Accepted for publication in *Neural Computation.*

Sequential Monte Carlo Point Process Estimation of Kinematics from Neural Spiking Activity for Brain Machine Interfaces

Yiwen Wang[1], António R. C. Paiva[1,] José C. Príncipe[1], Justin C. Sanchez[2]

[1]Department of Electrical & Computer Engineering, University of Florida, Gainesville, FL 32611, USA

[2]Department of Pediatrics, Neuroscience, and Biomedical Engineering, University of Florida, Gainesville, FL 32610, USA

E-mail: {wangyw,arpaiva,principe,justin}@cnel.ufl.edu

Abstract. Many Brain Machine Interfaces' (BMIs) decoding algorithms estimate hand movement from binned spike rates, which do not fully exploit the resolution contained in spike timing and may exclude rich neural dynamics from the modeling. More recently, an adaptive filtering method based on a Bayesian approach to reconstruct the neural state from the observed spike times has been proposed. However, it assumes and propagates a Gaussian distributed state posterior density, which is in general too restrictive. We have also proposed a Sequential Monte Carlo estimation methodology to reconstruct the kinematic states directly from the multi-channel spike trains. This paper presents a systematic testing of this algorithm in a simulated neural spike train decoding experiment and then in BMI data. Compared to a point process adaptive filtering algorithm with a linear observation model and a Gaussian approximation (the counterpart for point

processes of the Kalman filter), our Sequential Monte Carlo Estimation methodology exploits a detailed encoding model (tuning function) derived for each neuron from training data. However, this added complexity is translated in higher performance with real data. To deal with the intrinsic spike randomness in online modeling, several synthetic spike trains are generated from the intensity function estimated from the neurons and utilized as extra model inputs in an attempt to decrease the variance in the kinematic predictions. The performance of the Sequential Monte Carlo Estimation methodology augmented with this synthetic spike input provides improved reconstruction, which raises interesting questions and helps understand the overall modeling requirements better.

1. Introduction

Brain-Machine Interfaces (BMIs) exploit the spatial and temporal structure of neural activity to directly control a prosthetic device. In this framework [Wessberg, Stambaugh, Kralik, Beck, Laubach, Chapin, Kim, Biggs, Srinivasan, & Nicolelis, 2000; Serruya, Hatsopoulos, Paninski, Fellows, & Donoghue, 2002], neuronal activity (local field potentials and single unit activity) has been synchronously collected from microelectrode arrays implanted into multiple cortical areas while animals and humans have performed 3-D or 2-D target-tracking tasks. Several signal-processing approaches have been applied to extract the functional relationship between neural recordings and the animal's kinematic trajectories [Wessberg *et al.* 2000, Sanchez, Kim, Erdogmus, Rao, Principe, Wessberg, & Nicolelis, 2002; Kim, Sanchez, Erdogmus, Rao, Wessberg, Principe, & Nicolelis, 2003; Wu, Gao, Bienenstock, Donoghue, & Black, 2006; Brockwell, Rojas, &

Kaas, 2004; Shoham, Paninski, Fellows, Hatsopoulos, Donoghue, & Normann, 2005; ErgunBarbieri, Eden, Wilson, & Brown, 2007; Yu, Kemere, Santhanam, Afshar, Ryu., Meng, Sahani, & Shenoy, 2007; Srinivasan, Eden, Mitter, & Brown, 2007]. The models predict movements and control a prosthetic robot arm or computer to implement them. Many decoding methodologies use binned spike trains to predict movement based on linear or nonlinear optimal filters [Wessberg *et al*. 2000, Sanchez *et al*. 2002, Kim *et al.* 2003]. These methods avoid the need for explicit knowledge of the dynamical neural encoding properties, and standard linear or nonlinear regression fits the relationship directly into the decoding operation. Yet another probabilistic methodology can be derived probabilistically using a state-space model within a Bayesian formulation [Schwartz, Taylor, & Tillery, 2001; Wu *et al*., 2006; Brockwell *et al*., 2004; Shoham *et al.* 2005; Yu *et al.,* 2007; Srinivasan *et al.,* 2007]. From a sequence of noisy observations of the neural activity, the probabilistic approach analyzes and infers the kinematics as a state variable of the neural dynamical system. Since neural tuning relates the measurement of the neural activity to the animal's behavior, a physiologically realistic observation measurement model can be build. Consequently, a recursive algorithm based on all available statistical information will construct the posterior probability density function of each kinematic state given the neuron activity at each time step from the prior density of the state. The prior density in turn becomes the posterior density of previous time step updated with the discrepancy between an observation model and the neuron firings. Movements can be recovered probabilistically from the multi-channel neural recordings by estimating the posterior density. One of the general assumptions in these state-space models is that the model is linear and the posterior density is Gaussian, such as in the

Kalman filter as implemented in the free-arm-movement models [Wu *et al*., 2006], and the mixture-of-trajectories model [Yu *et al*., 2007]. Bootstrap filters overcome this assumption by Monte Carlo sampling of the posterior density and have been applied in motor BMIs using spike rates [Brockwell *et al*., 2004; Shoham *et al*. 2005], i.e. an instantaneous estimator of the neuron's intensity function.

The previous algorithms do not exploit spike timing structure due to binning (i.e. counting spikes on a time window) and may exclude rich neural dynamics in the modeling. One reason for this limitation is that these algorithms are designed for continuous random valued observations, and cannot be applied directly to point processes. Indeed, a spike train is a realization of a point process and is completely specified by the spike times; therefore the information is contained solely in the signal time structure [Brown *et al.* 1998, Barbieri *et al*. 2004; Eden et al. 2004; Ergun *et al*. 2007]. The fundamental question on how to adapt the Bayesian sequential estimation models to point processes can be answered by working with the probability of neural firing in a much shorter interval (~ 10 msec), which is a continuous random variable. Researchers used the well accepted Poisson model of spike generation [Tuckwell 1988; Rieke, Warland, Steveninck, & Bialek, 1997; Gabbiani & Koch 1998; and Reich, Victor, & Knight, 1998], making the firing rate dependent upon the system state. The Poisson distribution assumption is very common because it has been validated in numerous experimental setups, but it cannot account for multimodal firing histograms that are often found in neurons. The Poisson model has been improved with a time varying mean to yield what is called the inhomogeneous Poisson model [Gabbiani *et al.* 1998, Shadlen & Newsome 1998].

Initially, Diggle and colleagues [Diggle, Liang, & Zeger 1995] mentioned estimation from point process observations without a specific algorithm. Chan and Ledolter [1995] proposed a Monte Carlo Expectation-maximization (EM) algorithm using the Markov Chain sampling technique to calculate the expectation in the E-step of the EM algorithm. This method later became the theoretical basis to derive an EM algorithm for a point process recursive nonlinear filter [Smith & Brown, 2003]. The algorithm combined the inhomogeneous Poisson model on point process with the fixed interval smoothing algorithm to maximize the expectation of the complete data log likelihood. In this particular case, the observation process is a realization of a point process from an exponential family and the natural parameter is modeled as a linear function of the latent process.

A general adaptive filtering paradigm for point processes was recently proposed in [Brown, Nguyen, Frank, Wilson, & Solo, 2001] to reconstruct the hand position from the discrete observation of neural firings. This algorithm modeled the neural spike train as a history-dependent generalization of the inhomogeneous Poisson rate function [Eden *et al*. 2004, Daley & Vere-Jones, 1988] feeding a kinematic model through a nonlinear tuning function. The point process counterparts of the extended Kalman filter, recursive least squares, and steepest descent algorithms were derived and recently compared in the decoding of tuning parameters and states from the ensemble neural spiking activity [Eden, Frank, Solo, & Brown *et al*, 2004]. The stochastic state point process filter performs the best, because it provides an adjustable step size to update the state, which is estimated from the covariance information. However, this method still assumes that the posterior density of the state vector given the discrete observation is Gaussian distributed, which is

rather unlikely for a nonlinear system. More recently in [Ergun *et al.* 2007] the proposed prior density (where the particles are sampled from) is still assumed Gaussian distributed, although the propagation along time of the posterior is unconstrained. We proposed [Wang, Paiva, & Principe, 2006] a probabilistic filtering algorithm to reconstruct the state from the discrete observation (spiking events) by generating a sequential set of samples to estimate the distribution of the state posterior density without the Gaussian assumption. The posterior density is *non-parametrically* reconstructed, propagated and revised by the coming spike observation over time. The state at each time is determined either using the maximum a posteriori method or by collapsing the reconstructed posterior density using a Gaussian approximation of the posterior.

This paper further develops this algorithm in section 2 and builds an adaptive signal processing framework for Brain Machine Interfaces working directly in the spike domain, overcoming the issue of specifying the window size for binning. Such algorithm can be implemented online to reconstruct the kinematics from the neuron spike train observations. Section 3 illustrates the performance of both algorithms in a simulated neuron decoding example. We then apply this algorithm to predict the kinematics directly from spike trains for a Brain Machine Interface application. The kinematics reconstruction results are presented and compared with the point process analogue of the Kalman filter. This class of real-time spike domain algorithms produces results that are themselves stochastic processes because of the Poisson assumption to generate the spike timings in the encoding model. By creating artificial point processes with the same intensity function as the incoming spike trains, averaging across the population is possible. Unexpectedly, the trajectory fitting performance improved, which means that

for this motor task the variability of the filter response caused by the stochasticity in spike timing is more detrimental to performance than averaging the actual spike timings thru the intensity function. The impact of this finding for spike modeling in motor BMIs is addressed in the discussion and conclusion.

2. Theory

In this section, the design of adaptive filters for point processes under the Gaussian assumption is reviewed, and then the proposed methodology of a Sequential Monte Carlo (SMC) estimation is presented.

2.1 Adaptive Filtering for Point Processes with a Gaussian Assumption

One can model a point process using a Bayesian approach to estimate the system state by evaluating the posterior density of the state given the discrete observation [Eden *et al.* 2004]. This framework provides a nonlinear time-series probabilistic model between the state and the spiking event [Brown, Frank, Tang, Quirk, &Wilson 1998].

Given an observation interval $(0,T]$, $N(t)$ is defined as the counting process of events giving the total number of neuron spikes in the interval $(0,t]$, for $t \in (0,T]$. It can be modeled as an inhomogeneous Poisson process characterized by its conditional intensity function $\lambda(t \mid \mathbf{x}(t), \boldsymbol{\theta}(t), \boldsymbol{Z}(t))$, i.e. the instantaneous rate of events, defined as

$$\lambda(t \mid \mathbf{x}(t), \boldsymbol{\theta}(t), \boldsymbol{Z}(t)) = \lim_{\Delta t \to 0} \frac{\Pr(N(t+\Delta t) - N(t) = 1 \mid \mathbf{x}(t), \boldsymbol{\theta}(t), \boldsymbol{Z}(t))}{\Delta t} \tag{1}$$

where $\mathbf{x}(t)$ is the system state, $\boldsymbol{\theta}(t)$ is the parameter of the adaptive filter, and $\boldsymbol{Z}(t)$ is the history of all the states, parameters and the discrete observations up to time *t*. The

relationship between the Poisson process conditional intensity function λ, the state $\mathbf{x}(t)$, and the parameter $\boldsymbol{\theta}(t)$ is assumed to be a nonlinear observation model represented by

$$\lambda(t \mid \mathbf{x}(t), \boldsymbol{\theta}(t)) = f(\mathbf{x}(t), \boldsymbol{\theta}(t)) \tag{2}$$

The nonlinear function $f(\cdot)$ is assumed to be known or specified according to the application. Let us consider hereafter the parameters $\boldsymbol{\theta}(t)$ as part of the state vector $\mathbf{x}(t)$. Due to the Poisson assumption, the *observation* model is memoryless (i.e. it has no dependence on the previous discrete observations).

The previous continuous model can be converted to a discrete model with the individual time steps represented by $(t_{k-1}, t_k]$, where all the continuous value of the state and parameters are index with *k*. Given a binary observation vector ΔN_k for multi-neurons over the time interval $(t_{k-1}, t_k]$, the posterior density of the whole state vector $\mathbf{x}(t)$ at time t_k given the history $\boldsymbol{Z}_{k-1} = [\mathbf{x}_{1:k-1}, N_{1:k-1}]$ can be represented by Bayes' rule as

$$p(\mathbf{x}_k \mid \Delta N_k, \boldsymbol{Z}_{k-1}) = \frac{p(\Delta N_k \mid \mathbf{x}_k, \boldsymbol{Z}_{k-1}) p(\mathbf{x}_k \mid \boldsymbol{Z}_{k-1})}{p(\Delta N_k \mid \boldsymbol{Z}_{k-1})} \tag{3}$$

where $p(\Delta N_k \mid \mathbf{x}_k, \boldsymbol{Z}_{k-1})$ is the probability of observing spikes in the interval $(t_{k-1}, t_k]$. Considering the Poisson process made on the conditional intensity function, it is simplified as $p(\Delta N_k \mid \mathbf{x}_k)$, which is defined as

$$\Pr(\Delta N_k \mid \mathbf{x}_k) = (\lambda(t_k \mid \mathbf{x}_k, \boldsymbol{Z}_{k-1})\Delta t)^{\Delta N_k} \exp(-\lambda(t_k \mid \mathbf{x}_k, \boldsymbol{Z}_{k-1})\Delta t) \tag{4}$$

and $p(\mathbf{x}_k \mid \boldsymbol{Z}_{k-1})$ is the one-step prediction density given by the Chapman-Kolmogorov equation

$$p(\mathbf{x}_k \mid \boldsymbol{Z}_{k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \boldsymbol{Z}_{k-1}) p(\mathbf{x}_{k-1} \mid \Delta N_{k-1}, \boldsymbol{Z}_{k-1}) d\mathbf{x}_{k-1} \tag{5}$$

where the state $\mathbf{x}_k$ evolves according to the linear relation

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + \eta_k \tag{6}$$

F_k is the state evolution model represented by a D by D matrix, where D is the dimension of the state vector $\mathbf{x}_k$. It establishes the dependence on the previous state and η_k is zero-mean white noise with covariance Q_k. Substituting equations (4) and (5) in (3), the posterior density of the state $p(\mathbf{x}_k \mid \Delta N_k, \boldsymbol{Z}_{k-1})$ can be recursively estimated from the previous value based on the spike observation.

