NC ms 3018

Bayesian Population Decoding of Motor Cortical Activity using a Kalman Filter

Wei Wu[*] Yun Gao[*] Elie Bienenstock[*§] John P. Donoghue[§] Michael J. Black[¶]

[*]Division of Applied Mathematics, [§]Department of Neuroscience,

[¶]Department of Computer Science,

Brown University, Providence, RI 02912, USA

weiwu@dam.brown.edu, gao@dam.brown.edu, elie@dam.brown.edu,

john_donoghue@brown.edu, black@cs.brown.edu

May 3, 2005

Correspondence:

Wei Wu

Department of Organismal Biology and Anatomy

University of Chicago,

Chicago, IL 60637, USA

Phone: (773) 702-5594, FAX: (773) 702-0037

Email: weiwu@uchicago.edu

Keywords: neural decoding, primary motor cortex, Kalman filter, multi-electrode array, Bayesian inference, neural prosthesis.

Acknowledgments: This work was supported in part by: the DARPA BioInfoMicro Program, the NIH NINDS Neural Prosthesis Program and Grant #NS25074, and the NSF ITR Program award #0113679. We thank D. Mumford, E. Brown, M. Serruya, A. Shaikhouni, J. Dushanova, C. Vargas-Irwin, L. Lennox, D. Morris, D. Grollman, and M. Fellows for their assistance.

JPD is a cofounder and shareholder in Cyberkinetics, Inc., a neurotechnology company that is developing neural prosthetic devices.

The basic Kalman filter method was presented at the 16th Annual Conference on Neural Information Processing System, in Vancouver, Canada, December 2002. This submission adds significant new material and expands upon the conference paper. We repeat some of of results so that the current manuscript is self-contained.

Abstract

Effective neural motor prostheses require a method for decoding neural activity representing desired movement. In particular, the accurate reconstruction of a *continuous* motion signal is necessary for the control of devices such as computer cursors, robots, or a patient's own paralyzed limbs. For such applications we developed a real-time system that uses Bayesian inference techniques to estimate hand motion from the firing rates of multiple neurons. In this study, we used recordings that were previously made in the arm area of primary motor cortex in awake behaving monkeys using a chronically implanted multi-electrode microarray. Bayesian inference involves computing the posterior probability of the hand motion conditioned on a sequence of observed firing rates; this is formulated in terms of the product of a *likelihood* and a *prior*. The likelihood term models the probability of firing rates given a particular hand motion. We found that a linear Gaussian model could be used to approximate this likelihood and could be readily learned from a small amount of training data. The prior term defines a probabilistic model of hand kinematics and was also taken to be a linear Gaussian model. Decoding was performed using a Kalman filter which gives an efficient recursive method for Bayesian inference when the likelihood and prior are linear and Gaussian. In off-line experiments, the Kalman-filter reconstructions of hand trajectory were more accurate than previously reported results. The resulting decoding algorithm provides a principled probabilistic model of motor-cortical coding, decodes hand motion in real time, provides an estimate of uncertainty, and is straightfor-

ward to implement. Additionally the formulation unifies and extends previous models of neural coding while providing insights into the motor-cortical code.

1 Introduction

Recent research on developing neural motor prostheses has demonstrated the feasibility of direct neural control of computer cursor motion and other devices using implanted electrodes in non-human primates (Wessberg et al., 2000; Serruya et al., 2002; Taylor et al., 2002; Carmena et al., 2003). These results are enabled by a variety of mathematical *decoding* methods that produce an estimate of the subject's *state* (e.g. hand position) from a sequence of measurements (e.g. the firing rates of a population of cells). A number of algorithms have been proposed for decoding extracellularly recorded neural firing activity in the arm area of primary motor cortex (MI) to perform off-line reconstruction of hand motion or on-line control of cursors or robotic devices (see (Schwartz et al., 2001; Serruya et al., 2003) for a brief overview). Here we pose this problem as one of Bayesian inference in which the goal is to estimate the *a posteriori* probability of hand kinematics conditioned on an observed sequence of firing rates. The Bayesian approach formulates this posterior probability as the product of a *likelihood* term and an *a priori* probability. The likelihood term models the probability of the firing rates given the current hand motion and can be learned from training data. The prior combines a model of how the hand moves over time with an estimate of the kinematics at the previous time instant. The Kalman filter (Gelb, 1974; Kalman, 1960) provides an efficient recursive algorithm to optimally estimate the posterior probability when the likelihood and prior models are linear and Gaussian.

Previous methods for decoding MI activity include the population-vector algorithm (Georgopoulos et al., 1986; Moran and Schwartz, 1999b; Moran and Schwartz, 1999a;

Schwartz and Moran, 1999; Taylor et al., 2002), linear filtering (Paninski et al., 2004; Sanchez et al., 2003; Serruya et al., 2002; Wessberg et al., 2000), and artificial neural networks (Wessberg et al., 2000). Each of these methods can be viewed as a *direct* method that attempts to estimate the hand kinematics $\mathbf{x}$ as a function of the neural firing $\mathbf{z}$; that is

$$\mathbf{x} = f_1(\mathbf{z}).$$

In contrast, most models of neural coding can be viewed as *generative* models where the neural activity is a function of a behavior or stimulus $\mathbf{x}$; that is

$$\mathbf{z} = f_2(\mathbf{x}) + \text{noise}$$

where the noise might be assumed to be Poisson. The Bayesian approach presented here provides a clear and rigorous way of taking generative models of neural encoding and exploiting them to perform decoding.

Previous authors have explored the relationship between firing rates in MI and various aspects of hand motion (position, direction, speed, velocity, or acceleration) (Kettner et al., 1988; Georgopoulos et al., 1986; Moran and Schwartz, 1999b; Flament and Hore, 1988). While previous studies have typically viewed these behavioral variables in isolation, we found that decoding performance is improved when the encoding model *simultaneously* takes into account all these variables. This suggests the need for a richer model of neural coding than is typically considered. Additionally, our results demonstrate the importance of modeling correlated noise in the firing rates of multiple cells. Decoding performance drops significantly when cells are assumed to be conditionally independent.

Much of the prior work on motor cortical decoding has focused on relatively constrained "center-out" motions rather than the continuous motions considered here (Paninski et al., 2004). To cope with continuous motion we adopt a very simple prior model of hand motion that, in contrast to previous decoding methods, explicitly models the temporal evolution of the hand kinematics. Bayesian methods have been exploited previously to infer the 2D location of a rat from hippocampal place-cell activity (Brown et al., 1998; Twum-Danso and Brockett, 2001; Zhang et al., 1998). The application of Bayesian decoding to motor cortical data was proposed in (Gao et al., 2002) with various Kalman filter formulations being recently studied (Sanchez et al., 2002; Wu et al., 2002; Wu et al., 2003). Some of the results in the present paper were previously reported in Wu *et al.* (Wu et al., 2003).

The Kalman filter has a number of desirable properties for motor cortical decoding. The inclusion of prior information about the system state enables an efficient recursive formulation of the decoding algorithm and effectively smooths noisy estimates in a mathematically principled way; this is particularly important for decoding complex, natural, hand motions required for neural motor prostheses. We reconstructed hand trajectories from pre-recorded neural firing rates and found that the Kalman filter method was more accurate than previous approaches while also being computationally efficient. The detailed application of the method provides insight into neural coding in MI and is useful for examining optimal time lags between spiking activity and hand movement, the accuracy of different models of hand kinematics, the contributions of different neural population sizes, and the effect of temporal bin sizes for estimating neuronal firing rate. We describe the implementation and structure of a Kalman-filter method that is

computationally efficient to learn, requires little training data, provides real-time decoding, is applicable to the complex natural motions, and thus, seems well suited to prosthetic applications.

The structure of this paper is as follows: Section 2 summarizes the experimental paradigm used previously to obtain the neural recordings. This section also introduces the linear Gaussian model of motor cortical activity, our Bayesian framework, the underlying statistical assumptions, the Kalman filter decoding algorithm, and an approach for estimate optimal time lags. Section 3 describes the decoding results as well as experimental results related to various modeling choices that are important for accurate decoding. It also compares the Kalman filter with previous decoding methods (the linear filter and population vector methods). Section 4 discusses related work and offers conclusions. The Appendix provides the mathematical details of the Kalman filter algorithm, a comparison with the Wiener filter and a few additional algorithmic and modeling details with associated experimental results.

2 Materials and Methods

2.1 *Experimental Methods*

The neural data used here was pre-recorded and has been described elsewhere (Paninski et al., 2004; Serruya et al., 2002). Briefly, after initial task training, two macaque monkeys were implanted with silicon microelectrode arrays containing 100 platinized-tip probes (Cyberkinetics Inc., Foxboro, MA). Details of the array and recording protocols are described elsewhere (Maynard et al., 1997; Maynard et al., 1999). The devices were implanted in the arm area of primary motor cortex (MI) (see (Donoghue et al., 1998)

for details). All procedures were in accordance with protocols approved by Brown University Institutional Animal Care and Use Committee. Signals were amplified and digitized using commercial hardware (Plexon Inc, Dallas TX). As is common practice in the literature, waveforms crossing experimenter-determined thresholds were further processed to detect action potentials; the details differed for each of the two behavioral tasks described below. Action potentials were then counted within fixed time windows (bins) and the firing rate (number of spikes per unit time) within each bin was computed for each neuron. All encoding/decoding analysis was performed using these discrete approximations to the firing rate.

The behavioral paradigms for the two tasks below are described in (Paninski et al., 2004) and (Serruya et al., 2002). In each task, the monkey viewed a computer monitor while continuously moving a manipulandum on a $30cm \times 30cm$ tablet (with approximately a $25cm \times 25cm$ workspace) that was parallel to the floor. The position of the manipulandum (hand position) controlled the 2D motion of a feedback cursor on the monitor. The hand position and neural activity were recorded simultaneously. In addition to hand position we computed derivatives of the position using finite differences; these approximated velocity, acceleration, and higher-order hand kinematics. For example, given a time series of positions (x_k, y_k) at time t_k we approximated the velocity, $(v_{x,k}, v_{y,k})$ as $(\frac{x_k - x_{k-1}}{\Delta t}, \frac{y_k - y_{k-1}}{\Delta t})$ (where Δt is the time step length). It is well known that this approach results in noisy estimates of the derivatives with the effects of noise being more pronounced in the higher derivatives. Consequently we also experimented with using splines to smoothly interpolate the position data. Differentiating these splines produced less noisy derivatives but did not significantly improve decoding

performance. It also increased the complexity of the method and complicated real-time decoding. Consequently in all further analysis we used the simple, discrete, derivative approximations to represent the hand kinematics.

Pursuit Tracking Task. The Pursuit Tracking task was clearly described in (Paninski et al., 2004). Briefly, a target dot moved slowly on the monitor and the behavioral task required moving the feedback cursor with the manipulandum so that it tracked the target within a given distance range. On each trial, the target motion followed a unique random walk from a different random starting location. Each trial ended when the dot was out of the tracking range and lasted at most 10 seconds with the majority of trials lasting approximately 8 or 9 seconds. Short trials, in which the monkey was unable to track the target for more than 5 seconds, were judged unsuccessful and were not considered. Our subsequent analysis was based on the remaining 182 trials. The 2D histograms of the hand position, velocity and acceleration of over all trials are shown in Fig. 1 (first row). Note that these distributions are significantly different from those obtained during simple, stylized, movements found in center-out reaching tasks (Taylor et al., 2002). While the hand motions were more "general" than those in stereotyped reaching tasks, the motion was still constrained to follow a particular path.

After thresholding, detected waveforms were analyzed off-line using commercial software (Plexon Inc, Dallas TX). Twenty five well isolated individual units (neurons) were detected (Serruya et al., 2003) and the firing rate for each unit was computed in non-overlapping $50ms$ time bins. The hand kinematics were sub-sampled to match the $50ms$ time intervals.

Pinball Task. The Pinball task (Serruya et al., 2002) was designed to test a direct neural control task and differed from the Pursuit Tracking task in that the target did not move continuously but rather appeared in random locations on the monitor. The monkey was required to move the feedback dot with the manipulandum to "hit" the target (within a pre-specified distance). When the target was acquired, it disappeared and then reappeared in a new random location. Each time the target appeared, the monkey moved to hit the new location. The motions made by the monkey on the 2D plane were less constrained than in the Pursuit Tracking task since the exact motion used to reach the targets was under the control of the subject.

From this experiment, we obtained two sets of trials: one was approximately 3.5 minutes in length and was used as training data to learn our encoding model; the other was approximately 1 minute in length and was used as test data for decoding. The 2D histograms of position, velocity and acceleration of all data are shown in Fig. 1 (second row). Note that the distribution over position is more uniform than in the Pursuit Tracking task. Also note that the magnitude of the velocity and acceleration of the hand was significantly higher in this task.

