Design and validation of a real-time spiking-neural-network decoder for brain–machine interfaces

2013 J. Neural Eng. 10 036008

(http://iopscience.iop.org/1741-2552/10/3/036008)

J. Neural Eng. **10** (2013) 036008 (12pp) doi:10.1088/1741-2560/10/3/036008

Design and validation of a real-time spiking-neural-network decoder for brain–machine interfaces

Julie Dethier[1,6,7,8], **Paul Nuyujukian**[1,2,6], **Stephen I Ryu**[3,4], **Krishna V Shenoy**[1,2,3,5] **and Kwabena Boahen**[1,3]

[1] Department of Bioengineering, Stanford University, Stanford, CA 94305, USA
[2] Medical Scientist Training Program, Stanford University, Stanford, CA 94305, USA
[3] Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA
[4] Department of Neurosurgery, Palo Alto Medical Foundation, Palo Alto, CA 94301, USA
[5] Department of Neurobiology, Stanford University, Stanford, CA 94305, USA

E-mail: boahen@stanford.edu

Received 27 December 2012
Accepted for publication 19 March 2013
Published 10 April 2013
Online at stacks.iop.org/JNE/10/036008

Abstract

Objective. Cortically-controlled motor prostheses aim to restore functions lost to neurological disease and injury. Several proof of concept demonstrations have shown encouraging results, but barriers to clinical translation still remain. In particular, intracortical prostheses must satisfy stringent power dissipation constraints so as not to damage cortex. *Approach*. One possible solution is to use ultra-low power neuromorphic chips to decode neural signals for these intracortical implants. The first step is to explore in simulation the feasibility of translating decoding algorithms for brain–machine interface (BMI) applications into spiking neural networks (SNNs). *Main results*. Here we demonstrate the validity of the approach by implementing an existing Kalman-filter-based decoder in a simulated SNN using the Neural Engineering Framework (NEF), a general method for mapping control algorithms onto SNNs. To measure this system's robustness and generalization, we tested it online in closed-loop BMI experiments with two rhesus monkeys. Across both monkeys, a Kalman filter implemented using a 2000-neuron SNN has comparable performance to that of a Kalman filter implemented using standard floating point techniques. *Significance*. These results demonstrate the tractability of SNN implementations of statistical signal processing algorithms on different monkeys and for several tasks, suggesting that a SNN decoder, implemented on a neuromorphic chip, may be a feasible computational platform for low-power fully-implanted prostheses. The validation of this closed-loop decoder system and the demonstration of its robustness and generalization hold promise for SNN implementations on an ultra-low power neuromorphic chip using the NEF.

(Some figures may appear in colour only in the online journal)

1. Cortically-controlled brain–machine interfaces

1.1. The challenge

Neural motor prostheses aim to help disabled individuals to use computer cursors or robotic limbs by extracting neural signals from the brain and decoding them into useful control signals (figure 1). Several proof-of-concept demonstrations have shown encouraging results [1–10]. Recently, a cortically-controlled motor prosthesis capable of quick, accurate, and robust computer cursor movements by decoding neuronal activity from 96-channel microelectrode arrays in monkeys has been reported [11–13]. This prosthesis, similar to other designs (e.g. [14]), uses a Kalman-filter-based decoder ubiquitous in statistical signal processing. Such a filter and its variants have

[6] These authors contributed equally to this work.
[7] Research Fellow.
[8] Present address: FRS-FNRS, Systmod Unit, Montefiore Institute, University of Liege, Belgium.

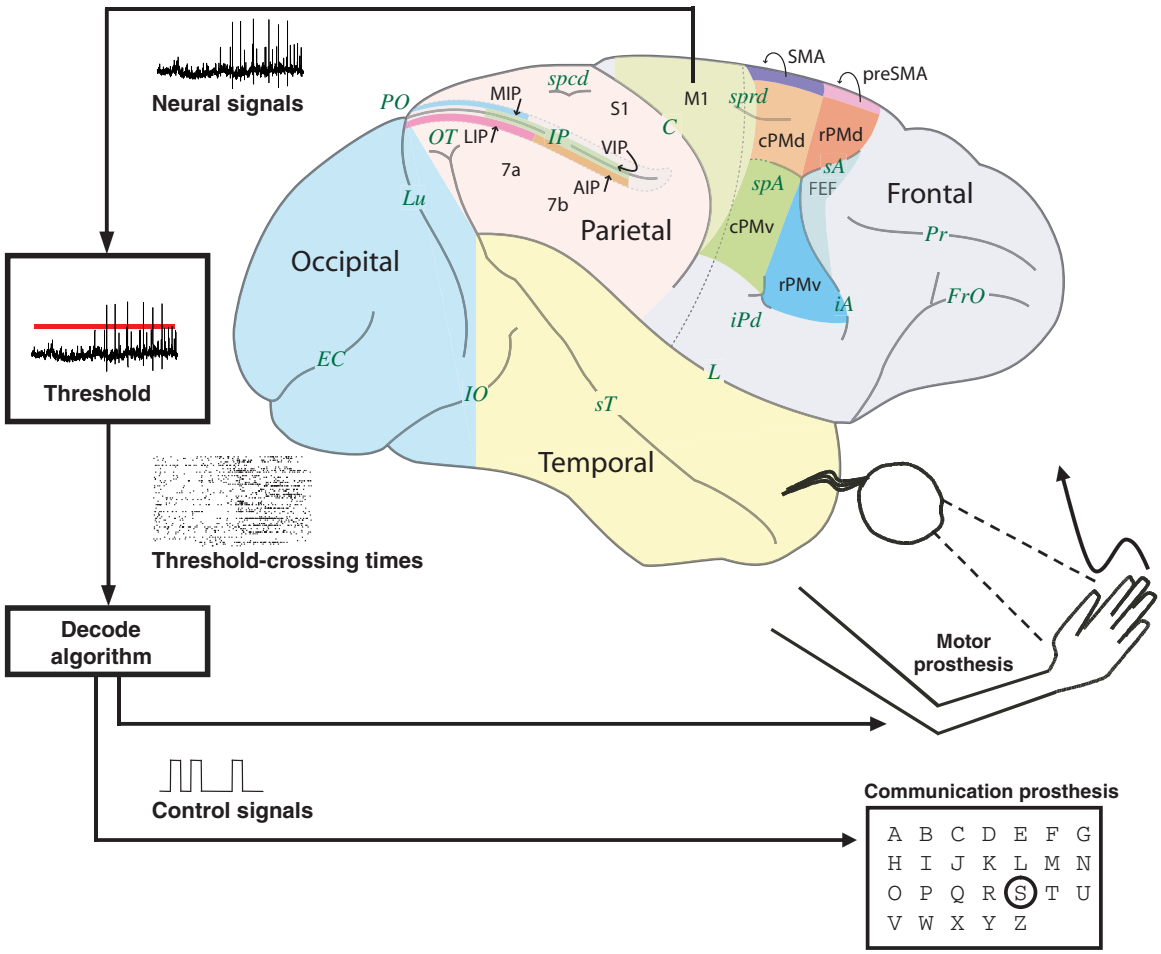

Figure 1. System diagram of a BMI. Electronic neural signals are measured from motor regions of the brain, converted to spike trains by detecting threshold-crossings, and translated (decoded) into control signals for a computer or prosthetic arm. Current implementations record neural signals with an implanted silicon microelectrode array, threshold and wirelessly transmit these signals with a battery-powered head-mounted unit, and decode them with a remote desktop computer.

demonstrated the highest levels of brain–machine interface (BMI) performance in both humans [14] and monkeys [13]. Even though successes with non-linear alternative decoder types, such as the unscented Kalman filter [15] and the population vector algorithm [16], are reported in the literature, this paper focuses on the linear version of the Kalman filter because of its ease of implementation in steady-state and the previous experience with the ReFIT-KF training algorithm [13].