Given observations modeled as a point process, the solution to the Chapman-Kolmogorov equations can be computed using a number of methods. In general, the update equations for the Gaussian approximation are non-linear (for example, the observation model defined in equation 2) and have to be solved at each step using a Newton's procedure to find the Gaussian approximation. In [Eden *et al*. 2004], the posterior density given by (3) and the noise term η_k in the state evolution equation (6) are assumed Gaussian distributed, for which the Chapman-Kolmogorov equation (5) becomes a convolution of two Gaussian functions. Under these assumptions, the estimation of the state at each time has a closed form expression given by (see [Eden *et al*. 2004] for details).

$$\mathbf{x}_{k|k-1} = F_k \mathbf{x}_{k-1|k-1} \tag{7a}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^{'} + Q_k \tag{7b}$$

$$(P_{k|k})^{-1} = (P_{k|k-1})^{-1} + [(\frac{\partial \log \lambda}{\partial \mathbf{x}_k})^{'} [\lambda \Delta t_k] (\frac{\partial \log \lambda}{\partial \mathbf{x}_k}) - (\Delta N_k - \lambda \Delta t_k) \frac{\partial^2 \log \lambda}{\partial \mathbf{x}_k \partial \mathbf{x}_k^{'}}]_{\mathbf{x}_{k|k-1}} \tag{7c}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + P_{k|k} [(\frac{\partial \log \lambda}{\partial \mathbf{x}_k})^{'} (\Delta N_k - \lambda \Delta t_k)]_{\mathbf{x}_{k|k-1}} \tag{7d}$$

The Gaussian assumption becomes a simplifying assumption that allows an analytical solution for (5) and therefore, for a closed form solution of (3) as (7).

Although the above set of equations may seem daunting they can be interpreted quite easily. First, (7a) establishes a prediction for the state based on the previous value. Then, (7b) and (7c) are used in (7d) to correct the previous estimate, after which the recursive calculation is repeated.

2.2 Sequential Monte Carlo Estimation for Point Processes

The Gaussian assumption applied to the posterior distribution in the algorithm is usually used as an approximation in practical situations; however, it is not a general case when the posterior distribution is multi-mode or highly skewed. Therefore, for the discrete observations case, a non-parametric approach is developed here which poses no constraints on the form of the posterior density.

Sequential Monte Carlo estimation [Doucet, de Freitas, & Gordon 2001] is a well known technique for implementing a recursive Bayesian filter by Monte Carlo simulations. The key idea is to construct the posterior density function required for Bayesian estimation from a set of random samples with associated weights and to calculate the estimations based on these weighted samples. As the number of samples increases, this Monte Carlo characterization becomes an equivalent representation of the posterior pdf. Here we apply this approach directly on discrete observations in the spike domain.

Suppose at time instant *k* the previous system state is $\mathbf{x}_{k-1}$. All we need is to estimate the state from the conditional intensity function, since the nonlinear relation $f(\cdot)$ in (2) is assumed known. Random state samples are generated using Monte Carlo simulations

[Carpenter, Clifford, & Fearnhead, 1999] in the neighborhood of the previous state according to (6). Then, Parzen windowing [Parzen 1962] with a Gaussian kernel estimates the posterior density. Due to the linearity of the integral in the Chapman-Kolmogorov equation and the weighted sum of Gaussians centered at the samples we are still able to evaluate the integral directly from samples. The process is recursively repeated for each time instant propagating the estimate of the posterior density as the prior density for the next time instant, and the state itself, based on the discrete events over time. Notice that due to the recursive approach the algorithm not only depends on the previous observation, but also on the whole history of the spike observation events that were involved in estimating the posterior density.

Let $\{\mathbf{x}_{0:k}^{i}, w_k^i\}_{i=1}^{N_S}$ denote a random measure [Arulampalam, Maskell, Gordon, & Clapp, 2002] of the posterior density $p(\mathbf{x}_{0:k} \mid N_{1:k})$, where $\{\mathbf{x}_{0:k}^{i}, i = 1,\cdots,N_S\}$ is the set of all state samples up to time *k* with associated normalized weights $\{w_k^i, i = 1,\cdots,N_S\}$, *i* is the sample index and N_S is the number of samples generated at each time, $N_{1:k}$ is the spike observation events up to time *k* modeled by an inhomogeneous Poisson Process as in section 2.1 Then, the posterior density at time *k* can be approximated by a weighted convolution of the samples with a Gaussian kernel as

$$p(\mathbf{x}_{0:k} \mid N_{1:k}) \approx \sum_{i=1}^{N_S} w_k^i \cdot k(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^{i}, \sigma) \tag{8}$$

where $k(x - \bar{x}, \sigma)$ is the Gaussian kernel with mean $\bar{x}$ and covariance σ. According to the principle of Importance Sampling [Bergman 1999; Doucet 1998; Doucet *et al.* 2001], the weights can be defined by

$$w_k^i \propto \frac{p(\mathbf{x}_{0:k}^i \mid N_{1:k})}{q(\mathbf{x}_{0:k}^i \mid N_{1:k})} \tag{9}$$

where $q(\mathbf{x}_{0:k} \mid N_{1:k})$ is a proposed importance density from where all samples $\mathbf{x}_{0:k}^i$ are generated from. Here, we assume the importance density obeys the properties of Markov Chains, which only depends on $\mathbf{x}_{k-1}$ and N_k, such that

$$\begin{aligned} q(\mathbf{x}_{0:k} \mid N_{1:k}) &= q(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, N_{1:k}) q(\mathbf{x}_{0:k-1} \mid N_{1:k-1}) \\ &= q(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \Delta N_k) q(\mathbf{x}_{0:k-1} \mid N_{1:k-1}) . \end{aligned} \tag{10}$$

At each iteration, the posterior density $p(\mathbf{x}_{0:k} \mid N_{1:k})$ can be estimated from the previous iteration according to Bayes rule and the properties of Markov Chains as:

$$\begin{aligned} p(\mathbf{x}_{0:k} \mid N_{1:k}) &= \frac{p(\Delta N_k \mid \mathbf{x}_{0:k}, N_{1:k-1}) p(\mathbf{x}_{0:k} \mid N_{1:k-1})}{p(\Delta N_k \mid N_{1:k-1})} \\ &= \frac{p(\Delta N_k \mid \mathbf{x}_{0:k}, N_{1:k-1}) p(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, N_{1:k-1})}{p(\Delta N_k \mid N_{1:k-1})} \times p(\mathbf{x}_{0:k-1} \mid N_{1:k-1}) \\ &= \frac{p(\Delta N_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{x}_{k-1})}{p(\Delta N_k \mid N_{1:k-1})} \times p(\mathbf{x}_{0:k-1} \mid N_{1:k-1}) \\ &\propto p(\Delta N_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{0:k-1} \mid N_{1:k-1}) \end{aligned} \tag{11}$$

Both terms are simplified in the weight updating equation (9) by the recursive presentation (10) and (11), therefore, the weight can be updated recursively as (12):

$$\begin{aligned} w_k^i &\propto \frac{p(\Delta N_k \mid \mathbf{x}_k^i) p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i) p(\mathbf{x}_{0:k-1}^i \mid N_{1:k-1})}{q(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i, \Delta N_k) q(\mathbf{x}_{0:k-1}^i \mid N_{1:k-1})} \\ &= \frac{p(\Delta N_k \mid \mathbf{x}_k^i) p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i, \Delta N_k)} w_{k-1}^i \end{aligned} \tag{12}$$

Frequently the importance density $q(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i, \Delta N_k)$ is chosen to be the prior density $p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)$, requiring the generation of new samples from $p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)$ by (6) as a prediction stage, where the noise is randomly drawn according to the noise distribution $p(\eta)$. The weight is then updated recursively with the information from the current observation $p(\Delta N_k \mid \mathbf{x}_k^i)$.

After the algorithm is applied for a few iterations, a phenomenon called degeneracy may arise, where all but one sample have negligible weight [Doucet, *et al*, 2001], implying that a large computational effort is taken to update the samples that have almost no contribution to estimate the posterior density. When a significant degeneracy appears, resampling is applied to eliminate the samples with small weight and to concentrate on samples with large weights according the samples *cdf*. Here Sequential Importance Resampling [Gordon, Salmond, & Smith 1993] is applied at every time index to avoid degeneration, so that the sample is *i.i.d.* from the discrete uniform density with weights $w_{k-1}^i = 1/N_S$. The weights then change proportionally given by

$$w_k^i \propto p(\Delta N_k \mid \mathbf{x}_k^i) \tag{13}$$

where $p(\Delta N_k \mid \mathbf{x}_k^i)$ is defined by equation (4) in this section. Using (6), (13) and the resampling step, the posterior density of the state $\mathbf{x}_k$ given the whole path of the observed events up to time t_k can be approximated as

$$p(\mathbf{x}_k \mid N_{1:k}) \approx \sum_{i=1}^{N_S} p(\Delta N_k \mid \mathbf{x}_k^i) \cdot k(\mathbf{x}_k - \mathbf{x}_k^i) \tag{14}$$

Equation (14) shows that the posterior density of the current state given the observation is modified by the latest probabilistic measurement of observing the spike event $p(\Delta N_k \mid \mathbf{x}_k^i)$, which is the update stage in the adaptive filtering algorithm.

Without a closed form solution for state estimation, we estimate all the information available through the posterior density of the state given the observed spike event $p(\mathbf{x}_k \mid N_{1:k})$ at every step. In this way, any interesting moments of the pdf can be obtained, for example, Maximum A Posteriori (MAP) can be applied to get the state estimation $\widetilde{\mathbf{x}}_k$, which picks out the sample $\mathbf{x}_k^{i^*}$ with maximum posterior density. We can also alternatively obtain the mean value of the posterior and the error covariance V_k by a technique called collapse [Wu, Black, Memford, Gao, Bienenstock, & Donoghue, 2004]:

$$\widetilde{\mathbf{x}}_k = \sum_{i=1}^{N_S} p(\Delta N_k \mid \mathbf{x}_k^i) \cdot \mathbf{x}_k^i \tag{15}$$

$$V_k = \sum_{i=1}^{N_S} p(\Delta N_k \mid \mathbf{x}_k^i) \cdot (\sigma + (\mathbf{x}_k^i - \widetilde{\mathbf{x}}_k)(\mathbf{x}_k^i - \widetilde{\mathbf{x}}_k)^T) \tag{16}$$

From (15) and (16), it is evident that with simple computation one can estimate the next state. Hence, the expectation by collapse is simple and elegant. Although this amounts to a Gaussian approximation, note that it is done after the arbitrary density is propagated through the filter.

The major drawback of the algorithm is computational complexity because the quality of the solution requires many particles $\{\mathbf{x}_{0:k}^i, i = 1, \cdots, N_S\}$ to approximate the posterior density. Smoothing the particles with kernels as in (14) alleviates the problem in

particular when collapsing is utilized, but still the computation is higher than calculating the mean and covariance of the pdf with a Gaussian assumption.

Although, strictly speaking, collapse of the posterior pdf for estimation of the kinematics implies a Gaussian approximation, this occurs only at the "output" stage of the algorithm to estimate the state. The propagation of the posterior pdf, however, is non-parametric and has no underlying assumptions. This means that information about the evolution of the kinematics contained in the, generally, non-Gaussian posterior is fully preserved (equations (8) through (14)). The examples shown in the next section demonstrate the importance of this approach for decoding of the kinematics.

We have to point out that both approaches (MAP or collapse) assume we know the state model F_k in (6) and the observation model $f(\cdot)$ in (2), which actually are unknown in real applications. The state model is normally assumed linear and the parameters are obtained from the data using least squares. The knowledge of the observation model is very important for decoding (deriving states from observations), because the probabilistic approach based on Bayesian estimation constructs the posterior density of each state given the spike observation at each time step from the prior density of the state. The prior density in turn is the posterior density of previous time step updated with the discrepancy between an observation model and the spike event. The observation model basically quantifies how each neuron encodes the kinematic variables (encoding step in Figure 1), and due to the variability of neural responses it should be carefully estimated from a training set. In a previous paper [Wang 2008] we proposed a statistical approach using *instantaneous kinematics*, instead of the conventional window based approach [Paninski, Shoham, Fellows, Hatsopoulos, and Donoghue 2004] to estimate the tuning function for

each neuron which will be used here in the testing of the SMC estimation model with real data. Figure 1 shows the schematic of the relationship between the encoding and decoding in point process SMC estimation.

