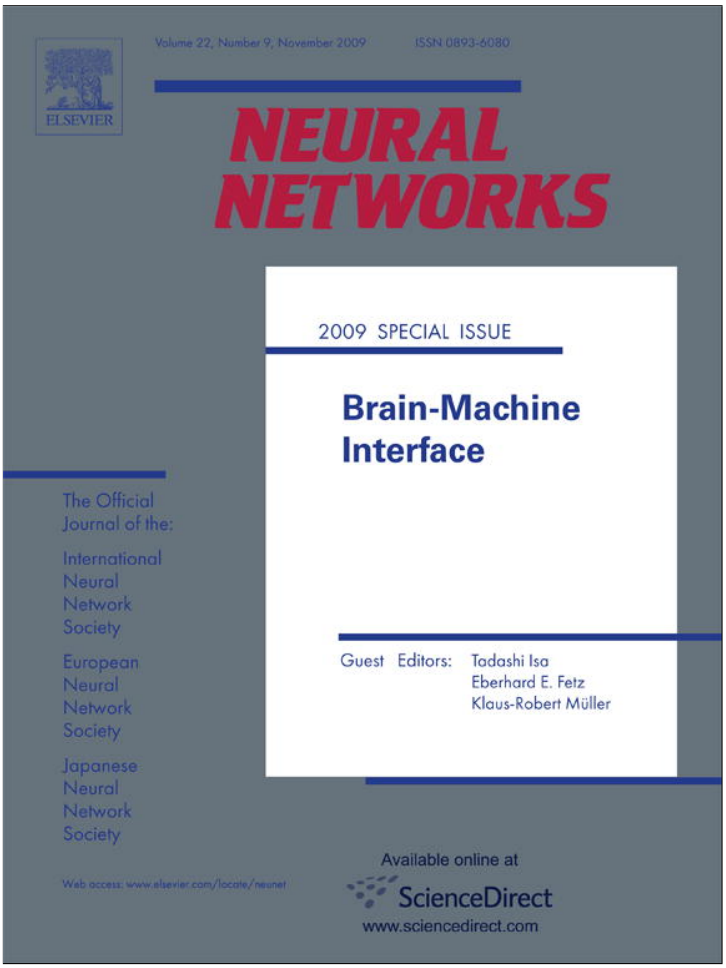

Neural Networks 22 (2009) 1203–1213

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2009 Special Issue

Bias, optimal linear estimation, and the differences between open-loop simulation and closed-loop performance of spiking-based brain–computer interface algorithms

Steven M. Chase [a,b,c,*], Andrew B. Schwartz [b,c], Robert E. Kass [a,c]

[a] Department of Statistics, Carnegie Mellon University, United States
[b] Department of Neurobiology, University of Pittsburgh, United States
[c] Center for the Neural Basis of Cognition, University of Pittsburgh and Carnegie Mellon University, United States

ARTICLE INFO

Article history:
Received 29 September 2008
Received in revised form 1 April 2009
Accepted 17 May 2009

Keywords:
Neural prosthetics
Decoding algorithms
Brain–machine interface

ABSTRACT

The activity of dozens of simultaneously recorded neurons can be used to control the movement of a robotic arm or a cursor on a computer screen. This motor neural prosthetic technology has spurred an increased interest in the algorithms by which motor intention can be inferred. The simplest of these algorithms is the population vector algorithm (PVA), where the activity of each cell is used to weight a vector pointing in that neuron's preferred direction. Off-line, it is possible to show that more complicated algorithms, such as the optimal linear estimator (OLE), can yield substantial improvements in the accuracy of reconstructed hand movements over the PVA. We call this open-loop performance. In contrast, this performance difference may not be present in closed-loop, on-line control.