These decoding systems can be implemented in one of three general reference designs (figure 2). Each design makes a tradeoff among available space for electronics, power budget (both acceptable dissipation and battery capacity), risk of infection, and cosmetic appearance. The first design implants only the recording electrodes under the skull, placing the electronics outside of the skin (see figure 2(A)). Examples of this approach are the Neurochip [17, 18] and the Hermes systems [19–23]. This design does not have stringent power constraints because the electronic systems are not in proximity of the brain and make little contact with the skin, so there is little risk for heating tissue. Systems in this class draw 20–200 mW and run on battery power for hours to weeks. The second design differs from the first one in that electronics are fully implanted beneath the skin, but are not in direct contact with the brain [24] (see figure 2(B)). Here again there is little risk of heating the brain, but the electronics may heat the cranium. However, since the system is implanted, space constraints limit the footprint of the electronics and the battery capacity—inductive power is a viable alternate power source. Systems of this design decrease the risk of infection due to the closure of the skin and the lack of a chronic wound margin. The third design is fully implanted under the skull [25, 26] (see figure 2(C)). This design has a further decreased infection risk relative to under-skin implantation because the dura is completely closed after the device is implanted, minimizing cerebrospinal fluid leak and maintaining the blood–brain barrier. This third design has the tightest space constraints while being the most cosmetically favorable of the three. Power for this design would be inductively delivered. In this scenario, because electronics are so close to brain tissue, heat dissipation is of greatest concern. The need to minimize power dissipation while providing a high level of computational ability makes the neuromorphic approach a potentially enabling technology for the second and third reference designs.

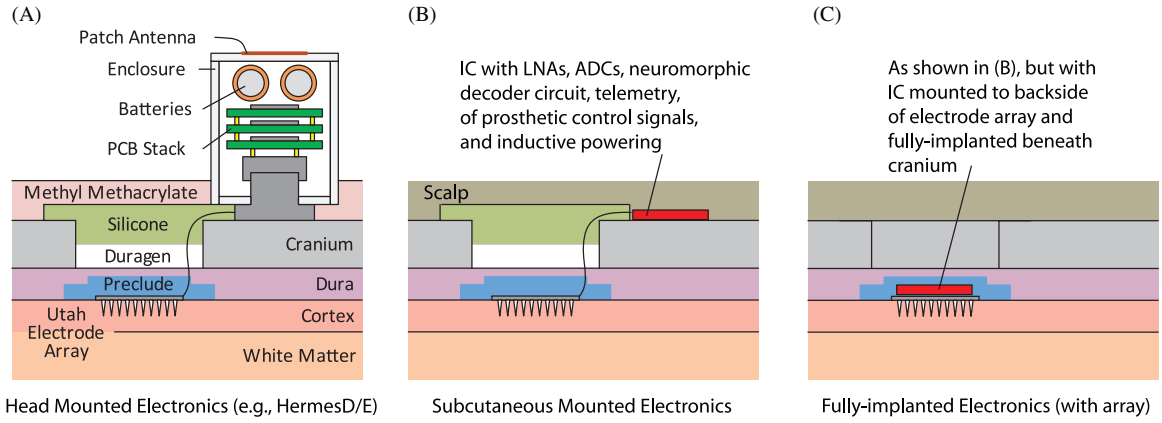

Figure 2. Cross-section of three reference designs for intracortical neural prostheses. (A) Externally head-mounted. The electrode array is the only intracortical device. The electronics are externally mounted, fixed against the cranium, with little concern of heating. This system can take advantage of a large power budget. (B) Fully-implanted beneath the scalp. The electronics, together with the rest of the electronics, are placed on the skull, but below the skin, allowing for wound closure. (C) Fully-implanted beneath the skull. The electronics are bonded to the backside of the electrode array and hermetically sealed. This is the only design where the dura is fully closed.

The lack of low-power electronic circuitry to run decoding algorithms is an obstacle to the successful clinical translation of implantable cortical motor prostheses. A design constraint set by the American Association of Medical Instrumentation for implanted medical devices specifies that a device should not increase the temperature of tissue chronically by more than 1 °C. This translates to a power dissipation limit of 40 mW cm^{-2} [27]. For the 4 × 4 mm^2 electrode array commonly used in neural prostheses applications in a 6 × 6 mm^2 hermetically sealed package, this places a power dissipation limit of about 10 mW for an implanted design [28]. A modern x86 processor consumes approximately 1.8 mW [29] just to perform 2D Kalman filtering on a 96-channel array. Thus this approach will not meet the demands of recording from higher electrode densities and controlling more degrees of freedom, which require substantially more computer-intensive decode/control algorithms.

In the first reference design (figure 2(A)), decoding occurs remotely, enabling the power budget to be allocated entirely to signal preprocessing (amplifying, filtering and digitizing), data preconditioning (syncing, scrambling, and coding), and wireless transmission. For 96 signals sampled at 31.25 KHz and digitized at 10 bits (30 Mb s^{-1}), a 120 nm-CMOS FPGA preconditioner consumes 14 mW (scaled from [22]) and a 65 nm-CMOS UWB transmitter consumes 0.35 mW (8.5 pJ b^{-1}) [30]. Assuming a custom preconditioner implementation consumes two orders of magnitude less power, the preconditioning and transmission power for 96 channels can be reduced to 0.49 mW. Extrapolating this number yields 5 mW for the 1000 channels thought to be needed for fast and robust control of a six degree-of-freedom robotic arm [31]. This figure can be reduced from 50% to 0.005% of the second and third reference designs' (figures 2(A) and (B)) 10 mW power budget if decoding is performed locally before wireless transmission. The data rate is reduced from 30 Mb s^{-1} to 3 Kb s^{-1}—six signals sampled at 50 Hz and digitized at 10 bits—with a proportionate reduction in power. A viable approach if the neural signals can be decoded using a lot less power than it takes to transmit them in raw form[9].

1.2. The neuromorphic alternative

The required power constraints could be met with an innovative ultra-low power technique: the *neuromorphic* approach. This approach follows the brain's organizing principles and uses large-scale integrated systems containing microelectronic analogue circuits to *morph* neural systems into silicon chips [34, 35]. It combines the best features of analogue and digital circuit design—efficiency and programmability—offering potentially greater robustness than either of them [34, 35]. With as little as 50 nW per silicon neuron when spiking at 100 Hz [36], these neuromorphic circuits may yield tremendous power savings over conventional digital solutions because they use an analogue approach based on physical operations to perform mathematical computations.

Before designing and fabricating such a dedicated neuromorphic decoding chip, we explored the feasibility of translating decoding algorithms into a spiking neural network (SNN) in software. Recent studies have highlighted the utility of neural networks for decoding in both off-line [37] and online [38] settings. Similarly, we encouragingly achieved off-line (open-loop) SNN performance comparable to that of the traditional floating point implementation [29]. We also realized simulation algorithm enhancements that enable the real-time execution of a 2000-neuron SNN on x86 hardware and reported preliminary closed-loop results obtained with a single monkey performing a single task [39].

In this study, we extended our preliminary closed-loop tests of the SNN Kalman-filter-based decoder from one to two monkeys and evaluated its performance on multiple tasks. We first describe the linear filter we used in our system, the Kalman filter already presented in [11–13], and how it functions as a

[9] Previous studies suggest that low-noise amplification, analogue-to-digital conversion, and spike sorting do not prevent scaling to 1000 channels as they can be performed with a total of 3.5 μW per channel if the input-referred noise specification is relaxed from 2.2 to 13 μV [32, 33, 21].

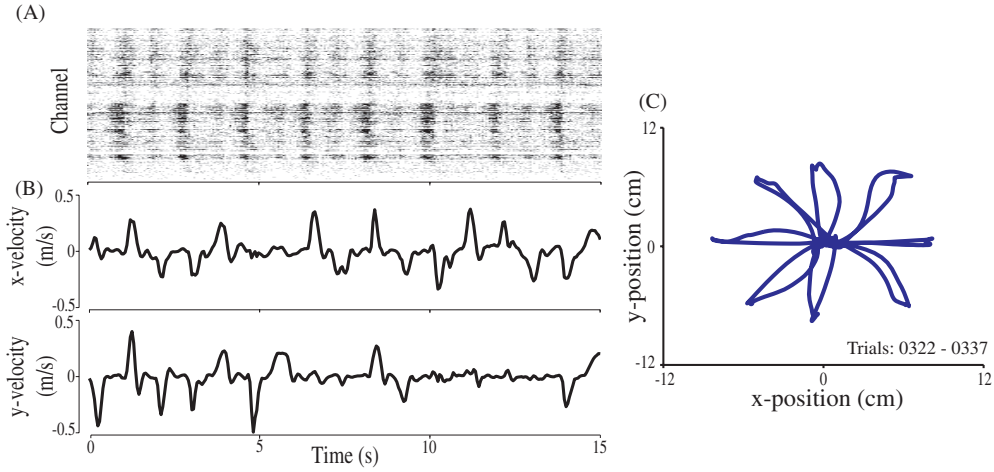

Figure 3. Neural and kinematic measurements (monkey J, 16 April 2011, 16 continuous trials) used to fit the Kalman-filter model. (A) The 192 channel recordings that were fed as input to fit the Kalman-filter matrices (grayscale refers to the number of threshold crossings observed in each 50 ms bin; black represents 4 spikes/bin). (B) Hand *x*- and *y*-velocity measurements that were correlated with the neural data to obtain the Kalman-filter matrices. (C) Hand position kinematics of the 16 continuous trials recorded using a Polaris Optical Measurement System (Northern Digital Inc.).

decoder. Next, the Neural Engineering Framework (NEF), the method we use to map this algorithm on to an SNN, is outlined, together with the optimizations that enabled the software-based SNN to operate in a closed-loop experimental setting. An analysis of the performance achieved in the closed-loop tests demonstrates the validity of this approach and concludes this paper.

