Bioinspired smooth neuromorphic control for robotic arms

Ioannis Polykretis\textsuperscript{1,2,3}, Lazar Supic\textsuperscript{1} and Andreea Danielescu\textsuperscript{1}

\textsuperscript{1} Accenture Labs, San Francisco, CA, United States of America
\textsuperscript{2} Computer Science Department, Rutgers University, New Brunswick, NJ, United States of America
\textsuperscript{3} IP was partially funded by the Onassis Foundation scholarship.

* Author to whom any correspondence should be addressed.
E-mail: ioannis.polykretis@accenture.com

Keywords: neuromorphic control, smooth control, robot arms, bioinspired, spiking neural networks, neuromorphic processor

Abstract

Beyond providing accurate movements, achieving smooth motion trajectories is a long-standing goal of robotics control theory for arms aiming to replicate natural human movements. Drawing inspiration from biological agents, whose reaching control networks effortlessly give rise to smooth and precise movements, can simplify these control objectives for robot arms. Neuromorphic processors, which mimic the brain’s computational principles, are an ideal platform to approximate the accuracy and smoothness of biological controllers while maximizing their energy efficiency and robustness. However, the incompatibility of conventional control methods with neuromorphic hardware limits the computational efficiency and explainability of their existing adaptations. In contrast, the neuronal subnetworks underlying smooth and accurate reaching movements are effective, minimal, and inherently compatible with neuromorphic hardware. In this work, we emulate these networks with a biologically realistic spiking neural network for motor control on neuromorphic hardware. The proposed controller incorporates experimentally-identified short-term synaptic plasticity and specialized neurons that regulate sensory feedback gain to provide smooth and accurate joint control across a wide motion range. Concurrently, it preserves the minimal complexity of its biological counterpart and is directly deployable on Intel’s neuromorphic processor. Using the joint controller as a building block and inspired by joint coordination in human arms, we scaled up this approach to control real-world robot arms. The trajectories and smooth, bell-shaped velocity profiles of the resulting motions resembled those of humans, verifying the biological relevance of the controller. Notably, the method achieved state-of-the-art control performance while decreasing the motion jerk by 19% to improve motion smoothness. Overall, this work suggests that control solutions inspired by experimentally identified neuronal architectures can provide effective, neuromorphic-controlled robots.

1. Introduction

Modern robot arms perform a wide variety of complex and repetitive tasks, both in isolated industrial environments \cite{7} and in close interaction with humans \cite{2}. The efficiency of the arms’ performance requires the determination of the best trajectory for the execution of the task \cite{4, 23}. This trajectory can be determined by optimizing several variables. Beyond the paramount objective of control precision, a crucial optimization variable is the smoothness of the end-effector’s trajectory \cite{13}. Control smoothness not only improves position control \cite{37, 68} but also slows down motor wear \cite{49} by avoiding abrupt load changes and ensures a safe and comfortable human-robot interaction by avoiding damage to the robot, its user, or their environment \cite{26}. Patient-assistive robotic arms and other human-robot collaboration contexts especially benefit from smooth robotic control as they require low end-effector velocity when the arms enter a safety bubble surrounding the user \cite{19}. For this, numerous methods aim to optimize control smoothness by introducing kinematic constraints to the motion velocity, acceleration, and jerk \cite{37, 68, 73}. However, these constraints further complicate the already challenging optimization problem.
Humans, however, demonstrate smooth trajectories when performing reaching movements [66] as early as their first postnatal months [51]. Primates and rodents also achieve similar reaching behaviors during goal-oriented movements [20, 55]. The resulting trajectories satisfy the optimality criteria discussed above [21, 64, 69] seemingly effortlessly and in the presence of limited energy and computational resources. The motion smoothness depends on specialized neuronal microcircuits that form recurrent loops and refine sensory feedback [20]. Inhibitory interneurons critically enrich the computational repertoire of these microcircuits by affecting not only other neurons but also the dynamics of the synapses that interconnect them. The specialized synapse-modulating inhibitory neurons target presynaptic axons and modify their neurotransmitter release probability [72]. The effect of inhibition at this presynaptic locus has a divisive effect on feedback signals and provides a versatile mechanism of sensory gain control [50]. This dynamic feedback gain provides accurate and stable control across diverse motion ranges in biological agents. Including such neuroscientific principles in robot control could potentially give rise to novel solutions that would be as effective as their conventional counterparts and computationally efficient, adaptable to a wide range of stimuli, and explainable in terms of behavior and malfunction [34, 47, 48].

Neuromorphic algorithms and hardware platforms [15, 22, 43, 46] also draw inspiration from the same neuroscientific principles, attempting to emulate computation in the brain [9, 63]. Consequently, this computational framework is ideal for integrating biological inspiration into robot control. Neuromorphic solutions already provide additional benefits, such as low latency, low power consumption, robustness, and potentially higher accuracy, which have been proven primarily outside of robotic applications [30, 62]. These benefits are critical for all robotic applications, but are especially critical for mobile real-time human-robotic applications, which have additional power constraints. Brain-inspired neuromorphic methods have successfully targeted a wide range of tasks: from dynamic navigation and mapping [42, 61] to trajectory generation [41] and joint control [31, 12, 75]. On a broader perspective, biological inspiration has provided valuable insights for the whole spectrum of robot control [71]: Full-scale behaviors such as locomotion [28, 48, 58] and swimming [14, 29] benefit from the knowledge of Central Pattern Generators [25]; Low-level joint control exploits neuroscientific knowledge of neuromotor control models to design stable and robust variants of the classical PID controller [59, 74]. However, existing methods, be they neuromorphic or merely brain-inspired, have yet to emulate the characterized neuronal microcircuits that drive smooth movements in animals to explore the optimization of the trajectory smoothness in robotic control.