The obvious difference between open and closed-loop control is the ability to adapt to the specifics of the decoder in use at the time. In order to predict performance gains that an algorithm may yield in closed-loop control, it is necessary to build a model that captures aspects of this adaptation process. Here we present a framework for modeling the closed-loop performance of the PVA and the OLE. Using both simulations and experiments, we show that (1) the performance gain with certain decoders can be far less extreme than predicted by off-line results, (2) that subjects are able to compensate for certain types of bias in decoders, and (3) that care must be taken to ensure that estimation error does not degrade the performance of theoretically optimal decoders.

1. Introduction

Recent improvements in neural recording technology make it possible to extract detailed recordings from ensembles of motor cortical neurons in real time. When coupled with an appropriate decoder, these recordings can be used to control the continuous motion of a cursor in virtual reality (Hochberg et al., 2006; Taylor, Tillery, & Schwartz, 2002) or an external device such as a robotic arm (Chapin, Moxon, Markowitz, & Nicolelis, 1999; Velliste, Perel, Spalding, Whitford, & Schwartz, 2008; Wessberg et al., 2000). This research has spawned an increased interest in what is known as the "decoding problem": what is the appropriate algorithm to turn neural signals into movement commands? For continuous cursor motion, choices range from the simple population vector algorithm (PVA) (Georgopoulos, Kettner, & Schwartz, 1988; Georgopoulos, Schwartz, & Kettner, 1986) through the optimal linear estimator (OLE) (Kass, Ventura, & Brown, 2005; Salinas & Abbott, 1994; Zhang, Ginzburg, McNaughton, & Sejnowski, 1998) to various versions of state space models and other more complicated implementations (Brockwell, Rojas, & Kass, 2004; Kemere, Shenoy, & Meng, 2004; Kim et al., 2003; Kulkarni & Paninski, 2008; Srinivasan, Eden, Willsky, & Brown, 2006; Wessberg et al., 2000; Wu, Gao, Bienenstock, Donoghue, & Black, 2006).

Several decoders have been tried in on-line control, with varying degrees of success. Neural prosthetic devices have been driven by PVA decoders (Taylor et al. (2002); Velliste et al. (2008)), linear filters based on position, instead of direction (Carmena et al. (2003); Hochberg et al. (2006); Serruya, Hatsopoulos, Paninski, Fellows, and Donoghue (2002)), and Kalman filters (Kim, Simeral, Hochberg, Donoghue, and Black (2008); Mulliken, Musallam, and Andersen (2008)). To our knowledge, only one study has compared the differences in on-line control with different algorithms (Kim et al., 2008), where it was concluded that velocity-based decoders were generally superior to position-based decoders. As decoding details and experimental paradigms differ greatly across labs, there

* Corresponding address: Department of Statistics, Carnegie Mellon University, 3025 E. Carson St.McGowan Institute, Rm. 245, 15203 Pittsburgh, PA, United States.
E-mail address: schase@andrew.cmu.edu (S.M. Chase).

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.05.005

is at this time no general consensus about which algorithms work best.

The relative performance of different decoders is more commonly assessed by comparing their ability to reconstruct trajectories off-line. For example, a monkey is trained to perform center-out or drawing movements with an arm while recordings are made from the motor cortex; these recordings are then used, after the fact, to reconstruct the movements made by the monkey. Comparing algorithms in this fashion is problematic, because of the assumption that open-loop decoding improvements will translate to closed-loop performance improvements, and further that open-loop decoding deficits will translate to closed-loop performance deficits. Two factors go into the reconstruction error that one computes from off-line data: bias and variance. These measures must be considered carefully, because they may not be the same under open-loop and closed-loop control. For example, it is well known that humans are capable of adapting to specific types of novel visuo-motor distortions (Bock, Abeele, & Eversheim, 2003; Ghahramani, Wolpert, & Jordan, 1996; Held & Freedman, 1963) or force field environments (Flash & Gurevich, 1991; Lackner & Dizio, 1994; Shadmehr & Mussa-Ivaldi, 1994). Also, it has been shown that humans can adapt to some types of perturbations more readily than others (Cunningham, 1989; Fukushi & Ashe, 2003; Hwang, Donchin, Smith, & Shadmehr, 2003; Krakauer, Pine, Ghilardi, & Ghez, 2000). If the decoded movement can be viewed as a distortion of the desired movement, then it may well be true that certain decoders may be more "learnable" than others.