2. Kalman-filter-decoder algorithm

In the 1960s, Rudolf E Kálmán described a type of linear filter that will be later called the *Kalman filter* [40]. This filter tracks the state of a dynamical system throughout time using a weighted average of two types of information: a value predicted from the output state's dynamics and a value measured from external observations, both subject to noise and other inaccuracies. More weight is given to the value with the lower uncertainty as calculated by the *Kalman gain*, **K**, which is computed as follows.

The system is modeled by the following set of equations:

$$\boldsymbol{x}_t = \mathbf{A}\boldsymbol{x}_{t-1} + \boldsymbol{w}_t \tag{1}$$

$$\boldsymbol{y}_t = \mathbf{C}\boldsymbol{x}_t + \boldsymbol{q}_t \tag{2}$$

where **A** is the state matrix modeling the output state's dynamics, **C** is the observation matrix, and $\boldsymbol{w}_t$ and $\boldsymbol{q}_t$ are additive, multivariate Gaussian noise sources that are modeled with $\boldsymbol{w}_t \sim \mathcal{N}(0, \boldsymbol{W})$ and $\boldsymbol{q}_t \sim \mathcal{N}(0, \boldsymbol{Q})$. The model provides an estimate of the output at time t by propagating the state at time $t-1$. This estimate is then corrected using the noisy observations at time t to arrive at a final estimate:

$$\boldsymbol{x}_t = (\mathbf{I} - \mathbf{KC})\mathbf{A}\boldsymbol{x}_{t-1} + \mathbf{K}\boldsymbol{y}_t = \mathbf{M}_x\boldsymbol{x}_{t-1} + \mathbf{M}_y\boldsymbol{y}_t \tag{3}$$

where $\mathbf{K} = (\mathbf{I} + \mathbf{WCQ}^{-1}\mathbf{C})^{-1}\mathbf{WC}^{\mathrm{T}}\mathbf{Q}^{-1}$ is the steady-state formulation of the Kalman gain. $\mathbf{M}_x$ and $\mathbf{M}_y$ are the two matrices in the steady-state Kalman-filter-update equations. It is this steady-state equation that we implemented in the SNN using the NEF.

For neural prosthetic applications [13], the system's state vector, $\boldsymbol{x}_t$, was the kinematic state of the cursor: $\boldsymbol{x}_t = \left[\mathrm{pos}_t^x, \mathrm{pos}_t^y, \mathrm{pos}_t^z, \mathrm{vel}_t^x, \mathrm{vel}_t^y, \mathrm{vel}_t^z\right]$ and the measurement vector, $\boldsymbol{y}_t$, was the neural spike rate (binned threshold-crossing counts of measured neural potential) of the recorded neurons. For our system, the state vector was limited to velocities in two directions $\left(\boldsymbol{x}_t = \left[\mathrm{vel}_t^x, \mathrm{vel}_t^y\right]\right)$ to limit the amount of computational resources required (Kim *et al* [14] compare position and velocity decoders and finds that velocity decoders tend to outperform position decoders in tetraplegic patients). This velocity was integrated to yield the 2D position. The bin width of the decoder, a tradeoff between rapidity and accuracy [13], was 50 ms long. To allow for a fixed offset compensation (baseline firing rates), a constant 1 was added to the state vector: $\boldsymbol{x}_t = \left[\mathrm{vel}_t^x, \mathrm{vel}_t^y, 1\right]$. The neural spike rate observations $\boldsymbol{y}_t$ came from 96-channel silicon electrode arrays implanted in areas of cortex responsible for arm movement (M1 and PMd).

The model parameters (**A**, **C**, **W** and **Q**) were fit by correlating recorded neural signals and measured arm kinematics, obtained during training trials. Arm measurements were captured by a Polaris Optical Measurement System (Northern Digital Inc), with a passive reflective bead tied to the tip of the monkey's finger. The measurements were sampled at 60 Hz and were plotted directly to the screen. For our application, we assumed that neural signals recorded under arm control were similar to neural signals recorded under brain control, and that the observed arm kinematics matched the desired neural cursor kinematics. Therefore, the parameters could be fit from observed arm kinematics (figure 3).

For neuroprosthethic applications, the Kalman filter converges to its steady-state matrices, $\mathbf{M}_x$ and $\mathbf{M}_y$, rapidly—typically less than 100 iterations [41]. The steady-state implementation differs from the full Kalman-filter implementation by less than 1% in the first few seconds, for both open-loop and closed-loop systems [41]. The steady-state update equations decrease execution time for typical decoding

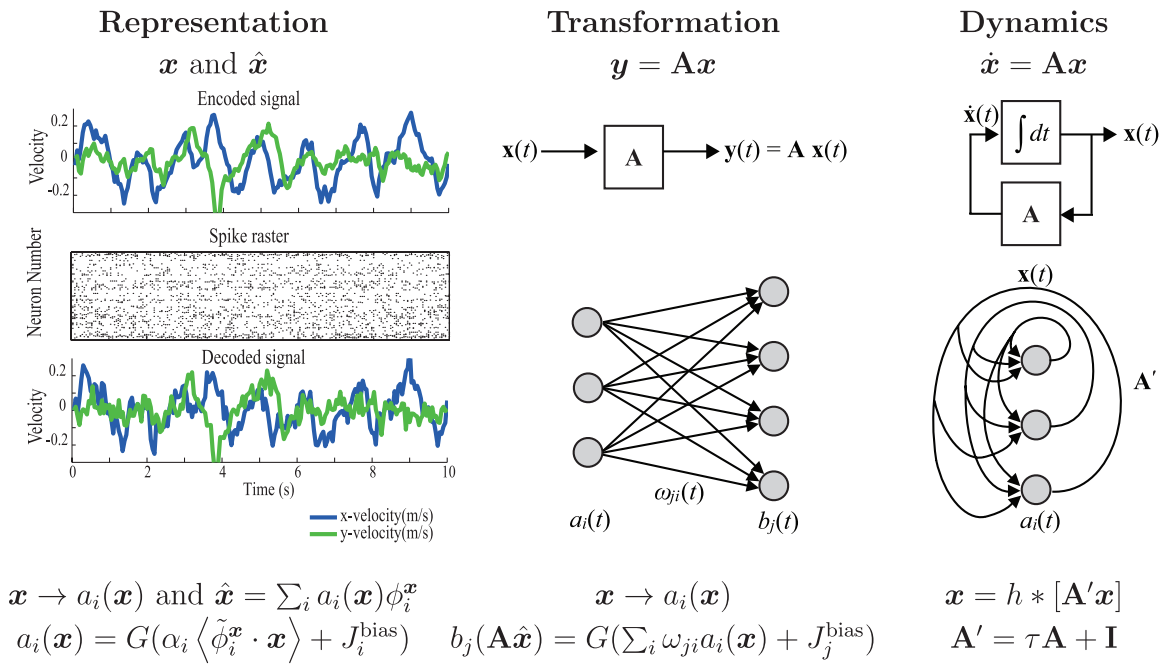

Figure 4. NEF's three principles. Top row represents the control-theory level and lower rows the neural level. *Representation.* Encoded signal, spikes raster, and decoded signal with a population of 200 leaky integrate-and-fire neurons. The neurons' tuning curves map the encoded signal ($\boldsymbol{x}$) to spike rates ($a_i(\boldsymbol{x})$); this nonlinear transformation is inverted by linear weighted decoding to retrieve the decoded signal ($\hat{\boldsymbol{x}}$). $G()$ is the neurons' nonlinear current-to-spike-rate function. *Transformation.* SNN with populations $a_i(t)$ and $b_j(t)$ representing $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$, respectively, with decoding, transformation, and encoding performed via the feedforward weights $\omega_{ji} = \alpha_j \langle \tilde{\phi}_j^{\boldsymbol{y}} \cdot \mathbf{A}\phi_i^{\boldsymbol{x}} \rangle$. *Dynamics.* The system's dynamics is captured in a neurally plausible fashion by replacing integration with the synapses' spike response, $h(t)$, and replacing the matrix with its neurally plausible equivalent, $\mathbf{A}' = \tau\mathbf{A} + \mathbf{I}$, to compensate.

by approximately a factor seven [41], a critical difference for real-time applications. Thus, efficiency is improved without any meaningful loss of accuracy.

