Dynamic movement primitives

Gregor Schöner
gregor.schoener@ini.rub.de
Neural motivation

- Notion that neural networks in the brain and spinal cord generated a limited set of temporal templates
- Whose weighted superposition is used to generate any given movement
Evidence for “primitives” in frog spinal cord

- electrical simulation in premotor spinal cord
- measure forces of resulted muscle activation pattern at different postures of limb
- interpolate force-field

[Bizzi, Mussa-Ivaldi, Gizter, 1991]
Evidence for “primitives” in frog spinal cord

- parallel force-fields in premotor areas vs. convergent force fields from interneurons…

[Bizzi, Mussa-Ivaldi, Gizter, 1991]
Evidence for “primitives” in frog spinal cord

Convergent force-fields occur more often than expected by chance

[Bizzi, Mussa-Ivaldi, Gizter, 1991]
Evidence for “primitives” in frog spinal cord

- superposition of force-fields from joint stimulation

[Bizzi, Mussa-Ivaldi, Gizter, 1991]

superposition of A and B

stimulating both A and B locations
Dynamic movement primitives

[ljspeert et al., Neural Computation 25:328-373 (2013)]
Base oscillator

- damped harmonic oscillator
- written as two first order equations
- has fixed point attractor

\[ \tau \ddot{y} = \alpha_z (\beta_z (g - y) - \dot{y}) + f, \quad \text{\( y \): position} \]

\[ \tau \dot{z} = \alpha_z (\beta_z (g - y) - z) + f, \quad \text{\( z \): velocity} \]

\[ \tau \dot{y} = z, \quad (z, y) = (0, g) \quad \text{\( g \): goal point} \]

[Ijspeert et al., Neural Computation 25:328-373 (2013)]
Forcing function

- base functions
- weighted
  superposition makes forcing function
- which are explicit functions of time!
- => non-autonomous
- and, through c_i, also staggered in time, so there is a “score” being kept in time

\[
\Psi_i(x) = \exp\left(-\frac{1}{2\sigma_i^2}(x - c_i)^2\right),
\]

\[
f(t) = \frac{\sum_{i=1}^{N} \Psi_i(t) w_i}{\sum_{i=1}^{N} \Psi_i(t)}
\]

Kernel Activation

![Graph showing Kernel Activation over time]
“Canonical system”

“phase” variable, $x$, to (seemingly) get rid of non-autonomous character of dynamics

but: ... $x$ is reset to an initial condition at each new movement initiation $x(0) = 1$

new: scale forcing functions with amplitude and with temporal distance from end of mov

\[ \tau \dot{x} = -\alpha_x x, \]

\[ f(x) = \frac{\sum_{i=1}^{N} \Psi_i(x) w_i}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0) \]

\[ y_0 \quad \text{initial position} \]

\[ g - y_0 \quad \text{amplitude} \]
Example 1D

- weights fitted to track dotted trajectory (=5th order polynomial)… with first goes in the negative direction
- 20 kernels…

**dotted: target**  
**solid: approximation**

**position** | **velocity** | **acceleration**
---|---|---

![Graphs](image)
Figure 1: Exemplary time evolution of the discrete dynamical system. The parameters $w_i$ have been adjusted to fit a fifth-order polynomial trajectory between start and goal point ($g = 1.0$), superimposed with a negative exponential bump. The upper plots show the desired position, velocity, and acceleration of this target trajectory with dotted lines, which largely coincide with the realized trajectories of the equations (solid lines). On the bottom right, the activation of the 20 exponential kernels comprising the forcing term is drawn as a function of time. The kernels have equal spacing in time, which corresponds to an exponential spacing in $x$.

Figure 1 demonstrates an exemplary time evolution of the equations. Throughout this letter, the differential equations are integrated using Euler integration with a 0.001 s time step. To start the time evolution of the equations, the goal is set to $g = 1$, and the canonical system state is initialized to $x = 1$. As indicated by the reversal of movement direction in Figure 1 (top left), the internal states and the basis function representation allow generating rather complex attractor landscapes.

Figure 2 illustrates the attractor landscape that is created by a two-dimensional discrete dynamical system, which we discuss in more detail in section 2.1.5. The left column in Figure 2 shows the individual dynamical systems, which act in two orthogonal dimensions, $y_1$ and $y_2$. The system starts at $y_1 = 0, y_2 = 0$, and the goal is $g_1 = 1, g_2 = 1$. As shown in the vector field plots of Figure 2, at every moment of time (represented by the phase variable $x$), there is an attractor landscape that guides the time evolution of the system until it finally ends at the goal state. These attractor properties play an important role in the development of our approach when coupling terms modulate the time evolution of the system.

