J Comput Neurosci
DOI 10.1007/s10827-009-0196-9

Comparison of brain–computer interface decoding algorithms in open-loop and closed-loop control

Shinsuke Koyama · Steven M. Chase · Andrew S. Whitford · Meel Velliste · Andrew B. Schwartz · Robert E. Kass

Received: 21 December 2008 / Revised: 2 July 2009 / Accepted: 9 October 2009

Abstract Neuroprosthetic devices such as a computer cursor can be controlled by the activity of cortical neurons when an appropriate algorithm is used to decode motor intention. Algorithms which have been proposed for this purpose range from the simple population vector algorithm (PVA) and optimal linear estimator (OLE) to various versions of Bayesian decoders. Although Bayesian decoders typically provide the most accurate off-line reconstructions, it is not known which model assumptions in these algorithms are critical for improving decoding performance. Furthermore, it is not necessarily true that improvements (or deficits) in off-line reconstruction will translate into improvements (or deficits) in on-line control, as the subject might compensate for the specifics of the decoder in use at the time. Here we show that by comparing the performance of nine decoders, assumptions about uniformly distributed preferred directions and the way the cursor trajectories are smoothed have the most impact on decoder performance in off-line reconstruction, while assumptions about tuning curve linearity and spike count variance play relatively minor roles. In on-line control, subjects compensate for directional biases caused by non-uniformly distributed preferred directions, leaving cursor smoothing differences as the largest single algorithmic difference driving decoder performance.

Keywords Neural decoding · Off-line reconstruction · Prosthetics · Bayesian inference

Action Editor: Alessandro Treves

S. Koyama (✉) · S. M. Chase · R. E. Kass
Department of Statistics, Center for the Neural Basis of Cognition, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: koyama@stat.cmu.edu

S. M. Chase
e-mail: schase@andrew.cmu.edu

R. E. Kass
e-mail: kass@stat.cmu.edu

S. M. Chase · M. Velliste · A. B. Schwartz
Department of Neurobiology, Center for the Neural Basis of Cognition, University of Pittsburgh, Pittsburgh, PA, USA

M. Velliste
e-mail: mev3@pitt.edu

A. B. Schwartz
e-mail: abs21@pitt.edu

A. S. Whitford
Department of Bioengineering, Center for the Neural Basis of Cognition, University of Pittsburgh, Pittsburgh, PA, USA
e-mail: asw35@pitt.edu

1 Introduction

Recent developments in experimental technology allow us to record neural activity from ensembles of motor cortical neurons in real time. When coupled with an appropriate decoder, the activity of these neurons can be used to establish a brain-computer interface, and directly drive the motion of, for example, a cursor on a computer screen or a robotic arm (Chapin et al. 1999; Serruya et al. 2002; Taylor et al. 2002; Lebedev and Nicolelis 2006; Velliste et al. 2008). Many decoding algorithms have been proposed for this purpose; choices range from the simple population vector algorithm (PVA) (Georgopoulos et al. 1986, 1988) and optimal linear estimator (OLE) (Salinas and Abbott 1994) to various versions of Bayesian decoders (Brockwell et al.

Published online: 11 November 2009

2004; Wu et al. 2006; Brockwell et al. 2007; Yu et al. 2007; Foldiak 1993).

The PVA characterizes each neuron's activity by preferred direction and performs optimally when the tuning functions are linear, the set of preferred directions are uniformly distributed, and spike counts in adjacent time bins are conditionally uncorrelated (e.g., Zhang et al. 1998). To improve performance when a set of preferred directions are not uniformly distributed, Salinas and Abbott suggested the OLE which corresponds to least-squares estimation (Salinas and Abbott 1994). The PVA may be considered a special case of least-squares estimation when the preferred directions are assumed to be uniformly distributed. Bayesian decoders make use of a fully probabilistic model comprising 1) an observation model, which describes how the observed neural activity relates to the time-evolving signal of interest, and 2) a state model, which describes how the signal changes from one time step to the next.

Bayesian decoders typically provide the most accurate off-line trajectory reconstructions (Brockwell et al. 2004; Wu et al. 2006; Yu et al. 2007). However, since Bayesian and linear decoders differ in several respects, it is not known which factors are critical for improving device performance. Furthermore, it is not necessarily true that improvements in off-line reconstruction will translate to improvements on-line control, and deficits in off-line reconstruction will translate to deficits in on-line control. The latter case is almost certainly wrong, because the subject might compensate for the specifics of the decoder in use at the time (Chase et al. 2009).

Neural decoders such as the PVA, OLE or Bayesian decoder are each derived as the optimal estimators under certain assumptions. In this study, we consider four assumptions underlying derivation of these decoders: (A) uniformity of a set of preferred directions, (B) tuning curve linearity, (C) spike count variance, and (D) smoothing process. We assess the performance of nine decoders (the PVA, plus eight decoders obtained by taking one of two alternative assumptions in each of conditions (B), (C) and (D)) in off-line reconstruction of hand control data, and three decoders (PVA, OLE and a Bayesian decoder) in a brain-control task. By comparing the performance of the decoders under each of these set of conditions, it is possible to attribute performance differences to the four modeling assumptions.

In off-line reconstruction, we find that assumptions about uniformity in the set of preferred directions and the way the cursor trajectories are smoothed are the most critical for improving decoding performance; perhaps surprisingly, assumptions about tuning curve linearity and spike count variance play relatively minor roles. In on-line control, subjects compensate for biases caused by non-uniformly distributed preferred directions, leaving cursor smoothing differences in decoders as the largest single algorithmic difference driving decoder performance.

2 Decoding algorithms

In this section, we briefly review the derivation of the decoding algorithms. Decoding algorithms consist of two components: an encoding model and a filtering process. The optimal estimator will depend on the modelling choices of both components.

2.1 Encoding model

The derivation of a decoding algorithm starts by assuming an encoding model. Let $y_{i,t}$ be the spike count of the i-th cell ($i = 1, \ldots, N$) at time t, and $\boldsymbol{v}_t \in \mathbb{R}^d$ (where d is the number of movement dimensions) be the to-be-decoded velocity of the intended movement at time t. Throughout this article we assume that the neural firing rates are conditionally independent given the velocity. An encoding model is specified by two components; one describes the dependency of mean spiking activity on velocity,

$$E(y_{i,t}) = f(\boldsymbol{v}_t, \boldsymbol{p}_i), \tag{1}$$

where $\boldsymbol{p}_i$ denotes the preferred direction of i-th cell: the other describes the variance of spiking activity, $Var(y_{i,t})$. We will specify these components for each of the decoding algorithms below.

2.1.1 Linear and constant variance model

If we take the mean model to be a cosine (linear) tuning function,

$$f(\boldsymbol{v}_t, \boldsymbol{p}_i) = b_i + m_i \boldsymbol{p}_i \cdot \boldsymbol{v}_t, \tag{2}$$

and assume the normalized spike counts, $z_{i,t} = \frac{y_{i,t} - b_i}{m_i}$, have a constant variance for all i and t, then it is well known in statistical analysis that ordinary least-squares regression gives the optimal estimator of $\boldsymbol{v}_t$, in a minimum-variance sense (Bickel and Doksum 2006). If we collect the normalized spike counts together into a vector $\boldsymbol{z}_t = (z_{1,t}, \ldots, z_{N,t})^T$ and the preferred directions into a corresponding matrix $P = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N)^T$, the ordinary least-squares estimate is expressed as

$$\hat{\boldsymbol{v}}_t = (P^T P)^{-1} P^T \boldsymbol{z}_t. \tag{3}$$

In the neuroscience literature, this is also called the *optimal linear estimator* (OLE) (Salinas and Abbott

1994).[1] If we further assume that the preferred directions are uniformly distributed in a unit sphere in $\mathbb{R}^d$, Eq. (3) reduces to

$$\hat{\boldsymbol{v}}_t = \frac{d}{N} P^T \boldsymbol{z}_t, \tag{4}$$

which corresponds to the population vector algorithm (PVA) estimate (Georgopoulos et al. 1986, 1988).

