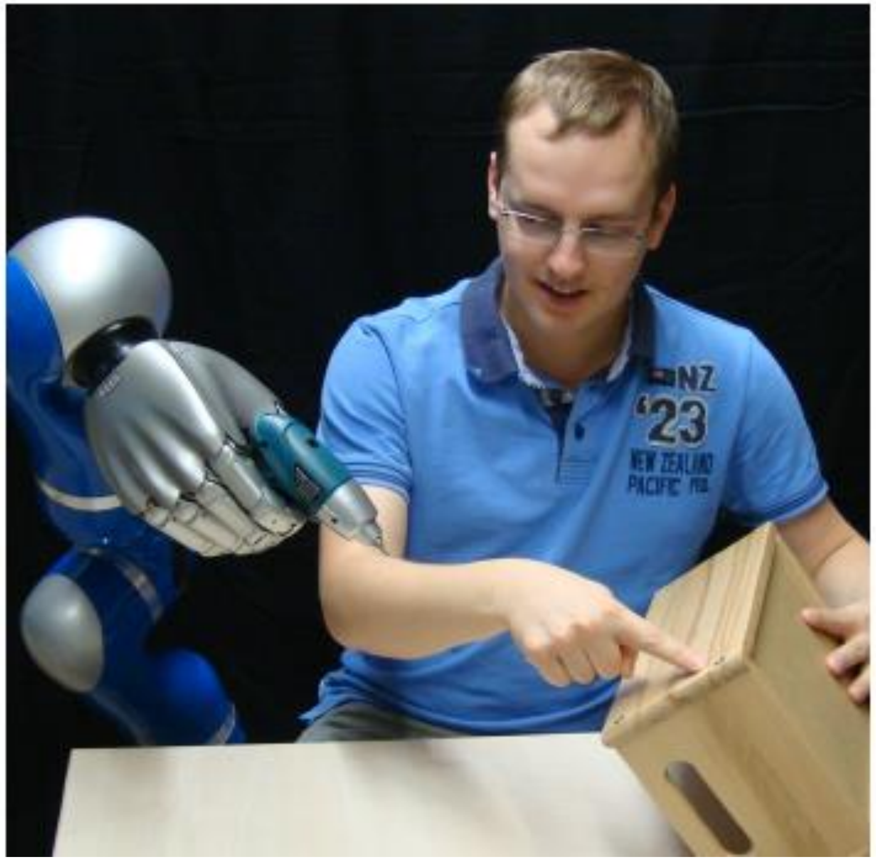
Learning Interaction Primitives



Prioritization of Movement Primitives

Robot Companions

- Inherently safe
- Assisting humans
- Couple with / react to human movement
- Huge variability of tasks
- Simple to teach new interaction patterns



Learning Collaborative Models

Correlating all robot and human DoFs

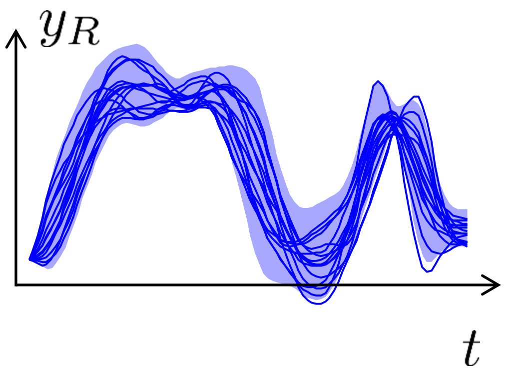
$$
\begin{matrix} \boldsymbol{w}^{[1]} = \\ \\ \boldsymbol{w}^{[M]} = \end{matrix}
\begin{pmatrix}
\boldsymbol{w}_1^{T[1]} \dots \boldsymbol{w}_Q^{T[1]} & \boldsymbol{w}_1^{T[1]} \dots \boldsymbol{w}_P^{T[1]} \\
\ddots & \vdots \\
\boldsymbol{w}_1^{T[M]} \dots \boldsymbol{w}_Q^{T[M]} & \boldsymbol{w}_1^{T[M]} \dots \boldsymbol{w}_P^{T[M]}
\end{pmatrix}_{[M \times K(P+Q)]}
$$

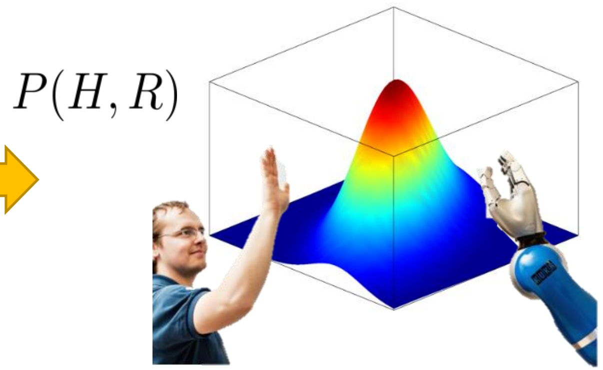
[Maeda, et al., AURO, IJRR]

Learning Collaborative Models

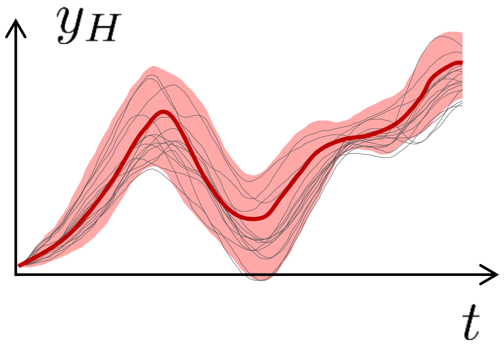
- Learn **joint distribution** of trajectory τ_h and robot trajectory τ_r from demonstrations