2.1.2 A Limit Cycle Attractor with Adjustable Attractor Landscape. Limit cycle attractors can be modeled in a similar fashion to the point attractor...
The space-time planning problem

- is to make sure the movement plan arrives at the target in a given time...

- the spatial goal is implemented by setting an attractor at the goal state

- the movement time is implicitly encoded in the tau/time scale of the “timing” variable...

  - but that relies on cutting off the timing variable, x, as some threshold level... as exponential time course never reaches zero...

  - quite sensitive to that threshold...
Periodic movement

- trivial phase oscillator (cycle time, tau) \( \tau \dot{\phi} = 1 \),

- trivial amplitude, \( r \) (constant), can be modulated by explicit time dependence

- forcing-function are functions of phase and amplitude

- base oscillator

\[
\begin{align*}
\dot{z} &= \alpha_z (\beta_z (g - y) - z) + f, \\
\tau \dot{y} &= z,
\end{align*}
\]
Example: rhythmic movement

\[ y(t) = \sin(2\pi t) + 0.25 \cos(4\pi t + 0.77) + 0.1 \sin(6\pi t + 3.07). \]

The plots show the desired position, velocity, and acceleration with dotted lines, but these are mostly covered by the time evolutions of \( y, \dot{y}, \text{and} \ddot{y}. \) The bottom plots show the phase variable and its derivative and the basis functions of the forcing term over time (20 basis functions per period).

The development of a stability proof follows standard arguments. The constants of equation 2.1 are assumed to be chosen such that without the forcing term, the system is critically damped. Rearranging equation 2.1 to combine the goal \( g \) and the forcing term \( f \) in one expression results in

\[
\tau \dot{z} = \alpha z \beta z (u - y) - \alpha z z,
\]

(2.8)

where \( u \) is a time-variant input to the linear spring-damper system. Equation 2.8 acts as a low-pass filter on \( u \). For such linear systems, with appropriate \( \alpha \) and \( \beta \), for example, from critically damping we employed in our work, it is easy to prove bounded-input, bounded-output (BIBO) stability (Friedland, 1986), as the magnitude of the forcing function \( f \) is bounded by virtue that all terms of the function (i.e., basis functions, weights, and other multipliers) are bounded by design. Thus, both the discrete and rhythmic system are BIBO stable. For the discrete system, given that \( f \) decays to zero, \( u \) converges to the steady state \( g \) after a transient time, such that the system will asymptotically converge to \( g \). After the transient time, the system will exponentially converge to \( g \) as only the linear spring-damper dynamics remain relevant (Slotine & Li, 1991). Thus, ensuring that our dynamical systems remain stable is a rather simple exercise of basic stability theory.
Scaling primitives

Figure 4: Illustration of invariance properties in the discrete dynamical systems, using the example from Figure 1. (a) The goal position is varied from $-1$ to 1 in 10 steps. (b) The time constant $\tau$ is changed to generate trajectories from about 0.15 seconds to 1.7 seconds duration.

Rhythmic systems can be established trivially with

$$\dot{z} \rightarrow \dot{z}_k, \quad \dot{y} \rightarrow \dot{y}_k, \quad \dot{x} \rightarrow \dot{x}_k, \quad \dot{\phi} \rightarrow \dot{\phi}_k.$$  \hspace{2cm} (2.10)

Figure 5 provides an example of why and when invariance properties are useful. The blue (thin) line in all subfigures shows the same handwritten cursive letter 'a' that was recorded with a digitizing tablet and learned by a two-dimensional discrete dynamical system. The letter starts at a StartPoint, as indicated in Figure 5a, and ends originally at the goal point Target$_0$.

Superimposed on all subfigures in red (thick line) is the letter 'a' generated by the same movement primitive when the goal is shifted to Target$_1$. For Figures 5a and 5b, the goal is shifted by just a small amount, while for Figures 5c and 5d, it is shifted significantly more. Importantly, for Figures 5b and 5d, the scaling term $g - y_0$ in equation 2.3 was left out, which destroys the invariance properties as described above. For the small shift of the goal in Figures 5a and 5b, the omission of the scaling term is qualitatively not very significant: the red letter "a" in both subfigures looks like a reasonable "a." For the large goal change in Figures 5c and 5d, however, the omission of the scaling term creates a different appearance of the letter "a," which looks almost like a letter "u." In contrast, the proper scaling in Figure 5c creates just a large letter "a," which is otherwise identical in shape to the original scale in space from $-1$ to 1 and time from 0.15 to 1.7 but: not trivially right.
Multi-dimensional trajectories

- rather than couple multiple movement generator (deemed “complicated”)...
- only one central harmonic oscillator and multiple transformations of that...