2.1.2 Log-linear model

Spike counts often exhibit Poisson-like variability, where the variance of the spike count is not constant, but rather is proportional to the mean. Log-linear models may be more adequate for the analysis of such data (McCullagh and Nelder 1989). In log-linear models, the two components of the classical linear model are replaced in the following way: first, the count distribution is assumed to be Poisson, or more generally, the variance is assumed to be

$$Var(y_{i,t}) = \sigma^2 E(y_{i,t}), \tag{5}$$

where σ^2, the dispersion parameter, is assumed to be constant over the data ($\sigma^2 = 1$ for Poisson case). Under-dispersion, $\sigma^2 < 1$, implies less variable spike counts, which may occur when using filtered spike counts (see Section 2.2 for filtered spike counts). Second, the dependence of $E(y_{i,t}) = f(\boldsymbol{v}_t, \boldsymbol{p}_i)$ on the velocity $\boldsymbol{v}_t$ is often assumed to be logarithmic,

$$\log f(\boldsymbol{v}_t, \boldsymbol{p}_i) = b_i + m_i \boldsymbol{p}_i \cdot \boldsymbol{v}_t \tag{6}$$

The main reason for taking the logarithmic form is that it ensures $f(\boldsymbol{v}_t, \boldsymbol{p}_i)$ remains positive, as required by the Poisson assumption. In addition, this form has an important property for neural encoding models: the inverse $f(\boldsymbol{v}_t, \boldsymbol{p}_i) = \exp(b_i + m_i \boldsymbol{p}_i \cdot \boldsymbol{v}_t)$ is similar to the von Mises tuning function which allows for a narrower tuning curve than the cosine tuning function (Amirikian and Georgopulos 2000). Decoding based on log-linear models can be done by maximizing the Poisson log likelihood.

2.1.3 Other combinations

The combination of the cosine tuning function Eq. (2) with the constant spike count variance or the log-firing rate Eq. (6) with the variance proportional to the mean are the standard choices in the framework of *generalized linear models*, since the linear (identity) function and the logarithm function are the canonical link for the Gaussian and Poisson distributions, respectively.[2] Of course we can chose either the log-firing rate with the constant spike count variance or the cosine tuning function with the variance proportional to the mean. In the latter case, however, we must be careful about the treatment of the mean of spike count to ensure it remains positive. In our analysis below, we simply rectify the mean of spike counts when the value of Eq. (2) goes under 1Hz.[3]

2.2 Filtering

Decoding $\boldsymbol{v}_t$ directly from raw spike counts in short time windows (.03s is typical for real-time decoding) often results in very noisy trajectory estimates unless there are a large number of cells, and thus the estimates should be smoothed. A common way of smoothing trajectories, employed by PVA and OLE, for instance, is to use filtered spike counts. In the analysis below, we use a 5-point boxcar filter, i.e. spike counts are averaged over 5 consecutive short time windows (Velliste et al. 2008; Chase et al. 2009).

State-space methods, used in Bayesian decoders, provide an alternative way of smoothing (Brown et al. 1998; Brockwell et al. 2004; Eden et al. 2004; Koyama et al. in press). In addition to an encoding model, which is called an *observation model*, the state-space method relies on the specification of a *state model* describing the evolution of the state we are trying to predict (here, velocity $\{\boldsymbol{v}_t\}$). For the purpose of obtaining a smooth velocity trajectory, we use a state model that constrains the sequence of states $\{\boldsymbol{v}_t\}$ so that they are likely to evolve with some degree of smoothness from one time step to the next. To see intuitively how the state-space method works, take a random walk, $p(\boldsymbol{v}_t|\boldsymbol{v}_{t-1}) = \mathcal{N}(\boldsymbol{v}_{t-1}, \eta^2 I)$ (where $\mathcal{N}(m, V)$ is a multivariate Gaussian

[1] If spike counts of the various neurons have different variances (and are not independent), the weighted least-squares:

$$\hat{\boldsymbol{v}}_t = (P^T \Sigma^{-1} P)^{-1} P^T \Sigma^{-1} \boldsymbol{z}_t$$

will be more efficient than the OLS. Here Σ^{-1} is the covariance matrix of $\boldsymbol{z}_t$ (Kutner et al. 2004). We consider only the OLS here, and instead, the non-constant variance is taken in log-linear models below.

[2] The canonical link function equates the linear predictor with the canonical parameter of the exponential family distribution, which allows $P^T \boldsymbol{y}_t$ to be a sufficient statistic for $\boldsymbol{v}_t$ (McCullagh and Nelder 1989).

[3] We took the threshold value to be 1 Hz instead of 0 because the algorithm becomes unstable as the variance gets close to 0 due to the fact that the inverse of variance is taken as the "weight" of the weighted least-squares; see Eq. (17) in Appendix A2.

Table 1 Summary of the decoding algorithms

Algorithm	Model	Filtering
PVA	Cosine tuning function	
	Constant spike count variance	
	Uniformly distributed preferred directions	
LGB/LGS	Cosine tuning function	Spike count filtering
	Constant spike count variance	(by a 5-point boxcar filter)
NGB/NGS	Non-cosine tuning function	
	Constant spike count variance	or
LPB/LPS	Cosine tuning function	
	Spike count variance $\propto$ mean	State-space method
NPB/NPS	Non-cosine tuning function	
	Spike count variance $\propto$ mean	

distribution with the mean vector m and the covariance matrix V, and I is the identity matrix) as the state model (Brockwell et al. 2004). The estimate of $\boldsymbol{v}_t$ is obtained from the conditional probability distribution of $\boldsymbol{v}_t$ given the observations, whose logarithm is computed by Bayes' theorem as

$$\log p(\boldsymbol{v}_t|\boldsymbol{y}_t) = p(\boldsymbol{y}_t|\boldsymbol{v}_t) - \frac{1}{2\eta}\|\boldsymbol{v}_t - \boldsymbol{v}_{t-1}\|^2 + \text{const}, \qquad (7)$$

where the second term comes from the state model. The maximum a posteriori (MAP) estimate of the velocity is obtained by maximizing Eq. (7) with respect to $\boldsymbol{v}_t$. As is seen in Eq. (7), the second term enforces the smoothness on $\boldsymbol{v}_t$ by penalizing $\boldsymbol{v}_t$ for being far from $\boldsymbol{v}_{t-1}$, where the degree of smoothness is controlled by the variance of the state model, η.

A conceptual difference between the two filtering processes is that in the former filtering and estimation of the velocity are performed separately, while in the latter filtering and estimation of the velocity are performed simultaneously within a probabilistic framework that uses an explicit model of how the output should evolve over time.

2.3 Summary of decoding algorithms

We have considered several alternative assumptions:

(A) The preferred directions are uniformly distributed or not.
(B) The tuning function is cosine or not (narrower).
(C) The variance of spike counts is constant or proportional to the mean.
(D) Trajectory smoothing is performed by averaging several spike counts or by a state-space method.

Excepting the assumption of uniformly distributed preferred directions, we have eight combinations by choosing one of the two alternative assumptions in (B), (C) and (D). The assumption of non-uniformly distributed preferred directions is then automatically incorporated in the optimal decoders (the maximum likelihood estimator for a spike count filter or the optimal Bayesian decoder for a state-space method). As already mentioned, PVA is a special case of the optimal linear Gaussian decoder with a spike count filter under the assumption of uniformly distributed preferred directions. We thus have $8 + 1 = 9$ decoders. We label these eight decoders (except PVA) with three capital letters, "XYZ", where X specifies condition (B) taking either "L" (linear, i.e. cosine tuning) or "N" (nonlinear, log-firng rate), Y specifies condition (C) taking either "P" (Poisson distribution, i.e. count variance proportional to the mean) or "G" (Gaussian distribution, i.e. constant count variance), and Z specifies condition (D) taking either "S" (state-space method) or "B" (spike count filtering with a boxcar filter). For example, "LGB" stands for the optimal decoder under the assumptions of linear tuning curve and constant spike count variance with a spike count filter, which corresponds to the "optimal linear estimator"(OLE). Note also that "LGS" corresponds to the Kalman filter. Table 1 summarizes the decoding algorithms. In Appendix A2, we show that these eight decoders are implemented in the same framework of exponential family regression. Our goal is to study how much impact the alternative assumptions have on decoding performance by contrasting the nine decoders.

3 Study design

3.1 Off-line trajectory reconstruction

The data we analyzed off-line was taken in the following way. A 96 channel electrode array was implanted in the motor cortex of a monkey to record neural activity (Blackrock Microsystems, Salt Lake City, UT; Maynard and Normann 1997). In all, 78 distinct cells were recorded simultaneously. Raw voltage waveforms

were amplified, filtered, thresholded, and template sorted to isolate the activity of individual cells using a Plexon MAP system (Plexon Inc., Dallas, TX).[4] A monkey in this experiment was presented with a virtual 3-D space, containing 8 possible targets which were located on the corners of a cube, and a cursor which was controlled by the subject's hand position. The task was to move the cursor to a highlighted target from the middle of a cube; the monkey received a reward upon successful completion.