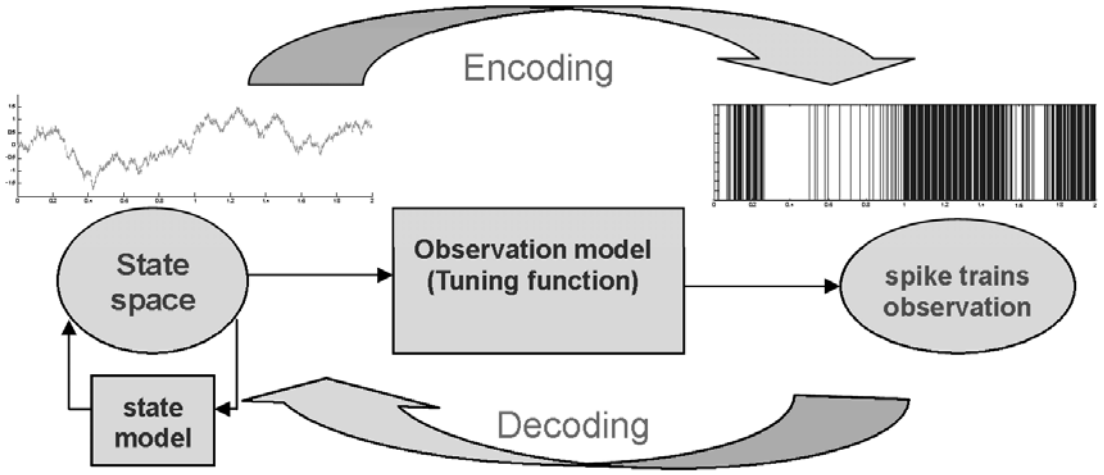

Figure 1 Schematic of relationship between encoding and decoding processes for Sequential Monte Carlo estimation of point processes

3. Applications

In this section, we present performance tests of the proposed point process SMC estimation and compare it to adaptive filtering, also on point processes, using a Gaussian assumption in simulated and real scenarios. First, both approaches will be tested on synthetic neuron decoding data, where each algorithm is fed to a known observation model. Secondly, both algorithms are applied to real neural data from a BMI experiment to estimate the kinematics from neural spike trains. The observation model linking the measurement of the noisy neural activity to the kinematics implicitly utilizes the tuning characteristics of each neuron, which is estimated from the data and will not be explained

here due to space limitations. For details on the methodology used to estimate the tuning function model of each neuron the reader is referred to Wang [2008].

3.1 Simulation of neuron decoding

In a conceptually simplified motor cortical neural model [Moran & Schwartz 1999], the one-dimensional velocity can be reconstructed from the neuron spiking events by a SMC estimation algorithm. This algorithm can provide a probabilistic approach to infer the most probable velocity as one of the components of the state. This decoding simulation updates the state estimation simultaneously and reconstructs the signal, which assumes interdependence between the encoding and decoding so that the accuracy of the receptive field estimation and the accuracy of the signal reconstruction are reliable.

Let us first explain how the simulated data was generated. The tuning function of the receptive field that models the relation between the velocity and the firing rate is assumed exponential and given by

$$\lambda(t_k) = \exp(\mu + \beta_k v_k) \tag{17}$$

where $\exp(\mu)$ is the background firing rate without any movement and β_k is the modulation in firing rate due to the velocity v_k. In practice in the electrophysiology lab, this function is unknown. Therefore, an educated guess needs to be made about the functional form, and the exponential function is widely utilized.

The desired velocity is a frequency modulated (chirp) triangle wave corrupted with additive Gaussian noise (variance 2.5×10^{-5}) at each 1ms time step, as shown in Figure 2. The design of the desired signal enables us to check if the algorithm could track the linear evolution and the different frequency of the "movement".

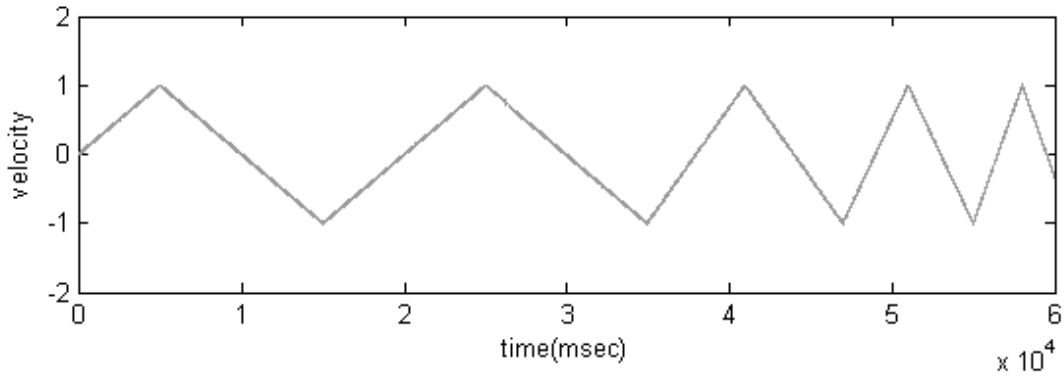

Figure 2 The desired velocity generated by triangle wave with Gaussian noise

The background-firing rate $\exp(\mu)$ and the modulation parameter β_k are set to be 1 and 3 respectively for the whole simulation time, 60s. A spike is drawn as a Bernoulli random variable with probability $\lambda(t_k)\Delta t$ within each 1ms time window [Brown, Barbieri, Ventura, Kass, & Frank 2002]. An example of a synthetic spike train is shown in Figure 3.

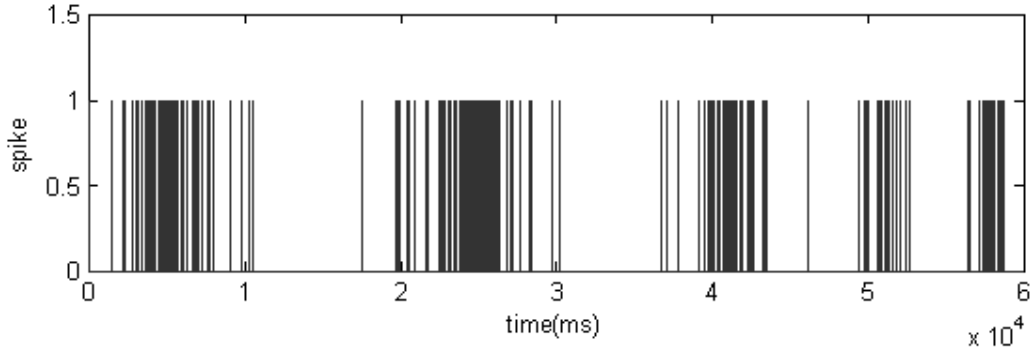

Figure 3 The simulated neuron spike train generated by an exponential tuning function

With the exponential tuning function operating on the velocity, we can see that when the velocity is negative, there are only a few spikes; whereas when the velocity is

positive, many spikes appear. The problem is to obtain from this spike train the desired velocity of Fig. 2, assuming the Poisson model of (17) and one of the sequential estimation techniques discussed.

To implement the SMC estimation for the point process, we regard both the modulation parameter β_k and velocity v_k as the state $\mathbf{x}_k = [v_k \ \beta_k]^T$. Here we set 100 samples to initialize the velocity v_0^i and modulation parameter β_k respectively with a uniform and with a Gaussian distribution. Note that too many samples would increase the computational complexity dramatically, while an insufficient number of samples would result on a poor description of the non-Gaussian posterior density. The new samples are generated according to the linear state evolution (6), where F_k is obtained from the data using least squares for v_k and 1 for β_k (implicitly assuming that the modulation parameter would not change very fast). The *i.i.d.* noise for velocity state in (6) was drawn from the distribution of the error between the true velocity and the linear predicted results by F_k. The *i.i.d.* noise for estimating the modulation parameter β_k is approximated by a zero mean Gaussian distribution with variance 10^{-7}. Notice that the noise variance should be small enough to track the unchanged β_k set in the data. The kernel size utilized in (14) to estimate the maximum of the posterior density (thru MAP) obeys Silverman's rule [Silverman 1981].

In order to obtain realistic performance assessments of the different models (MAP and collapse), the state estimations $\widetilde{v}_k \ \widetilde{\beta}_k$ for the duration of the trajectory are drawn 10 different times. The velocity reconstruction is shown in Figure 4. The Normalized Mean Square Error (MSE normalized by the power of the desired signal) between the desired

trajectory and the model output for the adaptive filtering with Gaussian assumption is 0.3633. NMSE for sequential estimation by MAP is 0.2710 and by collapse is 0.2532.

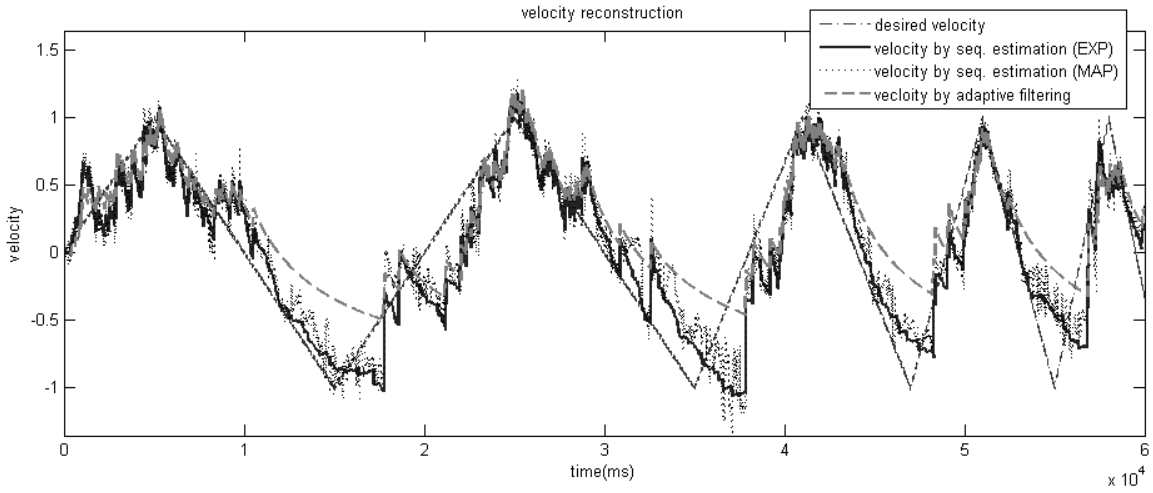

Figure 4 The velocity reconstruction by different algorithms

From Figure 4, we can see that compared with the desired velocity (dash-dotted line), all the methods obtain close estimation when there are many spikes, *i.e.* when the velocity is at the positive peaks of the triangle wave. This is because the high likelihood of spikes corresponds to the range of the exponential tuning function where the modulation of the high firing probability is easily distinguished and the posterior density is close to the Gaussian assumption. However, in the negative peaks of the desired velocity the sequential estimation algorithm (using collapse or MAP) performs considerably better. This is primarily because the modulation of the firing rate is nonlinearly compressed by the exponential tuning function, leading to non-Gaussian posterior densities, and thus violating the Gaussian assumption the adaptive filtering method relies on. Although there is nearly no neuronal representation for negative velocities and therefore both algorithms are inferring the new velocity solely on the previous state, the non-parametric estimation of the pdf in the sequential estimation

algorithm allows for more accurate inference. As an example, in Figure 5a, the posterior density at time 6.247s (when the desired velocity is close to the positive peak) is shown (dotted line). It has a Gaussian-like shape and therefore all methods provide similar estimations close to the true value (star). In Figure 5b, the posterior density at time 35.506s (when the desired velocity is close to the negative peak) is shown (dotted line), which is clearly non-symmetric with 2 ripples and is obviously also non-Gaussian distributed. The adaptive filtering on point process under a Gaussian assumption provides poor estimation (gray dotted), because the algorithm assumes and propagates a Gaussian pdf resulting in an accumulation of errors. The velocity estimated by the sequential estimation with collapse denoted by the circle is the closest to the desired velocity (star). Notice the sequential estimation with collapse collapse does better than MAP, and using the mean of posterior is equivalent to take it as being Gaussian. What this shows, however, is that the shape of the posterior is not really important for the estimation of the kinematics at every time step (or the estimation of the posterior is not reliable enough), but it is crucial to faithfully keep track of the evolution of the posterior at each step. Put differently, with a Gaussian approximation we disregard important information in the posterior when computing the evolution of the system. Notice also that in all cases the tracking performance gets progressively worse as the frequency increases. This is because the state linear model is fixed for the whole data set, which only tracks the velocity state at the average frequency. If a time-variant state model is used on a segment-by-segment basis, we could expect better reconstructions.