To map the steady-state version of this linear neural decoder on to an SNN, we used the NEF: a formal methodology for mapping control-theory algorithms onto SNNs.

3. Mapping onto spiking neural networks

The SNNs employed in the NEF are composed of highly heterogeneous spiking neurons characterized by a nonlinear multi-dimensional vector-to-spike-rate function—$a_i(\boldsymbol{x}(t))$ for the ith neuron—with parameters (preferred direction, maximum firing rate, and x-axis intercept) drawn randomly from a wide distribution (standard deviation comparable to mean) and with connection strengths programmed to perform the desired computations. To map control-theory algorithms onto SNNs, three principles—neural representation, transformation, and dynamics—apply [42–45].

3.1. Representation

Neural representation is defined by nonlinear encoding of a stimulus, $\boldsymbol{x}(t)$, as a spike rate, $a_i(\boldsymbol{x}(t))$, combined with optimal weighted linear decoding of $a_i(\boldsymbol{x}(t))$ to recover, in the stimulus space, an estimate of $\boldsymbol{x}(t)$, $\hat{\boldsymbol{x}}(t) = \sum_i a_i(\boldsymbol{x}(t))\phi_i^{\boldsymbol{x}}$, where $\phi_i^{\boldsymbol{x}}$ are the optimal decoding weights (figure 4 *representation*).

3.1.1. Nonlinear encoding. The nonlinear encoding process is exemplified by the *neuron tuning curve*, which captures the overall encoding process from a multi-dimensional stimulus, $\boldsymbol{x}(t)$, to a one-dimensional soma current, $J_i(\boldsymbol{x}(t))$, and finally to a firing rate, $a_i(\boldsymbol{x}(t))$:

$$a_i(\boldsymbol{x}(t)) = G(J_i(\boldsymbol{x}(t))), \tag{4}$$

where $G()$ is the nonlinear function describing the firing rate's dependence on the current's value. In the case of the leaky integrate-and-fire neuron model (LIF) that we used for this application, this function $G()$ is given by:

$$G(J_i(\boldsymbol{x}(t))) = [\tau^{\text{ref}} - \tau^{\text{RC}} \ln(1 - J_{\text{th}}/J_i(\boldsymbol{x}(t)))]^{-1}, \tag{5}$$

where J_i is the current entering the soma of the cell, i indexes the neuron, τ^{ref} is the absolute refractory period, τ^{RC} is the membrane RC time constant, and J_{th} is the threshold current. The LIF neuron has two behavioral regimes: sub-threshold and super-threshold. The sub-threshold regime is described by an RC circuit with time constant τ^{RC}. When the sub-threshold soma voltage reaches the threshold, V_{th}, the neuron emits a spike $\delta(t - t_n)$. After this spike, the neuron is reset and rests for τ^{ref} seconds before it resumes integrating. Ignoring the soma's RC time-constant when specifying the SNN's dynamics is reasonable because the neurons cross threshold at a rate that is proportional to their input current, which thus sets the spike rate instantaneously, without any filtering [42].

The conversion from a multi-dimensional stimulus, $\boldsymbol{x}(t)$, to a one-dimensional soma current, J_i, is performed by

assigning to the neuron a preferred direction, $\tilde{\phi}_i^x$, in the stimulus space and taking the dot-product:

$$J_i(\boldsymbol{x}(t)) = \alpha_i\langle\tilde{\phi}_i^x \cdot \boldsymbol{x}(t)\rangle + J_i^{\text{bias}}, \tag{6}$$

where α_i is a gain or conversion factor, and J_i^{bias} is a bias current that accounts for background activity. In the one-dimensional case, the preferred direction vector reduces to a scalar, either 1 or −1, resulting in a positive or negative slope, respectively (i.e. ON neurons that increase their firing rate as the value of the stimulus variable increases and OFF neurons that do the opposite).

3.1.2. Linear decoding. The linear decoding process converts spike trains back into a relevant quantity in the stimulus space. This process is characterized by the synapses' spike response, $h(t)$ (i.e. post-synaptic current waveform), and the decoding weights, ϕ_i^x, which are found by a least-squares method [42], described next.

A single noise term, η, amalgamates all sources of noise, as they all have the effect of introducing uncertainty into any signal sent by the transmitting neuron. With this noise term, the transmitted firing rate can be written as $a_i(\boldsymbol{x}(t)) + \eta_i$. That is, the firing rate that the receiving neuron actually perceives is the noiseless firing rate, $a_i(\boldsymbol{x}(t))$, plus some variation introduced into that neuron's activity by the noise sources, η_i. The noise sources are modeled by a random variable drawn from a normal distribution. Consequently, the mean square error between the actual stimulus, $\boldsymbol{x}(t)$, and its estimate, $\hat{\boldsymbol{x}}(t) = \sum_i (a_i(\boldsymbol{x}(t)) + \eta_i)\,\phi_i^x$, can be written as [42]:

$$E = \frac{1}{2}\langle[\boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t)]^2\rangle_{x,\eta,t} \tag{7}$$

$$= \frac{1}{2}\left\langle\left[\boldsymbol{x}(t) - \sum_i (a_i(\boldsymbol{x}(t)) + \eta_i)\,\phi_i^x\right]^2\right\rangle_{x,\eta,t} \tag{8}$$

where $\langle\cdot\rangle_{x,\eta}$ denotes integration over the range of $\boldsymbol{x}$ and η, the expected noise. We assume that the noise is independent and has the same variance for each neuron [42], yielding:

$$E = \frac{1}{2}\left\langle\left[\boldsymbol{x}(t) - \sum_i a_i(\boldsymbol{x}(t))\phi_i^x\right]^2\right\rangle_{x,t} + \sigma^2\sum_i\left(\phi_i^x\right)^2, \tag{9}$$

where σ^2 is the noise variance. This expression is minimized by choosing the decoding weights such that:

$$\phi_i^x = \sum_j^N \Gamma_{ij}^{-1}\Upsilon_j, \tag{10}$$

with $\Gamma_{ij} = \langle a_i(\boldsymbol{x})a_j(\boldsymbol{x})\rangle_x + \sigma^2\delta_{ij}$, where δ is the Kronecker delta function, and $\Upsilon_j = \langle a_j(\boldsymbol{x})\boldsymbol{x}\rangle_x$ [42]. One consequence of modeling noise in the neural representation is that the matrix Γ in (10) is invertible despite the use of a highly overcomplete representation. In a noiseless representation, Γ would be generally singular because, due to the large number of neurons, there would be a high probability of having two neurons with similar tuning curves leading to two similar rows in Γ.

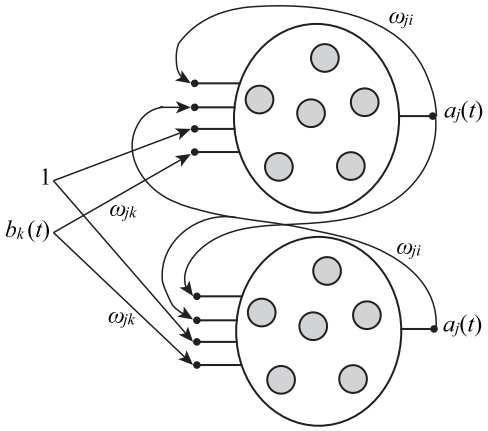

Figure 5. SNN implementation of a Kalman-filter-based decoder with populations $b_k(t)$ and $a_j(t)$ representing $\boldsymbol{y}(t)$ and $\boldsymbol{x}(t)$. Feedforward and recurrent weights, ω_{jk} and ω_{ji}, were determined by $\mathbf{B}'$ and $\mathbf{A}'$, respectively.

3.2. Transformation

Neural transformation is a special case of neural representation performed by using alternate decoding weights in the decoding operation. The transformation, $f(\boldsymbol{x}(t))$, is mapped directly into transformations of $a_i(\boldsymbol{x}(t))$ by using the appropriate linear decoders, $\phi_i^{f(\boldsymbol{x}(t))}$, to extract the function from the encoded information. For example, $\boldsymbol{y}(t) = \mathbf{A}\boldsymbol{x}(t)$ is represented by the spike rates $b_j(\mathbf{A}\hat{\boldsymbol{x}}(t))$ (figure 4 *transformation*), where unit j's input is computed directly from unit i's output using $\mathbf{A}\hat{\boldsymbol{x}}(t) = \sum_i a_i(\boldsymbol{x}(t))\mathbf{A}\phi_i^x$, an alternative linear weighting.