In our data each trial consisted of a time series of spike counts from the recorded cells, and a time series of recorded hand positions. Hand velocities were found by taking differences in hand position at successive $\Delta \approx 0.033$ s intervals. Each trial contained 23 time-steps on average. Our data set consisted of 104 such trials.

16 trials, consisting of 2 presentations of each of the 8 targets, were reserved for estimating the parameters of the model. The parameters of the cosine tuning function were determined by linear regression of the spike counts on the intended velocity, and in the same way, those of the log-linear model were determined by Poisson regression. The variance of the state model in the state-space method was estimated via maximum likelihood. The time lag between the arm movement and the neural activity of each cell was also estimated from the same training data. This was done by fitting a model over different values of time lag ranging from 0 to 3Δ s ($\Delta \approx 0.033$). The estimated optimal time lag was the value at which the model had the highest R^2. 66 cells whose modulation depth were 5 Hz or more were used for decoding. Having estimated all of the parameters, cursor motions were reconstructed from spike trains for the other 88 trials. The performance of the decoding methods was measured by the mean integrated squared error (MISE) between the actual and decoded cursor velocities.

3.2 Simulation

The simulations were meant to mimic the 3-D open-loop control experiments. We first generated Fisher tuning curves for $N = 60$ simulated cells by specifying the following parameters: the half-width of dropoff θ_{hw}, the modulation depth m_d, and the baseline firing rate b_s (Fig. 1; see also Appendix A1). The baseline firing rates and modulation depths were taken to be 5 Hz and 100 Hz, respectively, and the half widths were taken to be values between $\pi/4$ and $\pi/2$. The preferred

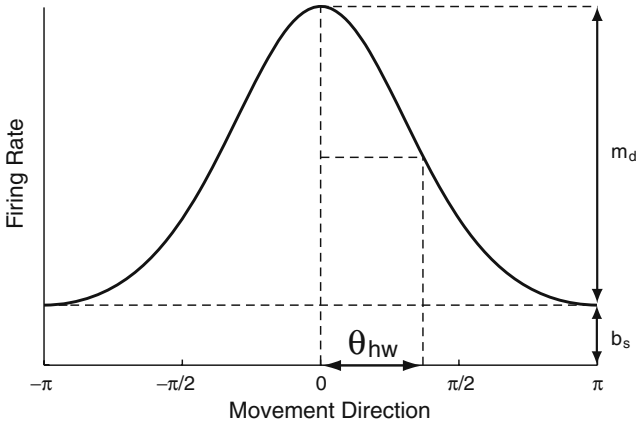

Fig. 1 Illustration of tuning curve. θ_{hw}, m_d and b_s are half-width of the dropoff shape, modulation depth and baseline firing rate, respectively

directions were drawn at random from a uniform distribution in a 3-D unit sphere. The intended movements were generated as velocity vectors pointing straight from the origin of the workspace to the center of the presented target, with a bell-shaped speed profile:

$$\|\boldsymbol{v}_t\| = \frac{1}{2}\left[1 + \sin\left(\frac{2\pi c_s t}{T} - \frac{\pi}{2}\right)\right], \tag{8}$$

where $T = 0.833$ s. Movements were generated over the time interval $[0, T]$. This time interval was chosen so that it consisted of 25 time-steps with 30 Hz binning. The eight targets were spaced at the corners of a cube. Binned spike counts were then generated at 30 Hz as Poisson realizations of the underlying rate parameter for the specified intended movements. Note that the movement commands simulated open-loop control, i.e., we did not simulate on-line corrections of errors in the trajectories.

The first 16 trials, consisting of 2 presentation of each of 8 targets, were used for estimating model parameters. These parameters were determined in the same way as in the real-data analysis. Once the parameters were estimated, 10 center-out trajectories were generated to each of the 8 targets, and the movement velocities were decoded by the 9 methods introduced in Section 2.3. The performance of the decoding methods was quantified by MISE between the intended and decoded velocities.

3.3 On-line closed loop control

The same monkey was used to perform the brain-control center-out task. In this experiment, the monkey was presented with a virtual 2-D space, and had to modulate the recorded neurons to move a cursor

[4]Some of the cells were well-isolated single-units, while others were multi-unit combinations.

from the center of the workspace to one of 8 targets spaced uniformly around a virtual circle. Both cursor and target had a radius of 8 mm; targets were spaced 85 mm from the center of the workspace. If the target was hit within 1750 ms of being displayed, the monkey received a liquid reward. After either success or failure, the cursor was placed back in the center following a 750 ms inter-trial interval.

We implemented the three real-time decoding algorithms of PVA, LGB and NPS. In each experimental session, two of the three algorithms were used for decoding; we call this experimental paradigm *dual control*. At the start of each session, a calibration session was run to determine the model parameters of each algorithm. In this session, decoding parameters were initially assigned to the cells at random. Then a single presentation of each of the 8 targets was made in random order. For the PVA and the LGB, after all targets were presented the tuning functions of the cells were estimated by linearly regressing the spike rates against target direction. Cells with modulation depths of less than a cutoff of 4 Hz were not used for control (this eliminated, on average, ~1/2 of the cells). For the NPS, the parameters of the log-linear model were determined by Poisson regression; cells with firing rates of less than 5 Hz were not used for control. After an initial estimate of the decoding parameters was obtained, they were set in the decoder and another round of 8 targets was presented. The regressions were then performed on the accumulated data. This procedure was repeated until good cursor control was achieved; typically this required a total of 4 to 5 rounds of target presentation (~32 to 40 movements), and never took more than 7 rounds of target presentation.[5] The number of cells used for decoding varied from 29 to 40 for the PVA, from 29 to 32 for the LGB, and from 35 to 45 for the NPS across sessions. To examine how accurate the cursor control can be with each decoder when going the same speed, the preferred direction vectors were multiplied by a constant speed factor. The speed factor was adjusted in each session so that the average cursor speeds decoded by the two algorithms approximately matched. The speed factor was also changed across sessions to examine the decoding performance at different cursor speeds; the average speed fell between 0.1 and 0.23 m/s. Once all the parameters were determined, each of the 8 targets were presented 15 times in random order, and the monkey performed the task by using one of the two decoding algorithms. The decoding algorithm was then switched to the other, and repeated the same number of target presentations. We analyzed the data from 10 NPS/PVA-, 6 NPS/LGB-, and 6 LGB/PVA-dual control sessions.

Since we did not have the "true" subject's intention of cursor control in on-line control experiments, the performance of the decoders was evaluated with four measures: 1) the length of the trajectory, 2) the trajectory variance across trials, 3) the speed asymmetry across the target positions, and 4) the hitting time. The lengths of the cursor trajectories were calculated by summing the distance the cursor moved in each time step from the trial start until the target was hit. For calculating the variance of the trajectory, the time axis of each trajectory was uniformly scaled to unity. That is, the time samples for an individual movement t of duration T were scaled by a factor $\alpha = 1/T$ to create a new set of time samples $t_s = t\alpha$. Each x and y component of the trajectory was then independently resampled using spline interpolation to a common time axis consisting of 100 evenly sampled points. Finally, the mean and variance of the cursor position was calculated at every time point. The speed asymmetry was defined as the standard deviation of the mean speed across targets. The hitting time was defined as the time at which the target was hit, relative to the time of target presentation. We chose these 4 measures to see how estimation errors in the decoding algorithms reflect the different aspects of the quality of cursor control; if a certain decoder causes a systematic bias in estimation of the cursor velocity (e.g. the directional bias caused by using PVA), it may increase the length of the trajectory and the speed asymmetry across the target positions. On the other hand, noisy estimation may make the trajectory longer and more variable across trials, and increase the time it takes to hit the target. Of course biases and variance in the estimates made by decoders can reflect all 4 measures and their effects cannot be isolated completely by these 4 measures.

4 Results

4.1 Off-line trajectory reconstruction

Results from off-line reconstruction of experimental hand-trajectory data are summarized in the first row of Table 2. We show in Fig. 2 typical trajectories reconstructed by PVA, LGB, and NPS. Overall, LGS and NPS show the best performance among the nine decoders, while PVA stands out as the worst of all of the decoders. Comparing within decoders having the

[5]During the dual control experiment, only one algorithm could control the cursor at a time. The controlling algorithm during the calibration phase was alternated between experimental sessions in order to encourage a more balanced comparison.