Robot





Human



$$p(\boldsymbol{w}; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{w} | \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$$

$$\boldsymbol{\Sigma}_w = \begin{bmatrix} \boldsymbol{\Sigma}_H & \boldsymbol{\Sigma}_{HR} \\ \boldsymbol{\Sigma}_{RH} & \boldsymbol{\Sigma}_R \end{bmatrix}$$

Coupling between Human and Robot

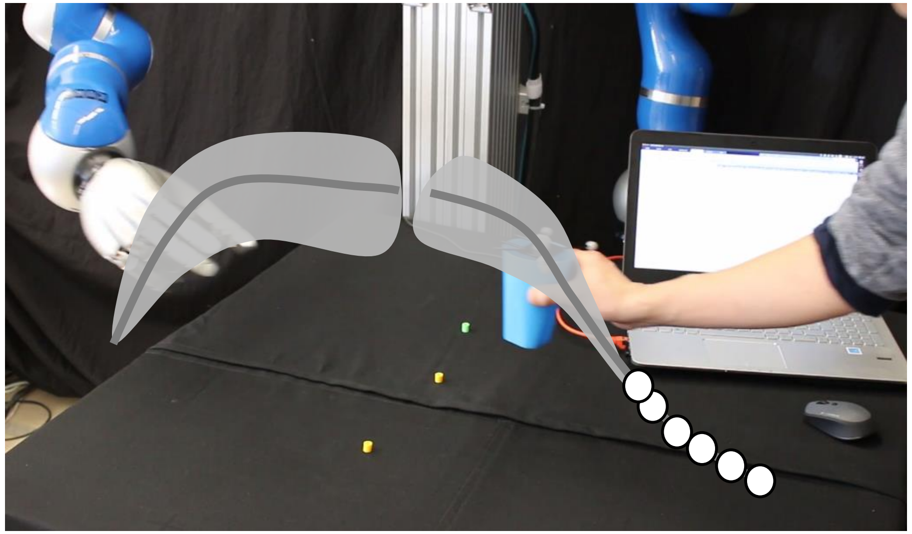
Maeda, G. et al. *Probabilistic Movement Primitives, for Coordination of Multiple Human-Robot Collaborative Tasks,* Autonomous Robots (AURO) 2016

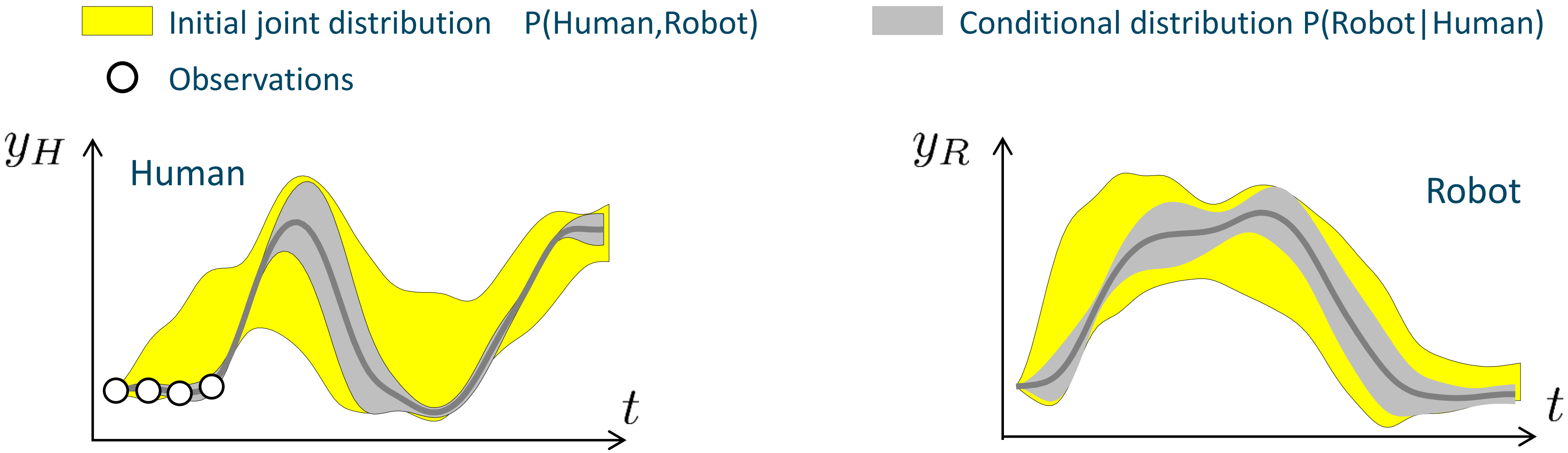
Coordinating motions with the human

Condition on observation of human:

$$p(\boldsymbol{w}|\boldsymbol{h}_{1:t}) = \mathcal{N}(\boldsymbol{\mu}_w^{\text{new}}, \boldsymbol{\Sigma}_w^{\text{new}})$$

