

Research Article

A Tensor-Product-Kernel Framework for Multiscale Neural Activity Decoding and Control

Lin Li,¹ Austin J. Brockmeier,² John S. Choi,³ Joseph T. Francis,⁴
Justin C. Sanchez,⁵ and José C. Principe²

¹ Philips Research North America, Briarcliff Manor, NY 10510, USA

² Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA

³ Joint Program in Biomedical Engineering, NYU Polytechnic School of Engineering and SUNY Downstate, Brooklyn, NY 11203, USA

⁴ Department of Physiology and Pharmacology, State University of New York Downstate Medical Center, Joint Program in Biomedical Engineering, NYU Polytechnic School of Engineering and SUNY Downstate, Robert F. Furchgott Center for Neural & Behavioral Science, Brooklyn, NY 11203, USA

⁵ Department of Biomedical Engineering, Department of Neuroscience, Miami Project to Cure Paralysis, University of Miami, Coral Gables, FL 33146, USA

Correspondence should be addressed to Lin Li; lin-li@philips.com

Received 22 August 2013; Revised 28 January 2014; Accepted 11 February 2014; Published 14 April 2014

Academic Editor: Zhe (Sage) Chen

Copyright © 2014 Lin Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Brain machine interfaces (BMIs) have attracted intense attention as a promising technology for directly interfacing computers or prostheses with the brain's motor and sensory areas, thereby bypassing the body. The availability of multiscale neural recordings including spike trains and local field potentials (LFPs) brings potential opportunities to enhance computational modeling by enriching the characterization of the neural system state. However, heterogeneity on data type (spike timing versus continuous amplitude signals) and spatiotemporal scale complicates the model integration of multiscale neural activity. In this paper, we propose a tensor-product-kernel-based framework to integrate the multiscale activity and exploit the complementary information available in multiscale neural activity. This provides a common mathematical framework for incorporating signals from different domains. The approach is applied to the problem of neural decoding and control. For neural decoding, the framework is able to identify the nonlinear functional relationship between the multiscale neural responses and the stimuli using general purpose kernel adaptive filtering. In a sensory stimulation experiment, the tensor-product-kernel decoder outperforms decoders that use only a single neural data type. In addition, an adaptive inverse controller for delivering electrical microstimulation patterns that utilizes the tensor-product kernel achieves promising results in emulating the responses to natural stimulation.

1. Introduction

Brain machine interfaces (BMIs) provide new means to communicate with the brain by directly accessing, interpreting, and even controlling neural states. They have attracted attention as a promising technology to aid the disabled (i.e., spinal cord injury, movement disability, stroke, hearing loss, and blindness) [1–6]. When designing neural prosthetics and brain machine interfaces (BMIs), the fundamental steps involve quantifying the information contained in neural

activity, modeling the neural system, decoding the intention of movement or stimulation, and controlling the spatiotemporal neural activity pattern to emulate natural stimulation. Furthermore, the complexity and distributed dynamic nature of the neural system pose challenges for the modeling tasks.

The development of recording technology enables access to brain activity from multiple functional levels, including the activity of individual neurons (spike trains), local field potentials (LFPs), electrocorticogram (ECoG), and electroencephalogram (EEG), collectively forming a multiscale

characterization of brain state. Simultaneous recording of multiple types of signals could facilitate enhanced neural system modeling. Although there are underlying relationships among these brain activities, it is unknown how to leverage the heterogeneous set of signals to improve the identification of the neural-response-stimulus mappings. The challenge is in defining a framework that can incorporate these heterogeneous signal formats coming from multiple spatiotemporal scales. In our work, we mainly address integrating spike trains and LFPs for multiscale neural decoding.

Spike trains and LFPs encode complementary information of stimuli and behaviors [7, 8]. In most recordings, spike trains are obtained by detecting transient events on a signal that is conditioned using a high-pass filter with the cutoff frequency set at about 300–500 Hz, while LFPs are obtained by using a low-pass filter with 300 Hz cutoff frequency [9]. Spike trains represent single- or multiunit neural activity with a fine temporal resolution. However, their stochastic properties induce considerable variability, especially when the stimulation amplitude is small; that is, the same stimuli rarely elicit the same firing patterns in repeated trials. In addition, a functional unit of the brain contains thousands of neurons. Only the activity of a small subset of neurons can be recorded and, of these, only a subset may modulate with respect to the stimuli or condition of interest.

In contrast, LFPs reflect the average synaptic input to a region near the electrode [10], which limits specificity but provides robustness for characterizing the modulation induced by stimuli. Furthermore, LFPs naturally provide population-level measure of neural activity. Therefore, an appropriate aggregation of LFPs and spike trains enables enhanced accuracy and robustness of neural decoding models. For example, the decoder can coordinate LFPs or spike patterns to tag particularly salient events or extract different stimulus features characterized by multisource signals. However, heterogeneity between LFPs and spike trains complicates their integration into the same model. The information in a spike train is coded in a set of ordered spike timings [11, 12], while an LFP is a continuous amplitude time series. Moreover, the time scale of LFPs is significantly longer than spike trains. Whereas recent work has compared the decoding accuracy of LFPs and spikes [13], only a small number of simple models have been developed to relate both activities [14]. However, the complete relationship between LFPs and spike trains is still a subject of controversy [15–17], which hinders principled modeling approaches.

To address these modeling issues, this paper proposes a signal processing framework based on tensor-product kernels to enable decoding and even controlling multiscale neural activities. The tensor-product kernel uses multiple heterogeneous signals and implicitly defines a kernel space constructed by the tensor product of individual kernels designed for each signal type [18]. The tensor-product kernel uses the joint features of spike trains and LFPs. This enables kernel-based machine learning methodologies to leverage multiscale neural activity to uncover the mapping from the neural system states and the corresponding stimuli.

The kernel least mean square (KLMS) algorithm is used to estimate the dynamic nonlinear mapping from the two

types of neural responses to the stimuli. The KLMS algorithm exploits the fact that the linear signal processing in reproducing kernel Hilbert spaces (RKHS) corresponds to nonlinear processing in the input space and is used in the adaptive inverse control scheme [19] designed for controlling neural systems. Utilizing the tensor-product kernel, we naturally extend this scheme to multiscale neural activity. Since the nonlinear control is achieved via linear processing in the RKHS, it bypasses the local minimum issues normally encountered in nonlinear control.

The validation of the effectiveness of the proposed tensor-product-kernel framework is done in a somatosensory stimulation study. Somatosensory feedback remains underdeveloped in BMI, which is important for motor and sensory integration during movement execution, such as proprioceptive and tactile feedback about limb state during interaction with external objects [20, 21]. A number of early experiments have shown that spatiotemporally patterned microstimulation delivered to somatosensory cortex can be used to guide the direction of reaching movements [22–24]. In order to effectively apply the artificial sensory feedback in BMI, it is essential to find out how to use microstimulation to replicate the target spatiotemporal patterns in the somatosensory cortex, where neural decoding and control are the critical technologies to achieve this goal. In this paper, our framework is applied to leverage multiscale neural activities to decode both natural sensory stimulation and microstimulation. Its decoding accuracy is compared with decoders that use a single type of neural activity (LFPs or spike trains).

In the neural system control scenario, this tensor-product-kernel methodology can also improve the controller performance. Controlling the neural activity via stimulation has raised the prospect of generating specific neural activity patterns in downstream areas, even mimicking natural neural responses, which is central both for our basic understanding of neural information processing and for engineering “neural prosthetic” devices that can interact with the brain directly [25]. From a control theory perspective, the neural circuit is treated as the “plant,” where the applied microstimulation is the control signal and the plant output is the elicited neural response represented by spike trains and LFPs. Most conventional control schema cannot be directly applied to spike trains because there is no algebraic structure in the space of spike trains. Therefore, most existing neural control approaches have been applied to binned spike trains or LFPs [25–31]. Here, we will utilize the kernel-based adaptive inverse controller for spike trains proposed in our previous work [19] as an input-output (system identification) based control scheme. This methodology can directly be extended to the tensor-product kernel to leverage the availability of multiscale neural signals (e.g., spike trains and LFPs) and improves the robustness and accuracy of the stimulation optimization by exploiting the complementary information of the heterogeneous neural signals recorded from multiple sources.

The adaptive inverse control framework controls patterned electrical microstimulation in order to drive neural

responses to mimic the spatiotemporal neural activity patterns induced by tactile stimulation. This framework creates new opportunities to improve the ability to control neural states to emulate the natural stimuli by leveraging the complementary information from multiscale neural activities. This better interprets the neural system internal states and thus enhances the robustness and accuracy of the optimal microstimulation pattern estimation.

The rest of the paper is organized as follows. Section 2 introduces kernels for spike trains and LFPs and the tensor-product kernel that combines them. The kernel-based decoding model and the adaptive inverse control scheme that exploit kernel-based neural decoding technology to enable control in RKHS are introduced in Sections 3 and 4, respectively. Section 5 discusses the somatosensory stimulation emulation scenario and illustrates the test results by applying tensor-product kernel to leverage multiscale neural activity for decoding and controlling tasks. Section 6 concludes this paper.

2. Tensor-Product Kernel for Multiscale Heterogeneous Neural Activity

The mathematics of many signal processing and pattern recognition algorithms is based on evaluating the similarity of pairs of exemplars. For vectors or functions, the inner product defined on Hilbert spaces is a linear operator and a measure of similarity. However, not all data types exist in Hilbert spaces. Kernel functions are bivariate, symmetric functions that implicitly embed samples in a Hilbert space. Consequently, if a kernel on a data type can be defined, then algorithms defined in terms of inner products can be used. This has enabled various kernel algorithms [18, 32–34].

To begin, we define the general framework for the various kernel functions used here, keeping in mind that the input corresponds to assorted neural data types. Let the domain of a single neural response dimension, that is, a single LFP channel or one spiking unit, be denoted by \mathcal{X} and consider a kernel $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. If κ is positive definite, then there is an implicit mapping $\phi : \mathcal{X} \rightarrow \mathcal{H}$ that maps any two sample points, say $x, x' \in \mathcal{X}$, to corresponding elements in the Hilbert space $\phi(x), \phi(x') \in \mathcal{H}$ such that $\kappa(x, x') = \langle \phi(x), \phi(x') \rangle$ is the inner product of these elements in the Hilbert space. As an inner product, the kernel evaluation $\kappa(x, x')$ quantifies the similarity between x and x' .