Table 2 MISEs [units of $10^{-2} \times (\mathrm{m/s})^2$] in estimating the true cursor velocity

	LGB	NGB	LPB	NPB	PVA
$n = 66$	1.46 ± 0.11	1.60 ± 0.12	1.85 ± 0.15	1.42 ± 0.11	2.33 ± 0.18
$n = 50$	1.73 ± 0.13	1.91 ± 0.16	2.12 ± 0.17	1.59 ± 0.13	2.66 ± 0.23
$n = 40$	1.84 ± 0.15	1.94 ± 0.15	2.29 ± 0.18	1.70 ± 0.14	2.98 ± 0.24
$n = 30$	2.22 ± 0.17	2.37 ± 0.19	3.01 ± 0.24	1.88 ± 0.15	3.78 ± 0.30
	LGS	NGS	LPS	NPS	
$n = 66$	1.02 ± 0.09	1.14 ± 0.09	1.16 ± 0.09	1.01 ± 0.08	
$n = 50$	1.14 ± 0.10	1.20 ± 0.11	1.31 ± 0.11	1.11 ± 0.10	
$n = 40$	1.26 ± 0.10	1.30 ± 0.11	1.43 ± 0.11	1.24 ± 0.11	
$n = 30$	1.39 ± 0.11	1.40 ± 0.12	1.58 ± 0.12	1.32 ± 0.11	

The means and the standard errors were obtained from 88 trials

same tuning function and variance model, the state-space method performs substantially better than the spike count filter. We also reconstructed the trajectories from different numbers of cells, $n = 50$, 40, and 30; these cells were selected randomly from the 66 cells recorded in the experiment. These trend are the same as the trends we observe when all 66 cells are used (the 2nd-4th rows of Table 2). In the following, we examine how much impact the alternative assumptions (A)–(D) described in Section 2.3 have on decoding performance.

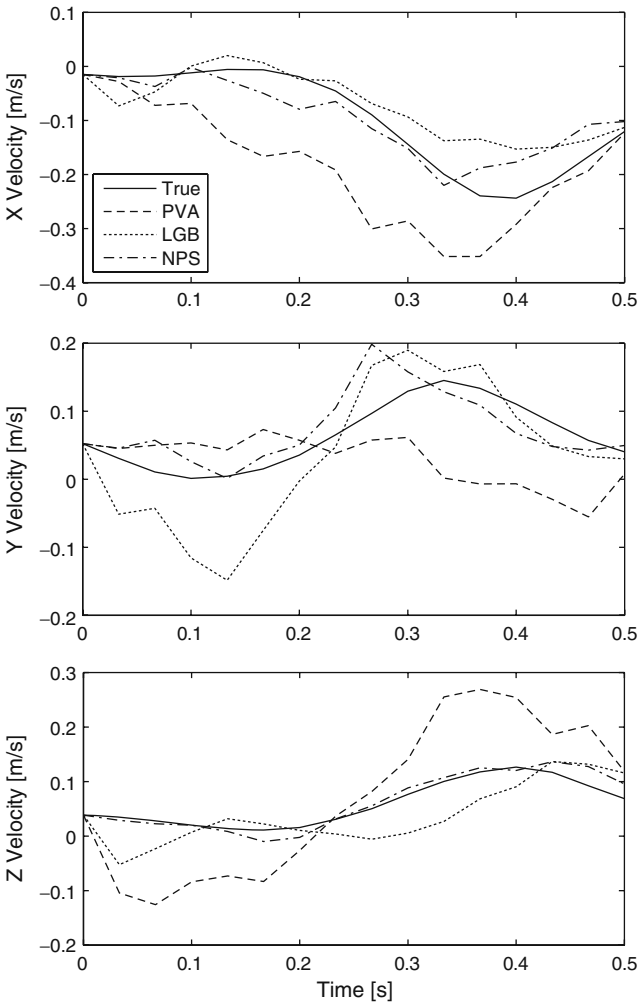

Fig. 2 Trajectories reconstructed by PVA, LGB and NPS for a trial. The *solid line* represents the true trajectories. The trajectories reconstructed by PVA and LGB are deviated from the true trajectory more than that reconstructed by NPS

4.1.1 Assumption of uniformly distributed preferred directions

The result that PVA stands out as the worst of all of the decoders implies that the assumption of uniformly distributed preferred directions is critical for the decoding performance, since this assumption differentiates PVA from all other decoders. Particularly, the effect of this assumption can be isolated by comparing the MISE of the PVA with that of the LGB. As seen in Table 2, the decoding performance of the PVA deteriorates more than that of other decoders as the number of cell decreases. Since the non-uniformity in a set of preferred directions becomes larger as the number of cell decreases, this result confirms that the performance of the PVA is affected by the non-uniformity more critically than the other decoders.

4.1.2 Smoothing process

The result that the state-space method performs substantially better than the spike count filter regardless of the choice of tuning function and spike count variance indicates that the smoothing process has a major impact on the decoding performance. The degree of smoothness both in the state-space method and in the spike count filter is determined by the memory length of the past trajectory (or data) that is averaged to produce the current state estimate. In the state-space method, the memory length is scaled by $1/\eta$ where η is the magnitude of the fluctuation of the state process in Eq. (7), which is determined by the maximum likelihood principle. In the spike count filter, on the other hand, we chose empirically a 5-point boxcar filter followed by the previous experimental setting (Chase

et al. 2009; Velliste et al. 2008), which may not be optimal in our data. We therefore reconstructed the trajectory from the same experimental data using the spike count filter with different filter lengths to examine if it improves the decoding performance (Fig. 3(a)). As seen in this figure, the best decoding performance was achieved when the filter length is about 5. Note also that the decoding performance was significantly improved compared with that without any filtering (that is, MISE for filter length 0). We thus conclude that the filtering process has a large impact on the decoding performance.

4.1.3 Tuning curve linearity and spike count variance

When using either the state-space method or spike count filtering as the smoothing process, the results in Table 2 show that the linear tuning model exhibits slightly (but not significantly) better performance than the nonlinear, log-firing rate, model when a constant spike count variance is assumed (that is, LGB and LGS are better than LPB and LPS, respectively). Conversely, the nonlinear tuning model is better than the linear tuning model when the variance of the spike count is assumed to be proportional to the mean (that is, NPB and NPS are better than NGB and NGS, respectively). Although the effect of the assumptions of tuning curve linearity and spike count variance are not well isolated, the effect of these two assumptions on the MISE are smaller than either that of uniformity in the set of preferred directions or that of the smoothing process. Thus, we can conclude that the assumption of tuning curve linearity (here, "sharpness" of the tuning curve) and the spike count variance play relatively minor roles in our data.

In order to examine how much impact these two assumptions could have on the decoding performance, we performed the simulations described in Section 3. To see how linearity in the tuning curve affects the decoding result, we plot the MISE between the input simulated velocities and the decoded simulated velocities as a function of the half-width in Fig. 3(b) for the spike count filter and in Fig. 3(c) for the state-space method. While NPB and NGB behave well across the entire range of half-widths, the performance of LPB and LGB are degraded as the half-width becomes very small; the same result is seen in Fig. 3(c). Thus, the assumption of tuning curve linearity could have a large impact on the decoding performance when the cells have very narrow tuning curves. The mean and standard deviation of half-width values that were measured in the 66 cells analyzed in Table 2 are 1.22 ± 0.14 rad, comparable to, though slightly larger than the results of Amirikian and Georgopulos (2000). Taking into account the simulation results shown in Fig. 3(b–c), it is concluded that the cells in the real-data do not have a strong enough nonlinearity to have a large impact on the performance of the linear decoders.