The recording of hand motion and the approximation of the temporal derivatives (velocity, acceleration, etc.) was exactly the same as in the Pursuit Tracking task. The waveforms, however, were sorted on-line into units using manually-set thresholds without a separate off-line sorting process. All the waveforms crossing the thresholds were treated as action potentials from a single unit potentially resulting in multi-unit data. The firing rate was computed for each such unit in $70ms$ time bins and there were 42 such units for this task.

The firing rates of the cells were significantly higher than in the Pursuit Tracking task (likely due to the faster hand motion). Quantitatively, the average hand speeds were approximately $14.67cm/s$ (Pinball) and $2.88cm/s$ (Pursuit Tracking), while the average firing rates over all cells were approximately 30spikes$/s$ (Pinball) and 10spikes$/s$ (Pursuit Tracking).

2.2 *Statistical Methods*

Our focus here is on a probabilistic, Bayesian, approach for inferring hand movement from the firing rate of neurons in MI. We want our approach to (1) have a sound probabilistic foundation; (2) explicitly model noise in the data; (3) indicate the uncertainty in estimates of hand position; (4) make the statistical assumptions about the data explicit; (5) require a minimal amount of "training" data; (6) provide on-line estimates of hand position with short delay (less than 200ms); and (7) provide insight into the neural coding of movement. To that end, we developed a Kalman-filtering method (Gelb, 1974; Welch and Bishop, 2001) that provides a rigorous and well understood framework that addresses these issues. This approach provides a control-theoretic model for the encoding of hand movement in motor cortex and for inferring, or decoding, this movement from the firing rates of a population of neurons. The approach generalizes and extends previous coding models.

2.2.1 Modeling Neural Coding of Hand Kinematics

Georgopoulos *et al.* (1982) observed that the firing rate of cells in MI were approximated by a cosine "tuning function." In particular the firing rate, z_k, of a cell at some

time t_k is related to movement *direction*, θ_k, by

$$z_k = h_0 + h_p cos(\theta_k - \theta_p) \tag{1}$$

where θ_p is the cell's "preferred" direction (i.e. direction of maximal response) and h_0 and h_p are scalar constants. Equivalently, this can be expressed as

$$z_k = h_0 + h_1 sin(\theta_k) + h_2 cos(\theta_k), \tag{2}$$

where h_1, h_2 are scalar parameters that can be fit to training data.

The above model has formed the foundation for much of the analysis of motor cortical encoding of motion. For general prosthetic applications, however, this model is insufficient as it only relates neural activity to movement *direction*. Since it does not capture the full kinematics of hand motion, decoding using this approach has focused on "center-out" tasks where movement direction is the key component of the motion.

Moran and Schwartz (Moran and Schwartz, 1999b) extended the above model to include the speed, $||v||$, of the hand

$$z_k = h_0 + ||v||(h_x sin(\theta_k) + h_y cos(\theta_k)). \tag{3}$$

This is equivalent to modeling the firing rate as a linear function of velocity in the x and y coordinates:

$$z_k = h_0 + h_x v_{x,k} + h_y v_{y,k}, \tag{4}$$

where $v_{x,k}$ and $v_{y,k}$ represent the hand velocity in the x and y directions respectively at time t_k.

A similar linear model was proposed to relate firing rate and hand position (Kettner et al., 1988):

$$z_k = f_0 + f_x x_k + f_y y_k, \tag{5}$$

where x_k and y_k represent hand position and f_0, f_x, f_y are the linear coefficients which are fit to training data.

We found that linear models of position and velocity (Equations (5) and (4) respectively) provided reasonable approximations to our data. In the Pursuit Tracking data, 23 of the total 25 cells are appropriately characterized by Equations (5), and 25 are by Equations (4) (multiple regression, F test, $p < 0.05$). In the Pinball data, these numbers are 39 and 41 of the total 42 cells (multiple regression, F test, $p < 0.05$). Some examples of linear fits are shown in Fig. 2 for illustration. In addition to position and velocity, the firing rate has also been shown to be related to hand acceleration (Flament and Hore, 1988).

Based on this previous work we assume (for now) that the population activity is related to the position, velocity and acceleration of the hand (this will be verified and extended later). We then define the system *state* to be a six-dimensional vector $\mathbf{x}_k = [x, y, v_x, v_y, a_x, a_y]_k^T$ representing the x-position, y-position, x-velocity, y-velocity, x-acceleration, and y-acceleration of the hand at time $t_k = k\Delta t$ where $\Delta t = 70ms$ (Pinball task) or $50ms$ (Pursuit Tracking task) in our experiments. Other models of the hand kinematics are possible and are explored in the experimental results.

Below we show how to estimate this system's state in a principled way that extends previous work, incorporates an explicit noise model, and models correlations between

the activity of different cells.

2.2.2 Bayesian Formulation

The above models all linearly relate components of the system state to the firing rates of individual neurons. Generalizing this idea, let observations $\mathbf{z}_k \in \mathbb{R}^C$ represent a $C \times 1$ vector containing the firing rates at time t_k for C observed neurons within a Δt time interval. Let $\mathbf{Z}_k = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k]^T$ represent the history of measurements up to time bin k.

We pose the problem of estimating the system state as one of Bayesian inference. Let $p(\mathbf{x}_k|\mathbf{Z}_k)$ be the *a posteriori* probability of the system state conditioned on the measurements. We make two Markov assumptions:

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{x}_{k-2}, \ldots, \mathbf{x}_1) = p(\mathbf{x}_k|\mathbf{x}_{k-1}), \tag{6}$$

and

$$p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{Z}_{k-1}) = p(\mathbf{z}_k|\mathbf{x}_k). \tag{7}$$

These state that, given the hand kinematics at time $k-1$, the hand kinematics at time k is conditionally independent of the previous hand motions and that, conditioned on the current state, the firing rates are independent of the firing rates at previous time instants. Then, using Bayes Theorem and simple algebra we can write the posterior probability as:

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \kappa\, p(\mathbf{z}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})\, d\mathbf{x}_{k-1}, \tag{8}$$

where $p(\mathbf{z}_k|\mathbf{x}_k)$ is called the *likelihood* of the state, $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is a *temporal prior* that models how the state evolves from one time instant to the next, and $p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})$ is

simply the posterior probability at the previous time instant. The term κ is a normalizing term, independent of $\mathbf{x}_k$, which ensures that posterior integrates to 1.

Decoding then involves estimating the posterior probability, $p(\mathbf{x}_k|\mathbf{Z}_k)$, at each time instant, which we can do recursively using Eq. (8). Having a representation of the full posterior has advantages over methods that estimate $\mathbf{x}_k$ with no representation of uncertainty. Given the posterior, one can compute an estimate of $\mathbf{x}_k$ in a variety of standard ways. For example, one can compute the expected value of $\mathbf{x}_k$, $E[\mathbf{x}_k|\mathbf{Z}_k]$, or the value that maximizes $p(\mathbf{x}_k|\mathbf{Z}_k)$ resulting in a maximum *a posteriori* (MAP) estimate. In the general case, the posterior probability, $p(\mathbf{x}_k|\mathbf{Z}_k)$, can be an arbitrary, non-Gaussian, multi-modal distribution (Gao et al., 2002; Cisek and Kalaska, 2002), and the integral in (8) can be numerically approximated using Monte Carlo integration techniques (Brockwell et al., 2004; Gao et al., 2002; Shoham, 2001). To achieve a real-time decoding algorithm, we assume that the likelihood and the prior are both Gaussian, and this leads to a closed-form recursive solution for estimating the posterior (which is also Gaussian); this is known as the Kalman filter (Kalman, 1960; Gelb, 1974; Welch and Bishop, 2001).

To fully specify the decoding model we must define the likelihood, the temporal prior, and the algorithm for estimating the posterior. These are described below.

Likelihood Model. The likelihood relates the hand kinematics to the neural-firing rates. We begin by defining a *generative model* for the activity of the population as

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{q}_k\,, \tag{9}$$

where $k = 1, 2, \cdots, M$, M is the number of time steps in the trial, and $\mathbf{H}_k \in \mathbb{R}^{C \times 6}$ is a matrix that linearly relates the six-dimensional hand state to the firing rates. We assume that the noise in the observations is zero-mean and normally distributed with covariance $\mathbf{Q}_k$; i.e. $\mathbf{q}_k \sim N(0, \mathbf{Q}_k), \mathbf{Q}_k \in \mathbb{R}^{C \times C}$.

Note that the assumption of zero-mean Gaussian noise in (9) is only applicable to centralized (zero-mean) firing rates. The raw data is neither zero-mean nor truly Gaussian. Consequently, before processing the firing rate data, it was first square-root transformed (Maynard et al., 1999; Moran and Schwartz, 1999b) to make it better modeled by a Gaussian. This pre-processing step is not absolutely necessary and only improves decoding performance slightly; nevertheless we will assume square-root transformed firing rates in the remainder of the paper. We then centralized the firing data as well as the hand kinematics so that they both had zero mean; this was simply done by computing the mean firing and mean kinematics in the training set and subtracting this from the data in both the training and test sets.

Previous work showed that the correlation in MI neurons is important for the encoding of movement parameters (Fetz et al., 1991; Hatsopoulos et al., 1998). Consequently, we take $\mathbf{Q}_k$ to be a full covariance matrix to model correlated noise in the firing rates. Specifically, noise here results from errors in the predictions from the generative model. This formulation of an explicit generative model with an explicit noise term generalizes previous work. In our experiments we found that the full system state and full covariance matrix produced the most accurate decoding results. The previous models above can be viewed as using a reduced system state and a diagonal covariance matrix. These simplified models resulted in degraded performance. This will be quantitatively

verified later for both tasks.

Temporal Prior. By modeling how the system state is expected to evolve over time, the temporal prior can reduce the effects of noise and smooth the estimates in a mathematically principled way. We take the simple approach of assuming that the state propagates in time according to a linear Gaussian model

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{w}_k, \tag{10}$$

where $\mathbf{A}_k \in \mathbb{R}^{6\times 6}$ is the coefficient matrix, and the noise term $\mathbf{w}_k \sim N(0, \mathbf{W}_k), \mathbf{W}_k \in \mathbb{R}^{6\times 6}$. As in the case of the the measurement model, $\mathbf{W}_k$ is taken to be a full covariance matrix. Equation (10) states that the hand kinematics (position, velocity and acceleration) at time $k+1$ is linearly related to the state at time k.

This is similar to the system model used in (Brown et al., 1998) for estimating a rat's position from the firing rates of hippocampal place cells. In contrast, here we also model higher order hand kinematics (velocity and acceleration). While more complex, non-linear, dynamical models could also be exploited, our focus here is on the likelihood term which represents our model of the neural code.

2.2.3 Learning and Decoding with the Kalman Filter

In practice, $\mathbf{A}_k, \mathbf{H}_k, \mathbf{W}_k, \mathbf{Q}_k$ may vary with time t_k, however, we make the common simplifying assumption that they are constant (independent of k). Thus, we can estimate all the constant parameters $\mathbf{A}, \mathbf{H}, \mathbf{W}, \mathbf{Q}$ from training data by maximizing the joint probability $p(\mathbf{X}_M, \mathbf{Z}_M)$, where both the hand kinematics $\mathbf{X}_M = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M]^T$ and the firing rates $\mathbf{Z}_M$ are known for the M time instants in the training set.

Given our first-order Markov assumptions, the joint probability distribution over states $\mathbf{X}_M$ and firing rates $\mathbf{Z}_M$ is

$$p(\mathbf{X}_M, \mathbf{Z}_M) = [p(\mathbf{x}_1) \prod_{k=2}^{M} p(\mathbf{x}_k|\mathbf{x}_{k-1})][\prod_{k=1}^{M} p(\mathbf{z}_k|\mathbf{x}_k)].$$

This is simply the product of the prior hand state $p(\mathbf{x}_1)$ at the first time instant, the probability of each new state conditioned on the previous one, and the likelihood at each time instant. Given training data we maximize this probability with respect to $\mathbf{A}, \mathbf{H}, \mathbf{W}, \mathbf{Q}$ as described in the Appendix.

Decoding then involves reconstructing (or inferring) the system state at each time instant given the prior estimate of the state and the new measurements of neural firing. Since the measurement and system equations, (9) and (10), are both linear and Gaussian, decoding can be performed using the Kalman filter (Kalman, 1960). The details of the algorithm for recursive Bayesian inference are provided in the Appendix. The Appendix also outlines the relationships between the Kalman filter, traditional linear filtering methods, and the Wiener filter.