A useful property is that this inner product induces a distance metric,

$$\begin{aligned} d(x, x') &= \|\phi(x) - \phi(x')\|_{\mathcal{H}} \\ &= \sqrt{\langle \phi(x) - \phi(x'), \phi(x) - \phi(x') \rangle} \\ &= \sqrt{\kappa(x, x) + \kappa(x', x') - 2\kappa(x, x')}. \end{aligned} \quad (1)$$

For normalized and shift-invariant kernels where, for all $x, \kappa(x, x) = 1$, the distance is inversely proportional to the kernel evaluation $d(x, x') = \sqrt{2 - 2\kappa(x, x')}$.

To utilize two or more dimensions of the neural response, a kernel that operates on the joint space is required. There are two basic approaches to construct multidimensional kernels from kernels defined on the individual variables: direct sum and tensor-product kernels. In terms of kernel evaluations, they consist of taking either the sum or the product of the individual kernel evaluations. In both cases, the resulting kernel is positive definite as long as the individual kernels are positive definite [18, 35].

Let \mathcal{X}_i denote the neural response domain of the i th dimension and consider a positive-definite kernel $\kappa_i : \mathcal{X}_i \times \mathcal{X}_i \rightarrow \mathbb{R}$ and corresponding mapping $\phi_i : \mathcal{X}_i \rightarrow \mathcal{H}_i$ for this dimension. Again, the similarity between a pair of samples x and x' on the i th dimension is $\kappa_i(x_{(i)}, x'_{(i)}) = \langle \phi_i(x_{(i)}), \phi_i(x'_{(i)}) \rangle$.

For the sum kernel, the joint similarity over a set of dimensions \mathcal{I} is

$$\kappa_{\Sigma}(\mathbf{x}, \mathbf{x}') = \sum_{i \in \mathcal{I}} \kappa_i(x_{(i)}, x'_{(i)}). \quad (2)$$

This measure of similarity is an average similarity across all dimensions. When the sum is over a large number of dimensions, the contributions of individual dimensions are diluted. This is useful for multiunit spike trains or multichannel LFPs when the individual dimensions are highly variable, which if used individually would result in a poor decoding performance on a single trial basis.

For the tensor-product kernel, the joint similarity over two dimensions i and j is computed by taking the product between the kernel evaluations $\kappa_{[i,j]}([x_{(i)}, x_{(j)}], [x'_{(i)}, x'_{(j)}]) = \kappa_i(x_{(i)}, x'_{(i)}) \cdot \kappa_j(x_{(j)}, x'_{(j)})$. The new kernel $\kappa_{[i,j]}$ corresponds to a mapping function that is the tensor product between the individual mapping functions $\phi_{[i,j]} = \phi_i \otimes \phi_j$ where $\phi_{[i,j]}(x_{(i,j)}) \in \mathcal{H}_{[i,j]}$. This is the tensor-product Hilbert space. The product can be taken over a set of dimensions \mathcal{I} and the result is a positive-definite kernel over the joint space: $\kappa_{\Pi}(\mathbf{x}, \mathbf{x}') = \prod_{i \in \mathcal{I}} \kappa_i(x_{(i)}, x'_{(i)})$.

The tensor-product kernel corresponds to a stricter measure of similarity than the sum kernel. Due to the product, if for one dimension $\kappa_i(x_{(i)}, x'_{(i)}) \approx 0$ then $\kappa_{\Pi}(\mathbf{x}, \mathbf{x}') \approx 0$. The tensor-product kernel requires the joint similarity; that is, for samples to be considered similar in the joint space they must be close in all the individual spaces. If even one dimension is dissimilar the product will appear dissimilar. If some of the dimensions are highly variable, then they will have a deleterious effect on the joint similarity measure. On the other hand, the tensor product is a more precise measure of similarity that will be used later to combine multiscale neural activity.

More generally, an explicit weight can be used to adjust the influence of the individual dimensions on the joint kernel. Any convex combinations of kernels are positive definite, and learning the weights of this combination is known as multiple kernel learning [36–38]. In certain cases of the constituent kernels, namely, that they are infinitely divisible [39], a weighted product kernel can also be applied [40]. However, the optimization of these weightings is not explored in the current work.

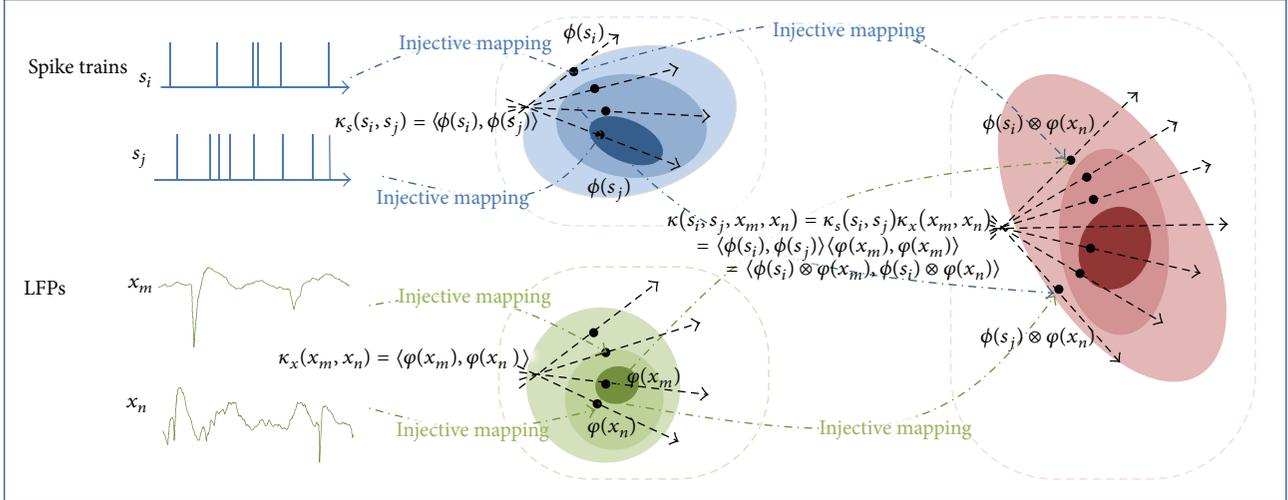


FIGURE 1: Schematic representation of the construction of the RKHS defined by the tensor-product kernel from the individual spike and LFP kernel spaces, along with the mapping from the original data. Specifically, s_i denotes a window of multiunit spike trains; x_n denotes a window of multichannel LFPs; $\kappa_s(\cdot, \cdot)$ denotes the spike train kernel with implicit mapping function $\phi(\cdot)$; and $\kappa_x(\cdot, \cdot)$ denotes the LFP kernel with implicit mapping function $\varphi(\cdot)$.

In general, a joint kernel space, constructed via either direct sum or tensor product, allows the homogeneous processing of heterogeneous signal types all within the framework of RKHSs. We use direct sum kernels to combine the different dimensions of multiunit spike trains or multichannel LFPs. For the spike trains, using the sum kernel across the different units enables an “average” population similarity over the space of spike trains where averages cannot be computed. Then a tensor-product kernel combines the two kernels: one for the multiunit spike trains and one for the multichannel LFPs; see Figure 1 for an illustration. The kernels for spike trains and LFPs can be selected and specified individually according to their specific properties.

In conclusion, composite kernels are very different from those commonly used in kernel-based machine learning, for example, for the support vector machine. In fact, here a pair of windowed spike trains and windowed LFPs is mapped into a feature function in the joint RKHS. Different spike train and LFP pairs are mapped to different locations in this RKHS, as shown in Figure 1. Due to its versatility, Schoenberg kernels defined for both the spike timing space and LFPs are employed in this paper and discussed below.

2.1. Kernel for Spike Trains. Unlike conventional amplitude data, there is no natural algebraic structure in the space of spike trains. The binning process, which easily transforms the point processes into discrete amplitude time series, is widely used in spike train analysis and allows the application of conventional amplitude-based kernels to spike trains [41] at the expense of losing the temporal resolution of the neural responses. This means that any temporal information in spikes within and between bins is disregarded, which is especially alarming when spike timing precision can be in the millisecond range. Although the bin size can be set small enough to preserve the fine time resolution, it will sparsify

the signal representation, increase the artifact variability, and cause high-dimensionality in the model, which requires voluminous data for proper training.

According to the literature, it is appropriate to consider a spike train to be a realization of a point process, which describes the temporal distribution of the spikes. Generally speaking, a point process p_i can be completely characterized by its conditional intensity function $\lambda(t | H_t^i)$, where $t \in \tau = [0, T]$ denotes the time coordinate and H_t^i is the history of the process up to t . A recent research area [42, 43] is to define an injective mapping from spike trains to RKHS based on the kernel between the conditional intensity functions of two point processes [42]. Among the cross-intensity (CI) kernels, the Schoenberg kernel is defined as

$$\begin{aligned} & \kappa(\lambda(t | H_t^i), \lambda(t | H_t^j)) \\ &= \exp\left(-\frac{\|\lambda(t | H_t^i) - \lambda(t | H_t^j)\|^2}{\sigma^2}\right) \\ &= \exp\left(-\frac{\int_{\tau} (\lambda(t | H_t^i) - \lambda(t | H_t^j))^2 dt}{\sigma^2}\right), \end{aligned} \quad (3)$$

where σ is the kernel size. The Schoenberg kernel is selected in this work because of its modeling accuracy and robustness to free parameter settings [44]. The Schoenberg kernel is a Gaussian-like kernel defined on intensity functions that is strictly positive definite and sensitive to the nonlinear coupling of two intensity functions [42]. Different spike trains will then be mapped to different locations in the RKHS. Compared to kernels designed on binned spike trains (e.g., spikernel [41]), the main advantage of the Schoenberg kernel is that the precision in the spike event location is better preserved and the limitations of the sparseness and

high-dimensionality for model building are also avoided, resulting in enhanced robustness and reduced computational complexity, especially when the application requires fine time resolution [44].

