Representing Movement Primitives as Implicit Dynamical Systems learned from Multiple Demonstrations

Robert Krug and Dimitar Dimitrov

Center for Applied Autonomous Sensor Systems (AASS)
Örebro University, Sweden
robert.krug@oru.se
Dynamical Movement Primitives (DMP) [Ijspeert et al., 2002]

- Feedback controllers in joint/task space . . .
- . . . formulated as one dynamical system per DoF: \( \dot{x}(t) = f(x(t), s(t)) \)
- Common phase variable \( s(t) \) to synchronize DoF
Feedback controllers in joint/task space . . .

. . . formulated as one dynamical system per DoF: \( \dot{x}(t) = f(x(t), s(t)) \)

Common phase variable \( s(t) \) to synchronize DoF

“On-the-fly” motion profile generation: \( x(t) = \int_0^t f(x(\tau), s(\tau)) d\tau \)
Outline

1. Motivation
2. Concept
3. Results
4. Contributions & Outlook
Why use primitive motion controllers?

Generate desired motions for a platform with many DoF

*Shadow Hand & Arm with 24 DoF*
Why use primitive motion controllers?

Generate desired motions for a platform with many DoF

- Controllers $\dot{x} = f(x, s)$ are state policies
  - Replaces explicit planning
  - Disturbance compensation

- Time synchronization of arbitrary many DoF

*Shadow Hand & Arm with 24 DoF*
Motivation

Why use primitive motion controllers?

Generate desired motions for a platform with many DoF

- Controllers $\dot{x} = f(x, s)$ are state policies
  - Replaces explicit planning
  - Disturbance compensation

- Time synchronization of arbitrary many DoF

- Motions resemble demonstrations

- Simple implementation

*Shadow Hand & Arm with 24 DoF*
Motivation

What’s the problem?

DMP [Ijspeert et al., 2002]: Stable spring excited by a learned control input $u$

\[
\dot{x}(t) = f(x, s) = Ax(t) + Bu(s; p),
\]

\[
x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \in \mathbb{R}^2
\]
What’s the problem?

DMP [Ijspeert et al., 2002]: Stable spring excited by a learned control input $u$

$$\dot{x}(t) = f(x,s) = Ax(t) + Bu(s;p), \quad x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \in \mathbb{R}^2$$

Problem: One-shot learning $\rightarrow$ undesirable behavior in regions not covered by the demonstration
Motivation

What’s the problem?

DMP [Ijspeert et al., 2002]: Stable spring excited by a learned control input $u$

$$\dot{x}(t) = f(x, s) = Ax(t) + Bu(s; p), \quad x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \in \mathbb{R}^2$$

**Problem:** One-shot learning $\rightarrow$ undesirable behavior in regions not covered by the demonstration

**Solution:** Capture different dynamics from multiple demonstrations [Ude et al., 2010][Forte et al., 2012]
What’s the problem?

DMP [Ijspeert et al., 2002]: Stable spring excited by a learned control input $u$

$$\dot{x}(t) = f(x, s) = A x(t) + B u(s; p), \quad x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \in \mathbb{R}^2$$

Problem: One-shot learning $\rightarrow$ undesirable behavior in regions not covered by the demonstration

Solution: Capture different dynamics from multiple demonstrations [Ude et al., 2010][Forte et al., 2012]

Presented approach $\rightarrow$ locally optimal combination:

$$\dot{x}(t) = A x(t) + B \sum_{d=1}^{D} \lambda_d(t) u_d(s; p_d)$$
Outline

1. Motivation
2. Concept
3. Results
4. Contributions & Outlook
Re-compute the dynamical system online

Optimize combination of pre-learned control inputs at each time step $k$ . . .

$$\dot{x}[k] = Ax[k] + B \sum_{d=1}^{D} \lambda_d[k] u_d[k]$$

. . . by minimizing a distance criterion between current and demonstrated states
Re-compute the dynamical system online

Optimize combination of pre-learned control inputs at each time step $k$ . . .

$$\dot{x}[k] = Ax[k] + B \sum_{d=1}^{D} \lambda_d[k] u_d[k]$$

. . . by minimizing a distance criterion between current and demonstrated states

- States evolve “in between” demonstrations . . .
- . . . or get “pulled” onto them with dynamics governed by $A$
- Encodes different dynamics
Re-compute the dynamical system online

Optimize combination of pre-learned control inputs at each time step $k$ . . .

\[ \dot{x}[k] = Ax[k] + B \sum_{d=1}^{D} \lambda_d[k] u_d[k] \]

. . . by minimizing a distance criterion between current and demonstrated states

- States evolve “in between” demonstrations . . .

- . . . or get “pulled” onto them with dynamics governed by $A$

- Encodes different dynamics

- First step towards Model Predictive Control with state constraints
How does it work?

States $x$ inside the convex hull of $x_d$
Outline

1. Motivation
2. Concept
3. Results
4. Contributions & Outlook
Generalization in simulation
Disturbance rejection in simulation
Evaluation on the Shadow Robot platform

Grasp motions recorded with a sensorized glove . . .

. . . and used to learn primitive controllers for the Shadow Hand
Evaluation on the Shadow Robot platform

Online Movement Planning/Control from Arbitrary Initial States
Outline

1. Motivation
2. Concept
3. Results
4. Contributions & Outlook
To sum up . . .

Contributions:

- Learn motion controllers from multiple demonstrations . . .
- . . . and form a (locally) optimal combination to generate movements
- Allows to encode fundamentally different dynamics
- Predictable behavior without explicit costly motion planning!
To sum up . . .

**Contributions:**

- Learn motion controllers from multiple demonstrations . . .
- . . . and form a (locally) optimal combination to generate movements
- Allows to encode fundamentally different dynamics
- Predictable behavior without explicit costly motion planning!

**Future work:**

- Optimize over a time window → Model Predictive Control
- Incorporate spatial & temporal state space constraints (obstacle avoidance . . .)
- Reactive on-line planning & control scheme [Anderson et al., 2012]
That’s it . . .
References