2.3 Estimating the Optimal Lag

The physical relationship between neural firing and arm movement implies the existence of a time lag between them (Moran and Schwartz, 1999b; Paninski et al., 2004). If an "optimal lag" can be found, it should improve the model's encoding ability and the accuracy of the decoding. Introducing a time lag in the model means that to estimate the state $\mathbf{x}_k$ at time t_k we should consider measurements, $\mathbf{z}_{k-i}$ at some previous (or future) instant in time t_{k-i} for some integer i.

2.3.1 Uniform Lag

The simplest assumption is that all cells exhibit the same lag. Finding the optimal lag then involves fitting the Kalman model for a variety of integer values i and choosing the one that gives the best encoding or decoding performance. Since we can easily bound the range of possible lags, this search is straightforward to perform.

In our experiments, the data was binned into 70 or 50ms time bins. We found that a uniform lag of 2 or 3 time bins, corresponding to 140 or 150ms, produced the most accurate decoding results. This result is similar to Moran and Schwartz (Moran and Schwartz, 1999b) where 145ms were chosen as the average time lag between firing activity and hand velocity.

2.3.2 Non-Uniform Lag

We found, however, that the best decoding results were obtained by allowing each cell to have its own lag (within a pre-defined range of possible lags consistent with known lags for cells in MI). To deal with different lags for each cell we extend our notation: let $l_i \in \{0, 1, \cdots, L\}$ be the lag of the i^{th} cell, $i = 1, 2, \cdots, C$ (C is the total number of cells). Let $\{l_j\} = \{l_1, \ldots, l_C\}$ be some set of lag times for the C units. As suggested by the optimal 140 or 150ms uniform time lag for both tasks, we took the maximum lag to be $L = 4$ in the Pinball task (corresponding to $0ms \leq$ lag $\leq 280ms$); and $L = 5$ in the Pursuit Tracking task (corresponding to $0ms \leq$ lag $\leq 250ms$). The k^{th} firing-rate vector is $\mathbf{z}_k = (z_{1,k}, z_{2,k}, \cdots, z_{C,k})$, in which each $z_{i,k}$ is the firing rate of cell i at time step $k - l_i$. For each possible assignment, $\{l_j\}$, of lags to cells one could train the Kalman-filter model. The Kalman filtering algorithm generates the error covariance

matrix $\mathbf{P}_k$ (for k large enough, it is approximately constant). Letting $mse(\{l_j\})$ be the sum of the first two components on the main diagonal of $\mathbf{P}_k$ (equivalent to the mean-squared error of position), our goal is to find the optimal assignment of lags to cells $\{l_j\}$; that is,

$$\text{argmin}_{\{l_j\}=\{l_i \in \{0,\dots,L\}; i=1,2,\cdots,C\}} mse(\{l_j\}).$$

A brute-force search of all possible combinations of lags would require estimating the Kalman model for L^C possibilities. This is impractical so, instead, we devise a fast, randomized, greedy searching approach which is able to obtain very stable approximations to the desired time lags. The algorithm is sketched in Table 1.

The algorithm works by optimizing the lag time for cells individually; that is, it solves for the lag of one cell while holding the lags of all the other cells fixed. The algorithm then cycles through all the cells in a random order updating their lag in this way. This process is repeated multiple times, and we found that it converges quickly in practice (after three or four passes through the cells).

This algorithm requires that the Kalman filtering algorithm be applied to the training data only LRC times. In our experiments we took the number of iterations, R, to be 5. In the Pinball experiment, for example, the number of cells, C, was 42 resulting in a computational cost that was much lower than the L^C operations required for exhaustive search.

1. Randomly choose initial lag $l_i \in \{0, 1, \cdots, L\}$ for each cell $i = 1, 2, \ldots, C$.
2. For $iteration$ = 1 to R

   % Randomize the order in which cells are considered

   % to minimize effects of dependencies between cells

   For C iterations

   select a cell index c at random from $\{1, \cdots, C\}$ without replacement

   hold all other lags constant (i.e. $l_m, m \neq c$).

   Update l_c by minimizing:

   $$l_c = \text{argmin}_{l_c \in \{0,1,\cdots,L\}} mse(\{l_1, \ldots, l_c, \ldots, l_C\}).$$

3. Return the final set of values $\{l_k\}$.

Table 1: Greedy algorithm for estimating individual lag times for each cell.

3 Results

The Kalman filter is well known and its implementation is quite standard. Our focus here is on the development of this method for neural prosthetic applications. This application to neural decoding requires a number of modeling choices which are explored below in detail. Optimizing these modeling choices may provide insight into neural coding in MI. Below, we report results for the two different continuous hand-motion tasks described in the Methods section, on two different animals.

The accuracy of the reconstructions is reported using two different criteria: the correlation coefficient (CC, describing the similarity) and the mean squared error (MSE, describing the Euclidean distance) between the reconstructed and true hand trajectories. Assume $(\hat{x}_t, \hat{y}_t)$ is the estimate for the true position (x_t, y_t), $t = 1, \cdots, T$, then MSE and CC are defined as follows:

$$MSE = \frac{1}{T}\sum_{t=1}^{T}((x_t - \hat{x}_t)^2 + (y_t - \hat{y}_t)^2),$$

and

$$CC = \left(\frac{\sum_t (x_t - \bar{x})(\hat{x}_t - \bar{\hat{x}})}{\sqrt{\sum_t (x_t - \bar{x})^2 \sum_t (\hat{x}_t - \bar{\hat{x}})^2}}, \frac{\sum_t (y_t - \bar{y})(\hat{y}_t - \bar{\hat{y}})}{\sqrt{\sum_t (y_t - \bar{y})^2 \sum_t (\hat{y}_t - \bar{\hat{y}})^2}} \right),$$

where $\bar{x}$ and $\bar{y}$ represent the means of the x and y positions respectively. Note that while we computed the full kinematic state vector, we characterized the error solely in terms of position, since position accuracy is the key criterion for many neural-control tasks. We note also that the MSE is more meaningful for prosthetic applications where the subject needs precise control of cursor position; we observed decoding results with

relatively high correlation coefficients that were sometimes far from the true 2D hand trajectory.

Below, we first explore the basic Kalman model and its decoding performance for the two different movement tasks. We then analyze its behavior with respect to a number of modeling choices. Finally, we compare it with other techniques (the population vector and linear filter).

3.1 Decoding

Pinball Task. To be practical, we must be able to train the model (i.e. estimate $\mathbf{A}$, $\mathbf{H}$, $\mathbf{W}$, $\mathbf{Q}$) using a small amount of data. Experimentally we found that approximately 2.5 minutes of training data sufficed for accurate reconstruction (this is similar to the result for fixed linear filters reported in (Serruya et al., 2002)). The exact amount needed will increase as the number of parameters in the model (neurons and kinematic variables) increases. We explore the effect of varying the amount of training data below.

After learning the Kalman model as described in the Methods section and Appendix, we evaluate it by reconstructing test trajectories off-line using approximately 1 minute of recorded neural data not present in the training set. At the beginning of a test trial, we made the assumption that hand kinematics were unknown and we let the predicted initial condition be equal to the average of the kinematics in the training data. The Kalman filter was then applied to reconstruct the hand kinematics. Some examples of the reconstructed hand trajectory are shown in Fig. 3 (A) while Fig. 4 shows the reconstruction of each component of the state variable (position, velocity and acceleration in x and y) for 1/3 of the test data. Note that, since the ground-truth velocity and accel-

eration curves were computed from the position data with simple finite differences, the ground truth for these variables is quite noisy.

Pursuit Tracking Task. The data from the pursuit tracking task consisted of 182 short trials from which we randomly selected 130 trials (approximately 17 minutes) as training data and took the remaining 52 trials (approximately 6.5 minutes) as test data.

In contrast to the Pinball task, here each test trial was of short duration, and consequently the choice of initial starting state for the hand kinematics had a large impact on the accuracy of the results. Whereas in the Pinball task we let the starting kinematics be equal to the mean state in the training data, here we took it to be the true initial condition. Starting from this state, the Kalman filter algorithm was applied to reconstruct the system state as before. Figure 3 (B) shows some examples of reconstructed hand trajectories. Numerical results are presented below for various lags.

3.2 Real-time Performance

The encoding and decoding software was written in Matlab (The Mathworks Inc., Natick, MA) and the experiments were run on a computer with a Pentium III 866MHz processor. For the Pursuit Tracking task with $17min$ of training data, the Kalman model took $1.70s$ to learn while the Pinball task with approximately $3.5min$ of training data took $0.14s$ to train.

Decoding was performed at a rate of $0.15ms$ per $50ms$ time bin for the Pursuit Tracking task and $1.54ms$ per $70ms$ time bin for the Pinball task. This decoding cost is insignificant relative to these bin sizes. Note that decoding for the Pinball data took

Method	$CC\ (x, y)$	$MSE\ (cm^2)$
Kalman ($0ms$ lag)	(0.78, 0.91)	7.01
Kalman ($70ms$ lag)	(0.80, 0.93)	6.25
Kalman (140ms lag)	**(0.82, 0.93)**	**5.87**
Kalman ($210ms$ lag)	(0.81, 0.89)	6.80
Kalman ($280ms$ lag)	(0.76, 0.82)	8.81
Kalman (non-uni lag, init 1)	**(0.84, 0.93)**	**4.75**
Kalman (non-uni lag, init 2)	(0.84, 0.93)	4.77

Table 2: **Pinball Task:** Reconstruction results for the recursive Kalman filter with varied time lags: the upper portion shows uniform lags for all cells while the bottom portion shows non-uniform lags. In the case of uniform time lags, we found that $140ms$ provided the best description of the relationship between firing pattern and hand movement. Optimizing the individual lag for each cell, however, produced relatively more accurate decoding in terms of mean-squared error.

longer due to the fact that the kinematic model had more parameters (six rather than four as discussed below) and there were more neurons recorded for this task (42 versus 25).

3.3 Optimal Lag

Pinball Task. It is convenient to choose time lags corresponding to multiples of the bin size, although one could always re-bin the data and compute a finer discretization of the time lag. We used uniform time lags of 0, 70, 140, 210, 280 ms for the training and testing of the Kalman filter (see Table 2). From Table 2 (similar to Table 1 in (Wu et al., 2003)) we see that the optimal uniform lag was approximately two time bins (or $140ms$). Note that at each time step, the firing rate was the number of spikes over

the previous $70ms$ bin, therefore binning the spikes introduces a lag even at what we are calling 0 lag in the binned data. The same lag property also exists in the Pursuit Tracking task (with bin size $50ms$).

In general, we observed that individual neurons do not all have the same optimal lag though, in practice, the range of possible lags is bounded ($0 \leq i \leq 4$ or $0ms \leq$ lag $\leq 280ms$). To simplify our data analysis, we assume that the optimal lag for all cells is less than 4 time bins ($280ms$) and exploit the greedy algorithm described in Table 1 to approximate the optimal lag for each neuron.

The algorithm requires an initial guess of the lag for each cell. We experimented with multiple random initial conditions and found that the greedy search algorithm produced lags that were almost the same in all cases. The results for two experiments are shown in Fig. 5 (A). These two suboptimal time-lag solutions gave similar decoding results (see Table 2).

These results suggest that a non-uniform time lag is superior to a uniform one for modeling the relationships between firing patterns and hand movement. In the interest of simplicity however, and for the remainder of the paper, except where noted, we choose a fixed uniform time lag of $140ms$ with which we evaluate all other aspects of the Kalman model.

Pursuit Tracking Task. For the Pursuit Tracking task, the decoding accuracy for different uniform lag times is shown in Table 4 (column labeled "pos, vel, accel"); the accuracy is reported in terms of the average MSE and CC over 52 test trials. We observed that $150ms$ was approximately the optimal uniform lag, which is consistent

with the $140ms$ lag found for the Pinball task. (Recall that data for the Pursuit Tracking task was binned into $50ms$ time bins in contrast to the $70ms$ bins in the Pinball task.)

Here we found less variation in decoding results with respect to time lag. Consequently the optimal non-uniform lag (see Fig. 5 (B)) showed no improvement over the best uniform lag. We hypothesize that this is due to the much slower hand motions present in this task; the hand is moving slowly enough that the difference in kinematic state between various lags is less important. For the remainder of the paper, except where noted, we choose a fixed uniform time lag of $150ms$ for the Pursuit Tracking task.

3.4 Kinematic Model

We assumed above (by default) that position, velocity, and acceleration of the hand were related to neural firing rates. Other kinematic models with more or fewer terms could be employed. To understand the relationship between the order of the kinematic model and neural activity, we tested a variety of models including position alone (zeroth order), position and velocity (first order), position, velocity, and acceleration (second order), and so on up to fifth order. Note that velocity and acceleration correspond to first and second derivatives of position respectively. The next three higher-order derivatives are referred to as jerk, snap, and crackle.