Here we use both simulations and experiments to compare the performance of two decoding algorithms, the PVA and the OLE. We chose these two decoders because the PVA has been extensively used in the literature and the OLE is the theoretically optimal decoder that can be implemented in exactly the same fashion as the PVA, with only a change of decoding parameters. We find that open-loop simulation results do not match the closed-loop experimental differences between these two decoders. However, by making simple assumptions about how the task may be learned, we can construct a general framework for modeling the closed-loop performance of the decoding algorithms. We find that the performance gains in closed-loop simulation better match the experimental differences between the decoders, indicating that subjects readily compensate for directional bias. Finally, we investigate several implementations of the OLE and find that, in practice, estimation noise can degrade the performance of theoretically optimal implementations beyond the performance of their less optimal counterparts.

2. Methods

The experiment and simulations are based on a center-out movement task in 2 dimensions. We model both the real and simulated neurons as being cosine-tuned to movement direction (Georgopoulos, Kalaska, Caminiti, & Massey, 1982; Truccolo, Friehs, Donoghue, & Hochberg, 2008), where the firing rates, λ, depend on the movement direction, $\vec{d}$, through the equation

$$\lambda = b_0 + m\left(\vec{p}\cdot\vec{d}\right). \tag{1}$$

Here, b_0 is the baseline firing rate of the cell, m is its modulation depth, and $\vec{p}$ is its preferred direction. The following sections detail the experimental recording procedures (Section 2.1), how we estimate the tuning curves of both recorded and simulated neurons (Section 2.2), how firing rates are converted into cursor movements (Section 2.3), how decoding parameters are estimated for the PVA and the OLE (Section 2.4), the general format of the simulations (Section 2.5) and finally how the two simulation control modes (open-loop control / closed-loop control) are implemented (Section 2.6).

2.1. Experimental data: Behavioral paradigm and recording

All procedures were performed in accordance with the guidelines of the Institutional Animal Care and Use Committee of the University of Pittsburgh. One male Rhesus monkey (*Macaca mulatta)* was implanted with a 96-channel array (Cyberkinetics Neurotechnology Systems, Inc., Foxborough, MA) visually placed in the proximal arm area of primary motor cortex. Recordings were amplified, filtered, and sorted on-line with a 96-channel Plexon MAP system (Plexon Inc., Dallas, TX). Some of the units recorded were well-isolated single cells, and some contained two or more cells that could not be easily isolated from one another, but which were nevertheless tuned to the intended movement direction as a group.

The monkey was trained to sit in a primate chair facing a mirror that reflected an image from a computer monitor mounted above. In the center-out task, the monkey had to modulate the recorded neurons to move a cursor 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. Data for this experiment were taken approximately 1.5 years post-implant; the monkey was well trained to perform the 2D center-out task under brain-control.

We compared the performance of the two decoders by running the experiment in blocks, where cursor control was switched back and forth between the two decoders. Typically, 20 successful repetitions of each of the 8 targets were required before control switched to the other decoder. The switch was made instantaneously during the inter-trial period when the cursor was re-centered at the origin. Only one calibration session (described in Section 2.2) was used to determine the decoding parameters for both decoders; the OLE decoder was in control during this session.

