The Emergence of Multiple Movement Units in the Presence of Noise and Feedback Delay

Michael Kositsky  
Department of Computer Science  
University of Massachusetts  
Amherst, MA 01003-4610  
{kositsky,barto}@cs.umass.edu

Andrew G. Barto

Abstract

Tangential hand velocity profiles of rapid human arm movements often appear as sequences of several bell-shaped acceleration-deceleration phases called submovements or movement units. This suggests how the nervous system might efficiently control a motor plant in the presence of noise and feedback delay. Another critical observation is that stochasticity in a motor control problem makes the optimal control policy essentially different from the optimal control policy for the deterministic case. We use a simplified dynamic model of an arm and address rapid aimed arm movements. We use reinforcement learning as a tool to approximate the optimal policy in the presence of noise and feedback delay. Using a simplified model we show that multiple submovements emerge as an optimal policy in the presence of noise and feedback delay. The optimal policy in this situation is to drive the arm’s end point close to the target by one fast submovement and then apply a few slow submovements to accurately drive the arm’s end point into the target region. In our simulations, the controller sometimes generates corrective submovements before the initial fast submovement is completed, much like the predictive corrections observed in a number of psychophysical experiments.

1 Introduction

It has been consistently observed that rapid human arm movements in both infants and adults often consist of several submovements, sometimes called “movement units” [21]. The tangential hand velocity profiles of such movements show sequences of several bell-shaped acceleration-deceleration phases, sometimes overlapping in the time domain and sometimes completely separate. Multiple movement units are observed mostly in infant reaching [5, 21] and in reaching movements by adult subjects in the face of difficult time-accuracy requirements [15]. These data provide clues about how the nervous system efficiently produces fast and accurate movements in the presence of noise and significant feedback delay. Most modeling efforts concerned with movement units have addressed only the kinematic aspects of movement, e.g., [5, 12].

We show that multiple movement units might emerge as the result of a control policy that is optimal in the face of uncertainty and feedback delay. We use a simplified dynamic model
of an arm and address rapid aimed arm movements. We use reinforcement learning as a tool to approximate the optimal policy in the presence of noise and feedback delay.

An important motivation for this research is that stochasticity inherent in the motor control problem has a significant influence on the optimal control policy [9]. We are following the preliminary work of Zelevinsky [23] who showed that multiple movement units emerge from the stochasticity of the environment combined with a feedback delay. Whereas he restricted attention to a finite-state system to which he applied dynamic programming, our model has a continuous state space and we use reinforcement learning in a simulated real-time learning framework.

2 The model description

The model we simulated is sketched in Figure 1. Two main parts of this model are the “RL controller” (Reinforcement Learning controller) and the “plant.” The controller here represents some functionality of the central nervous system dealing with the control of reaching movements. The plant represents a simplified arm together with spinal circuitry. The controller generates the control signal, $u$, which influences how the state, $s$, of the plant changes over time. To simulate delayed feedback the state of the plant is made available to the controller after a delay period $\Delta$, so at time $t$ the controller can only observe $s(t-\Delta)$. To introduce stochasticity, we disturbed $u$ by adding noise to it, to produce a corrupted control $\bar{u}$. The controller learns to move the plant state as quickly as possible into a small region about a target state $s_T$. The reward structure block in Figure 1 provides a negative unit reward when the plant’s state is out of the target area of the state space, and it provides zero reward when the plant state is within the target area. The reinforcement learning controller tries to maximize the total cumulative reward for each movement. With the above mentioned reward structure, the faster the plant is driven into the target region, the less negative reward is accumulated during the movement. Thus this reward structure specifies the minimum time-to-goal criterion.