This work addresses this gap by developing a compact SNN architecture that allows for smooth and accurate neuromorphic control. The proposed SNN architecture attempts to faithfully emulate the function of the specialized neuronal microcircuits that provide smooth joint control in primates and rodents [20]. It comprises antagonistic extensor and flexor motor neurons that modify the controlled variable and sensory proprioceptor (PPC) neurons that ensure smoothness by regulating the sensory feedback to the motor neurons. This biologically plausible architecture is directly deployable on Intel’s neuromorphic research processor, the Loihi. The method is evaluated (figure 1) in the control of two increasingly complex robotic arms: (i) a two-link arm (Reacher-v2) and (ii) a 6-DOF real-world robotic arm (Jaco). The controller matches the performance of a conventional PID controller when evaluated on standard controller metrics (rise time, settling time, and overshoot) [36] while also improving control smoothness. The combination of smooth control for each joint with a biologically inspired coupling of the joints gives rise to end-effector trajectories that resemble those of human arms. Therefore, this work demonstrates that drawing inspiration
from experimentally identified neuronal architectures can successfully emulate their function in neuromorphic-controlled robots.

2. Related work

Conventional control methods have matured over several decades and are now very effective for controlling joints of multi-link robotic arms. The theoretical underpinnings of these methods rely on systems of nonlinear differential equations or dynamical systems [38, 39]. The continuous, high-precision and synchronized computations that these methods rely on hinder their compatibility with modern neuromorphic processors. This incompatibility prevents them from exploiting the energy efficiency, low latency, and robustness provided by neuromorphic computing [16].

In an effort to combine the performance of conventional methods with the benefits of neuromorphic computing, recent solutions have attempted to translate PID controllers on both analog [31] and digital [57, 12] neuromorphic processors. Although these approaches achieve competitive control performance, they suffer in terms of computational efficiency and interpretability. These come as a necessary sacrifice when the precise, continuous values of conventional control methods need to be represented and used in the low-precision, discrete and asynchronous computations performed on neuromorphic hardware. Specifically, these approaches represent continuous values utilizing large neuronal populations, whose size scales as a function of the required control precision to allow for distinct value representation [57, 12, 75]. After encoding the continuous quantities, these methods perform the necessary computations by exploiting large neuronal arrays, further increasing the computational burden [24, 12, 77]. The large number of required components introduces significant parameter tuning, which is inefficient or even infeasible in hardware such as mixed signal neuromorphic processors with inherent inaccuracies [24]. Lastly, the resulting architectures are platform-specific [31], requiring a complete rethinking of their design to translate to different processors.

This work addresses some of these drawbacks by proposing a biologically inspired SNN for the smooth control of robotic joints. Our architecture draws inspiration from the experimentally identified neuronal connectivity that gives smooth reaching movements in rodents [20] and primates [55]. Mimicking the connectivity of its biological counterpart, our controller architecture is not only minimal in terms of computational resources but is also interpretable. Moreover, the core of our architecture can be deployed on any neuromorphic platform that includes spiking compartments with modifiable connectivity, requiring only minor design changes to implement the additional adaptation mechanisms (synaptic facilitation and presynaptic inhibition).

3. Biological background

To replicate the smoothness of reaching movements in biological agents with robot arms, we must first understand the key factors that contribute to it. As smoothness is related to movement continuity [6], the discrete endpoints of reaching movements may introduce discontinuity when combined with the inertial properties of the limbs. Therefore, the two sensory-dependent movement components, the initiation and termination, are crucial for the smoothness.

Movement initiation must be sensitive to subtle sensory cues to ensure responsiveness, but it also needs to be gradual to ensure continuity. One of the mechanisms that contributes to the satisfaction of these constraints is short-term synaptic plasticity [65]. Specifically, synaptic facilitation, i.e. the input-dependent, temporary increase of synaptic weights, allows for temporally separated presynaptic inputs, which would otherwise have no effects, to accumulate and induce postsynaptic spikes. This accumulation becomes crucial for motor neurons, which receive presynaptic stimuli that encode external (sensory) and internal (proprioceptive) error representations. In rate-encoding representations, errors of small magnitude correspond to low spiking frequencies. Therefore, the accumulation mediated by synaptic facilitation can activate motor neurons even at low presynaptic firing rates [44], and allows for the correction of small deviations, ensuring accurate control. Concurrently, this progressive transfer of presynaptic, error-encoding activity to motor neurons ensures gradual movement initiation.

Effective movement termination depends on finely regulated sensory feedback. Without that, the motor circuits, which are inherently prone to oscillation [5], would fail to achieve smooth limb trajectories. Extensive evidence highlights the role of inhibitory neurons in regulating sensory gain [53, 70], with the most prominent inhibitory neurotransmitter, gamma-aminobutyric acid (GABA), being a key factor [56]. Whereas the majority of inhibitory neurons directly target motor or premotor neurons [3], a small subset of GABAergic neurons target the axons of presynaptic sensory afferents and reduce their neurotransmitter release probability [72]. While postsynaptic inhibition is far less effective in gain scaling due to its subtractive
nature [10, 11], the divisive effect of presynaptic inhibition is a versatile mechanism of sensory gain control [50] with experimental evidence suggesting that GABA-mediated presynaptic inhibition regulates the sensory gain in primates and rodents [20, 55] to suppress forelimb oscillation and provide smooth reaching movements. The ablation of this GABA-ergic presynaptic inhibition highlights its importance since it results in damped harmonic oscillations [20].