Pinball Task. We trained and evaluated each of these models. To avoid overfitting, we learned each Kalman model using the training data set, then evaluated the decoding accuracy in test data set. The decoding results in Table 3 show that adding higher-order

terms increases the decoding accuracy but with diminishing returns. As the number of terms in the model increases, so does the need for training data and the risk of overfitting.

The optimal choice of kinematic model is an example of a model-selection problem. The Bayesian Information Criterion (BIC) (Rissanen, 1989) provides one approach to deal with the general case. Assume the hand kinematics $\mathbf{X}_M = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M]^T$ and the firing rates $\mathbf{Z}_M = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M]^T$ are known for the M time instants in the training set. Let

$$\alpha = -\log_{A,W,H,Q} p(\mathbf{X}_M, \mathbf{Z}_M), \qquad \text{and} \qquad \beta = \frac{\text{\# of parameters}}{2} \log M,$$

then from information theory α is the number of bits to describe the data and β is the number of bits to describe the model. BIC searches for minimal number of bits to describe the model and the data: i.e. the kinematic model which has the smallest $\alpha + \beta$. Table 3 shows that (position, velocity, acceleration, jerk) was the optimal kinematic model. In practice we have found a second-order model to be sufficient and to provide a good trade-off between accuracy and model complexity; this model is used in the experiments below (except where noted).

Pursuit Tracking Task. Table 4 shows the decoding accuracy for different kinematic models as well as different lags. For this data set we found that velocity is crucial while the acceleration and higher-order terms result in overfitting of the model and a reduction of accuracy. One hypothesis for the difference between the Pursuit Tracking and Pinball tasks with respect to the optimal kinematic model has to do with the particular type of motion present. In the Pursuit Tracking task, the range of accelerations was

# of orders	$CC\ (x, y)$	$MSE\ (cm^2)$	$\alpha + \beta$
0	(0.72, 0.87)	7.72	-80834
1	(0.82, 0.91)	6.31	-84049
2	**(0.82, 0.93)**	**5.87**	**-84411**
3	**(0.82, 0.93)**	**5.72**	**-84458**
4	(0.82, 0.93)	5.61	-84307
5	(0.82, 0.93)	5.60	-84081

Table 3: **Pinball Task:** Decoding and BIC results with respect to the order of the kinematic model. Order 0 corresponds to a system state containing just hand position. Order 1 combines position with the first derivative of position (i.e. velocity). Order 2 uses position, velocity, and acceleration. Order 3 adds the third derivative (jerk) while orders 4 and 5 add the additional derivatives (snap and crackle respectively). Experiments showed that the higher order system models result in more accurate decoding but with diminishing returns. The bold-face entries are optimal in terms of a BIC as well as for practical purposes.

much smaller than in the Pinball task. In particular, the magnitude of the acceleration was much smaller relative to the magnitude of the velocities (see Fig. 1). We observed increased firing rates as velocity and acceleration increased. The slow motions in the Pursuit Tracking task and the resulting low firing rates suggest that the effects of acceleration cannot be distinguished from noise. In all experiments below involving the Pursuit Tracking task, except where noted, we used a kinematic model containing only position and velocity.

Lag	position		position, velocity		pos, vel, accel	
	$CC\ (x,y)$	$MSE\ (cm^2)$	$CC\ (x,y)$	$MSE\ (cm^2)$	$CC\ (x,y)$	$MSE\ (cm^2)$
$0ms$	(0.43,0.39)	7.64	(0.79,0.68)	6.16	(0.78,0.65)	6.22
$50ms$	(0.46,0.40)	7.56	(0.79,0.68)	6.08	(0.78,0.66)	6.17
$100ms$	(0.49,0.40)	7.46	(0.79,0.68)	6.08	(0.79,0.65)	6.17
$150ms$	(0.51,0.41)	7.38	**(0.79,0.68)**	**5.99**	(0.79,0.64)	6.15
$200ms$	(0.53,0.41)	7.34	(0.79,0.67)	5.96	(0.78,0.64)	6.18
$250ms$	(0.54,0.40)	7.33	(0.78,0.67)	6.02	(0.78,0.64)	6.16
non-uni	(0,53,0.41)	7.14	(0.79,0.68)	6.00	(0.79,0.65)	6.19

Table 4: **Pursuit Tracking Task:** Reconstruction results for the recursive Kalman filter with varying time lag and different kinematic models. A uniform time lag of approximately $150ms$ was roughly optimal as was a kinematic model containing just position and velocity (see bold-face entry). Here there was no improvement when using a non-uniform lag.

3.5 Conditional Dependence of the Firing Rates

In the Methods section, we suggested that the firing rates of the neurons in MI are not independent conditioned on the hand kinematics and emphasized the importance of learning a full covariance matrix $\mathbf{Q}$. Conditional independence would imply that the likelihood could be written in terms of the probability of the firing rates, $z_{i,k}$, of the individual neurons, $i = 1 \ldots C$:

$$p(\mathbf{z}_k|\mathbf{x}_k) = \prod_{i=1}^{C} p(z_{i,k}|\mathbf{x}_k) = \prod_{i=1}^{C} \frac{1}{\sqrt{2\pi}q_i} \exp(-\frac{1}{2}(z_{i,k} - \mathbf{H}_i\mathbf{x}_k)^2/q_i^2)$$

where q_i^2 is the observation variance for cell i and $\mathbf{H}_i$ is the i^{th} row of $\mathbf{H}$. This model corresponds to having a diagonal covariance matrix $\mathbf{Q}$ with the q_i along the diagonal. Learning the parameters of the Kalman model proceeds as before but with the restriction that $\mathbf{Q}$ is diagonal.

	Pinball task		Pursuit Tracking task	
Method	$CC\ (x, y)$	$MSE\ (cm^2)$	$CC\ (x, y)$	$MSE\ (cm^2)$
full covariance	(0.82, 0.93)	5.87	(0.79, 0.68)	5.99
diagonal covariance	(0.82, 0.93)	6.91	(0.79, 0.67)	6.35

Table 5: Decoding accuracy for full and diagonal matrix for both Pinball and Pursuit Tracking tasks.

Table 5 shows the decoding results obtained under this conditional independence assumption. We found that the full covariance matrix provides a better probabilistic model of the neural data, and results in more accurate decoding. The importance of conditional dependence of the firing rates can be demonstrated using a t-test (Larsen and Marx, 2001). For each pair of cells, we obtained the p-value under the null hypothesis that the firing rates for this pair were conditionally independent. We found that, over all the pairs, 51% (Pinball) and 40% (Pursuit Tracking) of p-values were less than 0.05, therefore the null hypotheses of the independence of the corresponding pairs should be rejected. This alternative analysis also suggests the importance of representing a full covariance matrix $\mathbf{Q}$.

Note that estimating the full covariance matrix may be problematic for large populations of neurons. As the size of $\mathbf{Q}$ increases, more training data is needed to avoid overfitting. To cope with this problem one can use Principal Component Analysis (PCA) to reduce the dimensionality of the observation data. In the reduced-dimensional PCA space, the directions are decorrelated but, since the firing rates are not Gaussian, PCA does not make the directions conditionally independent. Consequently, it is still advantageous to fit a full covariance matrix; the advantage is that it can be fit to lower-dimensional data.

3.6 Number of Neurons

We explored the effect of varying the population size on decoding accuracy and, not surprisingly, found that larger populations result in more accurate decoding. For any $N = 1, \cdots, C$, ($C = 42$ in Pinball; $C = 25$ in Pursuit Tracking), we randomly selected subsets of N neurons from our population. We then fit the Kalman model to the training data, and reconstructed the test data. The random subsets were chosen 100 times and the decoding results were averaged (Fig. 6 (A)). We found that for both datasets the MSE and CC improved with increasing population size. Current recording technology is now able to produce 100 or more simultaneous recordings. The results here suggest that these larger populations may lead to decreases in MSE which, in turn, should make neural prosthetic control more accurate.

3.7 Amount of Training Data

We also explored the effect of the quantity of training data on decoding accuracy: larger amounts of training data resulted in more accurate decoding. Using the same test data, we selected different amounts of partial training data to learn the Kalman model. The decoding results are shown in Fig. 6 (B). We found that, in the Pinball task, approximately 2.5 minutes of training data was sufficient; while in the Pursuit Tracking task we used approximately 10 minutes of training data, although more data improved the results slightly.

Modeling Summary. A Bayesian framework such as the Kalman filter requires us to define a likelihood model and a prior model. The prior model here is given by the linear

Gaussian system model and we do not explore various generalizations of that model. Rather, the experiments above pertain to the likelihood and how it is formulated. While the standard Kalman framework constrains us to a linear Gaussian model, there are still many modeling choices necessary to match this framework to motor cortical activity. Based on the above experiments, a number of general observations can be made.

We observed that a full covariance matrix is critical for decoding accuracy. Time lags of roughly 150ms improved results and, while there was some benefit to non-uniform lag times, there was much to be gained from a simple uniform lag. The order of the kinematic model was dependent on the application motion but for general, unconstrained, and continuous motions it appeared that higher-order models (up to 4th order) were beneficial. Not surprisingly we also found that accuracy increases with the number of cells used. The Appendix also contains an analysis of the effect of varying the bin size. We found that larger time bins improved decoding accuracy but the optimal size depended on the task, with fast motions requiring smaller bins than slow motions.

In summary, the best decoding performance for the two tasks were obtained as follows:

Pinball, $CC = (0.84, 0.93)$, $MSE = 4.55$:

$140ms$ time bin, non-uniform lag, 2nd order model (pos, vel, accel).

Pursuit Tracking, $CC = (0.81, 0.70)$, $MSE = 4.66$:

$300ms$ time bin, $150ms$ uniform lag or non-uniform lag, 1st order model (pos, vel).

3.8 Comparison with Previous Decoding Methods

A variety of other methods have been proposed for performing motor cortical decoding. Below we compare the Kalman filter to the most popular methods, and then suggest further extensions of the Bayesian decoding framework.

3.8.1 Population-Vector Method

The population-vector approach (Georgopoulos et al., 1982; Georgopoulos et al., 1986) treats MI cells as having a "preferred" movement *direction* and models the hand movement direction as a weighted vector average of these preferred directions where the weight is proportional to the firing rate of the cell. The population-vector algorithm can be expressed in the following equation (Moran and Schwartz, 1999b):

$$PV_k = \sum_{i=1}^{C} \frac{z_{i,k} - \bar{z}_i}{z_{max_i} - \bar{z}_i} \cdot \frac{\vec{B}_i}{z_{max_i} - \bar{z}_i}, \tag{11}$$

where PV_k is the population vector at time $t_k = k\Delta t, k = 1, \cdots, M$, $z_{i,k}$ is the firing rate of cell i at time t_k, $\bar{z}_i$ and z_{max_i} are the average and maximum firing rates of cell i over all time steps, and $\vec{B}_i$ is the 2D unit-length vector pointing in the preferred direction of cell i, $i = 1, \cdots, C$. This formulation models only movement direction and not hand position and hence represents only a subset of the kinematic information modeled by the Kalman filter. To recover hand position one *integrates* the direction estimates over time and then normalizes the result to obtain an estimate of hand position (Schwartz, 1993).

The population-vector method has been used in the decoding of various hand movement tasks, including center-out reaching (Moran and Schwartz, 1999b), sinusoid trac-

ing (Schwartz, 1993), spiral tracing (Moran and Schwartz, 1999a) and figure-eight tracing (Schwartz and Moran, 1999). Recently, this approach has been applied to real-time neural control of 3D cursor (center-out) movement (Taylor et al., 2002).

In both the Pinball and Pursuit Tracking tasks, we learned the parameters of the population-vector model from training data; the preferred directions are shown in Fig. 7. We then estimated the hand position in test data using the population-vector method; the decoding results are shown in Table 6. We found that the population vector method does not accurately reconstruct the hand trajectory for these complex motions (particularly the more natural motions of the Pinball task).

The lack of accuracy may be due to at least two reasons. First, the approach assumes that the population uniformly samples the range of movement directions, yet for the small sample sizes available with current multi-electrode recording technology this may be difficult to satisfy (Gribble and Scott, 2002). For the data sets we considered, the movement directions were not uniformly distributed (Fig. 7). One of the key advantages of the full covariance matrix in the Kalman filter is that it accounts for the non-uniform distribution of encoding properties across the population. If two cells encode the same information, the errors in the likelihood term are correlated and the covariance matrix appropriately accounts for their conditional dependence; the linear filter method below does something similar. This weighting of the data is critical for sound inference. Another issue with the population-vector approach is that the integration of direction estimates to compute position information results in the compounding of errors over time.

Thus, the population-vector method may be appropriate for simple stylized motions

or motions of short duration, yet to model general, continuous, and complex motions, a more powerful model (richer kinematics and a probabilistic formulation) is preferable.