In order to be applicable, the methodology must lead to a simple estimation of the quantities of interest (e.g., the kernel) from experimental data. A practical choice used in our work estimates the conditional intensity function using a kernel smoothing approach [42, 45], which allows estimating the intensity function from a single realization and nonparametrically and injectively maps a windowed spike train into a continuous function. The estimated intensity function is obtained by simply convolving $s(t)$ with the smoothing kernel $g(t)$, yielding

$$\hat{\lambda}(t) = \sum_{m=1}^M g(t - t_m), \quad \{t_m \in \mathcal{T} : m = 1, \dots, M\}, \quad (4)$$

where the smoothing function $g(t)$ integrates to 1. Here $\hat{\lambda}(t)$ can be interpreted as an estimation of the intensity function. The rectangular and exponential functions [42, 46] are both popular smoothing kernels, which guarantee injective mappings from the spike train to the estimated intensity function. In order to decrease the kernel computation complexity, the rectangular function $g(t) = (1/\mathcal{T})(U(t) - U(t - \mathcal{T}))$ ($\mathcal{T} \gg$ the interspike interval) is used in our work, where $U(t)$ is a Heaviside function. This rectangular function approximates the cumulative density function of spikes counts in the window T and compromises the locality of the spike trains; that is, the mapping places more emphasis on the early spikes than the later ones. However, our experiments show that this compromise only causes a minimal impact on the kernel-based regression performance.

Let $s_i^n(t)$ denote the spike train for the i th sample of the n th spiking unit. The multiunit spike kernel is taken as the unweighted sum over the kernels on the individual units

$$\kappa_s(\mathbf{s}_i(t), \mathbf{s}_j(t)) = \sum_n \kappa_s(s_i^n(t), s_j^n(t)). \quad (5)$$

2.2. Kernels for LFPs. In contrast with spike trains, LFPs exhibit less spatial and temporal selectivity [15]. In the time domain, LFP features can be obtained by sliding a window on the signal, which describes the temporal LFP structure. The length of the window is selected based on the duration of neural responses to certain stimuli; the extent of the duration can be assessed by its autocorrelation function, as we will discuss in Section 5.3.1. In the frequency domain, the spectral power and phase in different frequency bands are also known to be informative features for decoding, but here we concentrate only on the time-domain decompositions. In the time domain, we can simply treat single channel LFPs as a time series and apply the standard Schoenberg kernel to the sequence of signal samples in time. The Schoenberg

kernel, defined in continuous spaces, maps the correlation time structure of the LFP $x(t)$ into a function in RKHS,

$$\begin{aligned} \kappa_x(x_i(t), x_j(t)) &= \exp\left(-\frac{\|x_i(t) - x_j(t)\|^2}{\sigma_x^2}\right) \\ &= \exp\left(-\frac{\int_{\mathcal{T}_x} (x_i(t) - x_j(t))^2 dt}{\sigma_x^2}\right) \\ &= \exp\left(-\left(\int_{\mathcal{T}_x} x_i(t) x_i(t) \right. \right. \\ &\quad \left. \left. + x_j(t) x_j(t) - 2x_i(t) x_j(t) dt\right)\right) \\ &\quad \times (\sigma_x^2)^{-1}, \end{aligned} \quad (6)$$

where $\mathcal{T}_x = [0, T_x]$.

Let $x_i^n(t)$ denote the LFP waveform for the i th sample of the n th channel. Similar to the multiunit spike kernel, the multichannel LFP kernel is defined by the direct sum kernel

$$\kappa_x(\mathbf{x}_i(t), \mathbf{x}_j(t)) = \sum_n \kappa_x(x_i^n(t), x_j^n(t)). \quad (7)$$

2.3. Discrete Time Sampling. Assuming a sampling rate with period τ , let $\mathbf{x}_i = [x_i^1, x_{i+1}^1, \dots, x_{i-1+T/\tau}^1, x_i^2, \dots, x_{i-1+T/\tau}^N]$ denote the i th multichannel LFP vector obtained by sliding the T -length window with step τ . Let $\mathbf{s}_i = \{t_m - (i-1)\tau, t_m \in [(i-1)\tau, (i-1)\tau + T] : m = 1, \dots, M\}$ denote the corresponding i th window of the multiunit spike timing sequence. The time scale, both in terms of the window length and sampling rate, of the analysis for LFPs and spikes is very important and needs to be defined by the characteristics of each signal. The tensor-product kernel allows the time scales of the analysis for LFPs and spike trains to be specified individually; that is, the window length T and sampling rate for spike trains and LFPs could be different. The suitable time scale can be estimated through autocorrelation coefficients of the signal as will be explained below.

3. Adaptive Neural Decoding Model

For neural decoding applications, a regression model with multiscale neural activities as the input is built to reconstruct a stimulus. The appeal of kernel-based filters is the usage of the linear structure of RKHS to implement well-established linear adaptive algorithms and to obtain a nonlinear filter in the input space that leads to universal approximation capability without the problem of local minima. There are several candidate kernel-based regression methods [32], such as support vector regression (SVR) [33], kernel recursive least squares (KRLS), and kernel least mean square (KLMS) [34]. The KLMS algorithm is preferred here because it is an online methodology of low computation complexity.

Input: $\{x_n, d_n\}, n = 1, 2, \dots, N$
Initialization: initialize the weight vector $\Omega(1)$: codebook (set of centers) $\mathcal{E}(0) = \{\}$ and coefficient vector $a(0) = []$
Computation:
For $n = 1, 2, \dots, N$
 (1) compute the output

$$y_n = \langle \Omega(n), \phi(x_n) \rangle = \sum_{j=1}^{\text{size}(\mathcal{E}(n-1))} a_j(n-1) \kappa(x_n, \mathcal{E}_j(n-1))$$

 (2) compute the error, $e_n = d_n - y_n$
 (3) compute the minimum distance in RKHS between x_n and each $x_i \in \mathcal{E}(n-1)$,
 $d(x_n) = \min_j (2 - 2\kappa(x_n, \mathcal{E}_j(n-1)))$
 (4) if $d(x_n) \leq \varepsilon$, then keep the codebook unchanged: $\mathcal{E}(n) = \mathcal{E}(n-1)$, and update the coefficient of the center closest to x_n :
 $a_k(n) = a_k(n-1) + \eta e(n)$, where $k = \underset{j}{\text{argmin}} \sqrt{2 - 2\kappa(x_n, \mathcal{E}_j(n-1))}$
 (5) otherwise, store the new center: $\mathcal{E}(n) = \{\mathcal{E}(n-1), x_n\}$, $a(n) = [a(n-1), \eta e_n]$
 (6) $\Omega(n+1) = \sum_{j=1}^{\text{size}(\mathcal{E}(n))} a_j(n) \phi(\mathcal{E}_j(n))$
end

ALGORITHM 1: Quantized kernel least mean square (QKLMS) algorithm.

The quantized kernel least mean square (Q-KLMS) is selected in our work to decrease the filter growth. Algorithm 1 shows the pseudocode for the Q-KLMS algorithm with a simple online vector quantization (VQ) method, where the quantization is performed based on the distance between the new input and each existing center. In this work, this distance between a center and the input is defined by their distance in RKHS, which for a shift-invariant normalized kernel for all $x\kappa(x, x) = 1$ is $\|\phi(x_n) - \phi(x_i)\|_2^2 = 2 - 2\kappa(x_n, x_i)$. If the smallest distance is less than a prespecified quantization size ε , the new coefficient ηe_n adjusts the weight of the closest center; otherwise a new center is added. Compared to other techniques [47–50] that have been proposed to curb the growth of the networks, the simple online VQ method is not optimal but is very efficient. Since, in our work, the algorithm must be applied several times to the same data for convergence after the first iteration over the data, we choose $\varepsilon = 0$, which merges the repeated centers and enjoys the same performance as KLMS.

We use the Q-KLMS framework with the multiunit spike kernels, the multichannel LFP kernels, and the tensor-product kernel using the joint samples of LFPs and spike trains. This is quite unlike previous work in adaptive filtering that almost exclusively uses the Gaussian kernel with real-valued time series.

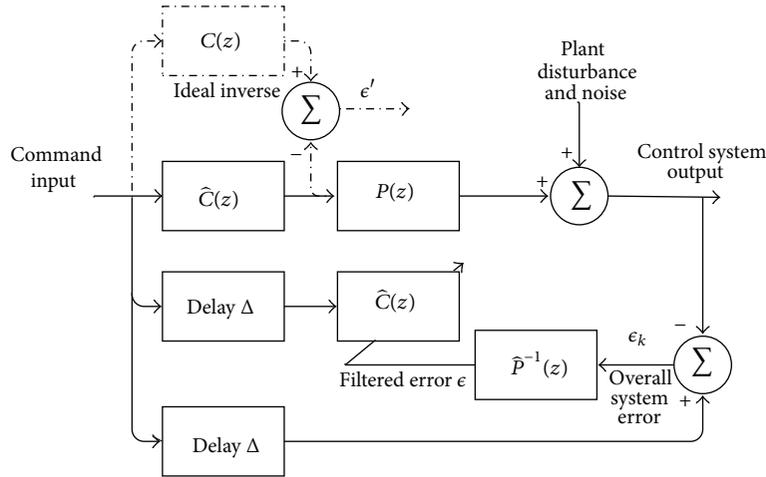
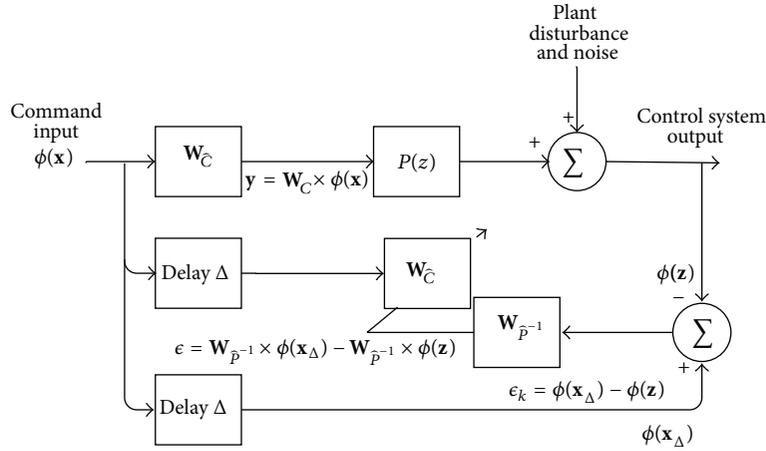
4. Adaptive Inverse Control of the Spatiotemporal Patterns of Neural Activity

As the name indicates, the basic idea of adaptive inverse control is to learn an inverse model of the plant as the controller in Figure 2(a), such that the cascade of the controller and the plant will perform like a unitary transfer function, that is, a perfect wire with some delay. The target plant output is used as the controller's command input. The controller parameters are updated to minimize the dissimilarity between the target

output and the plant's output during the control process, which enables the controller to track the plant variation and cancel system noises. The filtered- ε LMS adaptive inverse control diagram [51] shown in Figure 2(a) represents the filtered- ε approach to find $\widehat{C}(z)$. If the ideal inverse controller $C(z)$ is the actual inverse controller, the mean square of the overall system error ε_k would be minimized. The objective is to make $\widehat{C}(z)$ as close as possible to the ideal $C(z)$. The difference between the outputs of $C(z)$ and $\widehat{C}(z)$, both driven by the command input, is therefore an error signal ε' . Since the target stimulation is unknown, instead of ε' , a filtered error ε , obtained by filtering the overall system error ε_k through the inverse plant model $\widehat{P}^{-1}(z)$, is used for adaptation in place of ε' .