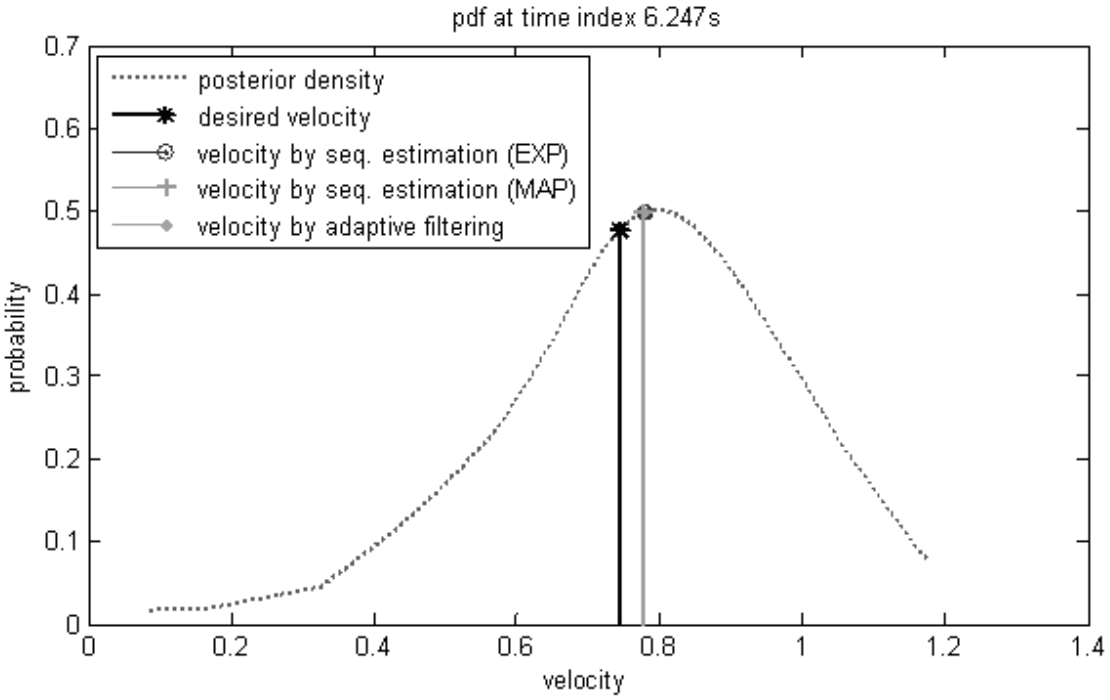

Figure 5a $p(v_k \mid \Delta N_k)$ at time 6.247s

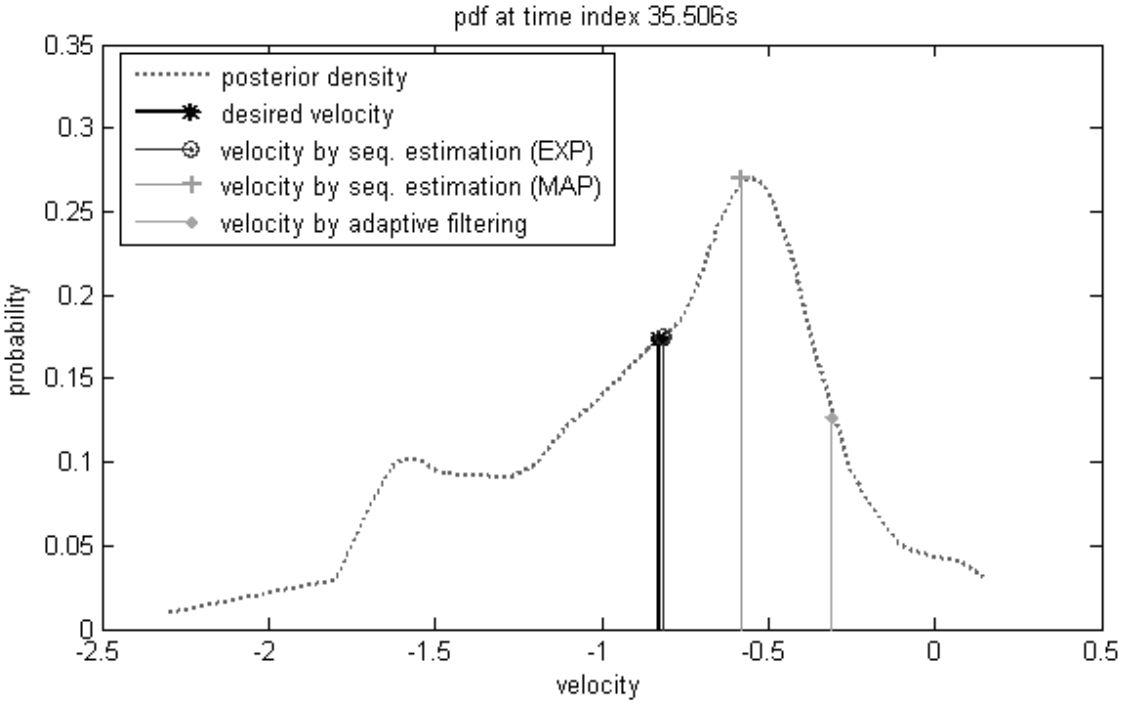

Figure 5b $p(v_k \mid \Delta N_k)$ at time 35.506s

In summary, Sequential Monte Carlo estimation on point processes seems promising for state estimation in BMIs.

3.2 In vivo neural decoding for BMIs

The Brain-Machine Interface paradigm was designed and implemented in Dr. Miguel Nicolelis laboratory at Duke University. Chronic, neural ensemble recordings were collected from the brain of an adult female Rhesus monkey, and synchronized with task behaviours. Microelectrode arrays were chronically implanted in five cortices: right dorsolateral premotor area (PMa), right primary motor cortex (MI), right primary somatosensory cortex (S1), right supplementary motor area (SMA), and the left primary motor cortex (MI).

After the surgical procedure, a multi-channel acquisition processor cluster (MAP, Plexon, Dallas, TX) was used in the experiments to record the neuronal action potentials simultaneously. Analog waveforms of the action potential were amplified and band pass filtered from 500 Hz to 5 kHz. The spikes of single neurons from each microwire were discriminated based on time-amplitude discriminators and a principal component analysis (PCA) algorithm [Nicolelis, Ghazanfar, Faggin, Votaw, & Oliveira, 1997; Wessberg *et al.* 2000]. The firing times of each spike were stored. Table 1 shows the assignment of the sorted neural activity to the electrodes for different motor cortical areas [Kim 2005].

Table 1. Assignment of the Sorted Neural Activity to the Electrodes

Monkey 1 (left handed)	Right PMa	Right MI	Right S1	Right SMA	Left MI
	1-66(66)	67-123(57)	124-161(38)	162-180(19)	181-185(5)

The monkey performed a two-dimensional target-reaching task to move the cursor on a computer screen by controlling a hand-held joystick to reach the target. The monkey was rewarded when the cursor intersected the target. The corresponding position (cm) of

the joystick was recorded continuously for an initial 30-min period at a 50 Hz sampling rate, referred to as the "pole control" period [Carmena, Lebedev, Crist, O'Doherty, Santucci, Dimitrov, Patil, Henriquez, & Nicolelis, 2003].

The decoding schematic for BMIs is shown in Fig. 1 as the arrow from right to left. The spike times from multiple neurons are the multi-channel point process observations. The signal processing begins by first translating the neural spike times into a sequence of 1 (spike) and 0 (no spike). A small enough time interval should be chosen to guarantee the Poisson hypothesis on the conditional intensity function, i.e. only a few intervals (ideally none) have more than one spike. If the interval is too small, however, the computational complexity is increased without any significant improvement in performance. One must also be careful when selecting the kinematic state (position, velocity, or acceleration) for the decoding model since the actual neuron encoding is unknown. The analysis presented here will consider a vector state with all three kinematic variables. The velocity is estimated as the difference between the current and previous recorded positions, and the acceleration is estimated by first differences from the velocity. For fine temporal resolution, all of the kinematics are interpolated and time synchronized with the neural spike trains.

It is interesting to note that in black box modeling, the motor BMI is posed as a decoding problem, *i.e*. a transformation from motor neurons to behavior. However, when we use the Bayesian sequential estimation, decoding is insufficient to solve the modeling problem. In order to implement *decoding* it is important to also model how each neuron *encodes* movement, which is exactly the observation model $f(\cdot)$ in (2). Therefore, one

sees that generative models do in fact require more information about the task and are therefore an opportunity to further investigate neural function.

3.2.1 Adaptive Filtering of Point Processes with a Gaussian Assumption for BMIs

The model developed here will be compared with an adaptive filter using a Gaussian assumption [Eden *et al.* 2004], which will be briefly described next. Adaptive filtering of point processes provides an analytical solution to state estimation; therefore, it requires a parametric model for the neuron tuning in closed form. Many different functional forms of tuning have been proposed, consisting mostly of linear projections of the neural modulation on 2 or 3 dimensions of kinematic vectors and bias. Moran and Schwartz [1999] also introduced a linear relationship from motor cortical spiking rate to speed and direction. Brockwell *et al.* [2003] assumed an exponential tuning function for their motor cortical data. Here we have tried both tuning functions. The exponential model produced experimentally a worse estimation than the linear model for this algorithm because it cannot actually characterize sufficiently well the tuning properties of the neurons in the data set [Wang 2008]. This experimental observation shows the importance of choosing an appropriate tuning model for the specific decoding methodology.

Notice that when a linear tuning function is selected for the observation model together with a Gaussian approximation for the posterior density, the end result is actually analoguous to a Kalman filter in the spike domain and will be called Kalman filter for point process (PP). Here the linear tuning function is estimated from 10000 samples of the training data as:

$$\lambda_t = \mathbf{h} \cdot \mathbf{x}_{t+lag} + \mathbf{B} \tag{18}$$

$$spike_t = Poisson(\lambda_t) \tag{19}$$

where λ_t is the firing probability for each neuron, obtained by smoothing the spike train with a Gaussian kernel. The kernel size is empirically set to 0.17 in this experiment [Wang 2008]. $\mathbf{x}_t$ is the instantaneous kinematics vector defined as $[\mathbf{p}_x\ \mathbf{v}_x\ \mathbf{a}_x\ \mathbf{p}_y\ \mathbf{v}_y\ \mathbf{a}_y]_t^T$ with 2-dimensional information of position, velocity and acceleration. The variable *lag* refers to the causal time delay between motor cortical neuron activity and kinematics due to the propagation effects of signals thru the motor and peripheral nervous systems. It was experimentally estimated to be 200 ms [Wang 2008], which agrees with prior work [Wu *et al.* 2006]. We extend the kinematics vector as $[\mathbf{p}_x\ \mathbf{v}_x\ \mathbf{a}_x\ \mathbf{p}_y\ \mathbf{v}_y\ \mathbf{a}_y\ 1]_t^T$ to include a bias $\mathbf{B}$, which can be regarded as part of the weights of the linear filter $\mathbf{H}$. The tuning function is then $\lambda_t = \mathbf{H} \cdot \mathbf{x}_{t+lag}$. The weight estimation of the linear filter $\mathbf{H}$ is given by

$$\mathbf{H} = (E[\mathbf{x}_{t+lag}{}^T \mathbf{x}_{t+lag}])^{-1} E[\mathbf{x}_{t+lag} \lambda_t] \tag{20}$$

Equation 20 represents the least squares solution for the linear adaptive filter, where $E[\mathbf{x}_{t+lag}{}^T \mathbf{x}_{t+lag}]$ gives the autocorrelation matrix R of the input kinematics vector considering a causal time delay. $E[\mathbf{x}_{t+lag} \lambda_t]$ gives the cross-correlation vector P between the input and the firing probability. The linear tuning function in (18) defines the first and second derivative terms in (7c) and (7d) as:

$$\frac{\partial \log \lambda_t}{\partial \mathbf{x}_{t+lag}} = \frac{\mathbf{H}^T}{\lambda_t} \tag{21}$$

$$\frac{\partial^2 \log \lambda_t}{\partial \mathbf{x}_{t+lag} \partial \mathbf{x}_{t+lag}^T} = -\frac{\mathbf{H} \cdot \mathbf{H}^T}{\lambda_t{}^2} \tag{22}$$

The kinematics vector is then derived as the state from the observation of multi-channel spikes train for the test samples by equation (7a-d).

3.2.2 Sequential Monte Carlo Estimation for Point Processes in BMIs

Closed form models for tuning may not be optimal for dealing with the real data because the tuning properties across the ensemble can vary significantly. In the literature, the point process filters have different encoding models according to their applications, either a linear approximation (Eden *et al*. 2004), or nonlinear (Brown et 1998, Barbieri *et al*. 2004). The accuracy of the tuning function estimation can directly affect the modeling in the Bayesian approach and the results of the kinematics estimation. One appropriate methodology is to estimate neural tuning using the training set data obtained in experiments. Based on the work by Simoncelli and colleagues [Simoncelli , Paninski, Pillow, and Schwartz,et al 2004] on Linear-Nonlinear-Poisson (LNP) model (figure 6), we developed an *instantaneous* encoding modeling for the neuron tuning function instead of performing the estimation on windows of data. Compared to the traditional windowed approach, our method builds a one-to-one mapping between the instantaneous kinematics vectors to the corresponding neural spiking time, which enables decoding online. Here, we briefly review the algorithms, but a full explanation can be found in [Wang 2008] including the model reasoning and testing. The tuning function between the kinematic vector and the neural spike train is exactly the observation model between the state and the observed data in our algorithm.

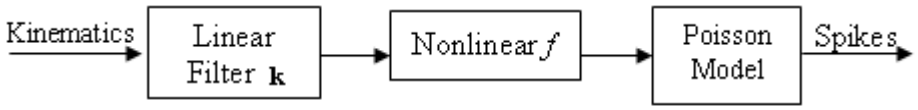

Figure 6. Block diagram of linear-nonlinear Poisson (LNP) model.

The instantaneous motor cortical neural activity can be modeled as

$$\lambda_t = f(\mathbf{k} \cdot \mathbf{x}_{t+lag}) \tag{23}$$

$$spike_t = Poisson(\lambda_t) \tag{24}$$

where, as before, $\vec{\mathbf{x}}_{t+lag}$ is the instantaneous kinematics vector defined as $[\mathbf{p}_x\ \mathbf{v}_x\ \mathbf{a}_x\ \mathbf{p}_y\ \mathbf{v}_y\ \mathbf{a}_y\ 1]^T_{t+lag}$ with 2 dimensional information of position, velocity, acceleration and bias with causal time delay depending on the data. For BMIs, the kinematic vector in Figure 6 must be read from the experiment for every spike occurrence since the task is dynamic, taking into consideration the causal delay between neural firings and kinematic outputs [Wang 2008]. The linear filter projects the kinematics vector **x** into its weight vector **k** (representing a preferred direction in space), which produces a scalar value that is converted by a nonlinear function f and applied to the Poisson model as the instantaneous conditional firing probability λ_t for that particular direction in space $p(spike \mid \mathbf{k} \cdot \mathbf{x})$. The filter weights are obtained optimally by least squares as $\mathbf{k} = (E[\mathbf{x}_{t+lag}{}^T \mathbf{x}_{t+lag}] + \alpha)^{-1} E_{\mathbf{x}_{t+lag}|spike_t}[\mathbf{x}_{t+lag}]$, where $E_{\mathbf{x}_{t+lag}|spike_t}[\mathbf{x}_{t+lag}]$ is the conditional expectation of the kinematic data given the spikes. The parameter α is a regularization parameter to properly condition the inverse. The optimal linear filter actually projects the multi-dimensional kinematic vectors along the direction where they differ the most from the spike triggered kinematic vectors.