![Figure 1: Sketch of the model used in our simulations. “RL controller” stands for a Reinforcement Learning controller.](image)

2.1 The plant

To model arm dynamics together with the spinal reflex mechanisms we used a fractional-power damping dynamic model [22]. The simplest model that captures the most critical dynamical features is a spring-mass system with a nonlinear damping:

$$m\ddot{x} + b\dot{x} + k(x - u) = 0.$$ 

Here, $x$ is the position of the mass attached to the spring, $\dot{x}$ and $\ddot{x}$ are respectively the velocity and the acceleration of the object, $m$ is the mass of the object (the mass of the spring is assumed equal to zero), $b$ is the damping coefficient, $k$ is the stiffness coefficient, and $u$ is the control signal which determines the resting, or equilibrium, position. Later in this paper, we call $u$ activation, referring to the activation level of a muscle pair. The
Table 1: Parameter values used in the simulations.

<table>
<thead>
<tr>
<th>description</th>
<th>value</th>
<th>description</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>the basic simulation time step</td>
<td>1 ms</td>
<td>threshold velocity radius</td>
<td>0.1 cm/s</td>
</tr>
<tr>
<td>the feedback delay, $\Delta$</td>
<td>200 ms</td>
<td>standard deviation of the noise</td>
<td>1 cm</td>
</tr>
<tr>
<td>initial position</td>
<td>0 cm</td>
<td>value function learning rate</td>
<td>0.5</td>
</tr>
<tr>
<td>initial velocity</td>
<td>0 cm/s</td>
<td>preferences learning rate</td>
<td>1</td>
</tr>
<tr>
<td>target position</td>
<td>5 cm</td>
<td>discount factor, $\gamma$</td>
<td>0.9</td>
</tr>
<tr>
<td>target velocity</td>
<td>5 cm</td>
<td>bootstrapping factor, $\lambda$</td>
<td>0.9</td>
</tr>
<tr>
<td>target position radius</td>
<td>0.5 cm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

values for the mass, the damping coefficient, and the stiffness coefficient were taken from Barto et al. [3]: $m = 1$ kg, $b = 3$ N (s/m), $k = 30$ N/m. These values provide movement trajectories qualitatively similar to those observed in human wrist movements [22].

The fractional-power damping in this model is motivated by both biological evidence [8, 14] and computational considerations. Controlling a system with such a concave damping function is an easier control problem than for a system with apparently simpler linear damping. Fractional-power damping creates a qualitatively novel dynamical feature called a stiction region, a region in the position space around the equilibrium position consisting of pseudo-stable states, where the velocity of the plant remains very close to zero. Such states are stable states for all practical purposes. For the parameter magnitudes used in our simulations, the stiction region is a region of radius 2.5 cm about the true equilibrium in the position space.

Another essential feature of the neural signal transmission can be accounted for by using a cascade of low-pass temporal filters on the activation level $u$ [16]. We used a second-order low-pass filter with the time constant of 25 ms.

2.2 The reinforcement learning controller

We used the version of the actor-critic algorithm described by Sutton and Barto [20]. A possible model of how an actor-critic architecture might be implemented in the nervous system was suggested by Barto [2] and Houk et al. [10]. We implemented the actor-critic algorithm for a continuous state space and a finite set of actions, i.e., activation level magnitudes $u$ evenly spaced every 1 cm between 0 cm and 10 cm. To represent functions defined over the continuous state space we used a CMAC representation [1] with 10 tilings, each tiling spans all three dimensions of the state space and has 10 tiles per dimension. The tilings have random offsets drawn from the uniform distribution. Learning is done in episodes. At the beginning of each episode the plant is at a fixed initial state, and the episode is complete when the plant reaches the target region of the state space. Table 1 shows the parameter values used in the simulations. Refer to ref. [20] for algorithm details.

2.3 Clocking the control signal

For the controller to have sufficient information about the current state of the plant, the controller internal representation of the state should be augmented by a vector of all the actions selected during the last delay period. To keep the dimension of the state space at a feasible level, we restrict the set of available policies and make the controller select a new activation level, $u$, in a clocked manner at time intervals equal to the delay period. One step of the reinforcement learning controller is performed once a delay period, which corresponds to many steps of the underlying plant simulation. To simulate a stochastic plant we added Gaussian noise to $u$. A new Gaussian disturbance was drawn every time a