If the plant has a long response time, a modeling delay is advantageous to capture the early stages of the response, which is determined by the sliding window length that is used to obtain the inverse controller input. There is no performance penalty from the delay Δ as long as the input to $\widehat{C}(z)$ undergoes the same delay. The parameters of the inverse model $\widehat{P}^{-1}(z)$ are initially modeled offline and updated during the whole system operation, which allows $\widehat{P}^{-1}(z)$ to incrementally identify the inverse system and thus make ε approach ε' . Moreover, the adaptation enables $\widehat{P}^{-1}(z)$ to track changes in the plant. Thus, minimizing the filter error obtained from $\widehat{P}^{-1}(z)$ makes the controller follow the system variation.

4.1. Relation to Neural Decoding. In this control scheme, there are only two models, $\widehat{C}(z)$ and $\widehat{P}^{-1}(z)$, adjusted during the control process, which share the same input (neural activity) and output types (continuous stimulation waveforms). Therefore, both models perform like neural decoders and can be implemented using the Q-KLMS method we introduced in the previous section. Since all the mathematical models


 (a) A filtered- ϵ LMS adaptive inverse control diagram


(b) An adaptive inverse control diagram in RKHS

FIGURE 2: Adaptive inverse control diagram.

in this control scheme are kernel-based models, the whole control scheme can be mapped into an RKHS space, which holds several advantages as follows. (1) No restriction is imposed on the signal type. As long as a kernel can map the plant activity to RKHS, the plant can be controlled with this scheme. (2) Both the plant inverse, $\hat{P}^{-1}(z)$, and the controller $\hat{C}(z)$ have linear structure in RKHS, which avoids the danger of converging to local minima.

Specifically, the controller $\hat{C}(z)$ and the plant inverse $\hat{P}^{-1}(z)$ are separately modeled with the tensor-product kernel that we described in Section 2, and the model coefficients are updated with Q-KLMS. This structure is shown in Figure 2(b). The model coefficients $\mathbf{W}_{\hat{C}}$ and $\mathbf{W}_{\hat{P}^{-1}}$ represent the weight matrix of $\hat{C}(z)$ and $\hat{P}^{-1}(z)$ obtained by Q-KLMS, respectively. As this is a multiple-input multiple-output model, $\mathbf{W}_{\hat{C}}$ and $\mathbf{W}_{\hat{P}^{-1}}$ are the concatenation of the filter weights for each stimulation channel.

The variables \mathbf{x} , \mathbf{y} , and \mathbf{z} denote the concatenation of the windowed target spike trains and LFPs as the command input of the controller, the estimated stimulation, and the

plant output, respectively. \mathbf{x}_Δ is delayed target signal, which is aligned with the plant output \mathbf{z} . $\phi(\cdot)$ represents the mapping function from input space to the RKHS associated with the tensor-product kernel.

The overall system error is defined as $\epsilon_k = \phi(\mathbf{x}_\Delta) - \phi(\mathbf{z})$, which means that the controller's parameter adaptation seeks to minimize the distance in the RKHS between the target spike train/LFP and the output of the plant inverse $\hat{P}^{-1}(z)$. In this way, since the inverse model $\hat{P}^{-1}(z)$ has a linear structure in RKHS, the filtered error for stimulation channel $j \in 1, \dots, M$ is

$$\epsilon(j) = \mathbf{W}_j^{\hat{P}^{-1}} \phi(\mathbf{x}_\Delta) - \mathbf{W}_j^{\hat{P}^{-1}} \phi(\mathbf{z}). \quad (8)$$

The controller model $\hat{C}(z)$ has a single input \mathbf{x} , corresponding to the concatenation of the spike trains and LFPs, and has an M -channel output \mathbf{y} , corresponding to the microstimulation. Q-KLMS is used to model $\hat{C}(z)$ with N input samples. The target spike trains are repeated among different trials, which means that the repeated samples will

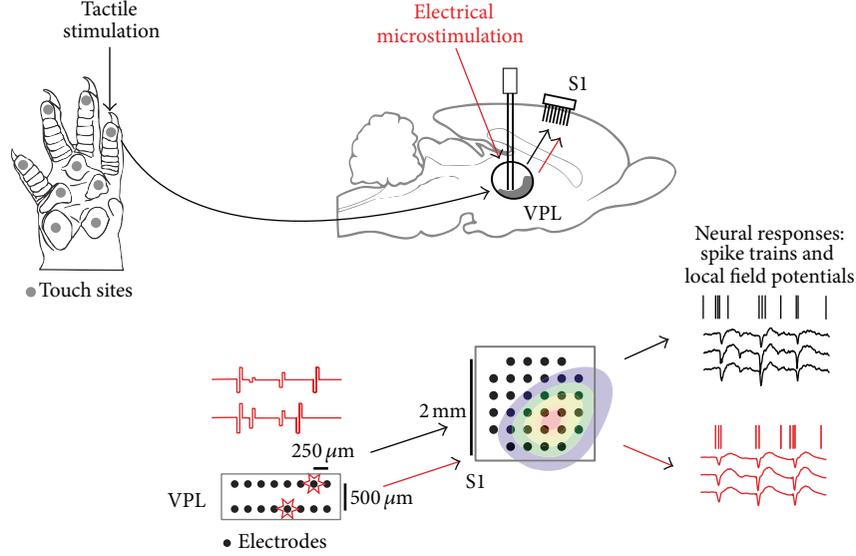


FIGURE 3: Neural elements in tactile stimulation experiments. To the left is the rat’s hand with representative cutaneous receptive fields. When the factor touches a particular “receptive field” on the hand, VPL thalamus receives this information and relays it to S1 cortex. To emulate “natural touch” with microstimulation, the optimized spatiotemporal microstimulus patterns are injected into the same receptive field on VPL thalamus through a microarray so that the target neural activity pattern can be replicated in somatosensory regions (S1) to convey the natural touch sensation to the animal.

be merged on the same kernel center of the first pass through the data by the quantization and thus the network size of the inverse controller is fixed (N centers). Only the coefficient matrix \mathbf{a} is updated with the filtered error ϵ during the whole control operation. The output of $\tilde{C}(z)$ can be calculated by

$$\begin{aligned}
 \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_M \end{bmatrix} &= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{12} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix} \begin{bmatrix} \phi(\tilde{\mathbf{c}}_1)' \\ \phi(\tilde{\mathbf{c}}_2)' \\ \vdots \\ \phi(\tilde{\mathbf{c}}_N)' \end{bmatrix} \phi(\mathbf{x}) \\
 &= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{12} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix} \begin{bmatrix} \kappa(\tilde{\mathbf{c}}_1, \mathbf{x}) \\ \kappa(\tilde{\mathbf{c}}_2, \mathbf{x}) \\ \vdots \\ \kappa(\tilde{\mathbf{c}}_N, \mathbf{x}) \end{bmatrix}, \quad (9)
 \end{aligned}$$

where $\tilde{\mathbf{c}}_n$ is the n th center and a_{mn} is the coefficient assigned to the n th kernel center for the m channel of the output.

5. Sensory Stimulation Experiment

5.1. Experimental Motivation and Framework. We applied these methods to the problem of converting touch information to electrical stimulation in neural prostheses. Somatosensory information originating in the peripheral nervous system ascends through the ventral posterior lateral (VPL) nucleus of the thalamus on its way to the primary

somatosensory cortex (S1). Since most cutaneous and proprioceptive information is relayed through this nucleus, we expect that a suitably designed electrode array could be used to selectively stimulate a local group of VPL neurons so as to convey similar information to cortex. Electrophysiological experiments [52] suggest that the rostral portion of the rat VPL nucleus carries a large amount of proprioceptive information, while the medial and caudal portions code mostly for cutaneous stimuli. Since the body maps for both VPL thalamus and S1 are known and fairly consistent, it is possible to implant electrode arrays in somatotopically overlapping areas of both regions.

We applied the proposed control method to generate multichannel electrical stimulation in VPL so as to evoke a naturalistic neural trajectory in S1. Figure 3 shows a schematic depiction of our experiment, which was conducted in rats. After implanting arrays in both VPL and S1, the responses to natural stimulation, delivered by a motorized factor, were recorded in S1. Then, we applied randomly patterned microstimulation in the VPL while recording the responses in S1. Using these responses, we then trained our controller to output the microstimulation patterns that would most accurately reproduce the neural responses to natural touch in S1. To solve this control problem, we first investigated how reliably the two types of stimulation, natural touch and electrical microstimulation, can be decoded.

5.2. Data Collection. All animal procedures were approved by the SUNY Downstate Medical Center IACUC and conformed to National Institutes of Health guidelines. A single female Long-Evans rat (Hilltop, Scottsdale, PA) was implanted with two microarrays while under anesthesia.