The architecture proposed in this paper emulates these two crucial mechanisms to achieve smooth end-effector trajectories. First, synaptic facilitation is implemented as a local plasticity mechanism. The synaptic weights temporarily increase in response to presynaptic activity to model how synaptic vesicles get gradually entrained in the facilitatory synapses [65]. Then, the mechanism of presynaptic inhibition is emulated as a separate disynaptic pathway. There, a presynaptic inhibitory neuron (PSI) resembles the effects of the GABA-ergic neurons on the sensory afferent terminals and induces a divisive effect on the depolarization of the motor neurons [20]. The implementation details are described in section 4.1.4.

4. Methods

The objective of the SNN controller is two-fold: (i) to ensure control accuracy by decreasing the control error, i.e. the difference between the current control variable value \( \theta \) and the desired control variable value, \( \theta^d \), and (ii) to ensure smooth control through a gradual initial increase of the control variable and a timely steady decline to avoid overshooting the target. The architecture of the controller is shown in figure 2. In the following sections, we first provide a detailed description of the components of the single-joint controller and then, we describe the neuromorphic deployment of the network. Finally, we explain the addition of joint coordination to support multi-DOF arms.

4.1. Architecture of the 1-DOF SNN controller

4.1.1. Input and feedback encoding

The proposed SNN controls a single variable by comparing its actual value \( \theta \), which is internally fed back, with the desired value \( \theta^d \), which is the only external input provided to the controller. The values of these two quantities are encoded into currents by the current converter modules (CCMs) and then stimulate the neurons of the network, driving their spiking activity. A pair of CCMs with opposite-signed inputs is employed for each quantity to allow for the encoding of both positive and negative magnitudes. We choose a Rectified Linear Unit transfer function for the CCMs to generate a positive current value when their input is positive while providing no output otherwise. In that way, the activity of the two CCM pairs is mutually exclusive since each quantity could be either positive or negative. The output currents of the CCMs are then passed to the sensory part of the network consisting of PPC neurons. The PPC neurons of the network responsible for sensing the control error and the rate of change in the control variable are then connected to the actuation component of the network, which updated the control variable to decrease the error.

4.1.2. Sensing and computing

All neurons in the proposed SNN are simulated using the leaky-integrate-and-fire (LIF) model, governed by the following dynamics. First, we integrate the input spikes into synaptic current:

\[
 u_i^{(t)} = u_i^{(t-1)} \cdot d_u + \sum_j w_{ij} \cdot \delta_j^{(t-1)}. \tag{1}
\]

Then, we integrate the synaptic current into the membrane voltage of the neuron, which then fires a spike when the voltage exceeded a threshold \( V_{th} \), as follows:

\[
 v_i^{(t)} = v_i^{(t-1)} \cdot d_v + u_i^{(t)}, \text{if } v_i^{(t-1)} < V_{th} \\
 v_i^{(t)} = 0 \text{ and } \delta_i^{(t-1)} = 1, \text{if } v_i^{(t-1)} \geq V_{th} \tag{2}
\]

where \( t \) is the time step, \( u_i \) is the neuron’s input current, \( v_i \) is the neuron’s voltage, \( d_u \) and \( d_v \) are current and voltage decay factors, \( w_{ij} \) are the connection weights between the presynaptic neuron \( j \) and the postsynaptic neuron \( i \), and \( \delta_j^{(t-1)} \) is a binary variable denoting the spikes of neuron \( j \) at time step \( t-1 \).

To estimate the control error (difference between actual and desired value) at each timestep, we employ a pair of PPC neurons (ePPC). These two neurons receive the current-encoded values of \( \theta \) and \( \theta^d \) as input and encode the sign and magnitude of the error in their spiking activity. To encode the sign, we connect the four CCMs to the pair of ePPC in a way that enforces the mutually exclusive firing of the neurons. For example, when \( 0 < \theta < \theta^d \), only the two middle CCMs produced non-zero output, with the output of the bottom one being larger in magnitude. Connecting the two CCMs with both ePPC but with opposite signs ensures that
Figure 2. Single-DOF SNN controller Architecture. CPU-based implementation (A) and deployment on Intel's Loihi (B). A pair of neurons (ePPC) compare the input target value ($\theta_d$) with the actual feed-back value of the controlled variable ($\theta$), and encode the difference in their activity. Then, they drive another pair of neurons (motor-neurons F/E) to increase $\theta$ and eliminate the difference. Short-term facilitation of the synaptic connections between the two pairs enforces gradual initiation of the elimination. A parallel pathway (PSI) also modifies the synaptic weights to adapt to a range of difference magnitudes. A third pair of neurons (dPPC) encodes the rate of change of $\theta$ to limit abrupt error corrections.

$ePPC^-$ is excited, while the bottom one is shunted. In that way, $ePPC^-$ ($ePPC^+$) is always active (inactive) when $\theta < \theta_d$. Following the same logic, we choose the rest of the connections between the CCM and the $ePPC$. The magnitude of the error is reflected by the firing rate of the respective $ePPC$. The aggregate current from the CCM that stimulates the $ePPC$ was proportional to the error, inducing a proportional firing rate in the neurons. In the above example, $ePPC^-$ is stimulated by a positive current proportional to $\theta_d$ and a negative current proportional to $\theta$. Therefore, its firing rate is proportional to $\theta_d - \theta$.