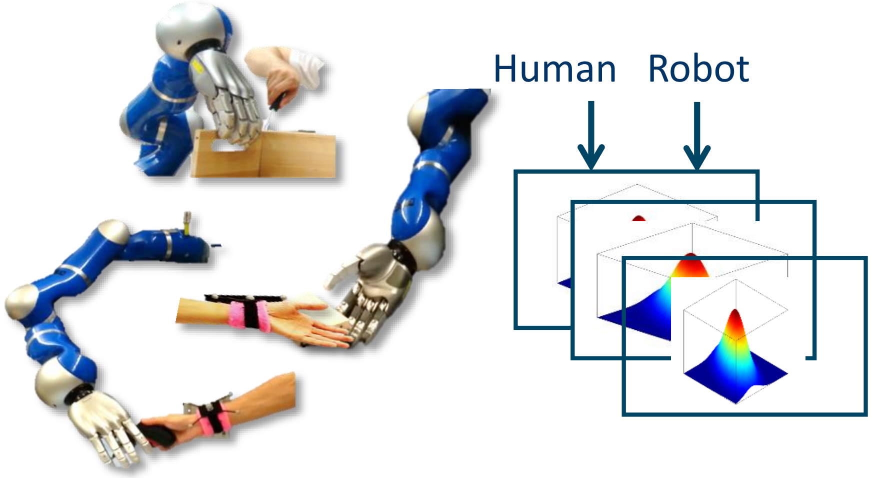
$$\boldsymbol{\mu}_w^{new} = \boldsymbol{\mu}_w + \boldsymbol{K}(\boldsymbol{h}_{1:t} - \boldsymbol{\Psi}_{1:t}\boldsymbol{\mu}_w)$$
$$\boldsymbol{\Sigma}_w^{new} = \boldsymbol{\Sigma}_w - \boldsymbol{K}(\boldsymbol{\Psi}_{1:t}\boldsymbol{\Sigma}_w)$$
$$\boldsymbol{K} = \boldsymbol{\Sigma}_w \boldsymbol{\Psi}_{1:t}^T(\boldsymbol{\Sigma}_h + \boldsymbol{\Psi}_{1:t}\boldsymbol{\Sigma}_w\boldsymbol{\Psi}_{1:t}^T)^{-1}$$





Training multiple interactions

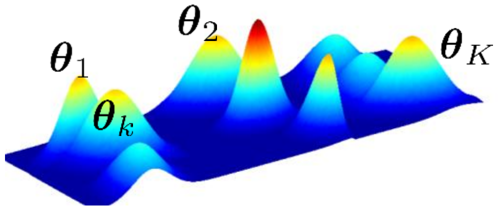
Model/training by imitation learning



Mixture of Interaction Primitives:

$$p(\boldsymbol{\tau}) = \sum_{k=1}^{K} \alpha_k p(\boldsymbol{\tau}|\boldsymbol{\theta}_k)$$

- Mixture coefficients: α_k
- Mixture components: $p(\boldsymbol{\tau}|\boldsymbol{\theta}_k) = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$