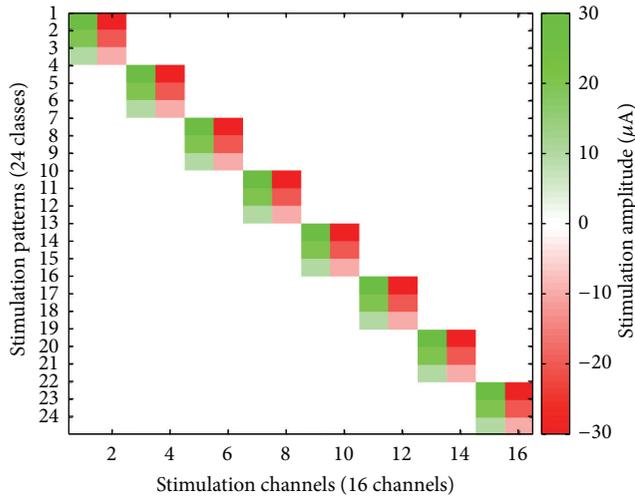


FIGURE 4: Bipolar microstimulation patterns applied in sensory stimulation experiment.

After induction using isoflurane, urethane was used to maintain anesthetic depth. The array in VPL was a 2×8 grid of 70% platinum 30% iridium $75 \mu\text{m}$ diameter microelectrodes (MicroProbes Inc.), with $500 \mu\text{m}$ between the rows and $250 \mu\text{m}$ interelectrode spacing within the rows. The microelectrodes had a 25:1 taper on the distal 5 mm with a tip diameter of $3 \mu\text{m}$. The approximate geometric surface area of the conducting tips was $1250 \mu\text{m}^2$. The shank lengths were custom designed to fit the contour of the rat VPL [52]. Both rows were identical and the shaft lengths for each row, from medial to lateral, were (8, 8, 8, 8, 8, 7.8, 7.6, 7.4) mm. The long axis of the VPL array was oriented along the rat's mediolateral axis.

The cortical electrode array (Blackrock Microsystems) was a 32-channel Utah array. The electrodes are arranged in a 6×6 grid excluding the 4 corners, and each electrode is 1.5 mm long. A single craniotomy that exposed the cortical insertions sites for both arrays was made, and, after several probing insertions with a single microelectrode (FHC) in an area 1 mm surrounding the stereotaxic coordinates for the digit region of S1 (4.0 mm lateral and 0.5 mm anterior to bregma) [53, 54], the Utah array was inserted using a pneumatic piston. The electrodes cover somatosensory areas of the S1 cortex and the VPL nucleus of the thalamus [52]. Neural recordings were made using a multichannel acquisition system (Tucker Davis).

Spike and field potential data were collected while the rat was maintained under anesthesia. The electrode voltages were preamplified with a gain of 1000, filtered with cutoffs at 0.7 Hz and 8.8 kHz, and digitized at 25 kHz. LFPs are further filtered from 1 to 300 Hz using a 3rd-order Butterworth filter. Spike sorting is achieved using k -means clustering of the first 2 principal components of the detected waveforms.

The experiment involves delivering microstimulation to VPL and tactile stimulation to the rat's fingers in separate sections. Microstimulation is administered on adjacent pairs

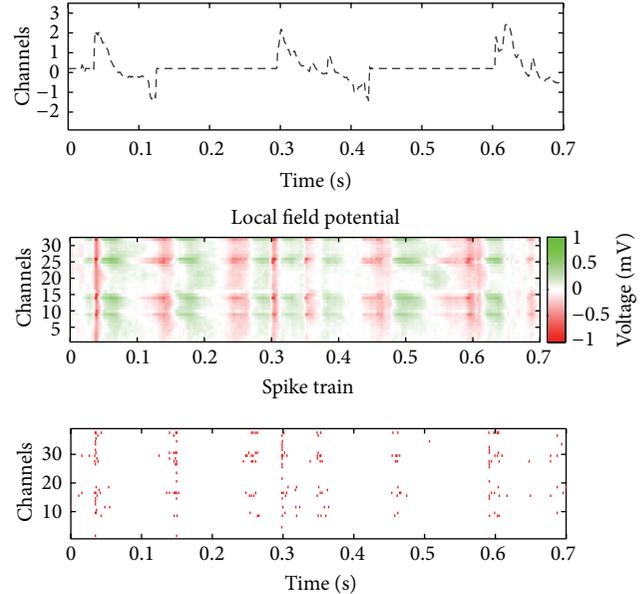


FIGURE 5: Rat neural response elicited by tactile stimulation. The upper plot shows the normalized derivative of tactile force. The remaining two plots show the corresponding LFPs and spike trains stimulated by tactile stimulation.

(bipolar configurations) of the thalamic array. The stimulation waveforms are single symmetric biphasic rectangular current pulses; each rectangular pulse is $200 \mu\text{s}$ long and has an amplitude of either $10 \mu\text{A}$, $20 \mu\text{A}$, or $30 \mu\text{A}$. Interstimulus intervals are exponentially distributed with mean interval of 100 ms. Stimulus isolation used a custom built switching headstage. The bipolar microstimulation pulses are delivered in the thalamus. There are 24 patterns of microstimulation: 8 different sites and 3 different amplitude levels for each site, as shown in Figure 4. Each pattern is delivered 125 times.

The experimental procedure also involves delivering 30–40 short 100 ms tactile touches to the rat's fingers (repeated for digit pads 1–4) using a hand-held probe. The rat remained anesthetized for the recording duration. The applied force is measured using a lever attached to the probe that pressed against a thin-film resistive force sensor (Trossen Robotics) when the probe tip contacted the rat's body. The resistive changes were converted to voltage using a bridge circuit and were filtered and digitized in the same way as described above. The digitized waveforms were filtered with a passband between 1 and 60 Hz using a 3rd-order Butterworth filter. The first derivative of this signal is used as the desired stimulation signal, which is shown in Figure 5.

5.3. Decoding Results. We now present the decoding results for the tactile stimulus waveform and microstimulation using Q-KLMS operating on the tensor-product kernel. The performance using the multiscale neural activity, both spike trains and LFPs, is compared with the decoder using single-type neural activity. This illustrates the effectiveness of the tensor-product-kernel-based framework to exploit the complementarity information from multiscale neural activities.

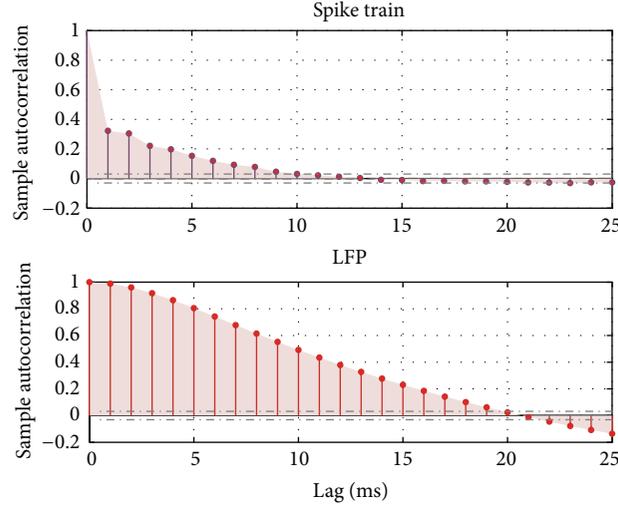


FIGURE 6: Autocorrelation of LFPs and spike trains for window size estimation.

5.3.1. Time Scale Estimation. The tensor-product kernel allows the time scales of the analysis for LFPs and spike trains to be specified individually, based on their own properties. In order to find reasonable time scales, we estimate the autocorrelation coefficients of LFPs and spike trains, which indicates the response duration induced by the stimulation. For this purpose, spike trains are binned with bin size of 1 ms. The LFPs are also resampled with sampling rate 1000 Hz. The autocorrelation coefficients of each signal average over channels are calculated by

$$\hat{\rho}_h = \frac{\sum_{t=h+1}^T (y_t - \bar{y})(y_{t-h} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}. \quad (10)$$

The 95% confidence bounds of the hypothesis that the autocorrelation coefficient is effectively zero are approximately estimated by $\pm 2SE\rho$, where

$$SE\rho = \sqrt{\frac{(1 + 2 \sum_{i=1}^{h-1} \rho_i^2)}{N}}. \quad (11)$$

The average confidence bounds for LFPs and spike trains are $[-0.032 \ 0.032]$ and $[-0.031 \ 0.031]$, respectively. The autocorrelation coefficients of LFPs fall into the confidence interval after 20 ms, while the autocorrelation coefficients of spike trains die out after 9 ms, as shown in Figure 6. Therefore, the decoder inputs are obtained by sliding the window with a size of $T_s = 9$ ms for spike trains and $T_x = 20$ ms for LFPs. In addition, the time discretization for the stimuli is 5 ms.

The learning rates for each decoder are determined by the best cross-validation results after scanning the parameters. The kernel sizes σ_s and σ_x are determined by the average distance in RKHS of each pair of training samples. The normalized mean square error (NMSE) between the estimated stimulus (\mathbf{y}) and the desired stimulus (\mathbf{d}) is utilized as an accuracy criterion.

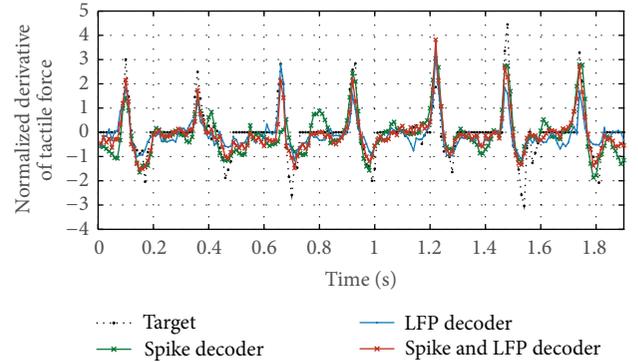


FIGURE 7: Qualitative comparison of decoding performance of the first tactile stimulation trial among LFP decoder, spike decoder, and spike and LFP decoder.

TABLE 1: Comparison among neural decoders.