Although the synthesized data was generated from Poisson processes, there is no clear evidence that the Poisson assumption performs better than the constant spike count variance model. Particularly, the Poisson assumption with the linear tuning function (LPB and LPS) tends to behave much worse than the other combinations. This may be due to our method of rectifying the link-function in the linear Poisson decoder; in the

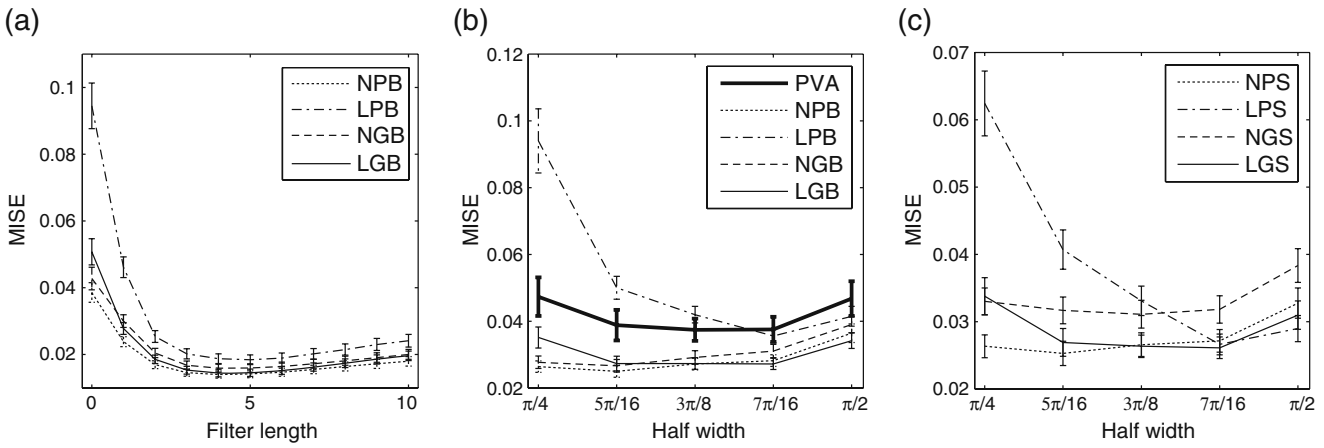

Fig. 3 Results of off-line reconstruction. (**a**) MISE as a function of filter length. The velocity was reconstructed with 66 cells. The best performance was achieved when the filter length is about 5. (**b**) MISE as a function of the half-width of the tuning curve. All 60 simulated cells had the same half-width represented in the horizontal axis. The performance of LGB and LPB are degraded relatively to that of NGB and NPB as the half-width becomes smaller. (**c**) the same as (**b**) for state-space method

Poisson model, the linear tuning function must be rectified in order to ensure the firing rate remains positive. Although we did not perform further systematic investigation on how it affects on the decoding performance, the rectification should be carefully designed case by case.

It would be worth mentioning that in the results of real-data analysis, LGS (i.e. Kalman filter) performs as well as NPS, which, as well as the fact that its implementation is much simpler than NPS, would encourage the use of the Kalman filter in practice. The Kalman filter, however, does not work well when the assumption of linearity is critical for the decoding performance; as seen in Fig. 3(c), MISE of LGS becomes larger than that of NPS when the half-width of tuning curve is less than $5\pi/16$. Although we examined only the sharpness of tuning curve as a nonlinearity, any kind of nonlinearity can deteriorate the performance of linear decoders if its effect is not negligible.

In summary, although the assumption of tuning curve linearity could potentially have a large impact on the decoding performance, we can conclude that both this assumption and the spike count variance model play relatively minor roles in determining trajectory reconstruction error in the data we analyzed.

4.2 On-line closed loop control

To see how well the results of off-line reconstruction represent those of on-line control, we performed on-line control experiments comparing decoding techniques. Trajectories controlled with the three algorithms are shown in Fig. 4 (in slow speed control) and in Fig. 5 (in high speed control) for visualization. It is seen from these figures how variable cursor trajectories are; cursor trajectories in PVA and LGB control become more variable across trials than those in NPS control do as the average speed is increased. Figure 6 quantifies the cursor control with the four measures, (a) trajectory length, (b) trajectory variance, (c) speed asymmetry and (d) hitting time, as a function of the average cursor speed. Overall, the NPS has smaller values in these four measures than either the PVA or the LGB. That is, cursor trajectories under NPS control are shorter (which implies straighter trajectories from the center to the targets), less variable across trials, exhibit relatively more symmetric speed profiles across target direction, and hit the target within a shorter time. Trajectories controlled with PVA and LGB behave about the same in terms of (a), (b) and (c), though trajectories controlled with the PVA hit the target with relatively shorter times than those controlled with the LGB. As the average speed increases, the values of trajectory length and variance also increase, which leads to an "optimal speed" in terms of least time to target. The average cursor speed that gives the least time to target is estimated to be 0.180 m/s for the NPS, 0.164 m/s for the LGB, and 0.175 m/s for the PVA (by fitting the data with a quadratic curve: Fig. 6(d)). These results may imply that the subject could control the cursor more accurately in NPS control than in PVA and LGB control.

5 Discussion

We used both simulations and experiments to compare the performance of several decoding algorithms. Each of these algorithms are optimal[6] under certain conditions. We found that assumptions about uniformity in the set of preferred directions and the trajectory smoothing method have the most impact on decoder performance in off-line reconstruction. From the view of statistical inference, these two assumptions account for systematic biases and variance in decoding errors, respectively. In theory, specification of the width of the tuning curve and allowing non-constant spike count variance may also have reduced the two errors, but these played relatively minor roles in the data we analyzed.

In on-line control, we saw that, first, there were no substantial differences in performance between the PVA and the LGB in terms of the trajectory length, trajectory variance or speed asymmetry. If the directional biases that result from the use of the PVA still had a large impact on the on-line control performance as seen in off-line reconstruction, the performance of the LGB would differ from that of the PVA. Thus, the experimental results indicate that the subject can compensate for the directional biases caused by non-uniformly distributed preferred directions. This result is consistent with that in Chase et al. (2009), in which PVA and LGB (they used the conventional labeling,

[6]The maximum likelihood estimator asymptotically achieves the theoretical lower bound (the "Cramer-Rao" lower bound given by the inverse of the Fisher information Schervish 1996) of the variance in the limit of large samples. In the Bayesian inference, the posterior expectation gives the optimal estimator under the squared error loss. In our analysis, we approximately computed the recursive Bayesian equations (see Appendix A2), which provides the first-order approximation of the posterior expectation in the "high-information" limit, where the posterior becomes very sharply peaked around the mode and the Laplace approximation is valid (Koyama et al. in press). Thus, the state-space estimates we computed are asymptotically optimal under the squared error loss.