3.8.2 Linear-Filter Method

Linear filters reconstruct hand position as a linear combination of the firing rates over some fixed time period (Carmena et al., 2003; Serruya et al., 2002; Warland et al., 1997; Wessberg et al., 2000); that is,

$$x_k = a + \sum_{i=1}^{C} \sum_{j=0}^{N} z_{i,k-j} f_{i,j}, \tag{12}$$

where x_k is the x-position (or, equivalently, the y-position) at time $t_k = k\Delta t, k = 1, \cdots, M$, a is a constant offset, $z_{i,k-j}$ is the firing rate of neuron i at time t_{k-j}, and $f_{i,j}$ are the filter coefficients. The coefficients can be learned from training data using simple least-squares regression. The method has been used for off-line decoding of MI activity (Paninski et al., 2004) and for real-time neural control (Carmena et al., 2003; Serruya et al., 2002; Wessberg et al., 2000).

The parameter N specifies the number of time bins used to estimate the hand position. Empirically, N is chosen so that the total data used is between $500ms$ and $2s$. In (Serruya et al., 2003), the authors stated that $N = 30$ is the optimal choice because "shorter filters did not perform as well and that longer filters did not provide much additional information". A larger N results in a large computational cost in estimating the filters (by inverting a large matrix), which becomes prohibitive when the number of cells is large. It also requires more training data to avoid overfitting. Consequently, we took $N = 30$ to be the maximum number of time bins.

Pinball task		
Method	$CC\ (x, y)$	$MSE\ (cm^2)$
population vector	(0.26, 0.21)	75.0
linear filter $(N = 14)$	(0.79, 0.93)	6.48
Kalman $\Delta t = 140ms$, non-uniform lag	**(0.84, 0.93)**	**4.55**
Pursuit Tracking task		
Method	$CC\ (x, y)$	$MSE\ (cm^2)$
population vector	(0.57, 0.43)	13.2
linear filter $(N = 30)$	(0.73, 0.67)	4.74
Kalman $\Delta t = 300ms$**,** $150ms$ **uniform lag**	**(0.81, 0.70)**	**4.66**

Table 6: Decoding results with different methods for both the Pinball and the Pursuit Tracking tasks.

To make the linear filter effective and compare it to the Kalman filter, we selected the N that gave the best decoding results, where we allowed N to range over $\{1, 2, \cdots, 30\}$ time bins. For the Pinball task, $N = 14$ (approximately $1sec$) gave the best decoding accuracy, while in Pursuit Tracking task, $N = 30$ (approximately $1.5sec$) was optimal (although there is little difference in the results for $N \geq 27$).

A $1sec$ worth (pinball) and $1.5sec$ worth (pursuit tracking) of firing-rate information before hand kinematics is also used in the method described in (Serruya et al., 2002; Serruya et al., 2003). Note that since the linear filter uses data over a long time window it does not benefit from the use of time-lagged data. Note also that it does not explicitly reconstruct velocity or acceleration. One could compute velocity with the linear filter (Carmena et al., 2003) but to estimate position one would need a way to combine linear-filter estimates for position and velocity; the Kalman filter automatically does this within an optimal Bayesian framework.

The linear-filter reconstruction of position for the Pinball data is shown in Fig. 8.

The results are visually similar to the Kalman results (Compare Fig. 4), yet Table 6 shows that the Kalman filter gives a more accurate reconstruction (higher CC and lower MSE). Figure 9 shows the linear-filter reconstruction for the same four trials as in the Pursuit Tracking task shown in Fig. 3 (B).

The linear filter is a discrete Wiener filter (Gelb, 1974) in the sense that, at each time instant, the hand position is a linear function of the firing rates of all the cells over the past N bins; the Appendix explores this relationship formally.

The linear-filter approach lacks an explicit generative model of neural activity and hence provides little insight into neural coding (apart from suggesting a linear relationship between hand position and firing rate). Both the linear filter and population vector lack an explicit prior model for how the system state evolves. This results in noisy estimates of the state and necessitates the use of large time windows for the linear filter. In practice, post hoc smoothing is often used to reduce the noise in the linear filter estimates yet this introduces undesirable lags in the reconstruction. The Kalman filter uses much less data at each time instant than the linear filter, but has an explicit temporal model that incorporates prior estimates in a recursive fashion. In particular, the probabilistic model (first order Markov) underlying the Kalman filter means that only a single bin of data is used at each time instant and these bins should never overlap in time.

3.8.3 Artificial Neural Networks

Artificial neural networks (ANNs) have also been used for neural decoding (Ghazanfar et al., 2000; Sanchez et al., 2003) and real-time prediction of hand trajectories from

neural ensembles in MI (Wessberg et al., 2000). ANNs can perform many types of statistical learning yet they do not provide explicit generative or probabilistic models that are open to inspection. The models can, however, be analyzed to varying degrees to try to tease out what they encode (Sanchez et al., 2003). Wide variability in the specific implementation and training of these methods makes it impractical to quantitatively compare our results with previous ANN implementations.

4 Discussion

Our focus in this paper has been on the development of a decoding algorithm appropriate for neural prosthetic applications. The Kalman filter is computationally efficient (real-time), requires little training data, and provides more accurate off-line decoding than previous methods such as population vectors or linear filtering. While the linear Gaussian model of neural coding is an approximation, it provides a reasonable trade-off between computational efficiency and accuracy. Its successful use in decoding may be thought of as resulting from its achieving a good bias/variance trade-off (Geman et al., 1992); equivalently, it provides a reasonable solution to the model-selection problem, given the complexity of the task and the size of the training data sets at hand. More powerful non-linear, and non-Gaussian, likelihood models can be constructed (Brockwell et al., 2004; Gao et al., 2002; Gao et al., 2003) and used for Bayesian decoding; the decoding task, however, becomes more difficult. For example, the likelihood can be formulated using Generalized Linear Models (Gao et al., 2003), Generalized Additive Models (Gao et al., 2003), mixture models (Wu et al., 2004a) and fully non-parametric models (Gao et al., 2002).

In (Gao et al., 2002) we introduced particle filtering to solve the general Bayesian decoding task in MI with non-linear, non-Gaussian, likelihoods (see also (Brockwell et al., 2004; Shoham, 2001)). The particle filtering method is more general than the Kalman filter proposed here. With sufficient computational resources, linear Gaussian decoding using a particle filter approaches the accuracy of the Kalman filter. Additionally, more complex likelihood models can give higher accuracy than the linear-Gaussian model (Gao et al., 2003). Current implementations of particle filtering however do not run in real-time and, hence, are not yet appropriate for neural prosthetic applications.

The method proposed here assumes that firing rates can be approximated by a linear model and that action potentials from multiple cells recorded by a single electrode can be cleanly separated (sorted). In practice however, automated spike sorting methods may incorrectly classify multiple waveforms as being produced by the same cell. This can reduce decoding accuracy since it violates the model assumptions. An extension of the Kalman filter reformulates the likelihood as a probabilistic mixture of linear Gaussian models (Wu et al., 2004a) in order to cope, to some extent, with errors of this type. An efficient algorithm known as the Switching Kalman filter (Murphy, 1998) can be used for decoding with this mixture model. Wood *et al.* (Wood et al., 2004) suggest this sorting may not be a signficant problem for neural prosthesis applications and that good decoding accuracy can be obtained using a Kalman filter with unsorted, multi-unit, data.

Our formulation also assumes that motor cortex codes movement in terms of firing rates. In the Bayesian formulation this assumption too can be relaxed and a point process model, taking into account spike timing, can be used for decoding (Brown

et al., 1998; Eden et al., 2004). The resulting decoding algorithm is more complex however than the simple Kalman filter presented here.

Here we focused on the likelihood (measurement model) as derived from studies of neural coding. To evaluate different choices for the likelihood we used a simple linear Gaussian temporal prior (system model). More complex, non-linear, models or higher-order Markov assumptions might lead to more accurate models of arm motion and more accurate decoding results.

The linear filter and Kalman filter both produce reasonable decodings of neural activity. Depending on the specifics of the model, the Kalman filter can be viewed as a linear filter (Weiner filter) in which all measurements back to the initial time step are taken into account yet the contribution of measurements decays exponentially with time. The Kalman filter however provides a number of benefits. It is easier to train (less computationally intensive) and, more importantly, formulates the problem as one of Bayesian inference. Since assumptions about the data and the noise are explicitly spelled out, the Kalman model provides more insight into the encoding model that is used. Moreover, by making the assumptions explicit, it suggests avenues for improving both modeling and decoding, by relaxing some of these assumptions.

We have studied off-line decoding accuracy rather than on-line control. It is reasonable to expect that algorithms that provide better off-line decoding will provide better on-line accuracy as well. One might also posit that better control algorithms may make the training of paralyzed human subjects easier. Results of on-line control studies, however, suggest that even algorithms such as the population-vector method, with its poor off-line accuracy, can be used to control devices (Taylor et al., 2002). It may be the case

that the brain adapts to a particular control algorithm and that improving the algorithm produces little gain in accuracy. Such a hypothesis remains to be tested.

Research on decoding for neural control has focused on cells in motor cortex. The rationale underlying this choice is that it will be easier for subjects to learn to control physical devices when this area of cortex is used, since it already represents information about motion. Even in paralyzed subjects imagined arm motions produce activity in MI, suggesting its appropriateness for human neural prostheses (Kennedy and Bakay, 1998). Other brain areas, however, may provide useful control signals for motor prosthetic applications. The applicability of the Kalman filter outside MI remains to be tested.

The results reported here involved only two animals and each animal was trained on, and performed, a slightly different task. More studies will be required to generalize these results to a broader range of tasks and conditions.

4.1 Conclusions

We have described a discrete linear Kalman filter that is appropriate for the neural control of 2D cursor motion. We showed that the model could be easily learned using a few minutes of training data, and provided real-time estimates of hand position given the firing rates of a population of cells in primary motor cortex. The estimated trajectories were more accurate than those produced by the population-vector and linear-filtering methods most commonly used in the literature. Analysis of the method was performed using two datasets involving complex, continuous, hand motions. Neural recordings from two different monkeys were obtained from chronically implanted microelectrode

arrays. The experiments suggested that a linear Gaussian model provided an approximate model relating the firing rates with continuous hand movement. Moreover, the model enabled the use of the Kalman filter to perform real-time recursive Bayesian decoding.

The Kalman filter proposed here provides a rigorous probabilistic approach with a well understood theory. By making the assumptions explicit and by providing an estimate of uncertainty, the Kalman filter offers significant advantages over previous methods. Unlike previous methods, this model estimates a full kinematic state vector (position, velocity and higher-order terms). It also provides an estimate of the uncertainty in the result in terms of an error-covariance matrix. Various experiments explored the use of this decoding model to choose the kinematic variables that gave the best encoding/decoding performance.

In contrast to the relatively constrained Pursuit Tracking task, we showed that, for the more natural 2D motions of the Pinball task, incorporating acceleration and higher-order terms into the model improved the accuracy of the decoding. The experiments suggest a number of general conclusions, which remain to be verified in more subjects and in on-line experiments: 1. A time lag of $140 - 150ms$ between firing activity and hand kinematics improves decoding results. 2. As the number of neurons in the population increases, one observes a corresponding increase in decoding accuracy. While even small populations produce reasonable results, the ability, in the not too distant future, to record from much larger populations is likely to increase accuracy even further. 3. A few minutes of training data is sufficient for learning the linear Gaussian model. The amount of data required may depend on the task. 4. With the Gaussian model, a

full covariance matrix usefully captures the conditional dependence between the firing rates of different neurons. Decoding accuracy is improved by using a full covariance matrix as compared to a diagonal matrix. 5. Firing rates in larger bin sizes, which are better approximated by a Gaussian model, improve the decoding accuracy up to a task-specific limit.

Currently, we are working to evaluate the performance of the Kalman filter for the closed-loop neural control of cursor motion in tasks such as described here. Our recent work demonstrates the feasibility of Kalman filter in the on-line neural control experiment (Wu et al., 2004b). More experiments with more animals, however, are needed to confirm those observations. Our future work will focus on the application of automated spike sorting methods that provide an estimate of uncertainty in the rate signal. This uncertainty can be naturally incorporated into the Kalman model by allowing an adaptive measurement covariance matrix $\mathbf{Q}_k$. Additionally, one may explore alternative measurement noise models, non-linear system models, and non-linear decoding methods.