3.3. Dynamics

Neural dynamics brings the first two principles together and adds the time dimension to the circuit. This principle aims at reuniting the control-theory and neural levels by modifying the matrices to render the system *neurally plausible*, thereby permitting the synapses' spike response, $h(t)$, to capture the system's dynamics.

For example, in control-theory, an integrator is written $\dot{\boldsymbol{x}}(t) = \mathbf{A}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{y}(t)$ with Laplace transform $\boldsymbol{x}(s) = 1/s[\mathbf{A}\boldsymbol{x}(s) + \mathbf{B}\boldsymbol{y}(s)]$. In the neural space, convolution replaces integration and the system takes the form $\boldsymbol{x}(t) = h(t) * [\mathbf{A}'\boldsymbol{x}(t) + \mathbf{B}'\boldsymbol{y}(t)]$ and its Laplace transform is $\boldsymbol{x}(s) = h(s)[\mathbf{A}'\boldsymbol{x}(s) + \mathbf{B}'\boldsymbol{y}(s)]$. The synapses' spike response takes the form $h(t) = \tau^{-1}\mathrm{e}^{-t/\tau}$ where τ is the synaptic time constant, and the transfer function is $h(s) = 1/(1 + s\tau)$. The two systems are therefore equivalent if $\mathbf{A}' = \tau\mathbf{A} + \mathbf{I}$ and $\mathbf{B}' = \tau\mathbf{B}$, so called *neurally plausible* matrices (figure 4 *dynamics*).

4. Spiking neural network decoder

To implement the Kalman filter in an SNN by applying the NEF, we used the three principles described in the previous section (figure 5). To render the system neurally plausible as explained in section 3.3, we started from a continuous time (CT) system in the control-theory space, and we therefore converted (3) from discrete time to CT:

$$\dot{\boldsymbol{x}}(t) = \mathbf{M}_x^{\text{CT}}\boldsymbol{x}(t) + \mathbf{M}_y^{\text{CT}}\boldsymbol{y}(t) \tag{11}$$

where $\mathbf{M}_x^{\mathrm{CT}} = (\mathbf{M}_x - \mathbf{I})/\Delta t$ and $\mathbf{M}_y^{\mathrm{CT}} = \mathbf{M}_y/\Delta t$ are the CT Kalman matrices and Δt is the discrete time step (50 ms).

From (11), by applying the dynamics' principle, we replaced integration with convolution by the synapse's spike response and the CT matrices with neurally plausible ones, which yielded:

$$\boldsymbol{x}(t) = h(t) * (\mathbf{A}'\boldsymbol{x}(t) + \mathbf{B}'\boldsymbol{y}(t)), \tag{12}$$

where $\mathbf{A}' = \tau\mathbf{M}_x^{\mathrm{CT}} + \mathbf{I} = \tau(\mathbf{M}_x - \mathbf{I})/\Delta t + \mathbf{I}$ and $\mathbf{B}' = \tau\mathbf{M}_y^{\mathrm{CT}} = \tau\mathbf{M}_y/\Delta t$.

The jth neuron's input current (see (6)) was computed from the system's current state, $\boldsymbol{x}(t)$, which was computed from estimates of the system's previous state ($\hat{\boldsymbol{x}}(t) = \sum_i a_i(t)\phi_i^{\boldsymbol{x}}$) and current measurements ($\hat{\boldsymbol{y}}(t) = \sum_k b_k(t)\phi_k^{\boldsymbol{y}}$) using (12). These quantities were decoded from the firing rates of the corresponding neural populations using the representation principle. This procedure yielded:

$$J_j(\boldsymbol{x}(t)) = \alpha_j\langle\tilde{\phi}_j^{\boldsymbol{x}} \cdot \boldsymbol{x}(t)\rangle + J_j^{\mathrm{bias}} \tag{13}$$

$$= \alpha_j\langle\tilde{\phi}_j^{\boldsymbol{x}} \cdot h(t) * \left(\mathbf{A}'\hat{\boldsymbol{x}}(t) + \mathbf{B}'\hat{\boldsymbol{y}}(t)\right)\rangle + J_j^{\mathrm{bias}} \tag{14}$$

$$= \alpha_j\left\langle\tilde{\phi}_j^{\boldsymbol{x}} \cdot h(t) * \left(\mathbf{A}'\sum_i a_i(t)\phi_i^{\boldsymbol{x}} + \mathbf{B}'\sum_k b_k(t)\phi_k^{\boldsymbol{y}}\right)\right\rangle + J_j^{\mathrm{bias}}. \tag{15}$$

This last equation can be written in a neural network form (figure 5):

$$J_j(\boldsymbol{x}(t)) = h(t) * \left(\sum_i \omega_{ji}a_i(t) + \sum_k \omega_{jk}b_k(t)\right) + J_j^{\mathrm{bias}} \tag{16}$$

where $\omega_{ji} = \alpha_j\langle\tilde{\phi}_j^{\boldsymbol{x}} \cdot \mathbf{A}'\phi_i^{\boldsymbol{x}}\rangle$ and $\omega_{jk} = \alpha_j\langle\tilde{\phi}_j^{\boldsymbol{x}} \cdot \mathbf{B}'\phi_k^{\boldsymbol{y}}\rangle$ are the recurrent and feedforward weights, respectively.

4.1. Efficient implementation

A software SNN implementation involves two distinct steps [39]: network creation and real-time execution. The network does not need to be created in real-time and therefore has no computational time constraints. However, executing the network has to be implemented efficiently for successful deployment in closed-loop experimental settings. To speed-up the simulation, we exploited the NEF mapping between the high-dimensional neural space and the low-dimensional control space. This mapping updated neuron interactions circuitously, using the decoding weights $\phi_j^{\boldsymbol{x}}$, dynamics matrix $\mathbf{A}'$, and preferred direction vectors $\tilde{\phi}_j^{\boldsymbol{x}}$ from (15), rather than directly using the recurrent and feedforward weights in (16). The circuitous approach yielded an almost 50-fold speedup [39], enabling real-time execution of a 2000-neuron network.

Other improvements to the basic SNN consisted of using two one-dimensional integrators instead of a single three-dimensional one, feeding the constant 1 into the two integrators continuously rather than obtaining it internally through integration, and connecting the 192 neural measurements directly to the recurrent pool of neurons, without using the $b_k(t)$ neurons as an intermediary (figure 5) [29].

Table 1. Model parameters.

Symbol	Range	Description
$\max G(J_j(\boldsymbol{x}))$	200–400 Hz	Maximum firing rate
$G(J_j(\boldsymbol{x})) = 0$	-1 to 1	Normalized x-axis intercept
J_j^{bias}	Satisfies first two	Bias current
α_j	Satisfies first two	Gain factor
$\tilde{\phi}_j^{\boldsymbol{x}}$	$\left\|\tilde{\phi}_j^{\boldsymbol{x}}\right\| = 1$	Preferred direction vector
σ^2	0.1	Gaussian noise variance
τ_j^{RC}	20 ms	RC time constant
τ_j^{ref}	1 ms	Refractory period
τ_j^{PSC}	20 ms	PSC time constant

4.2. Choice of parameters

Various parameters needed to be set for this network. They are recapitulated in table 1. Neural spike rates were computed in 50 ms time-bins and, therefore, the sum $\tau_j^{\mathrm{RC}} + \tau_j^{\mathrm{ref}} + \tau_j^{\mathrm{PSC}}$ had to be smaller than 50 ms, which was indeed the case. It was important for heterogeneity to be included [42, 43]. Therefore, neural parameters were randomly selected from a wide distribution. Specifically, the preferred direction vectors, $\tilde{\phi}_j^{\boldsymbol{x}}$, were randomly assigned to -1 and 1. The maximum firing rate, $\max G(J_j(\mathbf{x}))$, and the normalized x-axis intercept, $G(J_j(\mathbf{x})) = 0$, were drawn from a uniform distribution on [200, 400] Hz and $[-1, 1]$, respectively. The gain and bias current, α_j and J_j^{bias}, were chosen to satisfy these constraints. τ_j^{RC} was chosen at 20 ms, close to typical membrane time constant values, and τ_j^{ref} was set to 1 ms, again a typical value [44] . The time constant for the synapses' spike response dynamics, τ_j^{PSC}, was set to 20 ms, dynamics consistent with post-synaptic currents [44].