Property	Input		
	LFP and spike	LFP	Spike
NMSE (mean/STD)	0.48/0.05	0.55/0.03	0.63/0.11

5.3.2. Results for Decoding Tactile Stimulation. NMSEs of tac-tile stimulation are obtained across 8 trial data sets. For each trial, we use 20 s data to train the decoders and compute an independent test error on the remaining 2.5 s data. The results are shown in Table 1, where we can observe that the LFP and spike decoder significantly outperformed both the LFP decoder and the spike decoder with P value < 0.05 .

In order to illustrate the details of the decoding performance, a portion of the test results of the first trial are shown in Figure 7. It is observed that the output of the spike decoder fluctuates and misses some pulses (e.g., around 0.65 s) due to the sparsity and variability of spike trains. In contrast, the

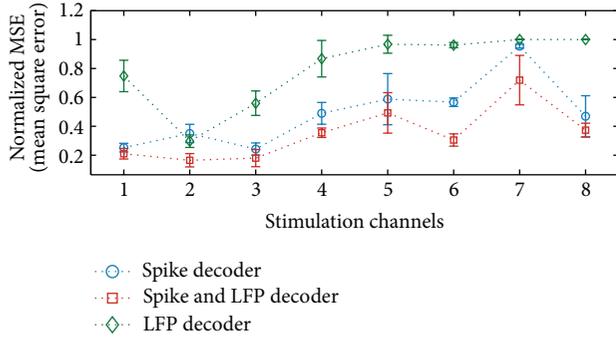


FIGURE 8: Performance comparison of the microstimulation reconstruction performance among spike decoder, LFP decoder, and spike and LFP decoder in terms of NMSE for each microstimulation channel.

output estimated by LFP decoder is smooth and more robust than the spike decoder, but the decoded signal undershoots the maximum force deflections. The LFP and spike decoder performed better than the LFP decoder by reincorporating the precise pulse timing information from spike trains.

5.3.3. Results for Decoding Electrical Microstimulation. We also implemented a decoder to reconstruct the microstimulation pattern. First, we mapped the 8 different stimulation configurations to 8 channels. We dismissed the shape of each stimulus, since the time scale of the stimulus width is only $200 \mu\text{s}$. The desired stimulation pattern of each channel is represented by a sparse time series of the stimulation amplitude.

NMSEs are obtained with ten subsequence decoding results. We used 120 s data to train the decoders and compute an independent test error on the remaining 20 s data. The spike and LFP decoder also outperformed both the LFP decoder and the spike decoder. The comparison of results is shown in Figure 8, which indicates that spike and LFP decoder is able to obtain the best performance amongst the stimulation channels, especially for channels 2, 4, 6, and 7. It is observed that stimulations on channels 4, 5, 6, 7, and 8 cannot be decoded from the LFP decoder at all, since the fine time information is averaged out in LFPs. For the spike trains decoder, the stimulation channels are not well discriminated. However, the combination of spike trains and LFPs enriched the stimulation information, which contributes to better discrimination of stimulation patterns among channels and also enables the model to capture the precise stimulation timing.

5.4. Open Loop Adaptive Inverse Control Results. The challenge of implementing a somatosensory prosthesis is to precisely control the neural response in order to mimic the neural response induced by natural stimulation. As discussed, the kernel-based adaptive inverse control diagram with tensor-product kernel is applied to address this problem. The adaptive inverse control model is based on a decoder which maps the neural activity in S1 to the microstimulation

delivered in VPL. We proceed to show how the adaptive inverse control model can emulate the neural response to “natural touch” using optimized microstimulation.

In the same recording, open loop adaptive inverse control via optimized thalamic (VPL) microstimulations is implemented. First, the inverse controller $C(z)$ is trained with 300 s of the data generated by recording the response to randomly patterned thalamic microstimulation. Then, the first 60 s of the neural response recorded during tactile stimulation at each touch site is used as the target pattern and control input. When this entire neural response sequence is fed offline to the controller, it generates a corresponding sequence of multichannel microstimulation amplitudes.

However, the generated microstimulation sequence needs further processing to meet the restrictions of bipolar microstimulation, before it applied to VPL. The restrictions and processing are the following.

- (1) The minimal interval between two stimuli 10 ms is suggested by the experimental setting. The mean shift algorithm [55] is used to locate the local maxima of a subsequence of stimuli (10 ms) for each single channel. The maximum amplitude and corresponding timing are used to set the amplitude and time of stimuli.
- (2) At any given time point, only a single pulse across all channels can be stimulated. Therefore, at each time point, only the maximum value across channels is selected for stimulation. The values at other channels are set to zero.
- (3) The maximum/minimum stimulation amplitude is set in the range $[8 \mu\text{A} - 30 \mu\text{A}]$, which has been suggested as the effective and safe amplitude range in previous experiments.

After this processing, the generated multichannel microstimulation sequence (60 s in duration) is ready to be applied to the microstimulator immediately following computation.

The neural response to the microstimulation is recorded and compared with the target natural response. Ideally, these two neural response sequences should be time-locked and be very similar. In particular, the portions of the controlled response in windows corresponding to a natural touch should match. As this corresponds to a *virtual touch* delivered by the optimized microstimulation, we define the term *virtual touch* to refer to the sequence of microstimulation patterns—the output of the controller—corresponding to a particular target natural touch.

Portions of the neural response for both natural and virtual touches are shown in Figure 9. The responses are aligned to the same time scale, even though they were not recorded concurrently. It is clear that the multiunit neural responses recorded during the controlled microstimulation share similar spatiotemporal patterns as those in the target set. Each virtual touch is achieved by a sequence of microstimulation pulses that evokes synchronized bursting across the neurons. In addition, microstimulation pulses are delivered in between touch times to mimic population spiking that is not associated with the touch timing.

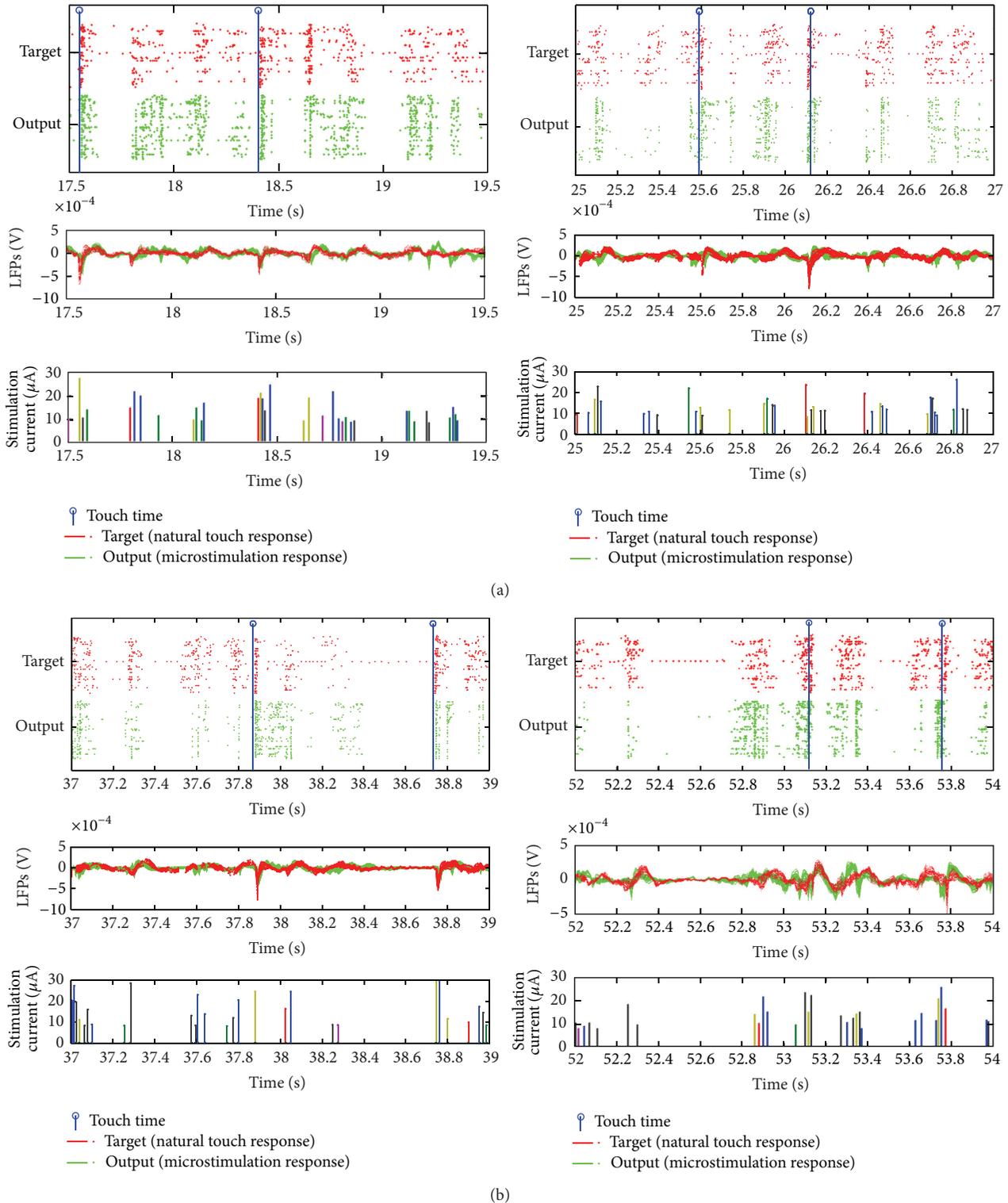


FIGURE 9: Neural responses to natural and virtual touches for touch on digit 1 (d1), along with the microstimulation corresponding to the virtual touches. Each of the four subfigures corresponds to a different segment of the continuous recording. In each subfigure, the timing of the touches, spatiotemporal pattern of spike trains and LFPs are shown in the top two panels; the bottom panel shows the spatiotemporal pattern of microstimulation, where different colors represent different microstimulation channels. The neural responses are time-locked, but not concurrently recorded, as the entire natural touch response is given as input to the controller which generates the optimized microstimulation patterns. When the optimized microstimulation is applied in the VPL, it generates S1 neural responses that qualitatively match the natural, that is, the target, response.