Appendix

A. Learning the Linear Gaussian Model

Training the Kalman model requires that we learn the matrices $\mathbf{A}, \mathbf{H}, \mathbf{W}, \mathbf{Q}$ from example data. We seek the coefficient matrices and covariance matrices that maximize

the joint probability $p(\mathbf{X}_M, \mathbf{Z}_M)$; that is,

$$\begin{aligned} \text{argmax}_{A,W,H,Q} p(\mathbf{X}_M, \mathbf{Z}_M) &= \text{argmax}_{A,W,H,Q} p(\mathbf{X}_M) p(\mathbf{Z}_M | \mathbf{X}_M) \\ &= \{\text{argmax}_{A,W} p(\mathbf{X}_M), \text{argmax}_{H,Q} p(\mathbf{Z}_M | \mathbf{X}_M)\}. \end{aligned}$$

Using the linear Gaussian properties of $p(\mathbf{X}_M)$ and $p(\mathbf{Z}_M | \mathbf{X}_M)$, the above maxmization has closed-form solutions:

$$\begin{aligned} \mathbf{A} &= \left(\sum_{k=2}^{M} \mathbf{x}_k \mathbf{x}_{k-1}^T \right) \left(\sum_{k=2}^{M} \mathbf{x}_{k-1} \mathbf{x}_{k-1}^T \right)^{-1}, \\ \mathbf{W} &= \frac{1}{M-1} \left(\sum_{k=2}^{M} \mathbf{x}_k \mathbf{x}_k^T - \mathbf{A} \sum_{k=2}^{M} \mathbf{x}_{k-1} \mathbf{x}_k^T \right), \\ \mathbf{H} &= \left(\sum_{k=1}^{M} \mathbf{z}_k \mathbf{x}_k^T \right) \left(\sum_{k=1}^{M} \mathbf{x}_k \mathbf{x}_k^T \right)^{-1}, \\ \mathbf{Q} &= \frac{1}{M} \left(\sum_{k=1}^{M} \mathbf{z}_k \mathbf{z}_k^T - \mathbf{H} \sum_{k=1}^{M} \mathbf{x}_k \mathbf{z}_k^T \right). \end{aligned}$$

B. Decoding (the Kalman-filter algorithm)

The assumptions of linear Gaussian models for the system and measurement equations allows us to exploit the Kalman filter algorithm for recursive Bayesian inference. The theory is well developed (Kalman, 1960) and the algorithm is summarized here. The algorithm comprises two steps:

I. Time update equations: At each time t_k, we use the system model to take the *a priori* state estimate, $\hat{\mathbf{x}}_{k-1}$, from the previous time t_{k-1}, and predict it forward to time

t_k:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1}. \tag{13}$$

Recall that the uncertainty in the system model is expressed in the covariance matrix $\mathbf{W}$ and this uncertainty must be incorporated into our *a priori* estimate of the posterior covariance:

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{W}, \tag{14}$$

where $\mathbf{P}_k^-$ is the *a priori* error covariance matrix at time t_k.

II. Measurement update equations: Using the prediction $\hat{\mathbf{x}}_k^-$ and firing rate $\mathbf{z}_k$, we update the state estimate using the measurement and compute the posterior error covariance matrix:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-), \tag{15}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^- , \tag{16}$$

where $\mathbf{P}_k$ represents the state error covariance after taking into account the neural data, and $\mathbf{K}_k$ is the Kalman *gain* matrix, given by:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{Q})^{-1}. \tag{17}$$

This $\mathbf{K}_k$ produces a state estimate that minimizes the mean-squared error of the reconstruction (see (Gelb, 1974) for details).

C. Comparison of the Kalman Filter and the Wiener Filter

Simple linear filtering methods that directly reconstruct the system state from a history of firing rates are commonly used for decoding. These methods compute the state as a linear combination of previous firing rates. In particular, the Wiener filter is an optimal linear filter that uses all previous firing data. The simplified Kalman filter developed here can be viewed as an efficient, recursive, version of the Wiener filter in which the modeling assumptions ($\mathbf{A}, \mathbf{H}, \mathbf{W}, \mathbf{Q}$) are made explicit.

We can obtain several basic properties of Kalman filter by studying its relationship with the Wiener filter. From equations (13) and (15), we have that

$$\begin{aligned}
\hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-) \\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\hat{\mathbf{x}}_k^- + \mathbf{K}_k\mathbf{z}_k \\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{K}_k\mathbf{z}_k .
\end{aligned}$$

Note that equations (14), (16) and (17) are independent of the new measurements of firing rates. If $\mathbf{A}, \mathbf{H}, \mathbf{W}, \mathbf{Q}$ are constant over time, the propagation of $\mathbf{P}_k^-, \mathbf{P}_k, \mathbf{K}_k$ by equation (14), (16), (17) converges to steady-state solutions (Kalman and Bucy, 1961). Let $\mathbf{K} = \lim_{k\to\infty} \mathbf{K}_k$, and $\mathbf{M} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{A}$. For k large enough (i.e. $k > 20$ in practice),

$$\begin{aligned}
\hat{\mathbf{x}}_k &\approx \mathbf{M}\hat{\mathbf{x}}_{k-1} + \mathbf{K}\mathbf{z}_k \\
&\approx \mathbf{M}^2\hat{\mathbf{x}}_{k-2} + \mathbf{M}\mathbf{K}\mathbf{z}_{k-1} + \mathbf{K}\mathbf{z}_k \\
&\approx \cdots \\
&\approx \mathbf{M}^{k-1}\hat{\mathbf{x}}_1 + \sum_{j=0}^{k-2} \mathbf{M}^j\mathbf{K}\mathbf{z}_{k-j} .
\end{aligned}$$

where $\mathbf{M}^j$ is the jth power of matrix $\mathbf{M}$ (i.e. $\mathbf{M}^2 = \mathbf{M} \cdot \mathbf{M}$).

The above equation shows that in the Kalman framework, the estimate at time step k is a linear function of the firing rates at all time instants from time t_2 to the present. This corresponds to the Wiener filter (Gelb, 1974) but the advantage of the Kalman implementation is that it computes the state estimate recursively, resulting in an efficient algorithm.

Note also that the coefficients of all firing rates decay exponentially with respect to the current time step. This shows three basic properties of the Kalman filter: (1) it estimates the state at each time step using all the previous and present measurements (firing rates); (2) the weights of the firing rates decay exponentially; those far from the present time have a weak effect on the state estimate; (3) for $k >> 1$, the state estimate is approximately independent of the initial state. This last point means that the choice of the initial state is relatively unimportant.

D. Effect of Bin Size on Decoding Accuracy

We studied the effect of varying the bin size on decoding accuracy and found that accuracy was improved by increasing the bin size beyond the $70ms$ and $50ms$ bins used in the analysis above. Table 7 and Fig. 10 (A) summarize the results. For the Pinball task, we varied the bin size, Δt, by multiples of $70ms$ from 0 to $700ms$. For the Pursuit Tracking task we considered bins ranging from 0 to $700ms$ in $50ms$ increments. In all cases we used non-overlapping time bins, as the use of overlapping bins results in a severe violation of the assumption of conditional independence underlying by the Kalman framework (Equation (7)): with overlapping bins, data is "reused", and the

Pinball task			Pursuit Tracking task		
Method	$CC\ (x, y)$	$MSE\ (cm^2)$	Method	$CC\ (x, y)$	$MSE\ (cm^2)$
$\Delta t = 70ms$	(0.82, 0.93)	5.87	$\Delta t = 50ms$	(0.79, 0.68)	5.99
$\Delta t = 140ms$	(0.83, 0.93)	5.29	$\Delta t = 100ms$	(0.80, 0.68)	5.63
$\Delta t = 210ms$	(0.81, 0.92)	5.45	$\Delta t = 150ms$	(0.81, 0.68)	5.27
$\Delta t = 280ms$	**(0.82, 0.93)**	**5.16**	$\Delta t = 200ms$	(0.81, 0.69)	5.03
$\Delta t = 350ms$	(0.78, 0.92)	5.95	$\Delta t = 250ms$	(0.81, 0.69)	4.89
$\Delta t = 420ms$	(0.78, 0.92)	5.81	$\Delta t = 300ms$	**(0.81, 0.70)**	**4.66**
$\Delta t = 490ms$	(0.75, 0.89)	7.25	$\Delta t = 400ms$	(0.79, 0.70)	4.59
			$\Delta t = 500ms$	(0.79, 0.71)	4.64

Table 7: Decoding results for varying bin sizes. The other model parameters are as described above, namely: Pinball task: uniform lag = $140ms$, kinematic model = (pos, vel, accel); Pursuit Tracking task: uniform lag = $150ms$, kinematic model = (pos, vel).

resulting estimates are no longer statistically valid.

For each test condition (bin size), the Kalman model was trained and hand kinematics were calculated every $\Delta t\, ms$. Table 7 shows that larger bins resulted in better decoding accuracy, up to a limit. One reason for this may be that, as the bin size grows, the variation in the firing rate is better approximated by the Gaussian model used in the Kalman filter.

In the case of the slow motions of the Pursuit Tracking task, larger bin sizes were appropriate. For the fast motions of the Pinball task, bin sizes beyond approximately $280ms$ resulted in a loss of accuracy. This suggests that, while larger bin sizes can increase accuracy, the ultimate size is limited and is related to the speed of motion.

Additionally, increased bin size had a negative effect on the detail of the recovered trajectories (Fig. 10 (B)). As bin size increases, the frequency of state estimates decreases, resulting in a coarser approximation to the underlying trajectory. In gen-

eral, larger bin sizes (up to some limits) produce more accurate results but at the cost of introducing a delay in estimating the system state. The constraints of a particular application will determine the appropriate bin size.

Note that if a uniform lag of, for example, $140ms$ time bins is used, we can exploit measurement data binned into $140ms$ time bins without introducing any delay (or output lag) in the estimate of the system state relative to the natural hand motion. For real-time prosthesis applications, this system delay should be less than $200ms$ which suggests that overall bin size minus the uniform lag time should be less than $200ms$. For the Pinball task, with a $140ms$ lag, this would mean a maximum bin size of approximately $280ms$. For the Pursuit Tracking task, with a $150ms$ lag, a maximum bin size of 250-300ms would be appropriate. While this increases accuracy, it comes at the cost of a more "jerky" reconstruction.

References

Brockwell, A. E., Rojas, A. L., and Kass, R. E. (2004). Recursive bayesian decoding of motor cortical signals by particle filtering. *Journal of Neurophysiology, 91*, 1899–1907.

Brown, E., Frank, L., Tang, D., Quirk, M., and Wilson, M. (1998). A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *Journal of Neuroscience, 18*, 7411–7425.

Carmena, J. M., Lebedev, M. A., Crist, R. E., O'Doherty, J. E., Santucci, D. M., Dim-

itrov, D. F., Patil, P. G., Henriquez, C. S., and Nicolelis, M. A. L. (2003). Learning to control a brain-machine interface for reaching and grasping by primates. *PLoS, Biology*, *1*, 001–016.

Cisek, P., and Kalaska, J. F. (2002). Simultaneous encoding of multiple potential reach directions in dorsal premotor cortex. *Journal of Neurophysiology*, *87*, 1149–1154.

Donoghue, J. P., Sanes, J. N., Hatsopoulos, N. G., and Gaal, G. (1998). Neural discharge and local field potential oscillations in primate motor cortex during voluntary movements. *Journal of Neurophysiology*, *79*, 159–173.

Eden, U., Frank, L., Barbieri, R., Solo, V., and Brown, E. (2004). Dynamic analysis of neural encoding by point process adaptive filtering. *Neural Computation*, *16*, 971–988.

Fetz, E., Toyama, K., and Smith, W. (1991). Synaptic interaction between cortical neurons. In A. Peters and E. Jones (Eds.), *Cerebral cortex*, vol. 9, 1–47. New York: Plenum.

Flament, D., and Hore, J. (1988). Relations of motor cortex neural discharge to kinematics of passive and active elbow movements in the monkey. *Journal of Neurophysiology*, *60*, 1268–1284.

Gao, Y., Black, M. J., Bienenstock, E., Shoham, S., and Donoghue, J. P. (2002). Probabilistic inference of hand motion from neural activity in motor cortex. *Advances in Neural Information Processing Systems 14* (pp. 213–220). Cambridge, MA: MIT Press.

Gao, Y., Black, M. J., Bienenstock, E., Wu, W., and Donoghue, J. P. (2003). A quantitative comparison of linear and non-linear models of motor cortical activity for the encoding and decoding of arm motions. *1st International IEEE/EMBS Conference on Neural Engineering* (pp. 189–192). Capri, Italy.

Gelb, A. (1974). *Applied optimal estimation*. MIT Press.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation, 4*, 1–58.

Georgopoulos, A., Kalaska, J., Caminiti, R., and Massey, J. (1982). On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neuroscience, 8*, 1527–1537.

Georgopoulos, A., Schwartz, A., and Kettner, R. (1986). Neural population coding of movement direction. *Science, 233*, 1416–1419.