To improve the control smoothness, we utilize another pair of PPC neurons (dPPC) that are sensitive to the rate of change of the control variable. These neurons introduce some push-pull dynamics: when $\theta$ diverges considerably from $\theta_d$, the controller attempts to quickly decrease the error, giving rise to a sizeable derivative value. This value is input to the respective $dPPC$ and activates it to counteract the error decrease and avoid overshooting. To incorporate the sensitivity of the $dPPC$ to the rate of change of the control variable, we first estimate the derivative of $\theta$ (measure its change over a time window and divide by the window length). Then, we encode its value into a current, which is fed to the two $dPPC$ with opposite signs. In that way, these neurons fire in a mutually exclusive manner to encode positive and negative rates of change, with their firing rate encoding the magnitude of the derivative.

4.1.3. Actuation
The actuation component of the network comprises of two antagonistic motor neurons, an extensor (E) and a flexor (F), that increase and decrease the control variable, respectively. The E integrates the spikes of the top $ePPC$, whose activity indicates a negative error ($\theta < \theta_d$), and responds with spikes that increase $\theta$ toward the target $\theta_d$. For this, we translate each E spike to an incremental increase of $\theta$. Concurrently, the E integrates the spikes of the left $dPPC$, whose activity indicates a decrease in $\theta$ and responds with spikes to counteract that decrease. The connectivity and the function of the F neuron is analogous for the decrease of the control variable.

4.1.4. Weight adaptation
To improve the motion smoothness without sacrificing control accuracy, we introduce two time-dependent, multiplicative factors $f^{(i)}$ and $g^{(i)}$ that adapt the static synaptic weights $W$ between the $ePPC$ and the motor neurons, as shown below:

$$w^{(i)} = W \cdot f^{(i)} \cdot g^{(i)}.$$  

(3)

First, we introduce a biologically plausible, activity-dependent facilitation of the synaptic weights [65] to maximize the smoothness of the motion. The efficacy $f^{(i)}$ of the synaptic connection between the the $i$th
ePPC and the \( i \)th motor neuron increases with each presynaptic spike until saturating at a maximum value and decays otherwise with a factor \( d_{\text{fac}} \), as shown in equation (4).

\[
f^{(t)}_{ij} = f^{(t-1)}_{ij} \cdot d_{\text{fac}} + U_{\text{fac}} \cdot s^{(t-1)}_j,
\]

where \( d_{\text{fac}} = 1 \) is the facilitation decay time constant (set to 500 ms\(^{-1}\)), \( U_{\text{fac}} \) is the facilitation factor increment (set to 0.005), and \( s^{(t-1)}_j \) is the spike of the presynaptic neuron. Due to the low initial synaptic efficacies, abrupt increases in the activity of the ePPC do not directly propagate to the motor neurons. The gradual increase of the synaptic efficacies results in a graded activation of the motor neurons, resulting in a smooth modification of the controlled amount. The parameter values for \( d_{\text{fac}} \) and \( U_{\text{fac}} \) are chosen to provide gradual movement initiation for the given arms, and can be modified over a range to increase/decrease this effect.

Second, we optimize the controller parameters according to the range of the desired movement to ensure accurate control for movements of significantly different magnitudes. Specifically, while small errors require strong synaptic connections from the ePPC to activate the motor neurons and ensure accurate control, large errors require weaker connections to avoid overshoot. To address this requirement, we introduce the biologically plausible adaptation mechanism of presynaptic inhibition [20], and scale the synaptic weights as a function of the error magnitude. Specifically, we utilize an additional neuron (PSI), whose activity reflects the activity of the ePPC and, therefore, the magnitude of the error. The PSI spikes modulate the connection weights through a factor \( g \) as described below:

\[
g^{(t)} = g_{\text{max}} - \left[ g^{(t-1)} \cdot d_{\text{ PSI}} + U_{\text{ PSI}} \cdot s^{(t-1)}_{\text{ PSI}} \right],
\]

where \( g_{\text{max}} \) is the maximum value of the modulatory factor \( g \) (set to 1), \( d_{\text{ PSI}} = 1 \) is the presynaptic inhibition decay time constant (set to 50 ms\(^{-1}\)), \( U_{\text{ PSI}} \) is the presynaptic inhibition increment (set to 0.005), and \( s^{(t-1)}_{\text{ PSI}} \) is the spike of the dedicated PSI. The parameter values for \( d_{\text{ PSI}} \) and \( U_{\text{ PSI}} \) are chosen to provide accurate control for joint movements in the maximum range allowed for the given arms, and can also be modified if this joint range were to be adapted to a different task. During the control, \( g \) decreases by \( U_{\text{ PSI}} \) with each PSI spike \( s^{(t-1)}_{\text{ PSI}} \) and otherwise decays to its maximum value \( g_{\text{max}} \), with a factor \( d_{\text{ PSI}} \). Using \( g \) as a multiplicative factor in equation (3), we emulate the divisive effect of presynaptic inhibition on the weights of biological synapses [20].

### 4.2. Neuromorphic realization

Due to the proposed algorithm’s biological plausibility, the SNN presented can be easily and directly ported to any of the state-of-the-art neuromorphic platforms [15, 22, 43, 46] (figure 2(B)). We deploy the SNN controller on Intel’s Loihi neuromorphic research chip [15], which we select because the chip’s hardware architecture includes customizable, tree-like neuron models that allow for easy deployment of the proposed mechanisms (e.g. facilitation) We implement PPC and actuation neurons as single Loihi compartments governed by the LIF dynamics (equations (1) and (2)) that Loihi inherently supports.