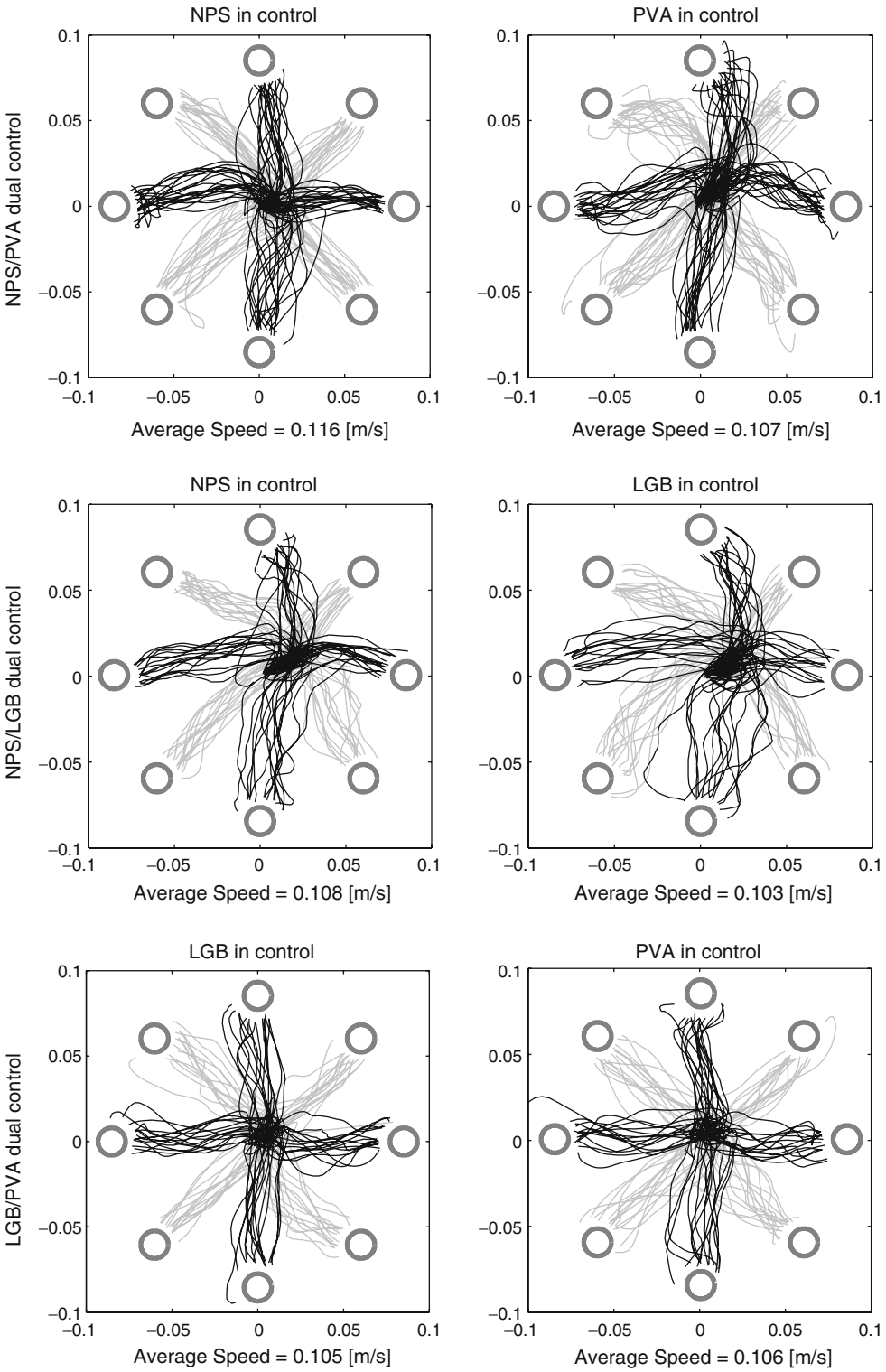

Fig. 4 Results comparing between cursor trajectories controlled under different decoders in slow speed control. Color changes across targets are to aid visibility. From top to bottom: results in NPS/PVA-, NPS/LGB-, and LGB/PVA-dual control. Trajectories in NPS control are less variable than either those in PVA control or those in LGB control, while trajectories in PVA control are about as variable as those in LGB control (compared in the same rows)

"OLE") were compared both in off-line and on-line control. Second, the NPS control outperformed both the LGB and the PVA (e.g. trajectory length for NPS control was about 90% of the trajectory length for PVA and LGB.) The LPS differs from the LGB in 1) allowing narrower tuning functions, 2) assuming a Poisson, as opposed to Gaussian, spike count variance model, and 3) smoothing trajectories through a state-space model, as opposed to filtering the spike counts directly. As is shown in the results of off-line reconstruction, 1) and 2) play relatively minor roles; we can thus attribute the performance differences to 3). In other words,

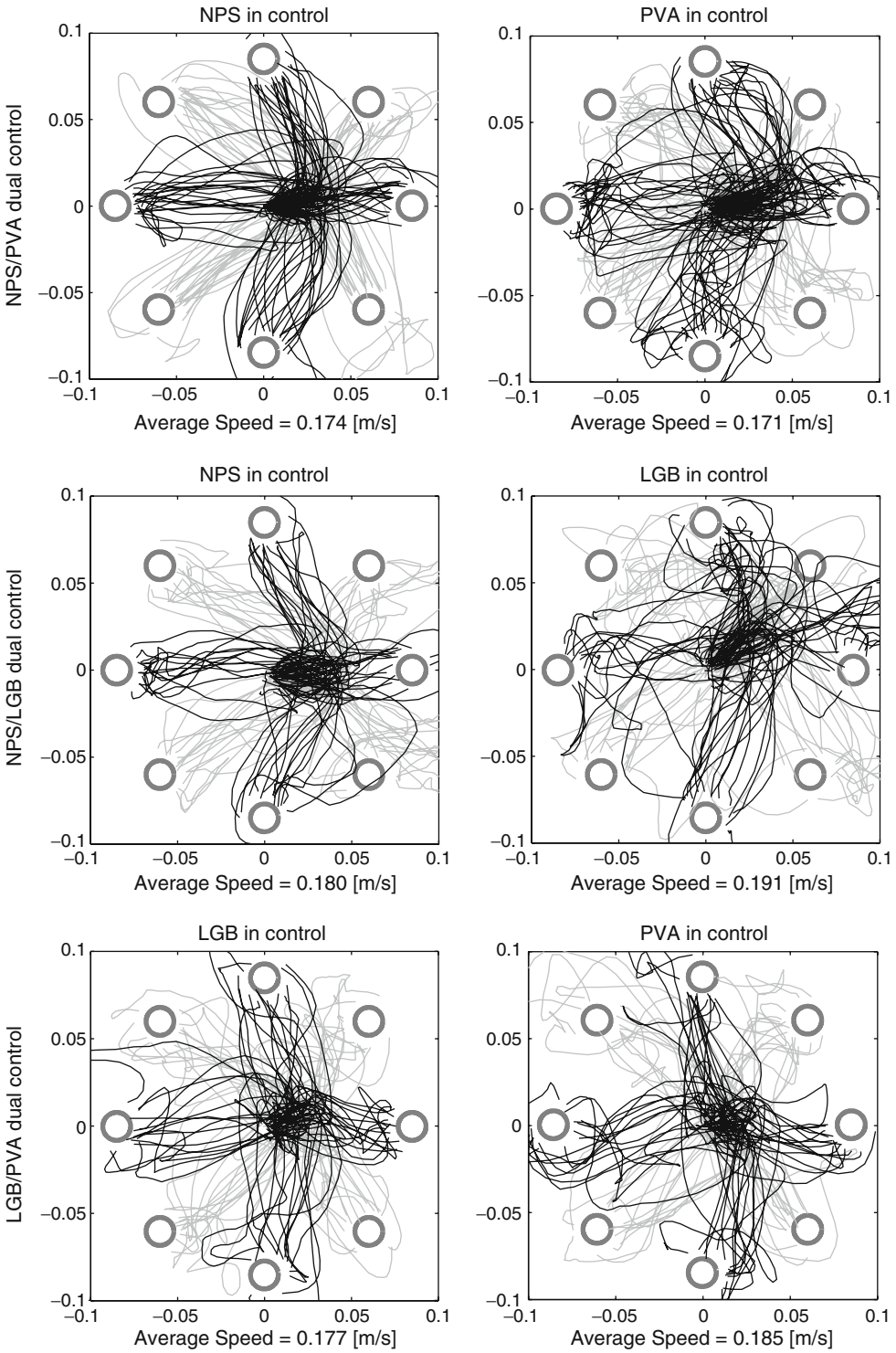

Fig. 5 Same as Fig. 4 in high speed control. Cursor trajectories in PVA and LGB control are more variable than those in NPS control as is the same as in Fig. 4, but trajectories are more variable than those in slow speed control

improvement of the performance in NPS control was made by the way trajectories were smoothed (i.e., reducing the variance) in the algorithm.

These results lead to a natural interpretation of what is "learnable" for a subject in on-line control: while subjects may compensate for systematic biases caused by decoders (Chase et al. 2009), they are much less likely to compensate for variance differences, although it should be noted that we did not specifically reward the subjects for either increased consistency or reduced cursor jitter.

As variance does not seem to be automatically compensated by a subject in on-line control, it should be reduced in a decoder to the greatest extent possible,