The nonlinear encoding function f for each neuron was estimated using an intuitive nonparametric technique [Chichilnisky, 2001; Simoncelli *et al*. 2004]. Given the linear filter vector $\mathbf{k}$, we drew the histogram of all the kinematics vectors filtered by $\mathbf{k}$, and smoothed the histogram by convolving with a Gaussian kernel. The same procedure was repeated to draw the smoothed histogram for the outputs of the spike-triggered velocity vectors filtered by $\mathbf{k}$. The nonlinear function f, which gives the conditional instantaneous firing rate to the Poisson spike-generating model, was then estimated as the ratio of the two smoothed histograms. Since f is estimated from real data by the nonparametric technique, it provides a more accurate nonlinear property than just assuming the exponential or Gaussian function. In practice, it can be implemented as a look up table for its evaluation in testing as

$$p(spike \mid \mathbf{k} \cdot \mathbf{x}^t_{test}) = \frac{\sum_j k(\mathbf{k} \cdot \mathbf{x}^t_{test} - \mathbf{k} \cdot \mathbf{x}^j_{spike,training})}{\sum_i k(\mathbf{k} \cdot \mathbf{x}^t_{test} - \mathbf{k} \cdot \mathbf{x}^i_{training})} \tag{25}$$

where k is the Gaussian kernel, $\mathbf{x}^t_{test}$ is a possible sample we generate at time t in the test data. $\mathbf{x}^i_{training}$ is one sample of velocity vector in the training data, and $\mathbf{x}^j_{spike,training}$ is corresponding spike-triggered sample. The causal time delay is obtained by maximizing the mutual information as a function of time lag for each neuron from 10000 continuous samples of the kinematic variables [Wang 2008].

The instantaneous LNP model was developed to enable online decoding for BMIs. In [Wang 2008] we compared the decoding performance of the Monet Carlo decoding algorithm using the traditional and the instantaneous LNP encoding models. Based on our analysis the instantaneous LNP model produces better decoding results and estimates a tuning curve with a shape between the exponential and linear curves, and has the best

decoding results among the 3 approaches.

In the decoding process, we further assume that the firing rates of all the neuron channels are conditionally independent. The whole Sequential Monte Carlo estimation (SMC) algorithm with the encoding and decoding process can be specified in the following steps:

Step 1: Preprocess and analysis.

1. Generate spike trains from stored spike times.
2. Synchronize all the kinematics with the spike trains.
3. Assign the kinematic vector $\mathbf{x}$ to be reconstructed.

Step 2: Model estimation (encoding).

1. Estimate the kinematic dynamics of the system model $F_k = (E[\mathbf{x}_{k-1}\mathbf{x}_{k-1}^T])^{-1} E[\mathbf{x}_{k-1}\mathbf{x}_k^T]$
2. For each neuron j, estimate the tuning function
    1) Linear model $\mathbf{k}^j = (E[\mathbf{x}^T\mathbf{x}] + \alpha I)^{-1} E_{\mathbf{x}|spike^j}[\mathbf{x}]$
    2) Nonlinear function $f^j(\mathbf{k}^j \cdot \mathbf{x}) = \dfrac{p(spike^j, \mathbf{k}^j \cdot \mathbf{x})}{p(\mathbf{k}^j \cdot \mathbf{x})}$
    3) Implement the inhomogeneous Poisson generator

S***tep 3***: Sequential Monte Carlo estimation of the kinematics (decoding)

For each time k, a set of samples for state $\mathbf{x}_k^i$ are generated, i=1:N

1. Predict new state samples $\mathbf{x}_k^i = F_k \mathbf{x}_{k-1}^i + \eta_k$, i=*1:N*
2. For each neuron j,
    1) Estimate the conditional firing rate $\lambda_k^{i,j} = f^j(\mathbf{k}^j \cdot \mathbf{x}_k^i)$, i=1:N
    2) Update the weights $w_k^{i,j} \propto p(\Delta N_k^j \mid \lambda_t^{i,j})$, i=1:N

3. Draw the weight for the joint posterior density $W_k^i = \prod_j w_k^{i,j}$, i=1:N

4. Normalize the weights $W_k^i = \frac{W_k^i}{\sum_i W_k^i}$, i=1:N

5. Draw the joint posterior density $p(\mathbf{x}_k \mid N_{1:k}) \approx \sum_{i=1}^{N} W_k^i \cdot k(\mathbf{x}_k - \mathbf{x}_k^i)$

6. Estimate the state $\mathbf{x}_k^*$ from the joint posterior density by MAP or expectation.

7. Resample $\mathbf{x}_k^i$ according to the weights W_k^i.

Notice that state vector $\mathbf{x}_k^*$ is estimated from the joint pdf one coordinate at a time. Although one of the advantages of using Monte Carlo estimation is to fully retrieve the joint space of the multi-dimensional pdf, our results show that many more samples are required to explore the joint space appropriately, hence we present results for the estimation of the posterior density for each dimension separately. The decoding performance would also enable us to analyze the 2D error separately.

There is still an important issue that needs to be addressed in this class of real time neural models. The SMC estimation for point processes contains two sources of stochasticity: the generation of the samples to reconstruct the posterior density and the very nature of the single neuron firings that is modeled as a Poisson point process. While the former was dealt with the Monte Carlo method (averaging several realizations), the later is still present in our results since the decoding method is based on a single spike from every neuron probed by the array to meet the real time constraint. If one could probe all the neurons of a given cell assembly responsible for a component of the movement, and average over the ensemble, this stochasticity would be eliminated,

however this is practically impossible. Therefore, the coarse sampling has two basic consequences: First, the SMC estimation model output will have an error produced by not observing all the relevant neural data. This problem will always be present due to the huge difference in the number of motor cortex neurons and electrodes. Second, even when a given neural assembly is probed by one or a few neurons, it is still not possible to achieve accurate modeling due to the stochasticity embedded in the time structure of single spike trains.

This means that every neuron belonging to the same neural assembly will display slightly different spike timing, although they share the same intensity function. Since each probed neuron drives an observation model in the BMI directly, there will be a stochastic term in the output of the BMI (kinematics estimation) that can only be removed by averaging over the neural assembly populations. However, we can attempt to decrease this variance by *estimating* the intensity function from the probed neuron and from it generate several *synthetic spike trains*, use them in the observation model and average the corresponding estimated kinematics. Since this averaging is done in the movement domain (and if the process would not incur a bias in the estimation of the intensity function) the time resolution of the intensity function would be preserved, while the variance would be decreased. We call this procedure *synthetic averaging* and it attempts to mimic the population effect in the cortical assemblies. Synthetic averaging is rather different and less severe from the time average that is operated in binning.

The synthetic spike trains are generated by an inhomogeneous Poisson process with a mean value given by the estimated intensity function obtained by kernel smoothing. This is repeated for each neuron in the array. During testing these synthetic spike trains play

the same role as the true spike trains to predict the kinematics on-line. Of course this will increase the computation time proportionally to the number of synthetic spike trains created. In a sense we are trying to use computer power to offset the limitations of probing relatively few neurons in the cortex. Since the errors in prediction have a bias and a variance which are not quantified, it is unclear how much better the performance will become with synthetic averaging, but this will be addressed in the validation.

3.2.3 Decoding Results

The point process adaptive filtering with linear observation model and Gaussian assumption (Kalman filter PP) and the proposed SMCE framework were both tested and compared in a BMI experiment for the 2-D control of a computer cursor using 185 cortical neurons [Nicolelis *et al.* 1997, Wessberg *et al.* 2000]. For each neuron in the ensemble, an optimum time interval of 10 ms was selected to construct the point process observation sequence. With this interval, 94.1% of the intervals with spikes had only a single spike. For each time interval and in each channel, 1 was assigned when there were one or more spikes, otherwise 0 was assigned.

After data preprocessing, all the parameters are estimated from 10000 training samples. The kinematic model F_k for both algorithms can be estimated using least squares. The non-zero coefficients in F_k are found to be close to 1 or –1, which verifies the consistency between the estimated acceleration, velocity and positions. For example, we checked the differentiated signal from estimated position x. After low-pass filtering to removed the noise due to the differentiation, it has a CC of 0.9117 with the estimated velocity x. Notice that the choice of parameters in the noise estimation (covariance Q_k in Kalman filter PP and the noise distribution $p(\eta)$ in SMCE) affects the algorithm

performance. However, since we have no access to the desired kinematics in the test data set, these parameters in both algorithms were obtained from the training data sets. For the Kalman filter PP, the noise in the kinematics model (6) is approximated by a Gaussian distribution with covariance Q_k. In the SMCE model, the noise distribution $p(\eta)$ is approximated by the histogram of $\eta_k = \mathbf{x}_k - F_k \mathbf{x}_{k-1}$. The resolution parameter was experimentally set at 100 to approximate the noise distribution. The regularization factor α in the tuning function was experimentally set at 10^{-7} for this analysis. The remaining parameters in SMC estimation include the kernel size σ selected at 0.02. The number of particles x_n is set for a reasonable compromise between computational time and estimation performance. By studying the decoding performances using different number of samples, we found out that when x_n is experimentally set at 1000, performance converges. This kernel size should be chosen carefully to not lose the characteristics of the tuning curve, while still minimizing ripples in the estimated density.

The Kalman filter PP has an analytical solution. We set the initial state $\mathbf{x}_0$ to be a zero vector and the state variance $P_{0|0}$ is estimated from the training data. Once the initial condition and parameters are set, the state estimation is determined uniquely by the spike observations. However, the SMCE approach introduces variations between realizations even with fixed parameters due to the estimation of the posterior distribution with the particles.

Figure 7 shows the 2D reconstruction of position for a segment of data, where the dot-dashed line is the desired signal, the dashed line is the reconstruction by Kalman filter PP, and the solid line is the reconstruction by SMC estimation. We can see that the

SMC estimation follows the desired signal with less error and more smoothly. We also notice that the performance for x and y are different, therefore we analyze the reconstruction error for x and y separately and then in 2D space.

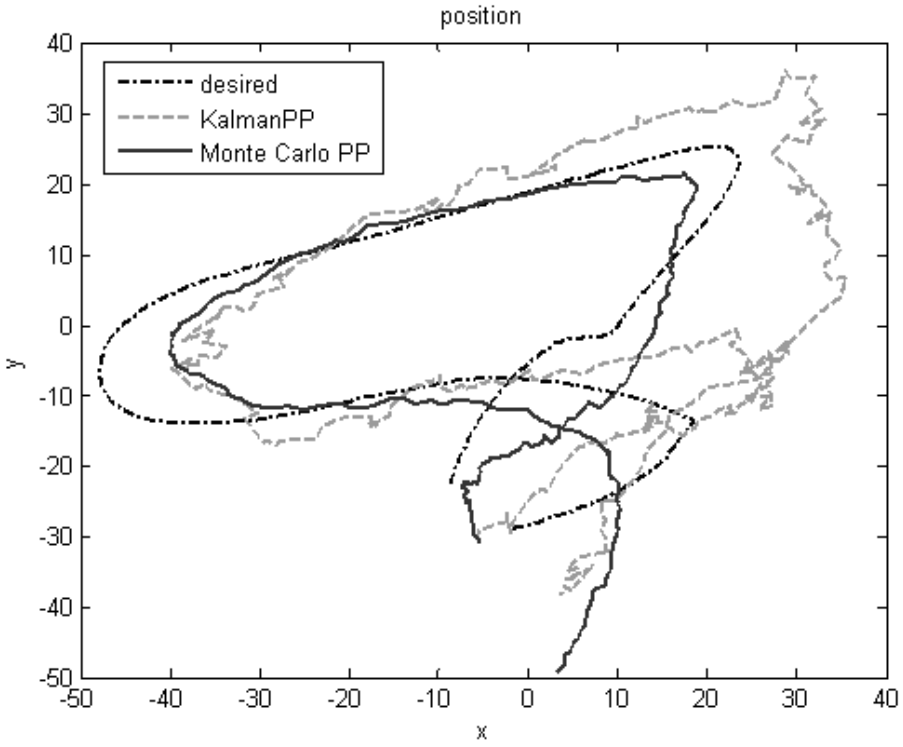

Figure 7 The 2D reconstruction of the position by 2 approaches

Table 2 shows reconstruction results on a 1000 sample (10 seconds) segment of a test segment (shown in Figure 7) of neural data. Correlation Coefficients (CC) and Normalized Mean Square Error (MSE normalized by the power of the desired signal) between the desired signal and the estimations for each kinematic variable and direction are evaluated for the Kalman filter PP as well as for the SMC estimation using 20 realizations of the posterior. For the second approach we also show the mean and the standard derivation among realizations, together with the best and the worst performance obtained by single realization.