One limitation of the Loihi hardware is that it does not directly support facilitatory synapses. To overcome this limitation, we develop a facilitatory module. We design this module as a two-compartment neuron that follows the imbalanced tree structure of multi-compartment neurons on Loihi. The leaf (figure 2(B), small orange circle) is a non-spiking compartment that receives and integrates the module’s input. The root (large black circle) is a spiking compartment that integrates the leaf compartment’s voltage. This cascaded integration of the input results in a gradual increase in the spiking activity of the root compartment, which induces gradual activation of the motor neurons, approximating the effects of the facilitatory synapses in the off-chip controller.

While this design introduces more Loihi compartments and connections than the number of neurons and synapses in the CPU-based implementation, the single-joint controller only requires 10 compartments and 8 connections (60 compartments and 66 connections including the connections providing joint coordination for the control of the real-world Jaco arm). Because of its network size, the hardware implementation only utilizes one out of the 128 available Loihi neurocores, while similar methods [12] have to distribute their networks over 38 cores. Therefore, the biological plausibility of the architecture minimizes the resources requirements of the design.

### 4.3. Multi-DOF control

To scale up the single-DOF control approach and allow for the control of real-world robotic arms with multiple DOF, we use the SNN as a control block and independently apply it to each DOF of the arm’s joints.
Figure 3. Coordination of independent joint controllers. The motor activation in the leading joints inhibits the activation of the subordinate joints. In this way, the motion initiation in the subordinate joints is temporally delayed, allowing those joints to exploit the inertia of the leading joint's motion, as observed in human subjects [17].

Figure 4. Loihi-robot interaction control loop. Loihi communicates with the Local Host through a Read and a Write channel controlled by the on-chip x86 core (SNIP). The neuron spikes are decoded and used to update the simulated servos. The servo states are read out and fed back to Loihi.

We draw inspiration from the experimentally observed joint coordination in human arms [8, 17] and apply these principles in the proposed arm controllers to improve the control behavior (figure 3). Specifically, we incorporate the leading joint hypothesis [17], which postulates that in human arms, the joints that are closer to the body lead the movement, and the ones further down the limb (subordinate) follow. In that way, the subordinate joints exploit the leading joint's inertia. To account for this, we stimulate the motor neurons of the subordinate joint with the spikes of the leading joint motor neurons. For the Reacher-v2, we define the first joint (shoulder) as leading and the second (elbow) as subordinate. For the Jaco arm, we define the first and second joints as leading and the remaining four as subordinate.

4.4. Controller-robot interaction
To allow the SNN to control the robot arms, we translate each spike of the motor neurons to an incremental change (positive for E and negative for F) in the angle of the respective joint. For the Reacher-v2, we first ran a calibration simulation to determine the nominal torques that are required to change each joint angle by the desired increment. Then, we translate each E/F spike during the arm control to a positive/negative torque increment of that magnitude. For the Jaco2 arm, we use the SNN to control the angle of each joint in the configuration space. Here, we predefine an angle increment and translate each spike of E/F to an increase/decrease of the respective angle by that increment. Then, we translate the instructed angles to torques using an analytic operational space controller [32].

The interaction between Loihi and the simulation environments is realized using the framework proposed in [60] to allow for real-time control of the robot arms (figure 4). For this, we use a Read channel controlled by the low-frequency x86 Loihi cores (SNIP) to read out the spiking activity of the motor neurons and translate it to joint movements. In real-time, estimates of $\theta$ and $d\theta/dt$ are fed as inputs to the corresponding PPC neurons through a Write channel, which sets the biases of these PPC neurons.
5. Results

5.1. Demonstrating smooth control and identifying its origins
Providing smooth control is one of the main goals of our design. As a proof of concept, we use our SNN controller to drive a discrete movement of the end effector of the Reacher-v2 arm (figure 5(A)). To quantify the smoothness of our control approach, we compute a well-established smoothness metric [27] the motion jerk (the second derivative of the motion velocity figure 5). We show the recorded trajectory and the evaluation metrics (joint velocity, acceleration, and jerk) after a single simulation because the absence of stochastic components in the proposed controller does not introduce any intra-session variability. Mimicking the behavior of its biological counterpart [20], the SNN controller gives rise to bell-shaped velocity profiles (figure 5(B)). The gradual acceleration and timely deceleration of the end-effector (figure 5(C)) confined the magnitude of the motion jerk (figure 5(D)), suggesting that the proposed SNN indeed provides smooth control of the robotic arm.

To determine the contribution of each component of the SNN to the desired control behavior, we sequentially ablate the mechanisms and evaluate the control dynamics. First, we deactivate the adaptation mechanism of PSI and use the modified SNN controller to drive the exact same discrete movement. While the initial phase of the movement is unaffected in the absence of PSI, the peak end-effector velocity is not adequately confined (figure 5(B), red vs. blue curves). The deceleration from this higher peak velocity is more abrupt (figure 5(C)), leading to an increase in the motion jerk (both in terms of absolute magnitude and integral). Next, we deactivate both the PSI and the synaptic facilitation mechanisms, and the remaining SNN drives the same discrete movement. In the absence of the two mechanisms, both the increase and the decrease in the end-effector speed are abrupt, and the peak velocity is even higher (figure 5(B), green vs. blue curve). The consequent larger values of acceleration (figure 5(C)) result in significantly increased motion jerk (figure 5(D)).