Ghazanfar, A., Stambaugh, C., and Nicolelis, M. (2000). Encoding of tactile stimulus location by somatosensory thalamocortical ensembles. *Journal of Neuroscience, 20*, 3761–3775.

Gribble, P. L., and Scott, S. H. (2002). Method for assessing directional characteristics of non-uniformly sampled neural activity. *Journal of Neuroscience Methods, 113*, 187–197.

Hatsopoulos, N., Ojakangas, C., Paninski, L., and Donoghue, J. (1998). Information about movement direction obtained from synchronous activity of motor cortical neurons. *Proceedings of the National Academy of Sciences* (pp. 15706—15711).

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering, 82*, 35–45.

Kalman, R. E., and Bucy, R. (1961). New results in linear filtering and prediction. *Trans. ASME, Journal of Basic Engineering, 83*, 95–108.

Kennedy, P. R., and Bakay, R. A. (1998). Restoration of neural output from a paralyzed patient by a direct brain connection. *NeuroReport, 9*, 1707–1711.

Kettner, R., Schwartz, A., and Georgopoulos, A. (1988). Primary motor cortex and free arm movements to visual targets in three-dimensional space. iii. positional gradients and population coding of movement direction from various movement origins. *Journal of Neuroscience, 8*, 2938–2947.

Larsen, R. J., and Marx, M. L. (2001). *An introduction to mathematical statistics and its applications*. Prentice Hall. Third edition.

Maynard, E., Hatsopoulos, N. G., Ojakangas, C. L., Acuna, B. D., Sanes, J. N., Normann, R. A., and Donoghue, J. P. (1999). Neuronal interactions improve cortical population coding of movement directions. *Journal of Neuroscience, 19*, 8083–8093.

Maynard, E., Nordhausen, C., and Normann, R. (1997). The Utah intracortical electrode array: A recording structure for potential brain-computer interfaces. *Electroencephalography and Clinical Neurophysiology, 102*, 228–239.

Moran, D., and Schwartz, A. (1999a). Motor cortical activity during drawing movements: Population representation during spiral tracing. *Journal of Neurophysiology, 82*, 2693–2704.

Moran, D., and Schwartz, A. (1999b). Motor cortical representation of speed and direction during reaching. *Journal of Neurophysiology*, *82*, 2676–2692.

Murphy, K. P. (1998). *Switching Kalman filter* (Technical Report 98-10). Compaq Cambridge Research Laboratory.

Paninski, L., Fellows, M., Hatsopoulos, N., and Donoghue, J. P. (2004). Spatiotemporal tuning of motor cortical neurons for hand position and velocity. *Journal of Neurophysiology*, *91*, 515–532.

Rissanen, J. (1989). *Stochastic complexity in statistical inquiry*. World Scientific.

Sanchez, J., Erdogmus, D., Principe, J., Wessberg, J., and Nicolelis, M. (2002). Comparison between nonlinear mappings and linear state estimation to model the relation from motor cortical neuronal firing to hand movements. *Proceedings of SAB Workshop on Motor Control in Humans and Robots: On the Interplay of Real Brains and Artificial Devices* (pp. 59–65). Edinburgh, Scotland.

Sanchez, J., Erdogmus, D., Rao, Y., Principe, J., Nicolelis, M., and Wessberg, J. (2003). Learning the contributions of motor, premotor, and posterior parietal cortices for hand trajectory reconstruction in a brain machine interface. *Proceedings of the 1st international IEEE EMBS Conference on Neural Engineering*.

Schwartz, A. (1993). Motor cortical activity during drawing movements: Population representation during sinusoid tracing. *Journal of Neurophysiology*, *70*, 28–36.

Schwartz, A., and Moran, D. (1999). Motor cortical activity during drawing move-

ments: Population representation during lemniscate tracing. *Journal of Neurophysiology*, *82*, 2705–2718.

Schwartz, A., Taylor, D., and Helms Tillery, S. (2001). Extraction algorithms for cortical control of arm prosthetics. *Current Opinion in Neurobiology*, *11*, 701–707.

Serruya, M., Hatsopoulos, N., Fellows, M., Paninski, L., and Donoghue, J. (2003). Robustness of neuroprosthetic decoding algorithms. *Biological Cybernetics*, *88*, 219–228.

Serruya, M. D., Hatsopoulos, N. G., Paninski, L., Fellows, M. R., and Donoghue, J. P. (2002). Brain-machine interface: Instant neural control of a movement signal. *Nature*, *416*, 141–142.

Shoham, S. (2001). *Advances towards an implantable motor cortical interface*. Doctoral dissertation, University of Utah.

Taylor, D., Helms Tillery, S., and Schwartz, A. (2002). Direct cortical control of 3D neuroprosthetic devices. *Science*, *296*, 1829–1832.

Twum-Danso, N., and Brockett, R. (2001). Trajectory estimation from place cell data. *Neural Networks*, *14*, 835–844.

Warland, D., Reinagel, P., and Meister, M. (1997). Decoding visual information from a population of retinal ganglion cells. *Journal of Neurophysiology*, *78*, 2336–2350.

Welch, G., and Bishop, G. (2001). *An introduction to the Kalman filter* (Technical Report 95–041). University of North Carolina at Chapel Hill.

Wessberg, J., Stambaugh, C., Kralik, J., Beck, P., L. M., Chapin, J., Kim, J., Biggs, S., Srinivasan, M., and Nicolelis, M. (2000). Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature, 408*, 361–365.

Wood, F., Fellows, M., Donoghue, J. P., and Black, M. J. (2004). Automatic spike sorting for neural decoding. *Proc. the 26th Annual Internaltional Conference of the IEEE EMBS* (pp. 4009–4012). San Francisco, CA.

Wu, W., Black, M. J., Gao, Y., Bienenstock, E., Serruya, M., and Donoghue, J. P. (2002). Inferring hand motion from multi-cell recordings in motor cortex using a Kalman filter. *SAB'02-Workshop on Motor Control in Humans and Robots: On the Interplay of Real Brains and Artificial Devices* (pp. 66–73).

Wu, W., Black, M. J., Gao, Y., Bienenstock, E., Serruya, M., Shaikhouni, A., and Donoghue, J. P. (2003). Neural decoding of cursor motion using a Kalman filter. *Advances in Neural Information Processing Systems 15* (pp. 133–140). MIT Press.

Wu, W., Black, M. J., Mumford, D., Gao, Y., Bienenstock, E., and Donoghue, J. P. (2004a). Modeling and decoding motor cortical activity using a switching Kalman filter. *IEEE Transactions on Biomedical Engineering, 51*, 933–942.

Wu, W., Shaikhouni, A., Donoghue, J. P., and Black, M. J. (2004b). Closed-loop neural control of cursor motion using a Kalman filter. *Proc. IEEE Engineering in Medicine and Biology Society* (pp. 4126–4129). San Francisco, CA.

Zhang, K., Ginzburg, I., McNaughton, B., and Sejnowski, T. (1998). Interpreting neu-

ronal population activity by reconstruction: Unified framework with application to hippocampal place cells. *Journal of Neurophysiology*, *79*, 1017–1044.

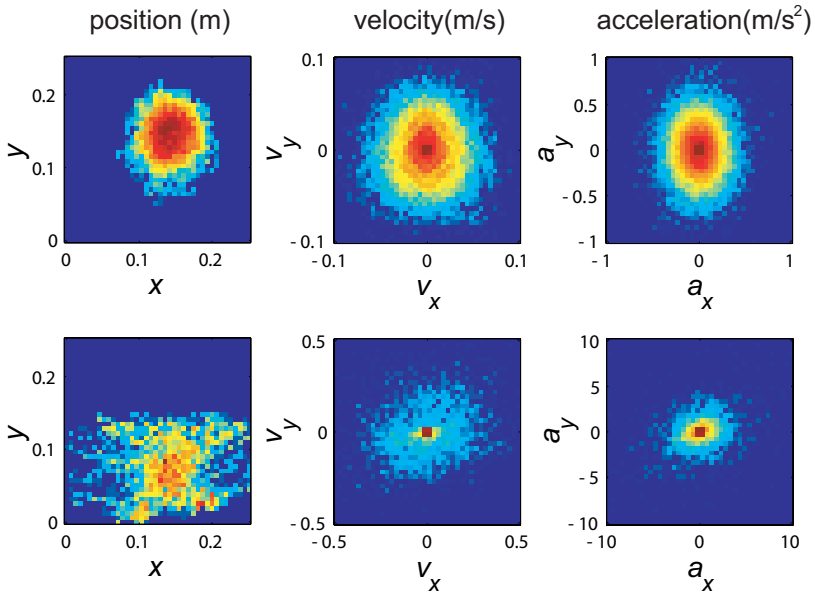

Figure 1: Normalized, log 2D-histograms of kinematics for both the Pursuit Tracking task (first row) and the Pinball task (second row). These plots show the log prior probability of the position, velocity and acceleration of the hand.

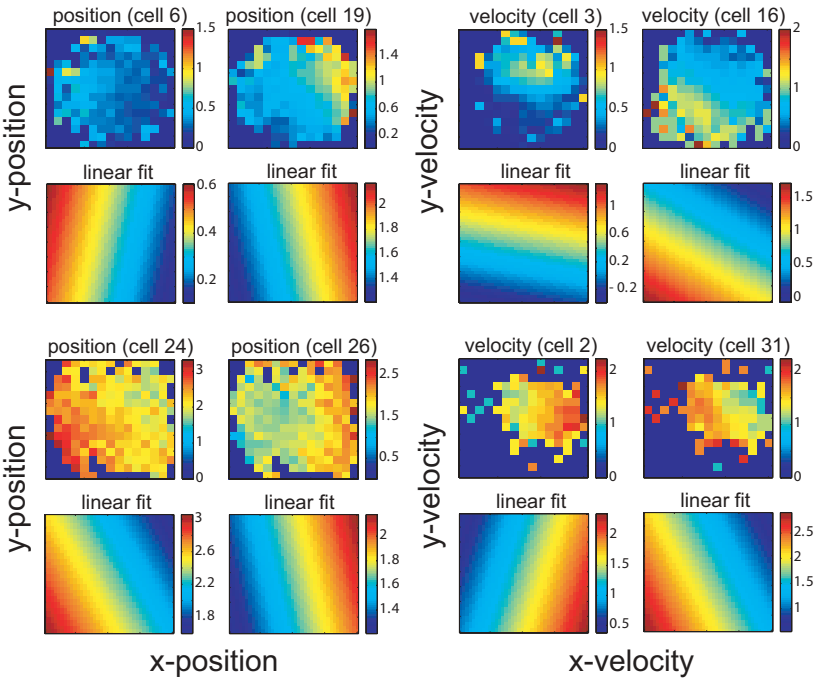

Figure 2: Empirical tuning functions and their linear approximation. The top two rows show data and models from the Pursuit Tracking task while the bottom two rows illustrate the Pinball data. For each task, the first row shows empirical firing rates and the second row shows the linear fit to this data. The empirical plots show the mean firing rate as a function of position (left two columns) or velocity (right two columns). The darkest blue areas correspond to kinematics that were never observed. The color coding (blue through red) represents the mean firing rate observed for a discrete region in the parameter space. The linear fit to this data is computed during the training of the Kalman-filter model. The linear fit is seen to provide a crude but reasonable approximation to the raw data.

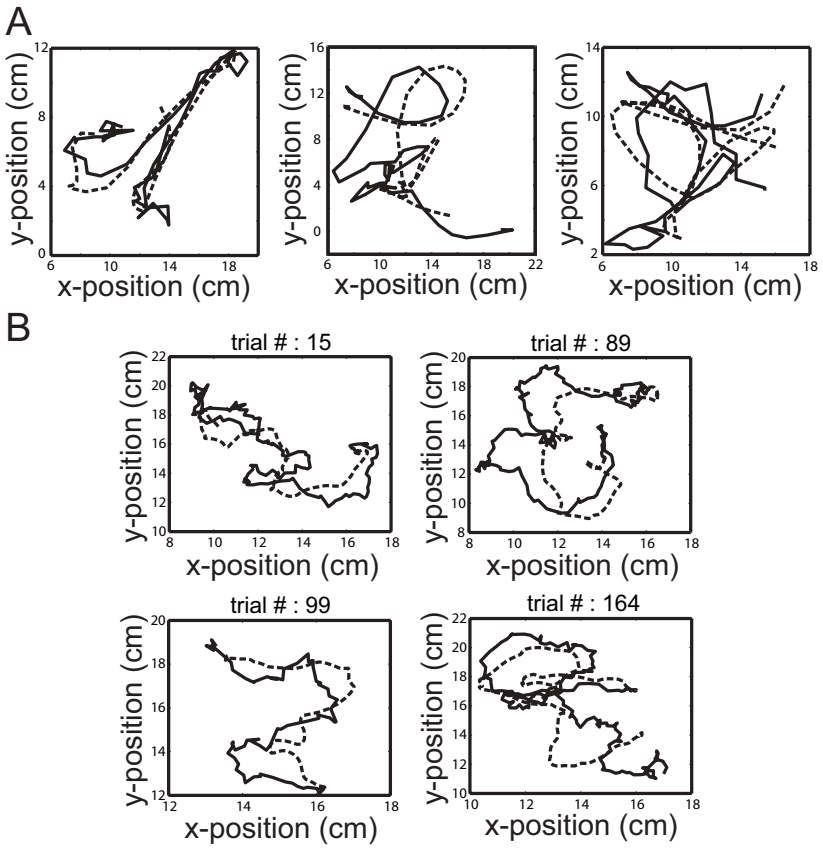

Figure 3: (A) **Pinball Task:** Reconstructed trajectories (portions of 1*min* test data – each plot shows 50 time instants ($3.5s$)): true target trajectory (dashed) and reconstruction using the Kalman filter (solid). (B) **Pursuit Tracking Task:** Reconstruction of hand position on a few test trials: true hand trajectory (dashed) and reconstruction using the Kalman filter (solid).