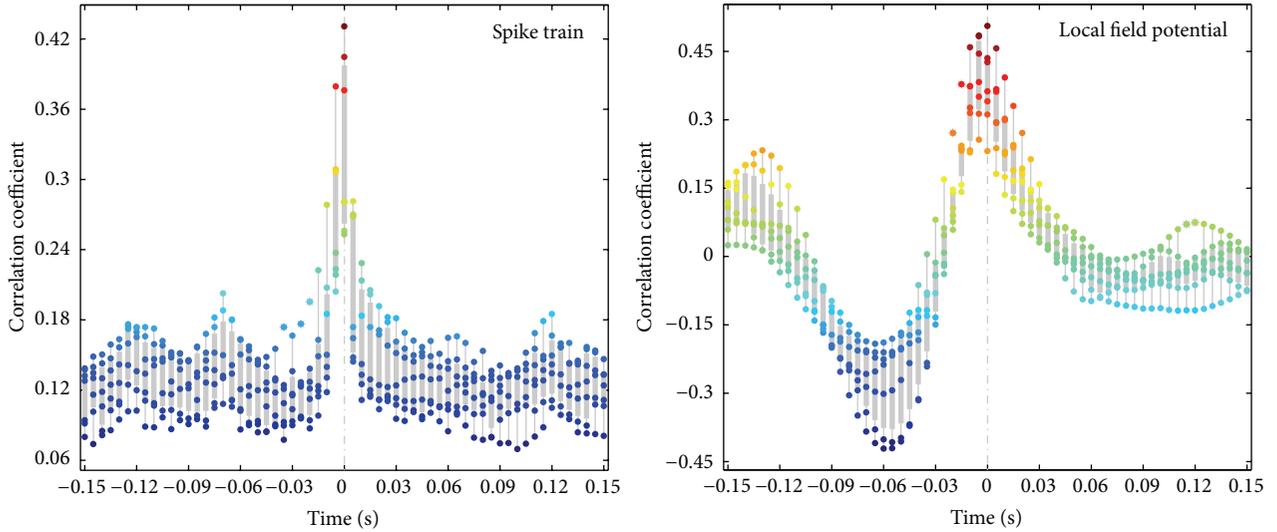


FIGURE 10: Correlation coefficients between the controlled neural system output and the corresponding target neural response stimulated by actual touch. Boxplot of correlation coefficients represents the results of 6 test trials. Each trial is corresponding to a particular touch site (digits: d1, d2, d4, p3, p1, and mp).

To evaluate performance, we concentrate on the following two aspects of virtual touches.

(i) *Touch Timing*. Whether the neural response to virtual touch is capable of capturing the timing information of the actual target touch is studied.

(ii) *Touch Site*. Whether the actual target touch site information can be discriminately represented by neural activity controlled by the microcirculation is studied.

For *touch timing*, we estimate the correlation coefficients (CC) over time between virtual touch responses and the corresponding target natural touch responses. To simplify the spike train correlation estimation, we bin the data using 5 ms bins. The correlation coefficients of both spike trains and LFPs are calculated. Figure 10 shows the boxplot plot of the correlation coefficients (CC) over 6 test trials, each of which corresponds to a particular natural touch site, a forepaw digit or pad (d1, d2, d4, p3, p1, or mp). It is observed that the maximum correlation coefficient is at lag zero for each trial, meaning that the virtual touch response is correctly time-locked. For each *touch site*, we estimate the similarity between the natural touch response and virtual touch response in the following two cases.

(i) *Matched Virtual*. Pairs consist of a virtual touch trial and a natural touch trial corresponding to the same touch site.

(ii) *Unmatched Virtual*. Pairs consist of a virtual touch trial and a natural touch trial corresponding to different touch sites.

We extract all the neural responses in the 300 ms window after touch onset and calculate the correlation coefficients between natural touch responses and virtual touch response across each pair of trials. The one-tailed Kolmogorov-Smirnov test (KS) is implemented to test the alternative

TABLE 2: Average and standard deviation of the correlation coefficient (CC) between natural touch spike train responses and virtual touch spike train responses (matched or unmatched). The P value is for the one-sided KS test between the matched and unmatched CC distributions.

Touch site	CC		P value
	Matched virtual	Unmatched virtual	
d1	0.42 ± 0.06	0.35 ± 0.06	0.00
d2	0.40 ± 0.05	0.37 ± 0.06	0.01
d4	0.40 ± 0.05	0.37 ± 0.05	0.02
p3	0.38 ± 0.05	0.37 ± 0.06	0.11
p2	0.40 ± 0.07	0.36 ± 0.05	0.00
mp	0.41 ± 0.07	0.37 ± 0.06	0.00

hypothesis that the distribution of the correlation coefficients for the *matched virtual* case is higher than the distribution for the *unmatched virtual* case (the null hypothesis is that the distributions are the same). The correlation coefficients and P value of KS test for spike trains and LFPs are shown in Tables 2 and 3. The similarity between natural touch responses and virtual touch responses in the *unmatched virtual* case is found to be significantly lower than the *matched virtual* case for most touch sites (P value < 0.05) except for touch site p3. Without psychophysical testing, it is unclear how effective the microstimulations are in producing true sensory sensations. Nonetheless, these are promising results to show the effectiveness of a controller utilizing the multiscale neural decoding methodology.

6. Conclusions

This work proposes a novel tensor-product-kernel-based machine learning framework, which provides a way to

TABLE 3: Average and standard deviation of the correlation coefficient (CC) between natural touch LFP responses and virtual touch LFP responses (matched or unmatched). The P value is for the one-sided KS test between the matched and unmatched CC distributions.

Touch site	CC		P value
	Matched virtual	Unmatched virtual	
d1	0.42 ± 0.20	0.28 ± 0.23	0.00
d2	0.46 ± 0.13	0.28 ± 0.22	0.00
d4	0.41 ± 0.19	0.26 ± 0.21	0.00
p3	0.38 ± 0.18	0.29 ± 0.22	0.07
p2	0.33 ± 0.19	0.26 ± 0.23	0.20
mp	0.34 ± 0.17	0.25 ± 0.21	0.00

decode stimulation information from the spatiotemporal patterns of multiscale neural activity (e.g., spike trains and LFPs). It has been hypothesized that spike trains and LFPs contain complementary information that can enhance neural data decoding. However, a systematic approach to combine, in a single signal processing framework, these two distinct neural responses has remained elusive. The combination of positive definite kernels, which can be defined in both the spike train space and the LFP space, seems to be a very productive approach to achieve our goals. We have basically used two types of combination kernels to achieve the multiscale combination: sum kernels to “average” across different spike channels, as well as across LFP channels, which combine evidence for the neural event in each modality, and product kernels across the spike and LFP modalities to emphasize events that are represented in both multiscale modalities. The results show that this approach enhances the accuracy and robustness of neural decoding and control. However, this paper should be interpreted as a first step of a long process to optimize the joint information contained in spike trains and LFPs. The first question is to understand why this combination of sum and product kernels works. Our analyses show that the sum kernel (particularly for the spike trains) brings stability to the neural events because it decreases the variability of the spike responses to stimuli. On the other hand, the product kernel requires that the neural event presents at both scales to be useful for decoding, which improves specificity. If we look carefully at Figure 6, we can understand the effect of decoding with the product kernel. Notice that the correlation times of spikes and LFPs are very different (LFPs have a much longer correlation time). Moreover, composite kernel definition can be naturally configured to different brain areas and even neuronal types with distinctive firing patterns. Each pattern will lead to different correlation profiles, which will immediately tune the properties of the kernels across brain areas and neural populations. If only LFPs are used, we can expect that the response time of the decoder will be very long and miss some events. The product kernel in fact limits the duration of the LFP kernel to that of the spike kernel and brings stability to the spike kernel. This explains exactly the decoding results. Therefore, results show that the proposed tensor-product-kernel framework can effectively integrate the information

from spikes and LFPs into the same model and enhance the neural decoding robustness and accuracy.

Furthermore, we applied the tensor-product-kernel framework in a more complex BMI scenario: how to emulate “natural touch” with microstimulation. Our preliminary results show that the kernel-based adaptive inverse control scheme employing tensor-product-kernel framework also achieves better optimization of the microstimulation than spikes and LFPs alone (results not shown). This result can be expected because the inverse controller is basically a decoder. However, we have to realize that not all the tasks of interest reduce to neural decoding, and we do not even know if neural control can be further improved by a different kernel design. This is where further research is necessary to optimize the joint kernels. For instance, we can weight both the channel information and the multiscale information to maximize the task performance using metric learning [40].

Overall, this tensor-product-kernel-based framework proposed in this work provides a general and practical framework to leverage heterogeneous neural activities in decoding and control scenario, which is not limited to spike trains and LFPs applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

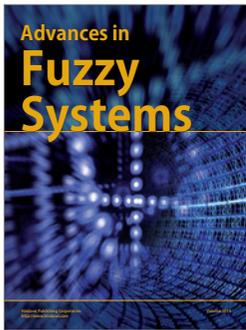
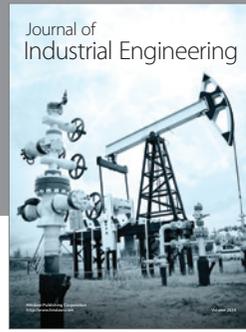
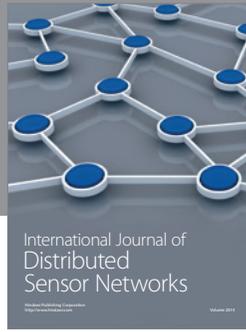
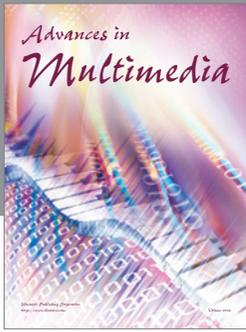
The authors would like to thank Ryan Burt for proofreading the paper. This work was supported in part by the US NSF Partnerships for Innovation Program 0650161 and DARPA Project N66001-10-C-2008.