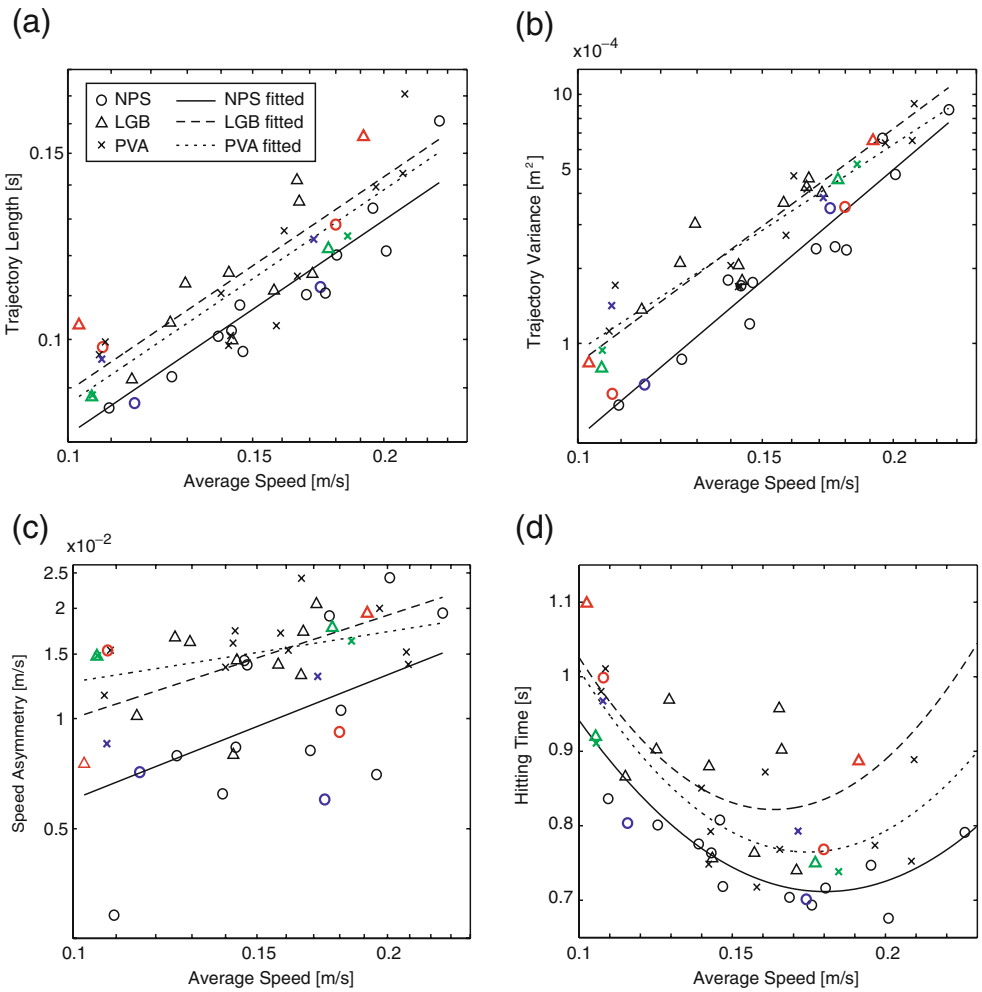

Fig. 6 A comparison of the on-line control performance under different decoders. (**a**) Average trajectory length as a function of the average speed. (**b**) Trajectory variance as a function of the average speed. (**c**) Speed asymmetry as a function of the average speed. The lines in (**a**), (**b**) and (**c**) were fitted to the data points in log-log scale. (**d**) Average hitting time as a function of the average speed. The quadratic curves were fitted to the data points in normal scale. Each data point was calculated with 120 trials (8 targets × 15 trials). *Blue*, *red* and *green* points, respectively, indicate the data of NPS/PVA-, NPS/LGB- and LGB/PVA-dual control shown in Figs. 4 and 5. Overall, the NPS has smaller values in these four measures than either the PVA or the LGB

without impeding control. In spike count filtering, noise reduction was performed by smoothing spike counts with a 5-point boxcar filter, where the filter length was chosen empirically, and the same filter length was used in different speed factors. There is a way to select an optimal filter length for estimating firing rate (Shimazaki and Shinomoto 2007), and using the optimal filter length will improve the estimation of trajectories. However, the optimal filter length may differ for each cell (so that the number of free parameters is the same as the number of cells) because it depends on the time scale and amplitude of rate modulation, and on the mean rate (Koyama and Shinomoto 2004). The state-space method, on the other hand, incorporates the inherent dynamic behavior of the trajectory (here smoothness) into the state model, and, in contrast to filtering spike counts, the smoothness is controlled by a single parameter η that is then determined by maximum likelihood.

In our case, there is not much difference between "Bayesian" (i.e. state-space) and "non-Bayesian" decoding. While it is supposed that an advantage of the former is its ability to integrate prior information into the inference, there is also a way in the latter to do it, through the use of a spike count filter. So, what is the substantial benefit of the state-space method? One potential advantage is its flexible framework. Although we have restricted our attention to velocity in this paper, more generally the state could involve position, acceleration, or other movement parameters, and it may be extended to more complicated state models (Wu et al. 2006; Yu et al. 2007). Furthermore, inherent dynamical properties of movement such as the two-third power law (Reina and Schwartz 2003) may be incorporated into the state model to realize natural movements during robotic arm control. Inference of the states as well as the model parameters can then be performed in the same probabilistic framework. However, a drawback of the state-space method is its computational complexity; the computational cost of posterior estimates becomes more expensive as the state-space model becomes more complicated, which will be critical for real-time neural

decoding. Therefore, the key to utilizing state-space methods in BCIs is in developing an efficient algorithm to perform approximate posterior inferences in real-time. Such an attempt has been made in Koyama et al. (in press), for example.

To summarize, we have assessed the performance of several decoders in simulations and experiments. In off-line reconstruction, we find that assumptions about uniformity in the set of preferred directions and the way the cursor trajectories are smoothed have the most impact on decoder performance; assumptions about tuning curve linearity and spike count variance play relatively minor roles. In on-line control, the subject compensated for directional biases caused by non-uniformly distributed preferred directions, leaving cursor smoothing differences as the largest single algorithmic difference driving decoder performance. These results may provide a helpful interpretation of when the off-line performance of a certain decoder will translate to on-line improvements in control.

Acknowledgements This work was supported by grants RO1 MH064537, RO1 EB005847 and RO1 NS050256.

Appendix A1: Fisher tuning function

The Fisher tuning function is defined as

$$f(\theta) = b + c\exp(\kappa\cos\theta), \tag{9}$$

where θ is the angle between the cell's preferred direction and the movement direction (Amirikian and Georgopulos 2000). Note that this corresponds to the log-linear model Eq. (6) when b is omitted. The parameters b, c and κ determine the tuning properties. These parameters are translated into three physiological parameters: the half-width of the dropoff shape θ_{hw}, the modulation depth m_d, and the baseline firing rate b_s, as

$$\theta_{hw} = \cos^{-1}\left[\frac{\log(\cosh\kappa)}{\kappa}\right], \tag{10}$$

$$m_d = 2c\sinh\kappa, \tag{11}$$

and

$$b_s = b + ce^{-\kappa}. \tag{12}$$

The half width takes $\theta_{hw} < \pi/2$ for $\kappa > 0$. For $\kappa \to 0$ (as $\theta_{hw} \to \pi/2$), Eq. (9) becomes the cosine tuning function,

$$f(\theta) = b_s + \frac{m_d}{2} + \frac{m_d}{2}\cos\theta. \tag{13}$$

Appendix A2: Exponential family regression

Here, we briefly describe the exponential family regression and its extension to the state-space method which provides a unifying framework for the neural decoders introduced in Section 2.3. We assume that the spike counts of the ith cell ($i = 1, \ldots, N$) follows the exponential family distribution,

$$p(y_{i,t}|\theta_i, \phi_i) = \exp[(y_{i,t}\theta_i - b(\theta_i))/a(\phi_i) + c(y_{i,t}, \phi_i)], \tag{14}$$

where θ_i is the canonical parameter related to the mean of $y_{i,t}$ and ϕ_i is the dispersion parameter (McCullagh and Nelder 1989). This family includes commonly used distributions such as the Poisson and Gaussian. The mean and variance of $y_{i,t}$ are given as $E(y_{i,t}) = b'(\theta_i)$ and $Var(y_{i,t}) = a(\phi_i)b''(\theta_i)$, respectively. Since the tuning function $f(\boldsymbol{v}_t, \boldsymbol{p}_i)$ relates the velocity to the firing rate as $E(y_{i,t}) = f(\boldsymbol{v}_t, \boldsymbol{p}_i)$, $\theta_i = \theta_i(\boldsymbol{v}_t)$ is a function of $\boldsymbol{v}_t$. Assuming that the spiking of N cells are independent of each other, the probability distribution of $\boldsymbol{y}_t = (y_{1,t}, \ldots, y_{N,t})$ is given by

$$p(\boldsymbol{y}_t|\boldsymbol{v}_t) = \prod_{i=1}^{N} p(y_{i,t}|\theta_i(\boldsymbol{v}_t), \phi_i). \tag{15}$$