Table 2 Results Comparison of the Kinematics Reconstructions by 2 methods for segment of data

Method	Criterion		Position		Velocity		Acceleration	
			x	y	x	y	x	y
Kalman filter PP	CC		0.74	0.83	0.74	0.68	0.42	0.18
	NMSE		0.81	1.51	0.50	0.77	0.95	1.13
SMCE	CC	mean $\pm$ std	0.81 ± 0.01	0.83 ± 0.01	0.79 ± 0.01	0.74 ± 0.01	0.45 ± 0.01	0.25 ± 0.01
		Best	0.83	0.84	0.80	0.74	0.47	0.26
		Worst	0.79	0.83	0.78	0.73	0.44	0.25
	NMSE	mean $\pm$ std	0.44 ± 0.03	0.98 ± 0.14	0.45 ± 0.02	0.55 ± 0.01	0.82 ± 0.01	1.03 ± 0.01
		Best	0.40	0.74	0.45	0.54	0.81	1.04
		Worst	0.43	1.30	0.44	0.54	0.81	1.02

Both approaches resulted in reasonable reconstructions of the position and the velocity. The position shows the best correlation coefficient with the true trajectory. This result may be due to the fact that the velocity and the acceleration were derived as differential variables, where the noise in the estimation might be magnified. Although the Kalman filter PP assumes a Gaussian posterior and a simple linear model for both the kinematic dynamic system and the tuning function, it obtains a reasonable reconstruction of the position and the velocity. For the position, CC= 0.7422 was for the x direction and CC= 0.8264 for the y direction. The velocity shows a CC = 0.7416 for x and CC = 0.6813 for y. For SMC estimation one obtains the tuning function nonlinearity for each neuron from the training data and estimates the kinematics without any restriction on the posterior density. The average correlation for the position along x is 0.8058 ± 0.0111 and along y is 0.8396 ± 0.0124. The average correlation for the velocity along x is 0.7945 ± 0.0104 and along y is 0.7381 ± 0.0057. We notice that although SMC estimation introduces differences on the reconstruction among realizations due to stochasticity, the

variance of the results is pretty small. Even the worst result we obtain is better than the Kalman filter PP in terms of both CC and NMSE.

Since the desired signal in the test set data is formally unknown, it is not reasonable to just pick the best realization to present the reconstruction results. Here, we choose the averaged performance among realizations as the reconstruction results by SMC estimation, and compare with the Kalman filter PP results.

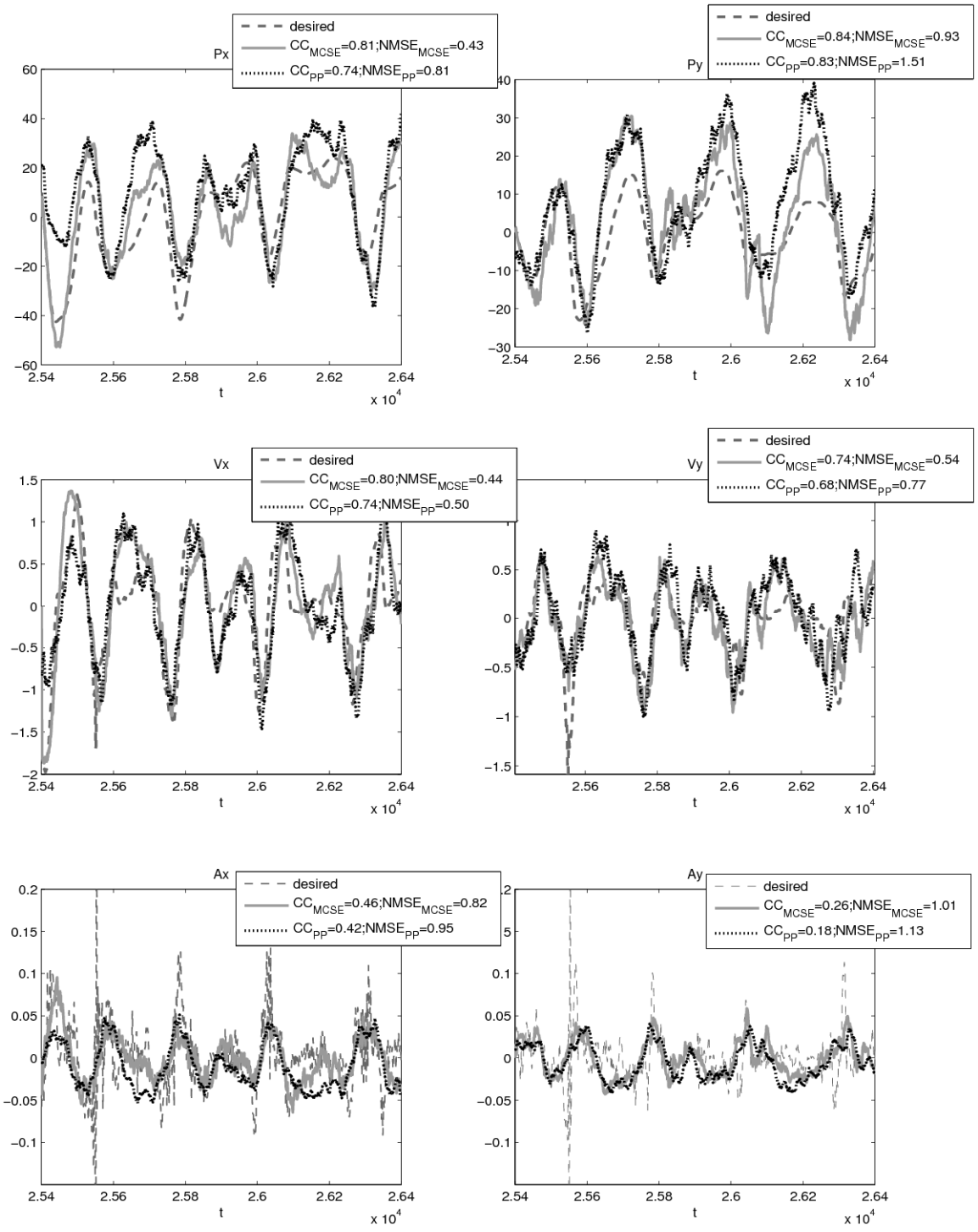

Figure 8 The reconstructed kinematics for a 2-D reaching task

Figure 8 shows the reconstructed kinematics using both algorithms from all 185 neurons for 1000 testing samples (10 seconds). The left and right panels depict

respectively the reconstructed kinematics for x-axis and y-axis. The 3 rows of plots from top to bottom display respectively the reconstructed position, the velocity and the acceleration. In each subplot, the dash line indicates the desired signal, the solid line indicates the estimation by SMCE, and the dotted line indicates the estimation by Kalman filtering PP. The Monte Carlo approach offers the most consistent reconstruction in terms of both correlation coefficient and normalized mean square error. Figure 9 zooms in the first 100 samples of the reconstructed kinematics to show better the modeling accuracy. The dash line is the desired signal the same as in figure 8. The solid line is the reconstructed kinematics by one trial of SMC estimation. The gray area in each plot represents the posterior density estimated by the algorithm over time where the darker areas represent a higher value. As the value of the posterior density decreases to 0, the color of the dots will fade to white. Figure 9 shows the SMC estimation effectiveness to generate samples whose density follows the trajectory. The desired signal falls almost always within the high probability range of the posterior density, which demonstrates the good tracking ability of SMC.

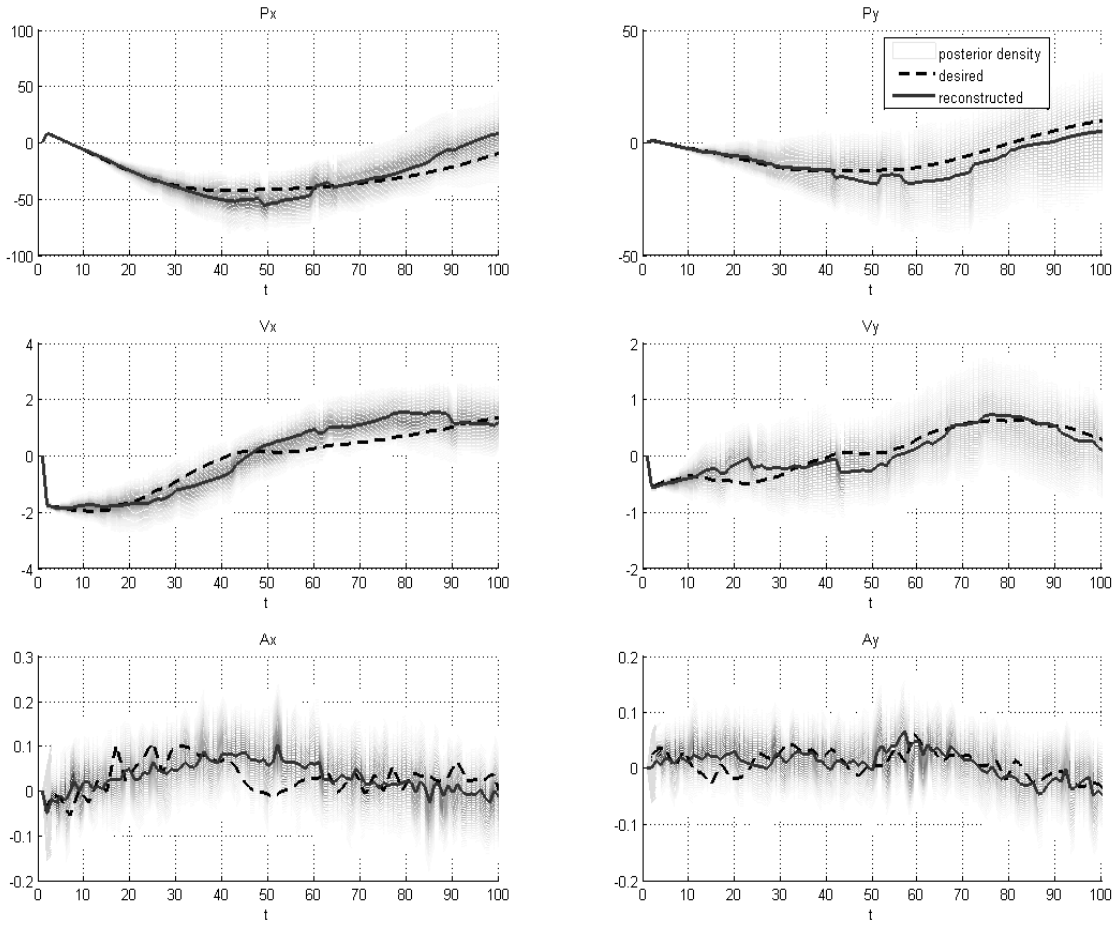

Figure 9 The posterior density of the reconstructed kinematics by SMCE

The simulation of both models with synthetic data provides important hints on how to interpret the results with real neural data. The linear tuning model of the Kalman filter PP provides less accuracy in the nonlinear region of the tuning function, which in turn affects the decoding performance. Moreover, the Kalman filter PP also assumes a Gaussian posterior density, therefore both algorithms provide similar velocity estimation along y when both assumptions are verified. When the estimation from the two algorithms are different (which often occurs at the peak of the desired signal), the SMC estimation model usually provides better performance, which is due to either its better

modeling of the neuron's nonlinear tuning and/or its ability to track the non-Gaussian posterior density better.

Figure 10 shows the posterior density of the position x given the spikes at the time index 216.25 seconds as an example of the non-Gaussian distribution. The pdf shape is estimated by Parzen windowing with 1000 samples. The kernel size to smooth the samples is chosen according to the Silverman's rule [Silverman 1981]. The pdf retrieved by Kalman filter PP is shown with a dashed line, and the pdf by SMCE is shown with the solid line. The pdf by SMCE in this image has multi-modes, which is certainly not Gaussian distributed. In order to statistically check the normality of posterior density along time, we performed Lilliefors test [Lilliefors, 1967] on the pdf shown in figure 8 for each kinematic variable. The test is performed as a goodness-of-fit test against the alternative that the kinematic samples do not come from a normal distribution. In contrast to Kolmogorov-Smirnov test, which requires that the null distribution be completely specified, the parameters of the null distribution (Gaussian) must be estimated from the samples. At $\alpha = 0.05$ significance level, 68% and 67% (respectively x and y) of the position pdf along time rejects the null hypothesis, which means they are not Gaussian distributed. For velocity, 39% and 49% (respectively x and y) of the pdf along time are not Gaussian distributed. For acceleration, 14% and 32% (respectively x and y) of the acceleration pdf along time are not Gaussian distributed. These results indicate the necessary to evaluate the pdf without Gaussian assumptions.

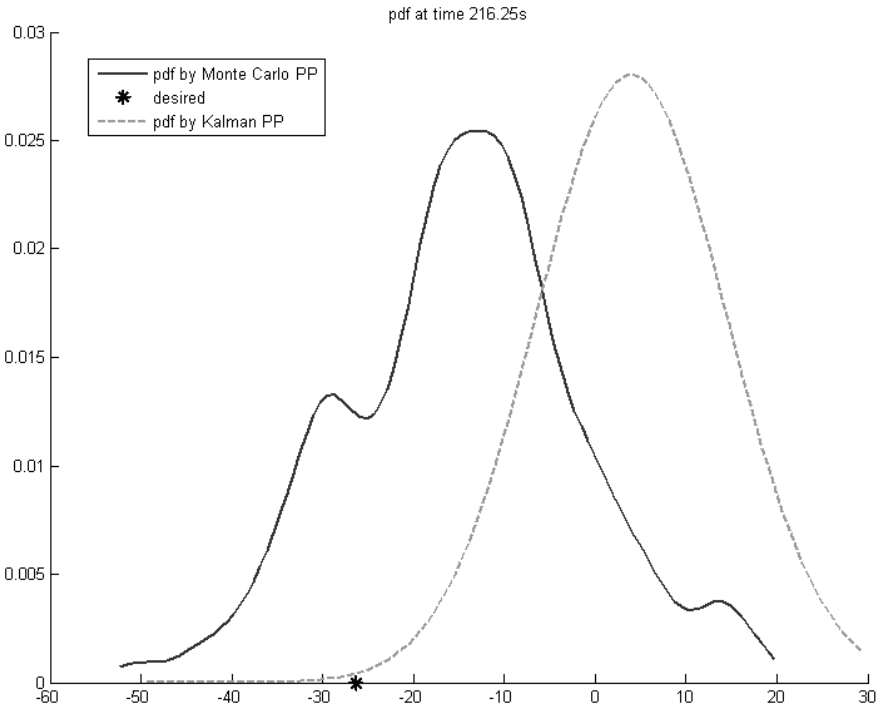

Figure 10 The posterior density of the reconstructed kinematics by SMCE and Kalman PP

We can also notice that the desired signal (star) is located within the main range of the pdf distribution by SMCE, while only the left tail of the Kalman filter PP pdf covers the desired signal with smaller probability density. The bias between the two pdf distributions is observed most of the time. It indicates a smaller accuracy of the linear tuning by Kalman filter PP and the error due to the less accurate pdf information propagated along time.