5.2. Biological relevance of the motion trajectories
The biological inspiration of the SNN architecture made us wonder whether the resultant arm movements are similar to those observed in human arms. To examine this, we apply the controller to the Reacher-v2 arm to drive a sequence of discrete movements and record the trajectories of its end-effector (figure 6). At predefined intervals and regardless of the completion of the previous movements, a new, random goal position is defined for the end-effector. This workspace position is provided to an Inverse Kinematics (IK)
Figure 6. End-effector trajectory for the 2-link arm when driven by the SNN controller to perform discrete movements. The curvature of the trajectories is in agreement with experimental evidence in human arms (see [33], figure 2).

A solver that calculates the joint configuration that would drive the robot arm to the desired position. Then, the resulting angles are fed to the SNN controllers of the respective joints to drive their movements. As expected, the SNN controller provides smooth control with a gradual initial increase and timely decrease in the end-effector speed (as shown by the increased point density around the targets). Interestingly, the shape of the trajectories resembles the sigmoidal trajectories with curvatures in the early and late stages of the movement interrupted by straight intermediate intervals as observed in human subjects during adulthood and shown in [33]. Considering that the proposed controller does not include motion planning beyond the calculation of the initial and final joint configurations, this natural emergence of slightly curved trajectories suggests that this human-like behavior results from the biologically plausible SNN architecture.

5.3. Application to multi-DOF robot arms

To demonstrate the applicability of the approach to real-world robotic arms, we scale up the proposed SNN to control a Jaco arm with 6 DOF. We use one controller block for each joint and introduce joint coordination as previously described. Here, the first joint of the arm acts as leading for the second, and both of them act as leading for the other four. In that way, we exploit the inertia of the first joint in the x-y plane and the inertia of the second joint along the z-axis. We utilize the SNN controller to drive the discrete movement of the arm’s end-effector between three points in the workspace and record its trajectory (figure 7(A)). As in the Reacher-v2 experiment, a new, random goal position for the end-effector is generated at predefined time instances and regardless of the completion of previous movements. Then, the IK solver for the Jaco arm transforms the workspace positions to joint configurations, and the target joints are fed to the respective controllers to drive the arm to the desired position. As discussed above, we only show the results of a single simulation since there was no variability across simulations. As in the case of the Reacher-v2 arm, the end-effector trajectory is smooth, with gradual deceleration when approaching the targets. This behavior of the end-effector is a result of the smooth control of each joint separately. As an example, we show the dynamics of one arm joint controlled by the proposed SNN (figure 7(B)). The joint moves accurately and smoothly to the desired angles without overshoot (figure 7(B), Angle) while preserving the bell-shaped angular speed profiles (figure 7(B), Velocity middle panel and continuous accelerations (figure 7(B), Acceleration) that keep the motion jerk to minimal levels (figure 7(B), Jerk). The spiking activity of the neurons in the respective SNN controller block explains the control dynamics of the single joint (figure 7(C)). Specifically, for each step movement, one of the ePPCs is activated by the position error and drives its respective motor neuron (E/F) to decrease it. As the joint starts moving toward the target, the respective dPPC is activated due to the change in the angular speed and, in turn, stimulates the antagonist motor neuron (F/E). This counteraction results in a timely joint deceleration when approaching the target, contributing to the smoothness of the control. Lastly, the PSI neuron contributes to the fine-tuning of the movement: This neuron is highly active in the early stages of the movement when the positional error is high. In that way, it downscales the synaptic weight between the ePPCs and the motor neurons to avoid the overshoot induced by strong E/F activation. As the error decreases towards the end of the movement, the activity of PSI fades, allowing for the gradual increase of the synaptic weight to its maximal value.
The larger weight is then sufficient to accurately correct the small positional errors without inducing overshoots.

5.4. Comparison with conventional PID control

After demonstrating that the proposed SNN controller is indeed applicable to real-world robotic arms with multiple DOF, we evaluate the approach by comparing it with a state-of-the-art PID controller that is widely used in similar applications. Specifically, we use as the baseline the operational space controller proposed in [32], which is optimized for the control of the Jaco arm in the Applied Brain Research control library.

The set of tuned PID parameters are $K_p = 20$ for the proportional component, $K_v = 30$ for the derivative component, and $K_i = 0.001$ for the integral component of the position, while $K_o = 180$ is set for the proportional control of the orientation. We then tune the SNN controller's parameters to approximate the control profiles (convergence time, accuracy).

We compare the step response of the two controllers when required to drive a single robot joint from an initial to a desired configuration (figure 8). As shown by the control performance of the first leading joint (figure 8), the SNN controller implemented on either a CPU or Loihi achieves comparable performance to

---

4 The Applied brain research control library can be found at: https://github.com/abr/abr_control. Copyright (c) 2017–2022 Applied Brain Research. Used with permission.
Table 1. Controller Metrics Comparison.

|                      | Overshoot ($\theta_{\text{max}} - \theta_{\text{des}} \%$) | Rise time (10%–90% $\theta_{\text{des}}$) | Settling time (<20%$\theta_{\text{des}}$) |
|----------------------|------------------------------------------------------------|------------------------------------------|------------------------------------------|
| PID                  | 0                                                         | 1136.9 ms                                | 945.5 ms                                 |
| SNN (CPU)            | 0                                                         | 724.3 ms                                 | 723.8 ms                                 |
| SNN (Loihi)          | 1.7                                                       | 804 ms                                   | 881 ms                                   |

Figure 9. Comparison of the end-effector’s jerk when driven by the SNN (black) and a PID (red) controller. The maximal jerk was lower for the SNN controller resulting in more smooth motion.

the PID. We quantify this comparison by measuring the overshoot, the rise time, and the settling time of the controllers (table 1) [36]. The SNN controllers exhibit a similar rise and settling time with the PID. While the SNN controllers are slightly faster than the PID, we clarify that the goal is not to outperform the conventional control method but to achieve comparable results with a biologically plausible and, therefore, neuromorphic-compatible approach. When implemented on Loihi, the SNN controller exhibits a small (1.7%) overshoot which is absent in the CPU implementation. This is expected due to the absence of presynaptic inhibition in the Loihi deployment, which further highlights the contribution of this adaptation mechanism to the control.