Let $l(\boldsymbol{v}_t) = \log p(\boldsymbol{y}_t|\boldsymbol{v}_t)$ be the log likelihood function of Eq. (15). The gradient of $l(\boldsymbol{v}_t)$ is then derived as

$$\nabla_{v_t} l(\boldsymbol{v}_t) = \sum_{i=1}^{N} \frac{y_{i,t} - f(\boldsymbol{v}_t, \boldsymbol{p}_i)}{Var(y_{i,t})} \nabla_{v_t} f(\boldsymbol{v}_t, \boldsymbol{p}_i), \tag{16}$$

where ∇_{v_t} denotes the gradient with respect to $\boldsymbol{v}_t$. The maximum likelihood estimate of $\boldsymbol{v}_t$ is then obtained by solving $\nabla_{v_t} l(\boldsymbol{v}_t) = 0$. Note that only the tuning function $f(\boldsymbol{v}_t, \boldsymbol{p}_i)$ and the spike count variance $Var(y_{i,t})$ appear in Eq. (16). When the tuning function is linear and the variance is constant, $\nabla_{v_t} l(\boldsymbol{v}_t) = 0$ is solved analytically leading to the LGB (i.e. the optimal linear estimator). Otherwise it is solved numerically (by the Newton-Raphson method, for example). Note also that it is seen from Eq. (16) that the maximum likelihood estimate corresponds to the weighted (nonlinear) least-squares estimate which minimizes the weighed squared error,

$$\sum_{i=1}^{N} \frac{[y_{i,t} - f(\boldsymbol{v}_t, \boldsymbol{p}_i)]^2}{Var(y_{i,t})}. \tag{17}$$

In state-space filtering, in addition to the likelihood of velocity Eq. (16), which is called the observation model, we take a state model describing the evolution

of the state, $\boldsymbol{v}_t$. In our analysis, the state model was taken to be a random walk,

$$p(\boldsymbol{v}_{t+1}|\boldsymbol{v}_t) = \frac{1}{(2\pi\eta)^{d/2}} \exp\left[-\frac{1}{2\eta}(\boldsymbol{v}_{t+1}-\boldsymbol{v}_t)^T(\boldsymbol{v}_{t+1}-\boldsymbol{v}_t)\right]. \tag{18}$$

The set of posterior distributions of $\boldsymbol{v}_t$ given the observation up to time t, for $t = 1, 2, \ldots$, is then computed by the recursive relationships,

$$p(\boldsymbol{v}_t|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t) \propto p(\boldsymbol{y}_t|\boldsymbol{v}_t)p(\boldsymbol{v}_t|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}), \tag{19}$$

where

$$p(\boldsymbol{v}_t|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}) = \int p(\boldsymbol{v}_t|\boldsymbol{v}_{t-1})p(\boldsymbol{v}_{t-1}|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})d\boldsymbol{v}_{t-1} \tag{20}$$

is called the predictive distribution. Note that in the limit of large variance, $\eta \to \infty$, (or when there is no dependence of $\boldsymbol{v}_t$ on $\boldsymbol{v}_{t-1}$) the state-space estimate (more precisely, the maximum a posteriori (MAP) estimate) converges to the maximum likelihood estimate obtained by solving $\nabla_{v_t} l(\boldsymbol{v}_t) = 0$.

One implementation of the recursive formula approximates the posterior distribution Eq. (19) as a Gaussian centered on its mode (Brown et al. 1998). Let $\boldsymbol{v}_{t|t}$ and $V_{t|t}$ be the (approximate) mode and covariance matrix for the posterior distribution Eq. (19), and $\boldsymbol{v}_{t|t-1}$ and $V_{t|t-1}$ be the mode and covariance matrix for the predictive distribution Eq. (20) at time t. Further, let $r(\boldsymbol{v}_t) = \log\{p(\boldsymbol{y}_t|\boldsymbol{v}_t)p(\boldsymbol{v}_t|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})\}$. The posterior distribution is then approximated as a Gaussian whose mean and covariance are $\boldsymbol{v}_{t|t} = \arg\max_{v_t} r(\boldsymbol{v}_t)$ and $V_{t|t} = -[\nabla\nabla_{v_t} r(\boldsymbol{v}_{t|t})]^{-1}$, respectively (where $\nabla\nabla_{v_t}$ represents the second derivative with respect to $\boldsymbol{v}_t$). Since the state model is a random walk, the predictive distribution Eq. (20) is also Gaussian, whose mean and covariance are computed as

$$\boldsymbol{v}_{t|t-1} = \boldsymbol{v}_{t-1|t-1}, \tag{21}$$

$$V_{t|t-1} = V_{t-1|t-1} + \eta I. \tag{22}$$

We take the initial state for filtering to be the center of the workspace. This approximate filter is a version of *Laplace-Gaussian filter* in which the posterior distribution is approximated to be a Gaussian by using Laplace's method (Koyama et al. in press). Note that we used the Gaussian approximation instead of particle filtering (Doucet et al. 2001) to compute the posterior estimates because under the constraint of computational cost for real-time decoding, the Gaussian approximation provides faster and more accurate posterior estimates than the particle filter (Koyama et al. in press).

References

Amirikian, B., & Georgopulos, A. P. (2000). Directional tuning profiles of motor cortical cells. *Neuroscience Research, 36*, 73–79.

Bickel, P. J., & Doksum, K. A. (2006). *Mathematical statistics* (2nd ed.). Englewood Cliffs: Prentice Hall.

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

Brockwell, A. E., Schwartz, A. B., & Kass, R. E. (2007). Statistical signal processing and the motor cortex. *Proceeding of the IEEE, 95*, 882–898.

Brown, E. N., Frank, L. M., Tang, D., Quirk, M. C., & 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*, 7411–7425.

Chapin, J. K., Moxon, K. A., Markowitz, R. S., & Nicolelis, M. A. L. (1999). Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nature Neuroscience, 2*, 664–670.

Chase, S. M., Schwartz, A. B., & Kass, R. E. (2009). Bias, optimal linear estimation, and the differences between open-loop simulation and closed-loop performance of brain-computer interface algorithms. *Neural Networks*. doi:10.1016/j.neunet.2009.05.005.

Doucet, A., de Freitas, N., & Gordon, N. (Eds.) (2001). *Sequential Monte Carlo methods in practice*. Berlin: Springer.

Eden, U. T., Frank, L. M., Barbieri, R., Solo, V., & Brown, E. N. (2004). Dynamic analyses of neural encoding by point process adaptive filtering. *Neural Computation, 16*, 971–998.

Foldiak, P. (1993). *Computation and neural systems* (chap. 9, pp. 55–60). Norwell, MA:Kluwer Academic Publishers.

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

Georgopoulos, A., Kettner, R., & Schwartz, A. (1988). Primate motor cortex and free arm movements to visual targets in three-dimensional space. ii. Coding of the direction of movement by a neural population. *Journal of Neuroscience, 8*, 2928–2937.

Koyama, S., P'erez-Bolde, L. C., Shalizi, C. R., & Kass, R. E. (in press). Approximate methods for statespace models. *Journal of the American Statistical Association*. Accepted for publication.

Koyama, S., & Shinomoto, S. (2004). Histogram bin width selection for time-dependent Poisson processes. *Journal of Physics A: Mathematical and General, 37*, 7255–7265.

Kutner, M. H., Nachtsheim, C. J., & Neter, J. (2004). *Applied linear regression methods* (4th ed.). Chicago: McGraw-Hill/Irwin.

Lebedev, M. A., & Nicolelis, A. L. (2006). Brain-machine interfaces: Past, present and future. *Trends in Neuroscience, 29*, 536–546.

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

McCullagh, P., & Nelder, J. (1989). *Generalized linear models*. London: Chapman and Hall.

Reina, G. A., & Schwartz, A. B. (2003). Eye-hand coupling during closed-loop drawing: Evidence of shared motor planning? *Human Movement Science, 22*, 137–152.

Salinas, E., & Abbott, L. F. (1994). Vector reconstruction from firing rates. *Journal of Computational Neuroscience, 1*, 89–107.

Schervish, M. (1996). *Theory of statistics*. New York: Springer.

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

Shimazaki, H., & Shinomoto, S. (2007). A method for selecting the bin size of a time histogram. *Neural Computation, 19*, 1503–1527.

Taylor, D. M., Tillery, H., Stephen, I., & Schwartz, A. B. (2002). Direct cortical control of 3D neuroprosthetic devices. *Science, 296*, 1829–1832.

Velliste, M., Perel, S., Spalding, M. C., Whitford, A. S., & Schwartz, A. B. (2008). Cortical control of a prosthetic arm for self-feeding. *Nature, 453*, 1098–1101. doi:10.1038/nature06996.

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 Computation, 18*, 80–118.

Yu, B. M., Kemere, C., Santhanam, G., Afshar, A., Ryu, S. I., Meng, T. H., et al. (2007). Mixture trajectory models for neural decoding of goal-directed movements. *Journal of Neurophysiology, 97*, 3763–3780.

Zhang, K., Ginzburg, I., McNaughton, B. L., & Sejnowski, T. J. (1998). Interpreting neuronal population activity by reconstruction: Unified framework with application to hippocampal place cells. *Journal of Neurophysiology, 79*, 1017–1044.