As we analyzed in the previous section, in order to deal with the intrinsic stochasticity due to the randomness of the spike trains, we proposed the synthetic averaging idea to mimic the neuron population effect. Instead of decoding only from current spike trains, we use a Poisson generator to obtain 20 sets of spike trains from each neuron as synthetic

plausible observations to represent the neuron ensemble firing with the same intensity function. This firing intensity function is estimated by kernel smoothing from each recorded spike train. The kernel size is experimentally set as 0.17 as the one used in the Kalman filter PP. In order to preserve the timing resolution as much as possible, the averaging is performed across the estimated kinematics of each group (including the output of the true spike train). Table 3 shows the comparison results of the performance by SMC estimation averaged among 20 realizations on recorded spike trains and the average decoding performance from 20 sets of re-generated spike trains (1 Monte Carlo trials for each set), and the averaged performance over Monte Carlo and synthetic data (20 sets re-generated spike trains, 20 Monte Carlo trials for each set) in the same segment of test data.

Table 3 Results Comparison of the Kinematics Reconstructions averaged among Monte Carlo trials and synthetic averaging

criterion	method	Position		Velocity		Acceleration	
		X	y	X	y	x	y
CC	Average among 20 Monte Carlo trials	0.811	0.837	0.799	0.741	0.456	0.255
	Average among 20 Synthetic spikes, 1 Monte Carlo trials each	0.844	0.833	0.805	0.733	0.468	0.212
	Average among 20 Synthetic spikes, 20 Monte Carlo trials each	0.843	0.852	0.822	0.737	0.443	0.233
NMSE	Average among 20 Monte Carlo trials	0.429	0.933	0.439	0.538	0.817	1.025
	Average among 20 Synthetic spikes 1 Monte Carlo trials each	0.326	0.821	0.359	0.494	0.787	1.013
	Average among 20 Synthetic spikes, 20 Monte Carlo trials each	0.319	0.768	0.330	0.484	0.808	0.990

Both approaches as well as the “deterministic” performance resulted in reconstruction with similar correlation coefficients. However, the average over synthetic data even with only one Monte Carlo trial for each generated spike train sets, shows smoother kinematics reconstruction with reduced NMSE compared to the averaged performance through 20 Monte Carlo trials on original spike trains. When we use 20 Monte Carlo

trials for each set of synthetic spike trains, NMSE reduces 26% for position along x, 18% for position along y, and on average 15% for all 6 kinematic variables. Therefore we can conclude that the reconstruction accuracy measured by NMSE (the gain difference noticeable in Figure 8) has a large component due to the variance intrinsic in the spike firing, but it seems not to affect the general trend of the reconstructed signal as measured by the CC.

This result demonstrates that using the simulated neuronal population attenuates the variability intrinsic in the coarse sampling of a given neural population, effectively trading computation for lack of more neural channels belonging to the same neural population. However, this procedure only reduces the kinematics estimation error that is due to the variance of the recorded spike trains as single random realizations of a stochastic process. It cannot cope with the lack of information produced by the coarse sampling of other neural population involved in the movement but not sampled at all. And in the process, the procedure creates a difficult to quantify modeling bias because the intensity function is estimated from a single neuron. Overall, however, the method gains more than it looses as measured by NMSE. When compared with binning (averaging in time), averaging in the kinematics domain preserves the natural stochasticity of the intensity function and still smoothes the reconstruction.

As for the figure of merit for reconstruction, the correlation coefficient has been the preferred metric to compare movement reconstruction between different experimental data sets in BMIs [Wessberg *et al. 2000*]. However, it may not be sufficient to evaluate the accuracy of BMI algorithm, since a bias in position means that a different point in the external space will be targeted, so the rating criterion should take this bias into

consideration to properly compare reconstruction models. Notice also that the correlation coefficient obtained from the acceleration is pretty low. However, if we qualitatively check the reconstruction results in Figure 8, the algorithm actually follows the trend of the desired signal closely. The problem with the NMSE for BMIs is that the results do not "look as good", with errors sometimes bigger than the power of the trajectory. This can be observed in Figure 8, where the reconstructed position seems to have a different scale than the desired trajectory. Therefore, to compare the algorithm's performance on the same data set, NMSE provides generally better insights on tracking accuracy of the animal's true movement trajectory.

We have to mention that the improvement of the decoding results comes at the cost of computational complexity. The full access of the posterior density within the nonlinear system needs many Monte Carlo samples. Figure 11 shows the decoding performance vs. the number of samples x_n for each kinematic variable. The left and right panels show respectively the reconstruction results (CC in cross and NMSE in circlet) for x-axis and y-axis. The 3 rows of plots from top to bottom display respectively the reconstruction results of position, the velocity and the acceleration. We can see that the NMSE shows clearer trends (decrease) than CC when there are more Monte Carlo samples. It indicates that when there are enough samples to fully explore the probability space, the algorithm is able to capture the detailed change in the kinematics without affecting the general trend of the reconstructed signal as measured by the CC. Although the performance can still be enhanced by increase the Monte Carlo sample number, this comes at the expense of higher computation in calculating the posterior density for each point. There should be a tradeoff between the performance improvement and bearable computational complexity.

The appropriate number of samples should be chosen as the performance starts to converge while the computation time is still acceptable for on-line implementation.

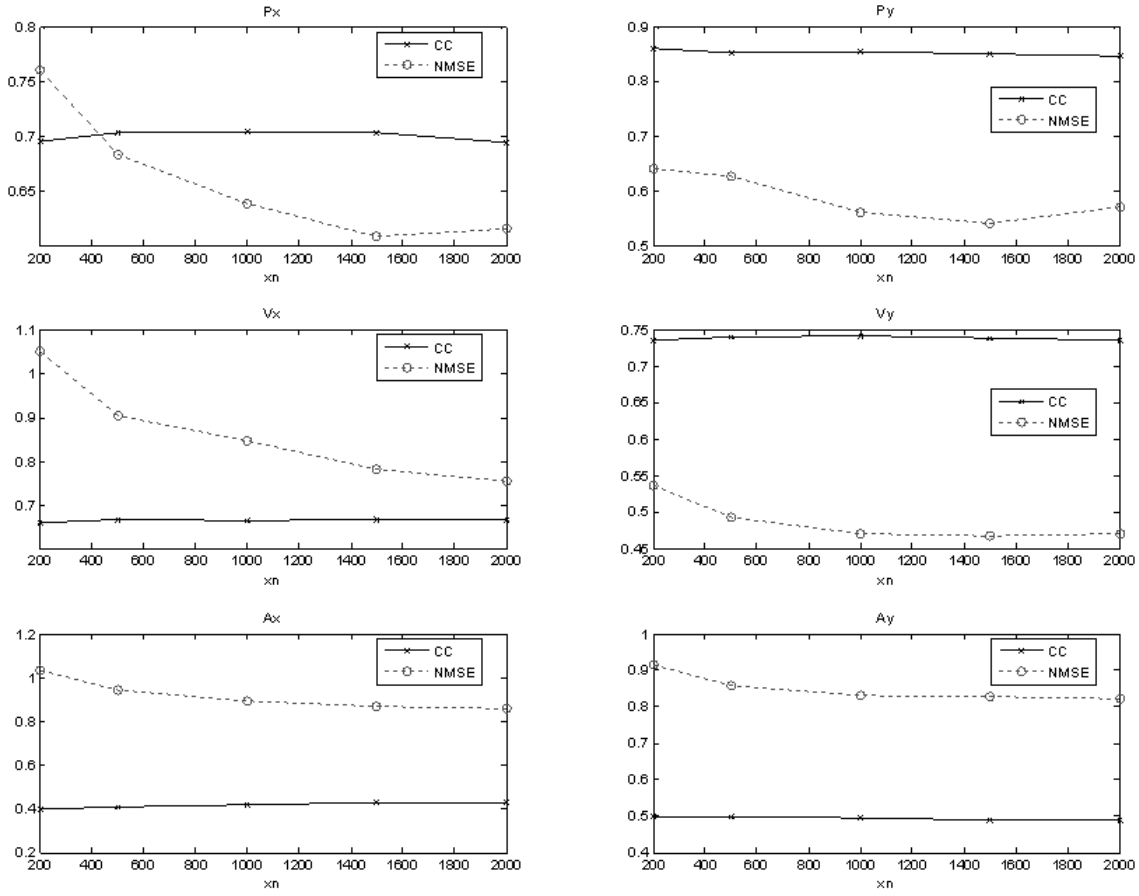

Figure 11 The decoding performances versus the number of Monte Carlo samples.

We further compared the statistical performance of both algorithms on 8000 samples of test neural data (80 sec). The performance averaged among the decoding results from 20 sets re-generated spike trains is chosen as the reconstruction result by SMCE. Due to the computation burden, we only run the decoding process for 1 Monte Carlo trial on each set of synthetic spike trains. NMSE for each dimension and the 2D error (RMSE normalized by the diameter of the 2D range) were both evaluated with an 800 sample-long window with 50% overlap. For each segment of data, pair-wise student *t*-test was

performed to see if the results are statistically different from the Kalman filter PP. The test is performed against the alternative specified by the right tail test $NMSE_{Kalman}>NMSE_{SMCE}$ and the right tail test $RMSE_{Kalman}>RMSE_{SMCE}$ for each kinematic variable. All the tests are performed on the null hypothesis at $\alpha = 0.05$ significance level. Under the null hypothesis, the probability of observing a value equal or higher in the test statistic, as indicated by the *p*-value, is shown in table 4.

Table 4 statistical performance of the Kinematics Reconstructions by 2 methods

	Method	Position		Velocity		Acceleration	
		x	y	x	y	x	y
NMSE	Kalman filter PP	0.897 ± 0.305	1.043 ± 0.245	0.673 ± 0.271	0.686 ± 0.172	0.891 ± 0.187	1.085 ± 0.385
	SMCE	0.466 ± 0.181	0.887 ± 0.301	0.436 ± 0.124	0.446 ± 0.133	0.755 ± 0.168	0.947 ± 0.380
t-test	H_1: $NMSE_{Kalman}>NMSE_{SMCE}$ (*p*_value)	1(0)	1(0.023)	1(0)	1(0)	1(0)	1(0)

The t-test verifies that the reconstruction with SMC estimation is statistically better than the Kalman filter PP for all kinematic variables. In terms of the averaged performance, the reconstruction of position improves 48% and 15% respectively for *x* and *y*; the reconstruction of velocity improves 34% and 35% respectively for *x* and *y*; the reconstruction of acceleration improves 14% and 12% respectively for *x* and *y*.

The statistical 2D error is also shown for the same data segment between the reconstructions and desired signal normalized by the diameter of the movement range for each kinematic variable in table 5. It provides an idea in space how good the performance should be. Not only the results by Kalman filter PP and SMC estimation, but also the linear decoding results (Wiener filter) on the binned spikes (100 ms window) are shown here as a reference. The pair-wise *t*-test was also performed against the alternative specified by the right tail test $RMSE_{Wiener}>RMSE_{SMCE}$ and $RMSE_{Kalman}>RMSE_{SMCE}$ for

each kinematic variable at $\alpha = 0.05$ significance level.

Table 5 statistical performance of the Kinematics Reconstructions by different approaches

	Method	Position		Velocity		Acceleration	
		x	y	x	y	x	y
RMSE -2D	Wiener filter on binned spike	0.2687 ± 0.0473		0.1739 ± 0.0215		0.1828 ± 0.0162	
	Kalman filter PP	0.2444 ± 0.0414		0.1395 ± 0.0238		0.1102 ± 0.0293	
	SMC estimation	0.1951 ± 0.0414		0.1156 ± 0.0163		0.0983 ± 0.0262	

All the t-tests successfully reject the null hypothesis, which verifies that reconstruction by SMC estimation is statistically better in 2D than the Kalman filter PP and Linear decoding method using binned spikes for all kinematic variables. In terms of the averaged performance compared to the Wiener filter, the reconstruction of position improves 27.36%, the reconstruction of velocity improves 32.74%, and the reconstruction of acceleration improves 46.18%. Comparing to the Kalman filter PP, the reconstruction of position improves 20.14%, the reconstruction of velocity improves 17.10%, and the reconstruction of acceleration improves 10.75%. In addition, both of the approaches in spike domain outperform the Wiener filter in continuous domain, which shows the potential advantage of decoding with the spike timing.