![Diagram](image)

- Canonical System
  - Transformation System 1
  - Transformation System 2
  - Transformation System 3
  - Transformation System n

<table>
<thead>
<tr>
<th>Transformation System 1</th>
<th>Transformation System 2</th>
<th>Transformation System 3</th>
<th>Transformation System n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Position</td>
<td>Position</td>
<td>Position</td>
</tr>
<tr>
<td>Velocity</td>
<td>Velocity</td>
<td>Velocity</td>
<td>Velocity</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Acceleration</td>
<td>Acceleration</td>
<td>Acceleration</td>
</tr>
</tbody>
</table>

Figure 6: Conceptual illustration of a multi-DOF dynamical system. The canonical system is shared, while each DOF has its own nonlinear function and transformation system.

2.1.6 Learning the Attractor Dynamics from Observed Behavior.

Our systems are constructed to be linear in the parameters \( w \), which allows applying a variety of learning algorithms to fit the \( w \). In this letter, we focus on a supervised learning framework. Of course, many optimization algorithms could be used too if only information from a cost function is available.

We assume that a desired behavior is given by one or multiple desired trajectories in terms of position, velocity, and acceleration triples \((y_{\text{demo}}(t), \dot{y}_{\text{demo}}(t), \ddot{y}_{\text{demo}}(t))\), where \( t \in [1,...,P] \).

Learning is performed in two phases: determining the high-level parameters \((g, y_0, \tau)\) for the discrete system or \((g, r, \tau)\) for the rhythmic system) and then learning the parameters \( w \).

For the discrete system, the parameter \( g \) is simply the position at the end of the movement, \( g = y_{\text{demo}}(t = P) \) and, analogously, \( y_0 = y_{\text{demo}}(t = 0) \).

The parameter \( \tau \) must be adjusted to the duration of the demonstration. In practice, extracting \( \tau \) from a recorded trajectory may require some thresholding in order to detect the movement onset and end. For instance, a velocity threshold of 2% of the maximum velocity in the movement may be employed, and \( \tau \) could be chosen as 1.05 times the duration.

We assume that the data triples are provided with the same time step as the integration step for solving the differential equations. If this is not the case, the data are downsampled or upsampled as needed.
Example 2D

- single “phase” \( x \)
- two base oscillator systems \( y_1, y_2 \)
- with two sets of forcing functions

Figure 2: Vector plot for a 2D trajectory where \( y_1 \) (top left) fits the trajectory of Figure 1 and \( y_2 \) (bottom left) fits a minimum jerk trajectory, both toward a goal \( g = (g_1, g_2) = (1, 1) \). The vector plots show \( (\dot{y}_1, \dot{y}_2) \) at different values of \( (y_1, y_2) \), assuming that only \( y_1 \) and \( y_2 \) have changed compared to the unperturbed trajectory (continuous line) and that \( x_1, x_2, \dot{y}_1, \) and \( \dot{y}_2 \) are not perturbed. In other words, it shows only slices of the full vector plot \( (\dot{z}_1, \dot{z}_2, \dot{y}_1, \dot{y}_2, \dot{x}_1, \dot{x}_2) \) for clarity. The vector plots are shown for successive values of \( x = x_1 = x_2 \) from 1.0 to 0.02 (i.e., from successive steps in time). Since \( \tau \dot{y}_i = z_i \), such a plot illustrates the instantaneous accelerations \( (\ddot{y}_1, \ddot{y}_2) \) of the 2D trajectory if the states \( (y_1, y_2) \) were pushed somewhere else in state space. Note how the system evolves to a spring-damper model with all arrows pointing to the goal \( g = (1, 1) \) when \( x \) converges to 0.
Learning the weights

\[
\tau \ddot{y} = \alpha_z (\beta_z (g - y) - \dot{y}) + f,
\]

\[
f_{\text{target}} = \tau^2 \ddot{y}_{\text{demo}} - \alpha_z (\beta_z (g - y_{\text{demo}}) - \tau \dot{y}_{\text{demo}}).
\]