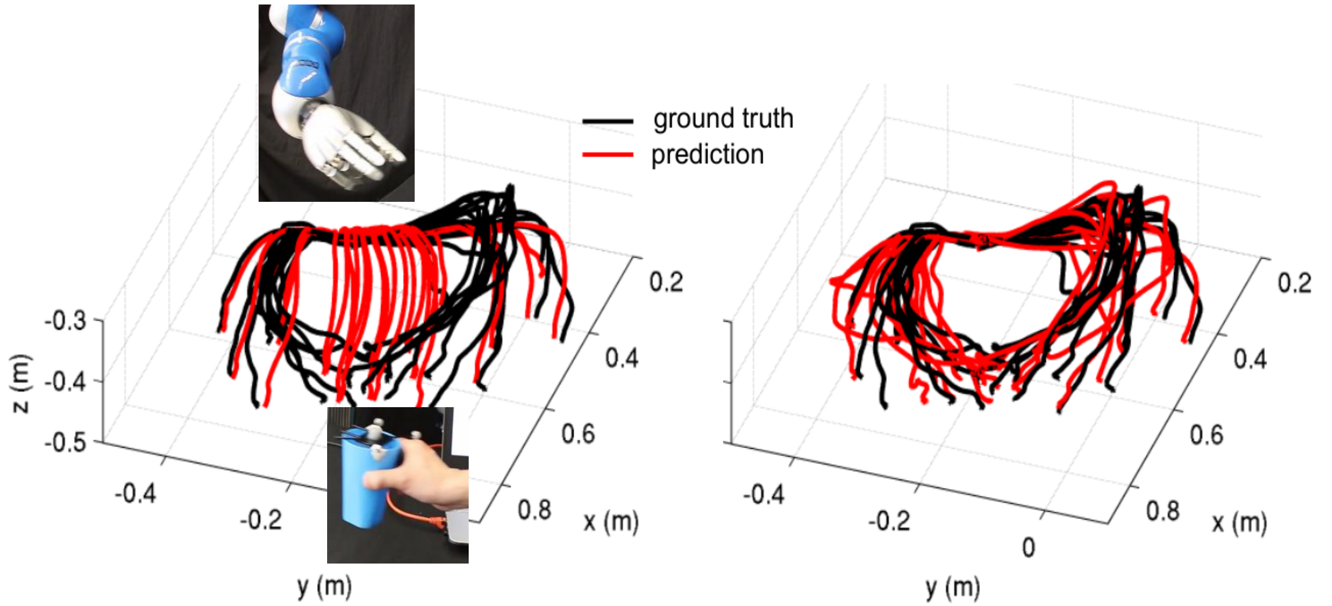
$$\theta_1,\ \theta_2,\ \dots\theta_N$$

Can be learned by EMM for GMMs

Multi-Modal Demonstrations

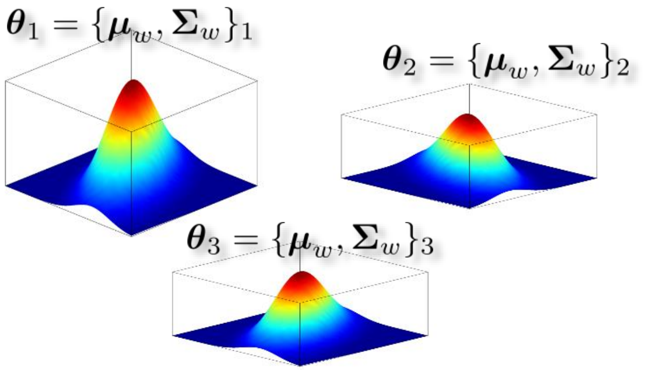
Mixture models also help to overcome Gaussian assumption

- Non-linear correlations
- Multi-modality



Identify current interaction

Interaction Primitives (trained independently or with EM)

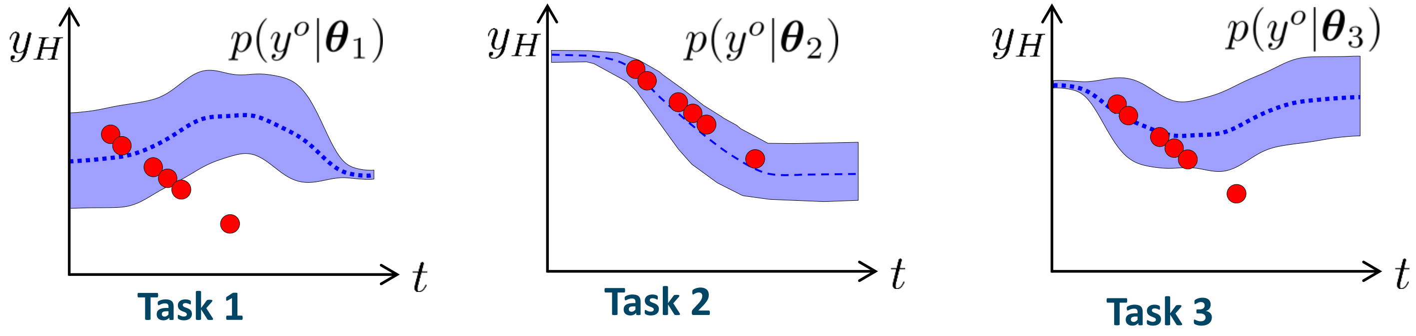


Identify task executed by the human

$$p(C_k|y^o) \propto p(y^o|\boldsymbol{\theta}_k) \; p(C_k)$$

Posterior Likelihood Prior

$$p(y^o|\boldsymbol{\theta}_k) = \mathcal{N}(y^o; \boldsymbol{\mu}_w^k, \boldsymbol{\Sigma}_w^k)$$

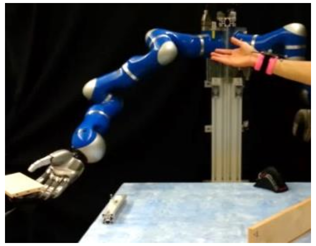


Teaching a Robot Assistant

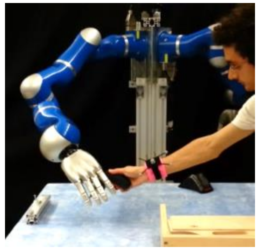
Box assembly task:

- **Learn interaction patterns** by kinesthetic teach in

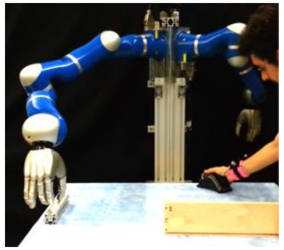
Plate handover

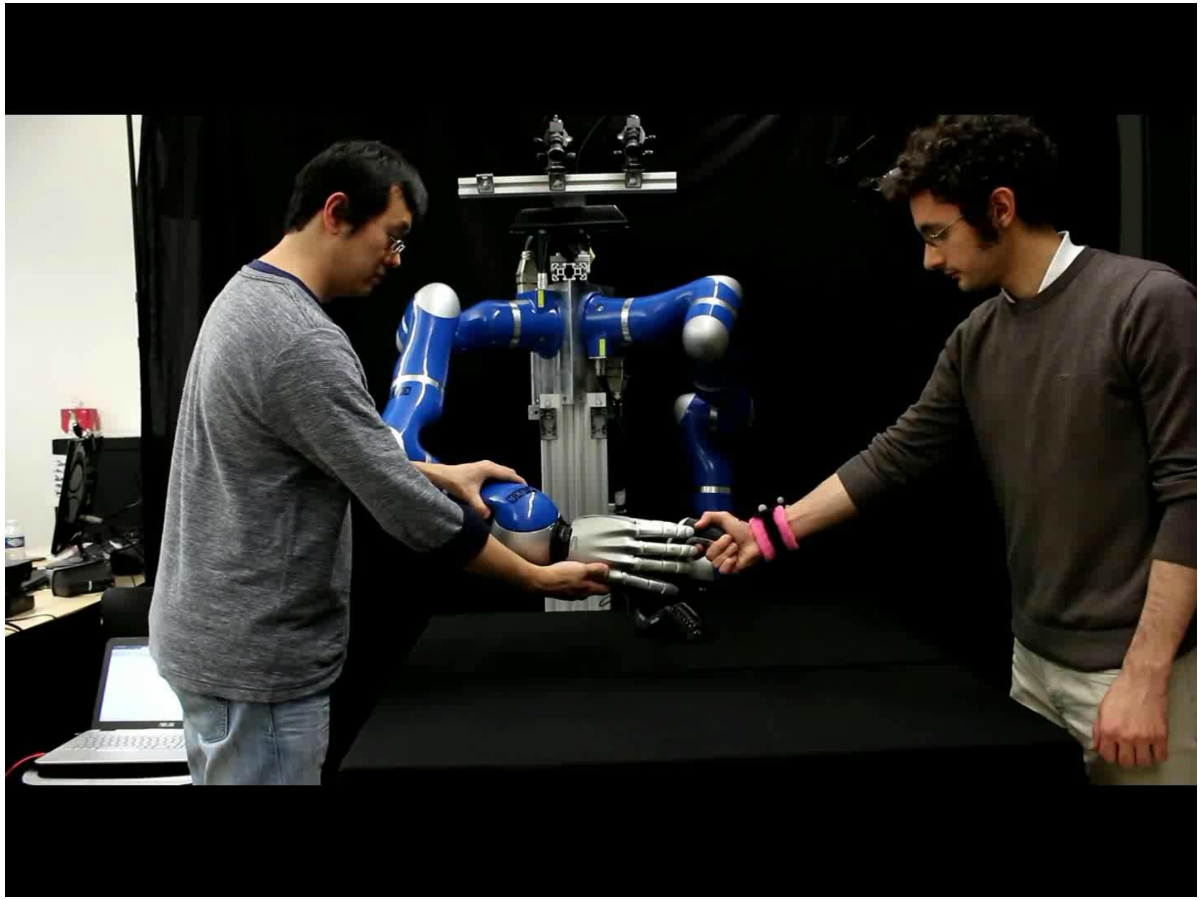


Holding tool



Screw handover





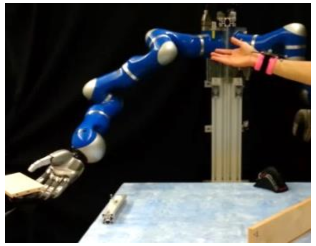
Maeda, G.; **Neumann, G.**; et al. *Probabilistic Movement Primitives, for Coordination of Multiple Human-Robot Collaborative Tasks,* Autonomous Robots (AURO)

Teaching a Robot Assistant

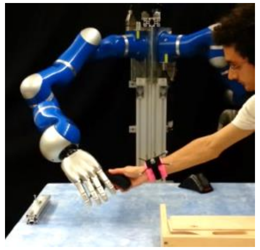
Box assembly task:

- **Learn interaction patterns** by kinesthetic teach in

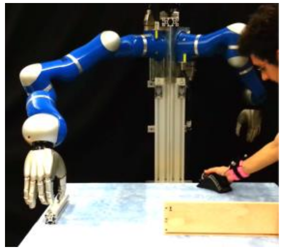
Plate handover



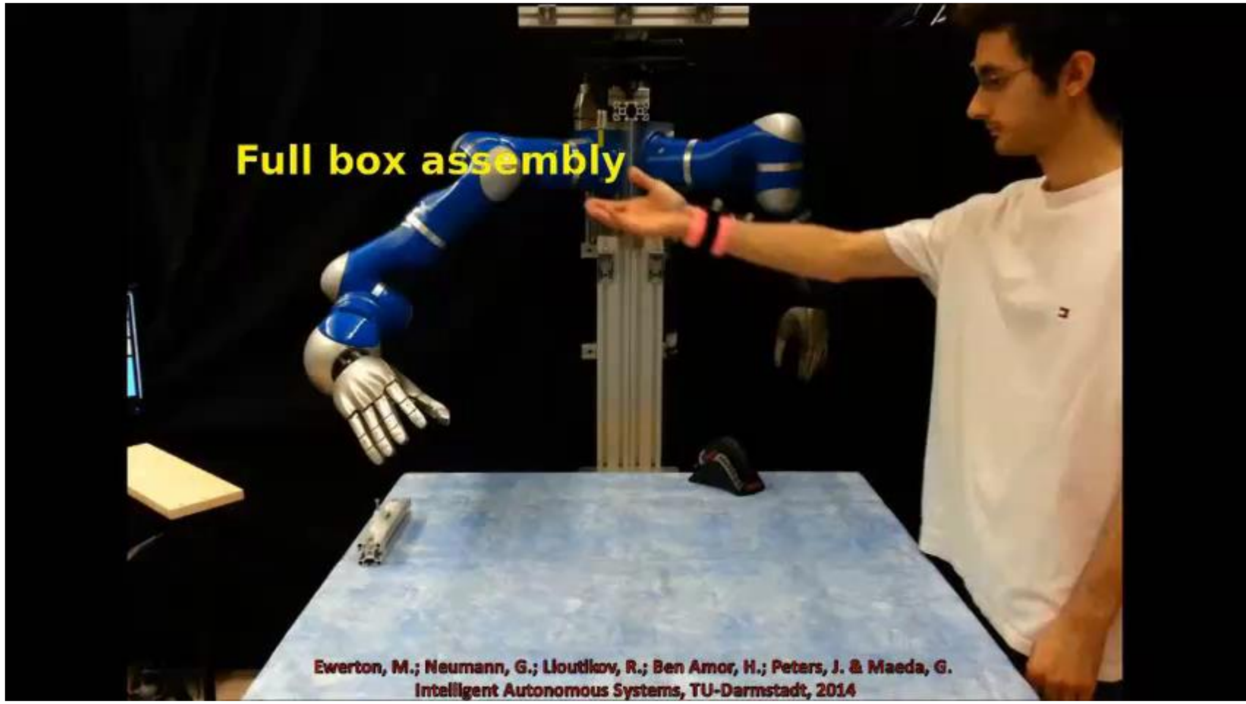
Holding tool



Screw handover

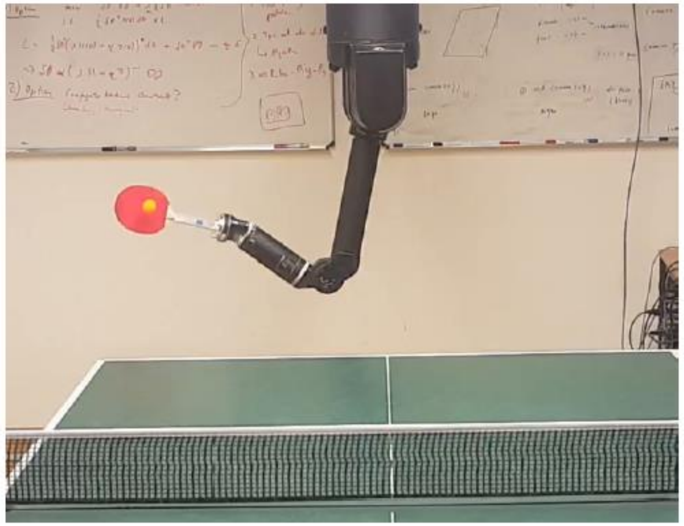


- **Couple robot movement** with human

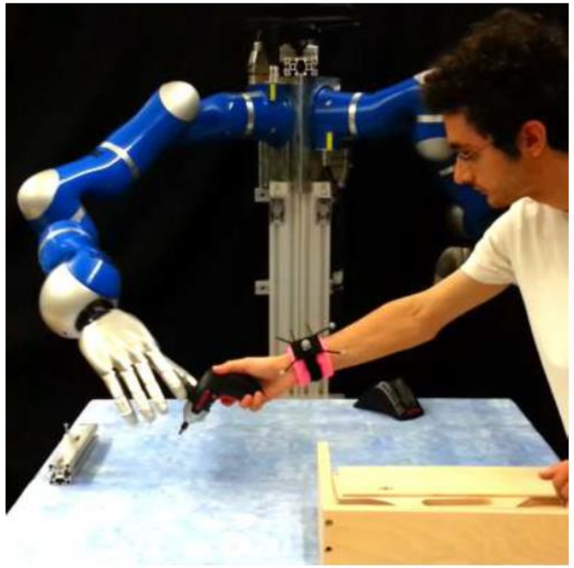


Maeda, G.; **Neumann, G.**; et al. *Probabilistic Movement Primitives, for Coordination of Multiple Human-Robot Collaborative Tasks,* Autonomous Robots (AURO)

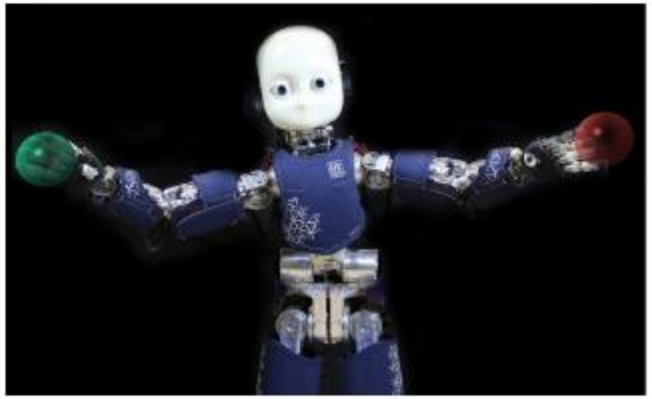
3 case studies



Robot Table Tennis



Learning Interaction Primitives



Prioritization of Movement Primitives

Combination of Skills

Modularity:

- Learn individual skills to achieve certain tasks
- Combine skills to solve a combination of tasks
- In theory: much smaller skill library needed

How can we combine skills in a useful ways?