With motion smoothness being the main goal for the proposed controller design, we further examine this aspect of its performance and compare it against the state-of-the-art PID controller. Both controllers drive the same discrete movement of the Jaco arm (section IV.D) and we evaluate the motion smoothness (figure 9). To do so, we use the well-established jerk metric (derivative of the acceleration) [35], which is minimal in smooth human arm movements [21]. When using the PID controller, an initial spike dominates the motion jerk, which quickly decays to zero subsequently. On the other hand, the SNN controller gives rise to a gradual increase in the motion jerk that persists longer, decreasing the maximum jerk by 19%. With jerk quantifying how abrupt the movement is, high instantaneous values deteriorate the movement smoothness more than smaller and persistent ones. Therefore, the proposed method indeed improves motion smoothness, approximating the minimum jerk profiles in human arms.

6. Discussion

In this paper, we presented an SNN controller which draws inspiration from biological findings on how primates perform smooth reaching movements [20]. We evaluated our controller on two robotic arms: (i) the simple but human-like two-link arm Reacher-v2, and (ii) the real-world 6-DOF Jaco arm. Our results show that the controller not only drove the arm to the target accurately but also provided smooth control, thus making it applicable for a wide variety of robotic applications, such as prosthetics [26], assistive arms [45], or industrial arms [73], that may require such control.

Adding to the mounting evidence that demonstrates the role of neuroscience in advancing robotics [18, 28, 40, 48, 52], we show here how the emulation of neuronal subnetworks with specialized functions can provide similar behavior on neuromorphic-controlled robots. Several recent studies have attempted to harness the advantages of neuromorphic platforms for low-level control [12, 75, 77]. Their performance approximates that of the conventional microcontrollers, showing that the non-Von Neumann architecture of the hardware is not a compounding factor. However, the deployment of conventional control methods on neuromorphic hardware disregards the underlying neuroscience principles. As a result, such methods only partially exploit the low complexity of the minimal brain networks. Here, we propose an SNN controller that emulates (i) the experimentally identified GABA-ergic pathways that regulate the gain of sensory feedback
afferents, and (ii) the short-term synaptic plasticity that enforces gradual motion initiation. The combination of these two mechanisms in our SNN architecture results in smooth joint control that resembles the dynamics of reaching movements in primates [35] and rodents [20]. This biological plausibility not only resulted in a minimal network in terms of size and complexity but also made it a perfect fit for deployment on neuromorphic hardware. Functionally, the controller inherited the behavior of its biological counterpart, as control smoothness emerged naturally through the gradual increase and timely and steady decline of the controlled variable.

While the biologically inspired design of our SNN controller resulted in the above mentioned architectural and functional advantages, it also gave rise to some limitations. First, our controller, similarly to other SNN solutions, lacks the concrete mathematical description of a PID controller. Drawing inspiration from biological networks controlling reaching movements, our SNN design is interpretable at the behavioral level. However, a thorough analysis of the control behavior is less straightforward when compared to conventional control methods. Second, our SNN controller introduced more tunable parameters than the three PID coefficients. Although the behaviorally interpretable nature of our design simplifies the tuning of the network hyperparameters, their increased number introduces additional complexity. Lastly, the binary nature of our spiking controller might introduce limited flexibility when compared with a continuous conventional controller. However, our method could be expanded with and benefit from previously proposed SNN controllers [31] that attempt to overcome this limitation.

Overall, this work supports ongoing efforts to develop real-time and robust neuromorphic solutions for robot control exhibiting desired behaviors [19]. The proposed methods could be combined with emerging solutions for the inverse-kinematics calculation of high-dimensional robotic arms [67, 76] to provide end-to-end neuromorphic control. Unlike other methods, here we aimed to derive computational principles from the knowledge of how the brain attains optimal motor behavior and introduce them to the design of neuromorphic robotic controllers, adding to the growing pool of non-cognitive neuromorphic applications [1]. In the light of our results, which combine low computational complexity and interpretability with a performance comparable to the state-of-the-art, the direction of developing brain-inspired solutions to robotic problems on neuromorphic hardware [54] is worth exploring further.

Data availability statement

The data cannot be made publicly available upon publication due to legal restrictions preventing unrestricted public distribution. The data that support the findings of this study are available upon reasonable request from the authors.