Noise was not explicitly added. It arose naturally from the fluctuations produced by representing a scalar through the filtering of spike trains, which has been shown to have effects similar to Gaussian noise [42]. For the purpose of computing the linear decoding weights (i.e. Γ), we modeled the resulting noise as Gaussian with a variance of 0.1 [42].

5. Off-line open-loop implementation

We first performed an off-line open-loop validation of our SNN decoder [29] by using a previously recorded BMI experiment that utilized a standard Kalman filter (SKF) with floating point computations. An adult male rhesus macaque (monkey L) was trained to perform a point-to-point arm movement task in a 3D experimental apparatus for a juice reward [13]. All animal procedures and experiments were approved by the Stanford Institutional Animal Care and Use Committee. A 96-electrode silicon array (Blackrock Microsystems) was implanted in the dorsal pre-motor (PMd) and motor (M1) cortex areas responsible for hand movement as guided by visual cortical landmarks. Array recordings ($-4.5\times$ RMS threshold crossing applied to each channel) yielded tuned activity for the direction of arm movements. For both monkeys (monkey L and J) the electrode array used in these experiments spanned approximately 4–6 mm of anterior–posterior distance on the pre-central gyrus associated with primary motor cortex

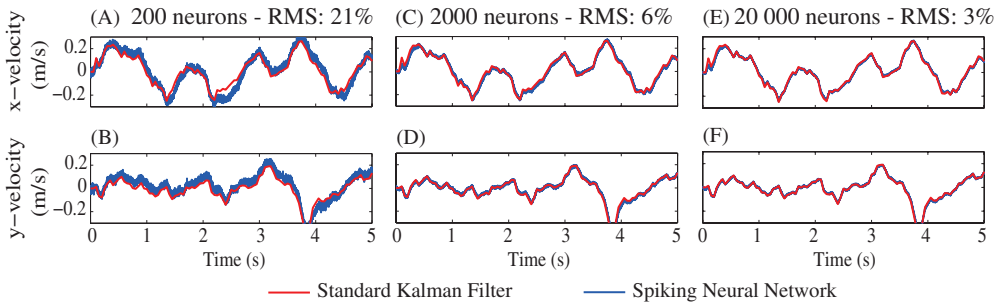

Figure 6. Comparing the *x* and *y*-velocity estimates decoded from 96 recorded cortical spike trains (5 s of data) by the SKF (red) and the SNN (blue). Networks with ((A) and (B)) 200, ((C) and (D)) 2000, and ((E) and (F)) 20 000 spiking neurons. RMS error normalized by maximum velocity is indicated above each plot.

(M1). Electrical stimulation and/or manual arm palpation further localized the area to the upper shoulder region/muscles (monkey J) and forearm (monkey L). It should be kept in mind that the border between M1 and the PMd cortex is not sharp neurophysiologically, and it is possible that the anterior aspect of either array could be within PMd. In addition, for monkey J, we also recorded simultaneously from a second array, which is the same as the first array except that it was implanted 1–2 mm anterior to the first array, and is thus nominally in PMd (see [46], supplementary figure 5, for an intraoperative photo of the arrays). For the purposes of the current experiments the distinction between these two areas is not of primary importance since both areas have robust movement-related activity and modulation.

As detailed in [13], a Kalman filter was fit by correlating the observed hand kinematics with the simultaneously measured neural signals (figure 3). The resulting model was used online to control an on-screen cursor in real time. This model and 500 of these trials (L20100308) served as the standard against which the SNN implementation's performance is compared. Starting with the steady-state Kalman-filter matrices derived from this experiment, we built an SNN using the NEF methodology and simulated it in *Nengo*, a freely available software, using the parameter values listed in table 1.

To highlight solely the computational differences between traditional floating point calculations and an SNN, the steady-state Kalman filter was run in both the SKF and SNN experimental blocks. This approach avoided any discrepancies in convergence of the Kalman filter that may have arisen from mathematical approximations by the SNN, leaving the deviation from floating point across time steps as the only source of computational variability. It also greatly simplifies the computational demands in the SNN implementation, enabling more efficient computations per neuron in the network. This approach required precomputing the steady-state Kalman-filter matrices, $\mathbf{M}_x$ and $\mathbf{M}_y$ shown in equation (3), and was prepared at the time of model fitting after regressing the four Kalman-filter matrices of equations (1) and (2).

The SNN performed better as we increased the number of neurons (figure 6). For 20 000 neurons, the *x* and *y*-velocity decoded from its two 10 000-neuron populations matched the standard decoder's prediction to within 3% (RMS error normalized by maximum velocity) [29]. There was a tradeoff

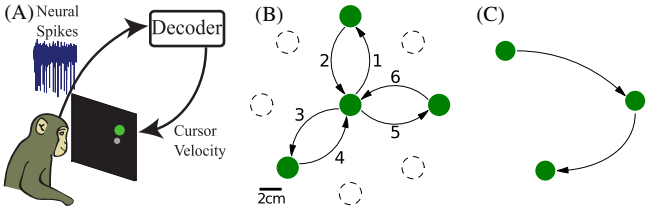

Figure 7. Experimental setup and tasks. (A) Data are recorded from silicon electrode arrays implanted in motor regions of cortex of monkeys performing a center-out-and-back (to one of eight targets) or pinball task for juice rewards with a 500 ms hold time. (B) Center-out and back task. (C) Pinball task.

between accuracy and network size. If the network size was decreased to 2000 neurons, the network's error increased to 6%; an even bigger decrease to 200 neurons led to an error of 21% [29]. Even with a 6% RMS error at 2000 neurons, we believed that this may provide sufficient accuracy to use in closed-loop experiments.

6. Online closed-loop performance

These off-line results encouraged us to test our SNN decoder in an online closed-loop setting. Despite the error, we suspected that the monkeys would actively compensate for any noticeable cursor deviation. Two adult male rhesus macaques (monkey L and J) were trained to perform a point-to-point arm movement task in a 3D experimental apparatus for a juice reward using the ReFIT-KF training protocol as detailed in the methods section of [13] (figure 7(A)). Unlike monkey L who only had one array, monkey J had two 96-electrode silicon arrays (Blackrock Microsystems) implanted, one in PMd and the other in M1. The Kalman filter was built as described in prior work [13]. The resulting models were used in a closed-loop system to control an on-screen cursor in real-time (figure 7(A), decoder block) and once again served as the base performance against which the SNN's performance was compared.

A 2000-neuron SNN decoder was built using the simulation algorithm enhancements mentioned earlier and simulated on an xPC Target (Mathworks) x86 system (Dell T3400, Intel Core 2 Duo E8600, 3.33 GHz). It ran in closed-loop, replacing the SKF as the decoder block in figure 7(A). Real-time execution constraints with our hardware limited the network size to no more than 2000 neurons.

6.1. Center-out and back task

Once the Kalman filter was trained and the SNN was built, we tested the two Kalman filter implementations (SKF and SNN) against each other. Each test was composed of 200 trials of target acquisition on a center-out-and-back task. The target alternated from the center of the workspace to one of eight peripheral locations chosen at random (see figure 7(B)). A successful trial is one in which the monkey navigates the cursor to the target and holds within the 4 cm square acquisition region for 500 ms during the allotted 3 s. Once a block of 200 trials was completed with one implementation, the decoder was switched to the other implementation and another block was collected. This ABA block switching was continued until the monkey was satiated and enabled an accurate comparison of just the relative difference in implementations. This ABA block style experimentation was repeated for at least three days with each monkey.

Success rates were higher than 94% on all blocks for the SNN implementation (94.9% and 99.6% for monkey L and J, respectively) and 98% for the SKF (98.0% and 99.7% for monkey L and J, respectively). Thousands of trials were collected and analyzed (5235 with monkey L and 5187 with monkey J). These reflect only center-out trials, not those that returned to the center from the periphery. The latter were not included in the analysis because the monkey anticipated the return to the center after navigating out to the periphery and thus initiated movement earlier than when the target location is unknown. About half of the trials, 2615 (2484), were performed with the SNN for monkey L (monkey J) and 2518 (2599) with the SKF (see figure 8 for BMI position kinematics examples). The average time to acquire the target was moderately slower for the SNN for both monkeys—1067 ms versus 830 ms for monkey L and 871 ms versus 708 ms for monkey J. Around 100 trials under hand control were used as a baseline comparison for each monkey.