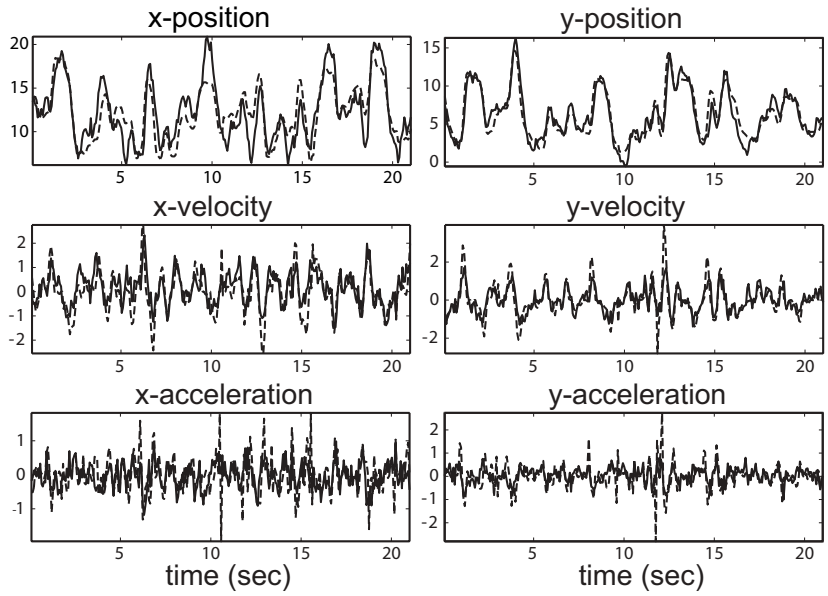

Figure 4: **Pinball Task:** Reconstruction of each component of the system state variable: true target motion (dashed) and reconstruction using the Kalman filter (solid). $20s$ from a 1*min* test sequence are shown. (From (Wu et al., 2003).)

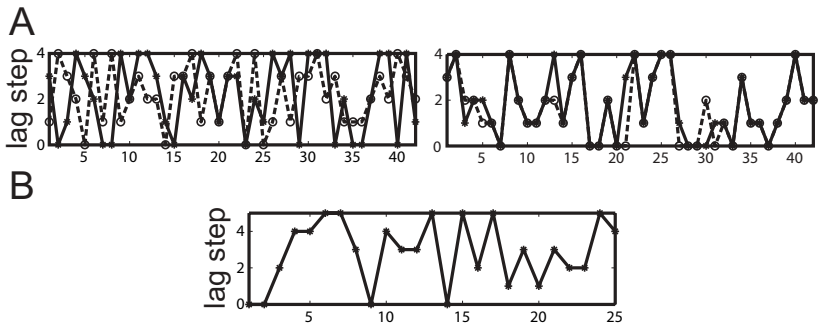

Figure 5: (A) Pinball Experiment: Optimization of individual lag times within a bound of $0ms \leq$ lag $\leq 280ms$. The vertical axis shows the lag in terms of the number of time bins. Each Unit on the horizontal axis corresponds to one of the 42 cells. The left plot shows two random initial lags: 1. dashed line with circles; 2. solid line with stars. The right plot shows final estimated lags from the two initial conditions. Note that for most cells these two solutions are the same. This suggests that the greedy search approach, with random initial starting lags, provides reasonably stable results. (B) Pursuit Tracking Experiment: Optimization of individual lag times using the greedy algorithm. The range of possible lag steps is bounded $0 \leq i \leq 5$, or $0ms \leq$ lag $\leq 250ms$.

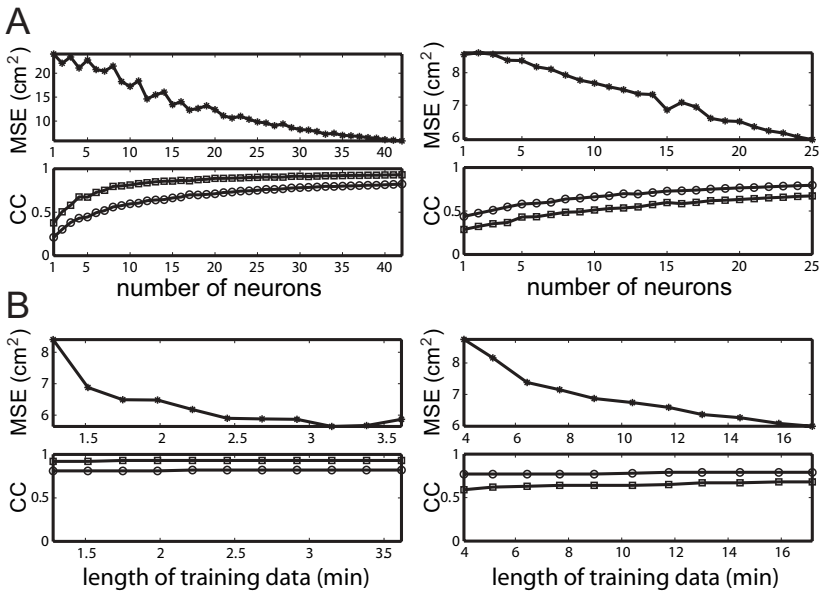

Figure 6: (A) Decoding accuracy with respect to number of neurons: The upper plot is the average of the MSE over 100 random subsets; the lower plot is the corresponding correlation coefficient for x (circle) and y (square). The left plot is for the Pinball task and the right one is for the Pursuit Tracking task. Note that, even with a single neuron, the value of the CC can be deceptively high while the MSE is large as expected. (B) Decoding accuracy with respect to amount of training data in both Pinball and Pursuit Tracking tasks: The illustrations of the four plots correspond to those in (A). The basic trend is that the larger the training data set, the higher the decoding accuracy, but the improvement appears to diminish as the amount of training data increases.

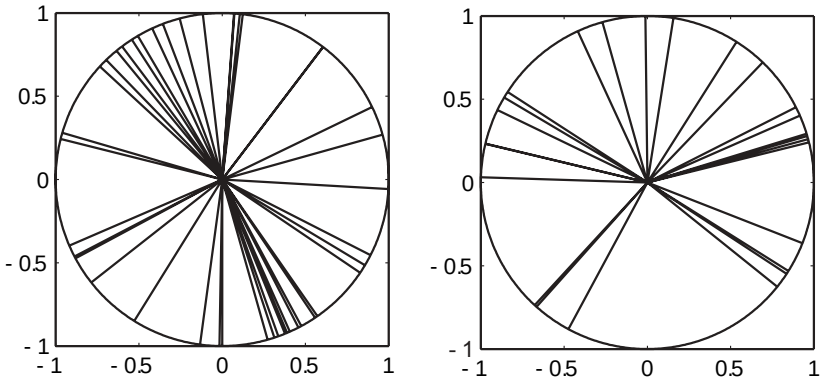

Figure 7: Preferred directions of all cells in Pinball (left) and Pursuit Tracking (right) tasks: the distributions of directions are not uniform over $[0, 2\pi]$.

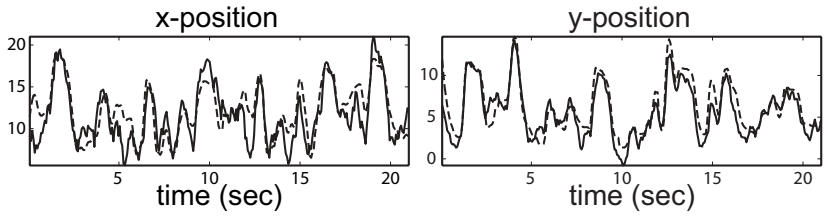

Figure 8: **Pinball Task:** Reconstruction of position using the linear filter: true target trajectory (dashed) and reconstruction using the linear filter (solid). (From (Wu et al., 2003).)

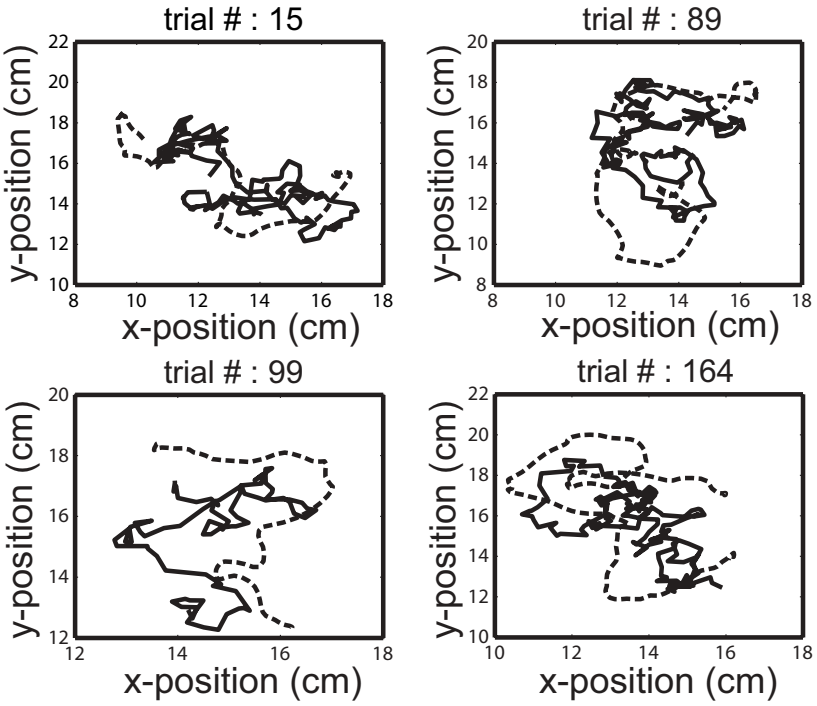

Figure 9: **Pursuit Tracking Task:** Reconstruction using the linear filter: true hand trajectory (dashed) and reconstruction using the linear filter (solid).

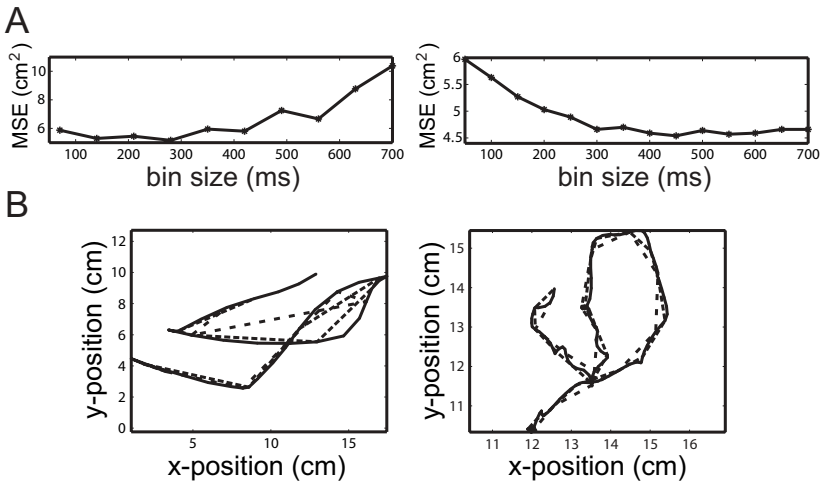

Figure 10: (A) Decoding accuracy (MSE) as a function of bin size for the Pinball task (left) and the Pursuit Tracking task (right). (B) Example reconstruction with varying bin size. (left) Pinball task: 70ms (solid), 280ms (tightly dashed) and 490ms (loosely dashed). (right) Pursuit Tracking task: 50ms (solid), 300ms (tightly dashed) and 500ms (loosely dashed).