ORCID iD

Ioannis Polykretis  https://orcid.org/0000-0003-1794-5716

References

[1] Aimone J et al 2022 A review of non-cognitive applications for neuromorphic computing Neuromorph. Comput. Eng. 2 032003
[2] Ajoudani A, Zanchettin A M, Ivaldi S, Albu-Schaffer A, Kosuge K and Khatib O 2018 Progress and prospects of the human–robot collaboration Auton. Robots 42 957–75
[3] Arber S 2012 Motor circuits in action: specification, connectivity and function Neuron 74 975–89
[4] Ata A A 2007 Optimal trajectory planning of manipulators: a review J. Eng. Sci. Technol. 2 32–54
[5] Baker S N 2007 Oscillatory interactions between sensorimotor cortex and the periphery Curr. Opin. Neurobiol. 17 649–55
[6] Balasubramanian S, Melendez-Calderon A, Roby-Brami A and Burdet E 2015 On the analysis of movement smoothness J. NeuroEng. Rehabil. 12 1–11
[7] Brogårdh T 2007 Present and future robot control development–an industrial perspective Annu. Rev. Control 31 69–79
[8] Buchanan J J 2004 Learning a single limb multijoint coordination pattern: the impact of a mechanical constraint on the coordination dynamics of learning and transfer Exp. Brain Res. 156 39–54
[9] Calimeri A, Maci E and Poncino M 2013 The human brain project and neuromorphic computing Funct. Neuro. 28 191
[10] Capaday C and Stein R 1987 Difference in the amplitude of the human soleus H reflex during walking and running J. Physiol. 392 513–22
[11] Capaday C and Stein R 1987 A method for simulating the reflex output of a motoneuron pool J. Neurosci. Methods 21 91–104
[12] Stagsted R K, Vitale A, Renner A, Larsen L B, Christensen A L and Sandamirskaya Y 2020 Event-based PID controller fully realized in neuromorphic hardware: a one DOF study 2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (IEEE) pp 10939–44
[13] Constantinides D and Croft E A 2000 Smooth and time-optimal trajectory planning for industrial manipulators along specified paths J. Robot. Syst. 17 233–49
[14] Crespi A, Badertscher A, Guignard A and Ijspeert A J 2005 AmphibBot I: an amphibious snake-like robot Robot. Auton. Syst. 50 163–75
[15] Davies M et al 2018 Loihi: a neuromorphic manycore processor with on-chip learning IEEE Micro 38 82–99
[16] DeWolff T, Patel K, Jaworski P, Leontie R, Hays J and Eliasmith C 2023 Neuromorphic control of a simulated 7-DOF arm using Loihi Neuromorph. Comput. Eng. 3 014007
[58] Strohmer B, Manoonpong P and Larsen L B 2020 Flexible spiking CPGs for online manipulation during hexapod walking Front. Neurorobot. 14 41
[59] Tan J, Liu K and Turk G 2011 Stable proportional-derivative controllers IEEE Comput. Graph. Appl. 31 34–44
[60] Tang G, Kumar N and Michmizos K P 2020 Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware (arXiv:2003.01157)
[61] Tang G, Shah A and Michmizos K P 2019 Spiking neural network on neuromorphic hardware for energy-efficient unidimensional SLAM 2019 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (IEEE) pp 4176–81
[62] Taunyazoz T, Sng W, See H H, Lim B, Kuang J, Angrani A F, Tee B and Soh H 2020 Event-driven visual-tactile sensing and learning for robots Proc. Robotics: Science and Systems
[63] Thakur C S et al 2018 Large-scale neuromorphic spiking array processors: a quest to mimic the brain Front. Neurosci. 12 891
[64] Todorov E 2004 Optimality principles in sensorimotor control Nat. Neurosci. 7 907–15
[65] Tsodyks M, Pawelzik K and Markram H 1998 Neural networks with dynamic synapses Neural Comput. 10 821–35
[66] Uno Y, Kawato M and Suzuki R 1989 Formation and control of optimal trajectory in human multijoint arm movement Biol. Cybern. 61 89–101
[67] Volinski A, Zaidel Y, Shalumov A, DeWolf T, Supic L and Tsur E E 2022 Data-driven artificial and spiking neural networks for inverse kinematics in neurorobotics Patterns 3 100391
[68] Wang H, Wang H, Huang J, Zhao B and Quan L 2019 Smooth point-to-point trajectory planning for industrial robots with kinematical constraints based on high-order polynomial curve Mech. Mach. Theory 139 284–93
[69] Whishaw I Q 1996 An endpoint, descriptive and kinematic comparison of skilled reaching in mice (Mus musculus) with rats (Rattus norvegicus) Behav. Brain Res. 78 101–11
[70] Windhorst U 1996 On the role of recurrent inhibitory feedback in motor control Prog. Neurobiol. 49 517–87
[71] Winters J M and Crago P E 2012 Biomechanics and Neural Control of Posture and Movement (Berlin: Springer)
[72] Wu L-G and Saggau P 1997 Presynaptic inhibition of elicited neurotransmitter release Trends Neurosci. 20 204–12
[73] Xiao Y, Du Z and Dong W 2012 Smooth and near time-optimal trajectory planning of industrial robots for online applications Ind. Robot 39 169–77
[74] Yin K, Cline M B and Pai D K 2003 Motion perturbation based on simple neuromotor control models 11th Pacific Conf. on Computer Graphics and Applications, 2003 Proc. (IEEE) pp 445–9
[75] Zaidel Y, Shalumov A, Volinski A, Supic L and Tsur E E 2021 Neuromorphic NEF-based inverse kinematics and PID control Front. Neurorobot. 15 631159
[76] Zhao J, Monforte M, Indiveri G, Bartolozzi C and Donati E 2022 Learning inverse kinematics using neural computational primitives on neuromorphic hardware
[77] Zhao J, Risi N, Monforte M, Bartolozzi C, Indiveri G and Donati E 2020 Closed-loop spiking control on a neuromorphic processor implemented on the iCub IEEE J. Emerg. Sel. Top. Circuits Syst. 10 546–56