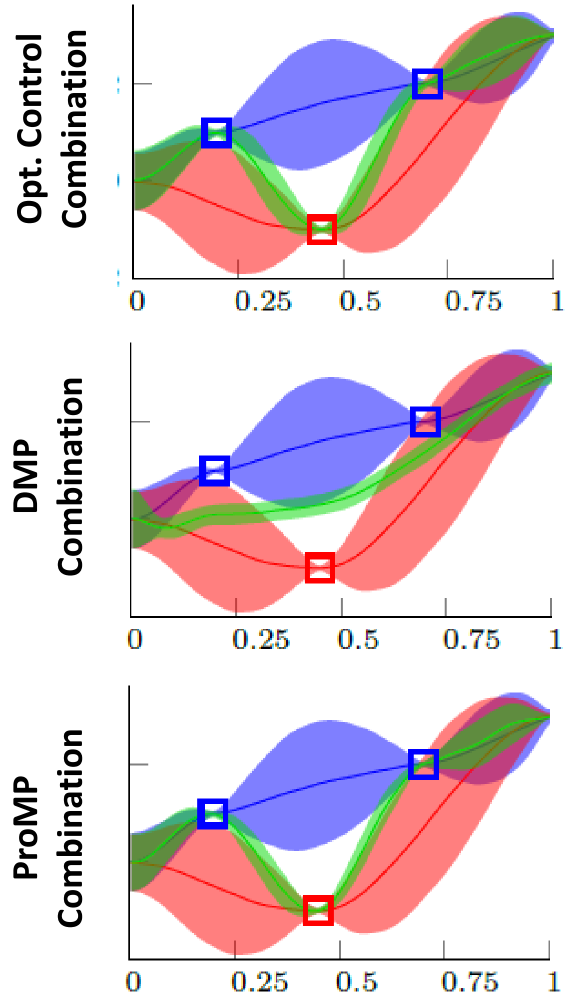
Coactivation

Coactivate primitives to solve a combination of tasks

- Implemented as **product of distributions**
- Area, in which all distributions have high probability

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

- $p_i(\boldsymbol{q}_t) \ldots$ i-th movement primitive
  $\alpha_i(t) \ldots$ activation factors

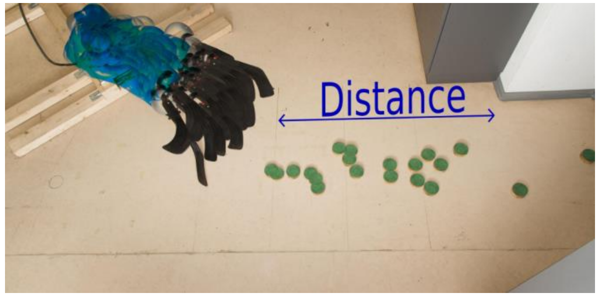


Example: Robot Hockey

7-link KUKA robot arm, playing hockey

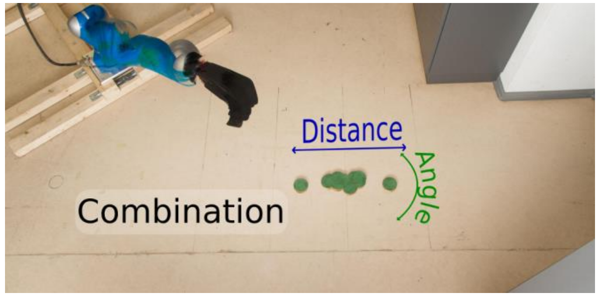
- Train 2 primitives with high variance in shooting angle or in distance

Demonstration 1



Demonstration 2

- Product of the primitives: **Combination of both tasks**



Combination in Controller Space

- ProMP provides stochastic variable stiffness controller
  - Joint-space ProMP: $p(\ddot{\boldsymbol{q}}) = \mathcal{N}\left(\boldsymbol{K}_t \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} + \boldsymbol{k}_t, \boldsymbol{\Sigma}_{\ddot{q}}\right) = \mathcal{N}\left(\boldsymbol{\mu}_{\ddot{q}}, \boldsymbol{\Sigma}_{\ddot{q}}\right)$
  - Task-space ProMP: $p(\ddot{\boldsymbol{x}}) = \mathcal{N}\left(\boldsymbol{\mu}_{\ddot{x}}, \boldsymbol{\Sigma}_{\ddot{x}}\right)$

- Variance can be propagated in task space

$$p(\ddot{\boldsymbol{x}}|\ddot{\boldsymbol{q}}, \Sigma_{\ddot{x}}) = \mathcal{N}\left(\boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}, \boldsymbol{\Sigma}_{\ddot{x}}\right)$$

- Bayes theorem

$$p(\ddot{\boldsymbol{q}}|\ddot{\boldsymbol{x}}) = \frac{p(\ddot{\boldsymbol{x}} = \boldsymbol{\mu}_{\ddot{x}}|\ddot{\boldsymbol{q}}, \boldsymbol{\Sigma}_{\ddot{x}})p(\ddot{\boldsymbol{q}})}{p(\ddot{\boldsymbol{x}})}$$