- base oscillator
- forcing function from sample trajectory
- weights by minimizing error \( J \)

\[
f(x) = \frac{\sum_{i=1}^{N} \Psi_i(x) w_i}{\sum_{i=1}^{N} \Psi_i(x)} x(g - y_0)
\]

\[
J_i = \sum_{t=1}^{P} \Psi_i(t) (f_{\text{target}}(t) - w_i \xi(t))^2
\]

\[
\xi(t) = x(t)(g - y_0) \quad \text{for discrete mov}
\]

\[
\xi(t) = r \quad \text{for rhythmic mov}
\]
Learning the weights

can be solved analytically

minimum of

\[
J_i = \sum_{t=1}^{P} \Psi_i(t)(f_{\text{target}}(t) - w_i \xi(t))^2,
\]

is

\[
w_i = \frac{s^T \Gamma_i f_{\text{target}}}{s^T \Gamma_i s},
\]

where (P=\# sample times in demo trajectories):

\[
s = \begin{pmatrix}
\xi(1) \\
\xi(2) \\
\vdots \\
\xi(P)
\end{pmatrix}
\]

\[
\Gamma_i = \begin{pmatrix}
\Psi_i(1) & 0 \\
0 & \Psi_i(2) \\
\vdots & \vdots & \ddots & \Psi_i(P)
\end{pmatrix}
\]

\[
f_{\text{target}} = \begin{pmatrix}
f_{\text{target}}(1) \\
f_{\text{target}}(2) \\
\vdots \\
f_{\text{target}}(P)
\end{pmatrix}
\]

\[
\xi(t) = x(t)(g - y_0)
\]

\[
\xi(t) = r
\]
Obstacle avoidance

- inspired by Schöner/Dose (in Fajen Warren form)
- obstacle avoidance force-let

\[
\begin{align*}
\tau \dot{z} &= \alpha_z (\beta_z (g - y) - z) + f + C_t, \\
\tau \dot{y} &= z. \\
C_t &= \gamma R \dot{y} \theta \exp(-\beta \theta), \\
\theta &= \arccos \left( \frac{(o - y)^T \dot{y}}{|o - y| |\dot{y}|} \right), \\
r &= (o - y) \times \dot{y}.
\end{align*}
\]

[actually this is: Reimann, Iossifidis, Schöner, 2010]
Obstacle avoidance

Figure 8: Illustration of obstacle avoidance with a coupling term. The obstacle is the large (red) sphere in the center of the plot. Various trajectories are shown, starting from different start positions and ending at the sphere labeled “goal.” Also shown is the nominal trajectory (green) that the discrete dynamical system creates when the obstacle is not present: it passes right through the sphere. Trajectories starting at points where the direct line to the goal does not intersect with the obstacle are only minimally curved around the obstacle, while other trajectories show strongly curved paths around the obstacle.

This looks intuitively natural, which is not surprising as it was derived from human obstacle-avoidance behavior (Fajen & Warren, 2003).

A more complex example of spatial coupling is given in Figure 9. Using imitation learning, a placing behavior of a cup on a target was coded in a discrete dynamical system for a 3D end effector movement of the robot, a Sarcos Slave 7 DOF robot arm. The first row of images shows the unperurbed behaviors. In the second row, the (green) target is suddenly moved to the right while the robot has already begun moving. This modification corresponds to a change of the goal parameter $g$. The third row of images demonstrates an avoidance behavior based on equation 3.2, when the blue ball comes too close to the robot’s movement. We emphasize that one single discrete dynamical system created all these different behaviors; there was no need for complex abortion of the ongoing movement or replanning. More details can be found in Pastor, Hoffmann, Asfour, and Schaal (2009).

3.2.2 Temporal Coupling. By modulating the canonical system, one can influence the temporal evolution of our dynamical systems without affecting...
But: human obstacle avoidance is not really like that…

=> Grimme, Lipinski, Schöner, 2012
Coordination

- In phase dynamics: couple to external timers...

- But: issue of predicting such events and aligning the prediction to achieve synchronicity...

\[
\tau \dot{x} = -\alpha_x x + C_c
\]

\[
\tau \dot{\phi} = 1 + C_c.
\]

\[
C_c = \alpha_c (\phi_{ext} - \phi).
\]
Conclusion

- DMP enable learning “movement styles” while enabling generalization to new movement targets
- DMP is a purely kinematic account
- => DMP is not addressing control
  - in that respect, analogy to force-fields is misleading
- DMP addresses timing, but account of coordination is limited
- DMP for different tasks and their combination… open