2.2. Estimating the tuning functions

At the start of each experimental session, a calibration session was run to determine the tuning functions of the recorded cells. To initialize the system, the decoding parameters were randomly chosen. Targets for the 2D center-out task were then presented, one at a time in random order, and left on the screen until a movement time-out period elapsed (typically, 1 s). Due to the randomized decoding parameters, the cursor did not move much during this time, but the firing rates were still modulated in response to target presentation. Once an entire cycle set consisting of one presentation of each of the 8 targets was completed, the spike rates were linearly regressed against target direction according to the equation

$$f_{k_i} = \beta_{0_i} + \beta_{x_i} d_{kx} + \beta_{y_i} d_{ky} + \varepsilon_{k_i}, \tag{2}$$

where f_{ki} represents the firing rate of the ith neuron to the kth target presentation, d_{kx} and d_{ky} give the x and y directions of the kth target, respectively, the β's represent fitted regression coefficients, and ε_{ki} represents an additive noise. Note that for these regressions, the target directions are scaled to be unit vectors. The true baseline firing rate, b_{0i}, is estimated as β_{0i}; the true modulation depth, m_i, is estimated as the magnitude of the vector $\vec{\beta}_i = \left(\beta_{x_i}, \beta_{y_i}\right)$; and the true preferred direction, $\vec{p}_i$, is estimated as $\vec{\beta}_i/m_i$. The estimates of the tuning parameters were then used to calculate the corresponding decoding parameters for each neuron, b^D_{0i}, m^D_i, and $\vec{p}^D_i$, according to the procedures outlined below. Once the new decoding parameters were calculated, another cycle set of targets was presented, and the process repeated until the monkey was able to complete the center-out task reliably. Typically, only 3–5 cycle sets of data were needed to achieve good control. Cells with modulation depths of less than 4 Hz were not used for control.

2.3. Converting firing rates to cursor movements

The PVA and the OLE can be implemented in the same way, with different decoding parameters. Once these parameters are known, they were used to construct cursor trajectories as follows. Spike counts were sampled at a regular rate of 30 Hz and converted to rates $f_i(t)$ by dividing by the sampling interval. Normalized rates $r_i(t)$ were computed through the equation

$$r_i(t) = \frac{f_i(t) - b_{0i}^D}{m_i^D}. \tag{3}$$

These normalized rates were then smoothed with a 5-point boxcar filter (i.e., by averaging the rates from the last 5 bins). The filtered, normalized rates were converted to cursor velocity, $\vec{v}(t)$, as:

$$\vec{v}(t) = k_s \frac{n_D}{N} \sum_{i=1}^{N} r_i(t)\, \vec{p}_i^D, \tag{4}$$

where n_D is the number of movement dimensions (in our case, 2) and N is the number of units being used for decoding. The speed factor, k_s, converts the magnitude of the population vector from a normalized range to a physical speed in mm/s; typical values ranged from 65 to 80. Finally, cursor position, $\vec{C}_P$, was updated every sampling interval as

$$\vec{C}_P(t) = \vec{C}_P(t - \Delta t) + \Delta t \vec{v}(t). \tag{5}$$

Trajectories always started at the origin.

2.4. Calculating the decoding parameters

Once the tuning curves of the cells were estimated it was possible to calculate the decoding parameters appropriate for each cell.

For the PVA, the decoding parameters are exactly the corresponding parameters of the tuning curve, that is, $b_{0i}^D = \beta_{0i}$, $m_i^D = \left|\vec{\beta}_i\right|$, and $\vec{p}_i^D = \vec{\beta}_i / \left|\vec{\beta}_i\right|$. This implementation has a straightforward interpretation: each cell pushes the cursor in the direction of its preferred direction, with the amount of the push being scaled by its normalized firing rate.