References

- [1] J. K. Chapin, K. A. Moxon, R. S. Markowitz, and M. A. L. Nicolelis, “Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex,” *Nature Neuroscience*, vol. 2, no. 7, pp. 664–670, 1999.
- [2] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, “Cortical control of a prosthetic arm for self-feeding,” *Nature*, vol. 453, no. 7198, pp. 1098–1101, 2008.
- [3] D. M. Taylor, S. I. H. Tillery, and A. B. Schwartz, “Direct cortical control of 3D neuroprosthetic devices,” *Science*, vol. 296, no. 5574, pp. 1829–1832, 2002.
- [4] M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue, “Instant neural control of a movement signal,” *Nature*, vol. 416, no. 6877, pp. 141–142, 2002.
- [5] J. M. Carmena, M. A. Lebedev, R. E. Crist et al., “Learning to control a brain-machine interface for reaching and grasping by primates,” *PLoS Biology*, vol. 1, no. 2, pp. 193–208, 2003.
- [6] J. DiGiovanna, B. Mahmoudi, J. Fortes, J. C. Principe, and J. C. Sanchez, “Coadaptive brain-machine interface via reinforcement learning,” *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 1, pp. 54–64, 2009.

- [7] A. Belitski, S. Panzeri, C. Magri, N. K. Logothetis, and C. Kayser, "Sensory information in local field potentials and spikes from visual and auditory cortices: time scales and frequency bands," *Journal of Computational Neuroscience*, vol. 29, no. 3, pp. 533–545, 2010.
- [8] J. R. Huxter, T. J. Senior, K. Allen, and J. Csicsvari, "Theta phase-specific codes for two-dimensional position, trajectory and heading in the hippocampus," *Nature Neuroscience*, vol. 11, no. 5, pp. 587–594, 2008.
- [9] M. J. Rasch, A. Gretton, Y. Murayama, W. Maass, and N. K. Logothetis, "Inferring spike trains from local field potentials," *Journal of Neurophysiology*, vol. 99, no. 3, pp. 1461–1476, 2008.
- [10] G. Buzsáki and A. Draguhn, "Neuronal oscillation in cortical networks," *Science*, vol. 304, no. 5679, pp. 1926–1929, 2004.
- [11] D. L. Snyder and M. I. Miller, *Random Point Processes in Time and Space*, Springer, 1991.
- [12] R. E. Kass, V. Ventura, and E. N. Brown, "Statistical issues in the analysis of neuronal data," *Journal of Neurophysiology*, vol. 94, no. 1, pp. 8–25, 2005.
- [13] R. D. Flint, E. W. Lindberg, L. R. Jordan, L. E. Miller, and M. W. Slutzky, "Accurate decoding of reaching movements from field potentials in the absence of spikes," *Journal of Neural Engineering*, vol. 9, no. 4, Article ID 046006, 2012.
- [14] R. C. Kelly, M. A. Smith, R. E. Kass, and T. S. Lee, "Local field potentials indicate network state and account for neuronal response variability," *Journal of Computational Neuroscience*, vol. 29, no. 3, pp. 567–579, 2010.
- [15] J. Liu and W. T. Newsome, "Local field potential in cortical area MT: stimulus tuning and behavioral correlations," *Journal of Neuroscience*, vol. 26, no. 30, pp. 7779–7790, 2006.
- [16] P. Berens, G. Keliris, A. Ecker, N. Logothetis, and A. Tolias, "Feature selectivity of the gamma-band of the local field potential in primate primary visual cortex," *Frontiers in Neuroscience*, vol. 2, Article ID 199207, 2008.
- [17] D. Xing, C.-I. Yeh, and R. M. Shapley, "Spatial spread of the local field potential and its laminar variation in visual cortex," *Journal of Neuroscience*, vol. 29, no. 37, pp. 11540–11549, 2009.
- [18] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, Ser. Adaptive Computation and Machine Learning*, MIT Press, 2002.
- [19] L. Li, I. M. Park, A. Brockmeier et al., "Adaptive inverse control of neural spatiotemporal spike patterns with a reproducing kernel Hilbert space (RKHS) framework," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 21, no. 4, pp. 532–543, 2013.
- [20] S. Monaco, G. Króliczak, D. J. Quinlan et al., "Contribution of visual and proprioceptive information to the precision of reaching movements," *Experimental Brain Research*, vol. 202, no. 1, pp. 15–32, 2010.
- [21] R. S. Johansson and G. Westling, "Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects," *Experimental Brain Research*, vol. 56, no. 3, pp. 550–564, 1984.
- [22] J. E. O'Doherty, M. A. Lebedev, P. J. Ifft et al., "Active tactile exploration using a brain-machine-brain interface," *Nature*, vol. 479, no. 7372, pp. 228–231, 2011.
- [23] J. E. O'Doherty, M. A. Lebedev, T. L. Hanson, N. A. Fitzsimmons, and M. A. Nicolelis, "A brain-machine interface instructed by direct intracortical microstimulation," *Frontiers in Integrative Neuroscience*, vol. 3, 2009.
- [24] N. A. Fitzsimmons, W. Drake, T. L. Hanson, M. A. Lebedev, and M. A. L. Nicolelis, "Primate reaching cued by multichannel spatiotemporal cortical microstimulation," *The Journal of Neuroscience*, vol. 27, no. 21, pp. 5593–5602, 2007.
- [25] X.-J. Feng, B. Greenwald, H. Rabitz, E. Shea-Brown, and R. Kosut, "Toward closed-loop optimization of deep brain stimulation for Parkinson's disease: concepts and lessons from a computational model," *Journal of Neural Engineering*, vol. 4, no. 2, pp. L14–L21, 2007.
- [26] J. Liu, H. K. Khalil, and K. G. Oweiss, "Model-based analysis and control of a network of basal ganglia spiking neurons in the normal and Parkinsonian states," *Journal of Neural Engineering*, vol. 8, no. 4, Article ID 045002, 2011.
- [27] A. J. Brockmeier, J. S. Choi, M. M. Distasio, J. T. Francis, and J. C. Principe, "Optimizing microstimulation using a reinforcement learning framework," in *Proceedings of the 33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS '11)*, pp. 1069–1072, September 2011.
- [28] A. Brockmeier, J. Choi, M. Emigh, L. Li, J. Francis, and J. Principe, "Subspace matching thalamic microstimulation to tactile evoked potentials in rat somatosensory cortex," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC '12)*, pp. 2957–2960, 2012.
- [29] Y. Ahmadian, A. M. Packer, R. Yuste, and L. Paninski, "Designing optimal stimuli to control neuronal spike timing," *Journal of Neurophysiology*, vol. 106, no. 2, pp. 1038–1053, 2011.
- [30] J. Moehlis, E. Shea-Brown, and H. Rabitz, "Optimal inputs for phase models of spiking neurons," *Journal of Computational and Nonlinear Dynamics*, vol. 1, no. 4, pp. 358–367, 2006.
- [31] D. Brugger, S. Butovas, M. Bogdan, and C. Schwarz, "Real-time adaptive microstimulation increases reliability of electrically evoked cortical potentials," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 5, pp. 1483–1491, 2011.
- [32] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, 2000.
- [33] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.
- [34] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering*, Edited by S. Haykin, John Wiley & Sons, 2010.
- [35] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2001.
- [36] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *The Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.
- [37] C. Cortes, M. Mohri, and A. Rostamizadeh, "Algorithms for learning kernels based on centered alignment," *Journal of Machine Learning Research*, vol. 13, pp. 795–828, 2012.
- [38] M. Yamada, W. Jitkrittum, L. Sigal, E. P. Xing, and M. Sugiyama, "High-dimensional feature selection by feature-wise kernelized lasso," *Neural Computation*, vol. 26, no. 1, pp. 185–207, 2013.
- [39] R. A. Horn, "On infinitely divisible matrices, kernels, and functions," *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 8, no. 3, pp. 219–230, 1967.
- [40] A. J. Brockmeier, J. S. Choi, E. G. Kriminger, J. T. Francis, and J. C. Principe, "Neural decoding with kernel-based metric learning," *Neural Computation*, vol. 26, no. 6, 2014.

- [41] L. Shpigelman, Y. Singer, R. Paz, and E. Vaadia, *Spikernels: Embedding Spiking Neurons in Inner Product Spaces*, vol. 15 of *Advances in Neural Information Processing Systems*, MIT Press, 2003.
- [42] A. R. C. Paiva, I. Park, and J. C. Príncipe, “A reproducing kernel Hilbert space framework for spike train signal processing,” *Neural Computation*, vol. 21, no. 2, pp. 424–449, 2009.
- [43] I. M. Park, S. Seth, M. Rao, and J. C. Príncipe, “Strictly positive definite spike train kernels for point process divergences,” *Neural Computation*, vol. 24, no. 8, pp. 2223–2250, 2012.
- [44] L. Li, *kernel based machine learning framework for neural decoding [Ph.D. dissertation]*, University of Florida, 2012.
- [45] H. Ramlau Hansen, “Smoothing counting process intensities by means of kernel functions,” *The Annals of Statistics*, vol. 11, no. 2, pp. 453–466, 1983.
- [46] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, MIT Press, 2001.
- [47] J. Platt, “A resource-allocating network for function interpolation,” *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.
- [48] L. Csató and M. Opper, “Sparse on-line gaussian processes,” *Neural Computation*, vol. 14, no. 3, pp. 641–668, 2002.
- [49] Y. Engel, S. Mannor, and R. Meir, “The kernel recursive least-squares algorithm,” *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.
- [50] W. Liu, I. Park, and J. C. Príncipe, “An information theoretic approach of designing sparse kernel adaptive filters,” *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1950–1961, 2009.
- [51] B. Widrow and E. Walach, *Adaptive Inverse Control*, Edited by E. Cliff, Prentice-Hall, 1995.
- [52] J. T. Francis, S. Xu, and J. K. Chapin, “Proprioceptive and cutaneous representations in the rat ventral posterolateral thalamus,” *Journal of Neurophysiology*, vol. 99, no. 5, pp. 2291–2304, 2008.
- [53] G. Paxinos and C. Watson, *The Rat Brain in Stereotaxic Coordinates*, Academic Press, 6th edition, 2006.
- [54] G. Foffani, J. K. Chapin, and K. A. Moxon, “Computational role of large receptive fields in the primary somatosensory cortex,” *Journal of Neurophysiology*, vol. 100, no. 1, pp. 268–280, 2008.
- [55] K. Fukunaga and L. D. Hostetler, “The estimation of the gradient of a density function, with application in pattern recognition,” *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, 1975.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