Although the speed of both implementations was comparable, as evidenced by the traces being nearly on top of each other up until the first acquire time, the SNN has more difficulty settling into the target (figure 9). Whereas the time at which the target was first successfully acquired (average indicated by trace becoming thicker) is comparable for both BMI implementations, the time at which the target was successfully last acquired (average indicated by trace's cutoff) occurs latter for SNN. That is, the monkey spent more

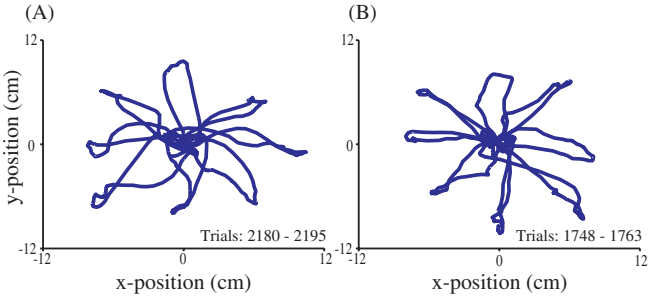

Figure 8. Center-out and back task (monkey J, 16 April 2011). (A) BMI position kinematics of 16 continuous trials for the SKF implementation. (B) BMI position kinematics of 16 continuous trials for the SNN implementation.

time wandering in and out of the acquisition region, before subsequently successfully staying inside it for the required hold time. This longer *dial-in time* (indicated by length of thick trace) suggests that the SNN provides less precise control when attempting to stop.

Towards the end of the day's experiments, the SNN decoder would occasionally fall to edge of the workspace and the monkey would lose interest in the task. The cursor was reset to the center of the screen so the monkey would continue the block. This happened infrequently and counted against the monkey's success rate on center-out trials. The off-line results shown earlier (see figure 6) suggest that this difference in usability, as well as the difference in performance, is a result of the network's neuron count, which was limited by the real-time execution capacity of the x86 hardware. These performance issues could be improved by using more neurons, as would be the case in a neuromorphic chip. Nevertheless, even with only 2000 neurons, the success rate and acquire times of the SNN decoder are comparable to that of the SKF decoder.

6.2. Pinball task

Another important measure when evaluating decoders is generalization and stability. To test this, we instructed a new pinball task under the SNN decoder. In this task, targets appeared randomly in a 16 cm square workspace without any systematic structure to target placement (see figure 7(C)). The monkey received a reward by navigating

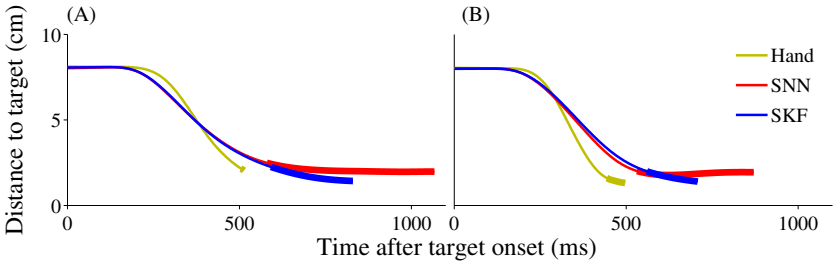

Figure 9. SNN (red) performance compared to SKF (blue) (hand trials are shown for reference (yellow)). The SNN achieves similar results as the SKF implementation. Plot of distance to target versus time after target onset for different control modalities. The thicker traces represent the average time when the cursor first enters the acceptance window until successfully beginning the 500 ms hold time. Results for monkey L (A) and results for monkey J (B).

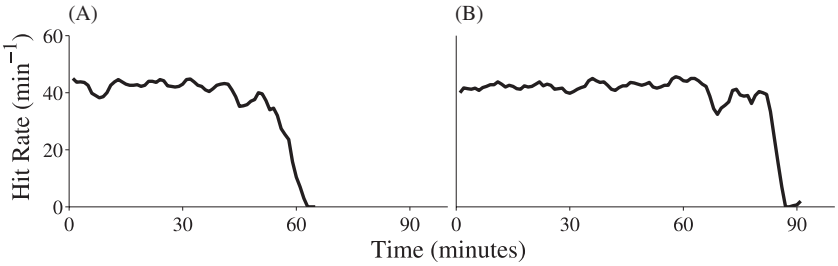

Figure 10. Sustained performance plots for both monkeys using the SNN decoder under the pinball task. The sharp falloff represents a loss of interest in the task, which occurred when the monkey was satiated. Results for monkey L (A) and for monkey J (B).

to the target and holding within the 4 cm acquisition region for 500 ms. Successfully performing this task highlights the SNN decoder's generalization. There was no block structure in this set of experiments. The monkeys were started on this task and were allowed to run continuously without interruption until satiated. Successfully performing this task for long durations highlights the SNN decoder's stability over an extended period of time.

Both monkeys sustained performance at around 40 targets per minute for over an hour on the pinball task (figure 10). On the day tested, monkey L lost interest in the task after approximately 61 min and monkey J after approximately 85 min. The sustained performance of the SNN decoder across both monkeys in a more generalized task demonstrates the robust performance of the system over uninterrupted periods, suggesting that the decoder is capable of sustained performance across long stretches of time. This performance is comparable to the hit rate and session duration achieved under hand control [38].

7. Conclusions and future work

The SNN decoder's performance was comparable to that produced by a SKF. The 2000-neuron network had success rates higher than 94% on all blocks but took moderately longer to hold the cursor still over targets. Performance was sustainable for at least an hour under a generalized acquisition task. As the Kalman filter and its variants are the state-of-the-art in cortically-controlled motor prostheses [11–14], these simulations increase confidence that similar levels of performance can be attained with a neuromorphic chip, which can potentially overcome the power constraints set by clinical applications. A neuromorphic chip could implement a 10 000-neuron network while dissipating a fraction of a milliwatt, likely increasing the performance of the system compared to the simulated SNN shown here.

This demonstration is an important proof-of-concept that highlights the feasibility of mapping existing control theory algorithms onto SNNs for BMI applications. For BMIs to gain widespread clinical deployment, they must packaged in low-power, completely implantable, wireless systems. The computational demands of BMI decoding are high, and thus present a problem for low-power applications. Implementing the SNN decoder presented here onto neuromorphic chips may be a possible answer, performing complex and demanding computations at a fraction of the power draw of conventional processors. Translating the SNN from software to ultra-low-power neuromorphic chips is the next step in the development of a fully-implantable neuromorphic chip for cortical prosthetic applications. Currently, we are exploring this mapping with Neurogrid, a hardware platform with 16 programmable neuromorphic chips that can simulate up to a million spiking neurons in real-time [35].

Acknowledgments

The authors would like to thank the support of Joline M Fan and Jonathan C Kao for assistance in collecting monkey data; Mackenzie Mazariegos, John Aguayo, Clare Sherman, and Erica Morgan for veterinary care; Hua Gao for help for figure 2; Kimberly Chin, Sandra Eisensee, Evelyn Castaneda, and Beverly Davis for administrative support. We also thank Chris Eliasmith and Terry Stewart for valuable help with Nengo.

This work was supported in part by the BAEF and FRS-FNRS (JD), Stanford NIH Medical Scientist Training Program (MSTP) and Soros Fellowship (PN), DARPA Revolutionizing Prosthetics program (N66001-06-C-8005, KVS), two NIH Director's Pioneer Awards (DP1-OD006409, KVS; DPI-OD000965, KB), and an NIH/NINDS Transformative Research Award (R01NS076460, KB).