4. Discussion and Conclusion

Although spike trains are very telling of neuronal function, they are also very removed from the macroscopic time scales of behavior. Therefore, a central question in modeling brain function in behavior experiments is how to optimally bridge the time scale between spike events (milliseconds) and the time scale of behavior (seconds). Most often, the relatively rudimentary method of time averaging (binning spikes) is used to bridge this

gap, but excludes the rich information embedded in the high resolution of the spike representation. Model-based methodologies including an encoding model linking the firing times to state variables as the ones presented in this paper seem to be a much more principled way to model the hierarchy of scales present in the nervous system. However, these models are intrinsically stochastic with the encoding models in use today, so they pose difficulties for real time operation of BMI models.

Here it was shown how a SMCE framework could be used as a probabilistic approach to reconstruct the kinematics directly from the multi-channel neural spike trains. We characterize the neuron tuning (encoding) properties to describe the functional relationship between each neuron firing and movement, using a parametric instantaneous linear-nonlinear-Poisson model [Wang 2008]. Comparing the decoding performances by linear and exponential tuning model within the same decoding algorithm, the instantaneous LNP model provides more reasonable tuning curves and better decoding results. However, further development and validation of this encoding model is an important aspect to consider because it directly affects the decoding performance.

With this encoding information, a novel signal processing algorithm based on Sequential Monte Carlo estimation was applied directly to point processes to convert the decoding problem of a Brain Machine Interface system into a problem of state sequential estimation. These signal-processing techniques directly draw information from timing of discrete event without a Gaussian assumption. In simulations, we showed that SMC estimation provided a better approximation of the evolution of the posterior density without any constrains when compared with adaptive filtering of point process under a Gaussian assumption. The results show the important of directly propagating the full

posterior over time instead of the Gaussian approximation commonly used, since the latter disregards important information in the posterior about the kinematics evolution. In our approach the final state is still derived from the non-parametrically reconstructed posterior by collapse, which indirectly assumes a Gaussian pdf model. Minor improvements can perhaps be gained if a more powerful estimation of the peak of the posterior is utilized to estimate the state. Although proposed for application in BMI, the modeling methodology is a general non-parametric approach that can infer continuous signals from point process without constraints, which potentially can be utilized in many other neuroscience applications (e.g. visual cortex processing), in communications (network traffic) and in process optimization. On the other hand, since pdf information is fully stored and propagated for each time index, the computation complexity is one of the trade-offs that the user must consider. We were able to pin point and quantify for motor BMIs the performance cap associated with the Gaussian assumption. Towards this goal, we compared performance with the Kalman filter PP applied to a cursor control task, and concluded that the SMC PP framework showed statistically better results between the desired and estimated trajectory. From the t-test results, the SMC estimation is significantly better than the Kalman filter PP and also the traditional linear decoding using spike rates. This seems to indicate that the real advantage is in the better neural modeling achieved with the tuning curve and the more accurate tracking due to the non-parametric estimation of the posterior.

Although the results are interesting, the signal processing methodologies for spike train modeling need to be further developed. They are substantially more complex than the ones for random processes, and many parameters are assumed and/or need estimation

with significant design expertise, and at the end of the day, the results are still intrinsically stochastic due to the randomness of the generated spike trains in the encoding model. In order to achieve more reliable results, we propose an averaging idea to artificially generate several sets of spike trains from the estimated firing intensity probability of the neurons to simulate the population effects in the cortex. The model is augmented with synthetically generated spike train realizations for the reconstruction of the kinematics. The performance is averaged among the decoding results in the movement domain to bypass the possible distortion in the nonlinear tuning function due to the binning. The synthetic averaging is an attempt to reduce the variance of the spike timing introduced by single realization of the neuron recordings but it also helps bridge the time-resolution difference between neuron activity and the kinematics, because the time scale of the intensity function defines the time resolution of the results. Since the synthetic averaging provided smoother kinematics reconstruction that agreed better with the measured movement, the variance of the estimation seems to be the limiting factor, i.e. it overshadowed the model bias created by the synthetic spike train generation. In other words, these results show that the time resolution of the spike trains seems an overkill for the kinematic reconstruction in these types of cursor tracking tasks, otherwise, when the synthetic copies were created with a stochastic timing consistent with the intensity function, the results would have been worse. If the goal of the analysis included a mapping of spike trains to electromyographic activity, where time resolution is essential, the conclusion may have been different. Therefore, spike timing models are more versatile spanning multiple time scales but the goal of the physiologic experiment ultimately determines the time resolution necessary. For motor BMIs a neural tuning

model for the time scale of the intensity function seems essential to reduce computation.

Another direction for further development is to remove the need to discretize the operation of the particle filter, i.e. remove the small binning interval used to convert the spike train into a binary string. This could potentially reduce the computational complexity due to the sparsity of spike events, however it is still not totally clear how the method proposed in [Del Moral P., Doucet A & Jasra A, 2006] affects the observation model nor how the non-uniform sampling will affect the mathematical analysis.

Acknowledgments

This work was supported by NSF grants ECS-0422718, CISE-0541241 and CNS-0540304 and the Children's Miracle Network. A. R. C. Paiva was supported by Fundação para a Ciência e a Tecnologia under grant SFRH/BD/18217/2004. The authors would like to thank Dr. Miguel Nicolelis for the use of the monkey data collected in his laboratory at Duke University.

References

Arulampalam, M. S., Maskell, S, Gordon, N., and Clapp T., (2002) A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian Tracking. *IEEE Trans. on Sig. Proc*. 50(2): p. 174-188.

Brockwell, A. E., Rojas A. L., and Kaas, R. E., (2003), Recursive Bayesian decoding of motor cortical signals by particle filtering. *Journal of Neurophysiology*, 91: p. 1899-1907.

Barbieri, R., Frank, L. M., Nguyen, D. P., Quirk, M. C., Solo, V., Wilson, M. A., and Brown, E. N., (2004) Dynamic analyses of information encoding in neural ensembles. *Neural Computation*, 16: 277-307

Bergman, N. (1999) *Recursive Bayesian estimation: navigation and tracking applications*, Linkoping University: Linkoping, Sweden.

Brown, E. N., Frank, L. M., Tang, D., Quirk, M. C., and Wilson, M. A. (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*, 18:7411-25

Brown, E. N., Nguyen, D. P., Frank, L. M., Wilson, M. A., and Solo, V. (2001). An analysis of neural receptive field plasticity by point process adaptive filtering. *PNAS.* 98(12): p. 12 261-12 266.

Brown, E. N., Barbieri, R., Ventura, V., Kass, R., Frank, L. M., (2002) The Time-Rescaling Theorem and Its Application to Neural Spike Train Data Analysis. 325-346.

Carmena, J. M., Lebedev, M. A., Crist, R. E., O'Doherty, J. E., Santucci, D. M., Dimitrov, D. F., Patil P. G, Henriquez, C. S., Nicolelis, M. A. L. (2003), learning to control a brain machine interface for reaching and grasping by primates, *PLoS Biology*, vol. 1, issue 2, 193-208

Carpenter, J., Clifford, P., and Fearnhead, P. (1999) Improved particle filter for nonlinear problems. *IEEE Proc. Radar Sonar Navigation*. 146: p. 2-7.

Chan, K. S., & Ledolter J. (1995) *Monte Carlo estimation for time series models involving counts*. J. Am. Stat. Assoc., 90, 242–252

Chichilnisky E. J. (2001), A simple white noise analysis of neuronal light responses, *Network: Comput. Neural Syst.* 12: 199-213

Daley, D., & Vere-Jones, D. (1988). An introduction to the theory of point process. New York: Springer-Verlag.

Diggle, P. J., Liang, K-Y., Zeger S. L (1995), *Analysis of longitudinal data*. Oxford: Clarendon

Del Moral P., Doucet A & Jasra A (2006), Sequential Monte Carlo Samplers, *J. Royal Statistical Society B*, 68 (3): 411-436,.

Doucet, A., (1998) *On sequential simulation-based methods for bayesian filtering*, University of Cambridge.

Doucet, A., de Freitas, N., and Gordon N. (2001) *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2001.

Eden, U. T., Frank, L. M., Solo, V., Brown, E. N. (2004). Dynamic Analysis of Neural Encoding by Point Process Adaptive Filtering. *Neural Comp*.,16: p. 971-998

Ergun, A., Barbieri, R, Eden, U. T., Wilson, M. A., Brown, E. N.,(2007) Construction of point process adaptive filter algorithms for neural systems using sequential Monte Carlo methods. *IEEE Transactions on Biomedical Engineering*, 54(3):419-428.

Gabbiani F, Koch C. (1998), Principles of spike train analysis. In: Koch C, Segev I, editors. *Methods in Neuronal Modeling: From Ions to Networks*, 2nd edition. Cambridge MA: MIT, 313–60

Gordon, N. J., Salmond, D. J., and Smith A., F., M., (1993) Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings on Radar and Signal Processing*, 140: p. 107-113.

Kim, S. P., Sanchez, J. C., Erdogmus, D., Rao, Y. N., Wessberg, J., Principe, J. C., & Nicolelis M. A. (2003) Divide-and-conquer approach for brain machine interfaces: nonlinear mixture of competitive linear models. *Neural Networks*, 16, pp. 865-871

Kim S. P. (2005), *Design and analysis of optimal encoding models for brain machine interfaces*, PhD. Dissertation, University of Florida

Lilliefors, H.W. (1967) On the Komogorov-Smirnov test for normality with mean and variance unknown, *Journal of the American Statistical Association*, 62, 399-402.

Moran, D.W. and Schwartz, A. B., (1999) Motor cortical representation of speed and direction during reaching. *Journal of Neurophysiology*,. 82(5): p. 2676-2692.

Nicolelis, M. A. L., Ghazanfar, A. A., Faggin, B., Votaw, S., & Oliveira, L. M. O. (1997) Reconstructing the engram: simultaneous, multiple site, many single neuron recordings. *Neuron* 18, pp. 529-537.

Paninski, L., Shoham, S., Fellows, M. R., Hatsopoulos, N. G., and Donoghue, G. P., (2004) Superlinear population encoding of dynamic hand trajectory in primary motor cortex. *J. Neurosci*. 24(39): p. 8551-8561.

Parzen, E., (1962) On the estimation of a probability density function and the mode. *Ann. Mathematical Statistics*. 33(2): p. 1065-1076.

Reich, D. S., Victor, J. D., Knight, B. W. (1998). The power ratio and the interval map: spiking models and extracellular recordings. *J Neurosci*., 18:10090–104

Rieke, F., Warland, D., Steveninck, R. R., Bialek W. (1997), *Spikes: Exploring the Neural Code*. Cambridge, MA: MIT

Sanchez, J. C., Kim, S. P., Erdogmus, D., Rao, Y. N., Principe, J. C., Wessberg, J., & Nicolelis, M. A. (2002) Input-output mapping performance of linear and nonlinear models for estimating hand trajectories from cortical neuronal firing patterns. *Proc. Of Neural Net. Sig. Proc*., pp. 139-148

Schwartz, A. B., Taylor D. M., and Tillery, S. I. H.. (2001). Extraction algorithms for cortical control of arm prosthetics. *Current Opinion in Neurobiology*. 11(6): p. 701-708.

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*, vol. 416, pp. 141-142

Shadlen, M. N., Newsome, W. T. (1998). The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. *J Neurosci*, 18:3870–96

Shoham, S., Paninski, L. M., Fellows, M. R., Hatsopoulos, N. G., Donoghue, J. P., Normann, A., (2005) Statistical encoding model for a primary motor cortical brain-machine interface. *IEEE Trans Biomed Eng*. 52(7):1312-22.

Silverman, B. W., (1981) Using Kernel Density Estimates to Investigate Multimodality *J. Roy. Stat. Soc., Ser. B*. 43 97-99

Simoncelli, E. P., Paninski, L., Pillow, J., and Schwartz, O., (2004) Characterization of neural responses with stochastic stimuli. 3rd ed. *The New Cognitive Neuroscience*. MIT Press.

Smith, A. C., Brown, E. N. (2003), State-space estimation from point process observations. *Neural Computation*, 15, 965-991

Srinivasan, L., Eden, U. T., Mitter, S. K., Brown, E. N. (2007) General purpose filter design for neural prosthetic devices. *Journal of Neurophysiology,* 98(4): 2456-2475.

Tuckwell, H. (1988), *Introduction to Theoretical Neurobiology*, vol. 2. New York: Cambridge University Press

Wang, Y., Paiva, A. R. C., and Principe, J.C. (2006) A Monte Carlo Sequential Estimation for Point Process Optimum Filtering. Neural Networks, 2006 *IJCNN '06*. International joint conference on.16-21 July 2006: 1846-1850

Wang, Y. (2008) Point Process Monte Carlo Filtering for Brain Machine Interfaces, PhD dissertation, University of Florida

Wessberg J, Stambaugh CR, Kralik JD, Beck PD, Laubach M, Chapin JK, Kim J, Biggs SJ, Srinivasan MA, Nicolelis MA (2000). Real-time prediction of hand trajectory by ensembles of cortical neurons in primates, *Nature*, 408:361-365

Wu, W., Black, M. J., Memford, D., Gao, Y., Bienenstock, E., and Donoghue J. P., (2004) Modeling and decoding motor cortical activity using a switching Kalman filter. IEEE Transactions on Biomedical Engineering, 51(6): p. 933-942.

Wu, W., Gao, Y., Bienenstock, E., Donoghue, J. P., Black, M. J. (2006), Bayesian population decoding of motor cortical activity using a Kalman filter, *Neural Comput*.18: 80-118

Yu, B. M., Kemere, C., Santhanam, G., Afshar, A., Ryu, S. I., Meng, T. H., Sahani, M., Shenoy, K. V. (2007) Mixture of trajectory models for neural decoding of goal-directed movements. *J Neurophysiol.* 97(5):3763-80