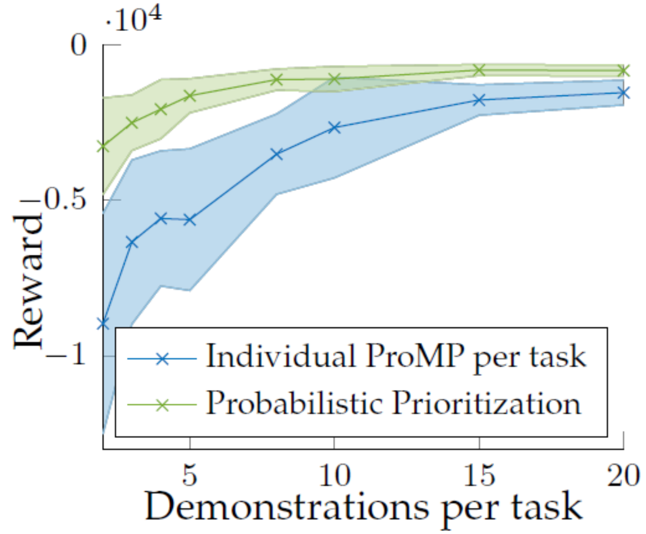
Prioritized Combination of ProMPs

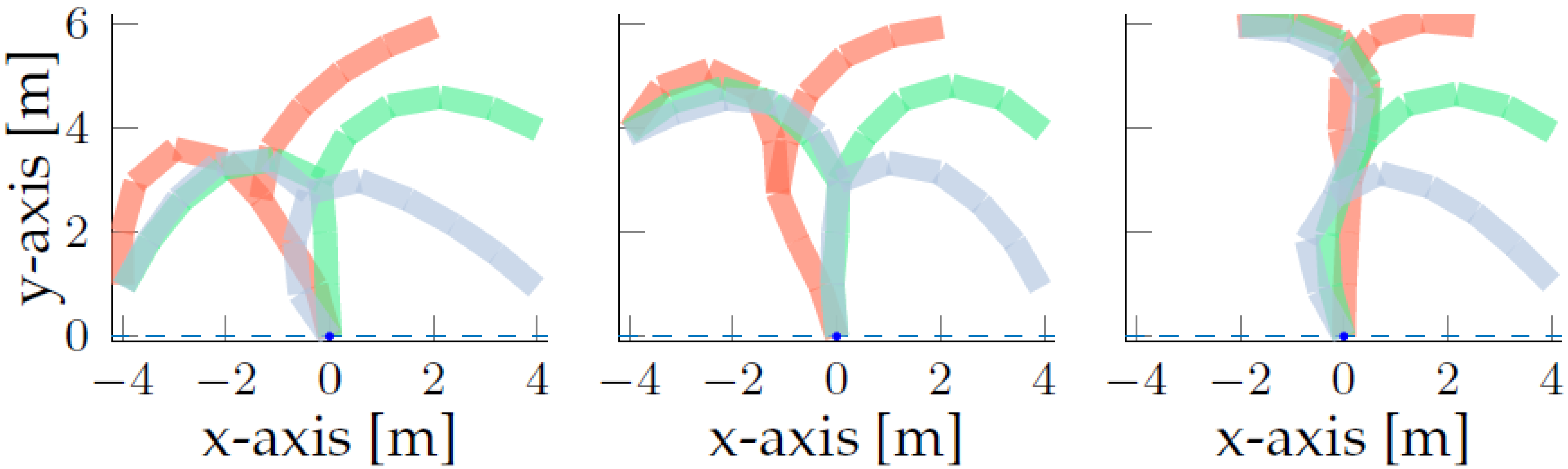
- Yields well known prioritized control laws
- Variances define soft-priorities between "tasks"

[Paraschos et al, Probabilistic Prioritization of Movement Primitives, RAL 2017]

Illustration for learning multiple tasks

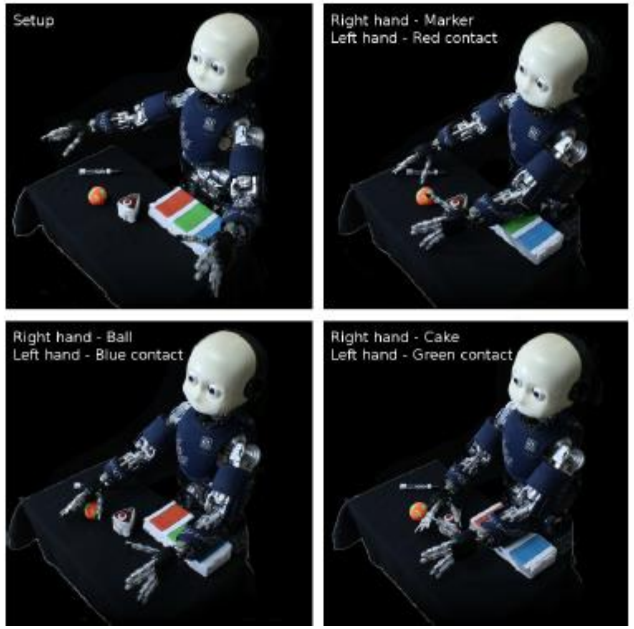
- Each end-effector has three tasks
- Nine task-combinations in total
- Using our approach, we can learn the tasks per end-effector
- Results in more sample efficient learning





I-Cub experiments

- Robot executes **multiple** tasks concurrently
- Joint stabilization control law
- Upper-body control (torso, left and right arms)



	Left Task Err. (cm)	Right Task Err. (cm)
Blue — Marker	2.38 ± 0.91	3.09 ± 1.22
Blue — Ball	2.34 ± 0.96	3.18 ± 1.10
Blue — Cake	2.05 ± 0.71	3.56 ± 1.45
Green — Marker	2.21 ± 0.64	1.70 ± 0.81
Green — Ball	2.47 ± 0.89	2.28 ± 1.26
Green — Cake	2.97 ± 0.84	3.85 ± 1.02
Red — Marker	3.67 ± 0.76	2.89 ± 1.66
Red — Ball	2.82 ± 0.75	2.43 ± 1.13
Red — Cake	3.31 ± 1.26	4.23 ± 1.62

Conclusion

Trajectory distributions are a powerful representation:

- Conditioning
- Movement coupling
- Variable stiffness
- Joint correlations

However:

- We have to know execution time / phase
- Estimate phase online?
- Only local policy

Important open issues

- Perceptual coupling (vision, tactile, etc)
- Forceful interactions
- Include online model learning
- Selection and switching of primitives