References

[1] Taylor D M, Tillery S I and Schwartz A B 2002 Direct cortical control of 3D neuroprosthetic devices *Science* **296** 1829–32

[2] Carmena J M, Lebedev M A, Crist R E, O'Doherty J E, Santucci D M, Dimitrov D F, Patil P G, Henriquez C S and Nicolelis M A 2003 Learning to control a brain–machine interface for reaching and grasping by primates *PLoS Biol.* **1** 193–208

[3] Musallam S, Corneil B D, Greger B, Scherberger H and Andersen R A 2004 Cognitive control signals for neural prosthetics *Science* **305** 258–62

[4] Hochberg L R, Serruya M D, Friehs G M, Mukand J A, Saleh M, Caplan A H, Branner A, Chen D, Penn R D and Donoghue J P 2006 Neuronal ensemble control of prosthetic devices by a human with tetraplegia *Nature* **442** 164–71

[5] Santhanam G, Ryu S I, Yu B M, Afshar A and Shenoy K V 2006 A high-performance brain-computer interface *Nature* **442** 195–8

[6] Velliste M, Perel S, Spalding M C, Whitford A S and Schwartz A B 2008 Cortical control of a prosthetic arm for self-feeding *Nature* **453** 1098–101

[7] Ganguly K, Dimitrov D F, Wallis J D and Carmena J M 2011 Reversible large-scale modification of cortical networks during neuroprosthetic control *Nature Neurosci.* **14** 662–7

[8] O'Doherty J E, Lebedev M A, Ifft P J, Zhuang K Z, Shokur S, Bleuler H and Nicolelis M A 2011 Active tactile exploration using a brain–machine–brain interface *Nature* **479** 228–31

[9] Ethier C, Oby E R, Bauman M J and Miller L E 2012 Restoration of grasp following paralysis through brain-controlled stimulation of muscles *Nature* **485** 368–71

[10] Hochberg L R *et al* 2012 Reach and grasp by people with tetraplegia using a neurally controlled robotic arm *Nature* **485** 372–5

[11] Nuyujukian P, Gilja V, Chestek C A, Cunningham J P, Fan J M, Yu B M, Ryu S I and Shenoy K V 2010 Generalization and robustness of a continuous cortically-controlled prosthesis enabled by feedback control design *Meeting of Society for Neuroscience 2010 (San Diego, CA)* Program No. 20.7

[12] Gilja V, Chestek C A, Diester I, Henderson J M, Deisseroth K and Shenoy K V 2011 Challenges and opportunities for next-generation intra-cortically based neural prostheses *IEEE Trans. Biomed. Eng.* **58** 1891–9

[13] Gilja V *et al* 2012 A high performance neural prosthesis enabled by control algorithm design *Nature Neurosci.* **15** 1752–7

[14] Kim S P, Simeral J D, Hochberg L R, Donoghue J P and Black M J 2008 Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia *J. Neural Eng.* **5** 455–76

[15] Li Z, O'Doherty J E, Hanson T L, Lebedev M A, Henriquez C S and Nicolelis M A L 2009 Unscented Kalman filter for brain–machine interfaces *PLoS One* **4** e6243

[16] Chase S M, Kass R E and Schwartz A B 2012 Behavioral and neural correlates of visuomotor adaptation observed through a brain-computer interface in primary motor cortex *J. Neurophysiol.* **108** 624–44

[17] Jackson A, Mavoori J and Fetz E E 2006 Long-term motor cortex plasticity induced by an electronic neural implant *Nature* **444** 56–60

[18] Zanos S, Richardson A G, Shupe L, Miles F P and Fetz E E 2011 The Neurochip-2. An autonomous head-fixed computer for recording and stimulating in freely behaving monkeys *IEEE Trans. Neural Syst. Rehabil. Eng.* **19** 427–35

[19] Santhanam G, Linderman M D, Gilja V, Afshar A, Ryu S I, Meng T H and Shenoy K V 2007 HermesB: a continuous neural recording system for freely behaving primates *IEEE Trans. Biomed. Eng.* **54** 2037–50

[20] Chestek C A, Gilja V, Nuyujukian P, Kier R J, Solzbacher F, Ryu S I, Harrison R R and Shenoy K V 2009 HermesC: low-power wireless neural recording system for freely moving primates *IEEE Trans. Neural Syst. Rehabil. Eng.* **17** 330–8

[21] Hua G, Walker R M, Nuyujukian P, Makinwa K A A, Shenoy K V, Murmann B and Meng T H 2012 HermesE: a 96-channel full data rate direct neural interface in 0.13 μ m CMOS *IEEE J. Solid-State Circuits* **47** 1043–55

[22] Miranda H, Gilja V, Chestek C A, Shenoy K V and Meng T H 2010 HermesD: a high-rate long-range wireless transmission system for simultaneous multichannel neural recording applications *IEEE Trans. Biomed. Circuits Syst.* **4** 181–91

[23] Gilja V, Chestek C A, Nuyujukian P, Foster J D and Shenoy K V 2010 Autonomous head-mounted electrophysiology systems for freely-behaving primates *Curr. Opin. Neurobiol.* **20** 676–86

[24] Borton D A, Yin M, Aceros J and Nurmikko A 2013 An implantable wireless neural interface for recording cortical circuit dynamics in moving primates *J. Neural Eng.* **10** 026010

[25] Harrison R R, Watkins P T, Kier R J, Lovejoy R O, Black D J, Greger B and Solzbacher F 2007 A low-power integrated circuit for a wireless 100-electrode neural recording system *IEEE J. Solid-State Circuits* **42** 123–33

[26] Harrison R R 2008 The design of integrated circuits to observe brain activity *Proc. IEEE* **96** 1203–16

[27] Wolf P D 2008 Thermal considerations for the design of an implanted cortical brain–machine interface (BMI) *Indwelling Neural Implants: Strategies for Contending with the In Vivo Environment* ed W M Reichert (Boca Raton, FL: CRC Press) chapter 3

[28] Kim S, Tathireddy P, Normann R A and Solzbacher F 2007 Thermal impact of an active 3-D microelectrode array implanted in the brain *IEEE Trans. Neural Syst. Rehabil. Eng.* **15** 493–501

[29] Dethier J, Gilja V, Nuyujukian P, Elassaad S A, Shenoy K V and Boahen K 2011 Spiking neural network decoder for brain–machine interfaces *Proc. 5th Int. IEEE EMBS Conf. on Neural Engineering (Cancun, Mexico)* pp 396–9

[30] Miranda H and Meng T H 2010 A programmable pulse UWB transmitter with 34% energy efficiency for multichannel neurorecording systems *CICC: Custom Integrated Circuits Conf.* pp 1–4

[31] Laubach M, Wessberg J and Nicolelis M A L 2000 Cortical ensemble activity increasingly predicts behaviour outcomes during learning of a motor task *Nature* **405** 567–71

[32] Zumsteg Z S, Kemere C, O'Driscoll S, Santhanam G, Ahmed R E, Shenoy K V and Meng T H 2005 Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems *IEEE Trans. Neural Syst. Rehabil. Eng.* **13** 272–9

[33] Harrison R R and Charles C 2003 A low-power low-noise CMOS amplifier for neural recording applications *IEEE J. Solid-State Circuits* **38** 958–65

[34] Boahen K 2005 Neuromorphic microchips *Sci. Am.* **292** 56–63

[35] Silver R, Boahen K, Grillner S, Kopell N and Olsen K L 2007 Neurotech for neuroscience: unifying concepts, organizing principles, and emerging tools *J. Neurosci.* **27** 11807–19

[36] Arthur J V and Boahen K 2011 Silicon neuron design: the dynamical systems approach *IEEE Trans. Circuits Syst.* I **58** 1034–43

[37] Fang H, Wang Y and He J 2010 Spiking neural networks for cortical neuronal spike train decoding *Neural Comput.* **22** 1060–85

[38] Sussillo D, Nuyujukian P, Fan J M, Kao J C, Stavisky S D, Ryu S I and Shenoy K V 2012 A recurrent neural network for closed-loop intracortical brain–machine interface decoders *J. Neural Eng.* **9** 026027

[39] Dethier J, Nuyujukian P, Eliasmith C, Stewart T, Elassaad S A, Shenoy K V and Boahen K 2011 A brain–machine interface operating with a real-time spiking neural network control algorithm *Advances in Neural Information Processing Systems* vol 24, ed J Shawe-Taylor *et al* (Cambridge: MIT Press) pp 2213–21

[40] Kalman R E 1960 A new approach to linear filtering and prediction problems *Trans. ASME* D **82** 35–45

[41] Malik W Q, Truccolo W, Brown E N and Hochberg L R 2011 Efficient decoding with steady-state Kalman filter in neural interface systems *IEEE Trans. Neural Syst. Rehabil. Eng.* **19** 25–34

[42] Eliasmith C and Anderson C H 2003 *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems* (Cambridge: MIT Press)

[43] Singh R and Eliasmith C 2006 Higher-dimensional neurons explain the tuning and dynamics of working memory cells *J. Neurosci.* **26** 3667–78
[44] Eliasmith C 2005 A unified approach to building and controlling spiking attractor networks *Neural Comput.* **17** 1276–314
[45] Eliasmith C 2007 How to build a brain: from function to implementation *Synthese* **159** 373–88
[46] Churchland M M, Cunningham J P, Kaufman M T, Foster J D, Nuyujukian P, Ryu S I and Shenoy K V 2012 Neural population dynamics during reaching *Nature* **487** 51–6