As explained in the Results section, the PVA assumes a uniform distribution of preferred directions. The OLE corrects for any non-uniformity in this distribution by using a different set of decoding preferred directions (the baseline firing rate and modulation depth decoding parameters remain the same). Consider the following derivation. From Eqs. (1) and (2), if we gather the normalized firing rates of all cells in response to a direction $\vec{d}$ into a single vector $\vec{r}$, we have

$$\vec{r} = \mathbf{B}\vec{d}, \tag{6}$$

where $\mathbf{B}$ is formed by gathering all of the preferred direction vectors $\vec{\beta}_i/m_i$ into a single N × 2 matrix. We are interested in the reverse problem of calculating an intended direction from the sampled firing rates, i.e., solving

$$\vec{d} = \mathbf{P^D}\vec{r}. \tag{7}$$

It follows, by inversion of Eq. (6), that the OLE decoding parameters $\mathbf{P^D}$ are

$$\mathbf{P^D} = \alpha \left(\mathbf{B}'\mathbf{B}\right)^{-1} \mathbf{B}', \tag{8}$$

where $\mathbf{X}^{-1}$ denotes the inverse of matrix $\mathbf{X}$ and $\mathbf{X}'$ denotes matrix transposition. Note that unlike the PVA, the decoding preferred directions for the OLE are not necessarily of a uniform length. To ensure that the average overall speed of the cursor is the same when using the two decoders, we apply a scaling factor, α, so that the average length of the decoding preferred directions in the OLE is 1, the same as for PVA. Also note that the OLE decoding parameters equal the PVA decoding parameters if the preferred directions of the recorded cells are uniformly distributed, such that $\left(\mathbf{B}'\mathbf{B}\right)^{-1}$ is diagonal. Alternate derivations of the OLE can be found in (Kass et al. (2005); Salinas and Abbott (1994)).

The linear regression solution used in Eq. (8) is valid only when the input covariates are independent, identically distributed random variables. In other words, this solution is valid only when the recorded neurons have the same noise variance and have no correlations beyond the signal correlation inherited from their tuning. If this is not true, the OLE decoding preferred directions should be calculated using weighted linear regression as

$$\mathbf{P^D} = \alpha \left(\mathbf{B}'\boldsymbol{\Sigma}^{-1}\mathbf{B}\right)^{-1} \mathbf{B}'\boldsymbol{\Sigma}^{-1}, \tag{9}$$

where $\boldsymbol{\Sigma}$ represents the covariance matrix of the residuals of the linear regression fits of the individual neurons. The calculation in Eq. (9) relies on estimating more parameters than the calculation in Eq. (8), and so may suffer more from estimation error due to limited sample sizes. We will explore this issue in the Results section. To differentiate these implementations, we call the OLE of Eq. (8) the "minimal OLE", which corrects only for bias due to non-uniformly distributed preferred directions (Fig. 1(C)). When $\boldsymbol{\Sigma}$ is a diagonal matrix, consisting of the variance in the spike rate of each neuron after accounting for direction modulations, the OLE additionally accounts for the different amount of noise provided by each neuron. We call this implementation the "variance-only OLE". Finally, when $\boldsymbol{\Sigma}$ is the full covariance matrix, it also accounts for noise correlations in the spike rates; we call this the "full OLE" implementation.

2.5. Simulation setup

The simulations were meant to mimic the experimental approach as closely as possible. To start, we generated cosine-tuning curves for N simulated cells by drawing preferred directions at random from a uniform distribution over $[0, 2\pi]$, baseline firing rates from a uniform distribution over [5, 10], and modulation depths from a uniform distribution over [4, 8]. While these parameters largely produced only positive firing rates, we enforced non-negative rates by clipping them to zero. Once the true tuning curves were established, we generated binned spike counts at 30 Hz as Poisson realizations of the underlying rate parameter for the specified intended direction in that bin. From then on, spike trains were treated just as they were in experiments.

Simulated experiments started with a calibration session to estimate the tuning parameters of the simulated units, allowing for situations in which the estimated tuning curves could differ from the true tuning curves. The estimated tuning parameters were used to generate decoding parameters that were in turn used to translate firing rates into cursor motion. Once the tuning curves were estimated and the decoding parameters set, 20 simulated brain-control center-out trajectories were generated to each of 16 targets spaced uniformly around a circle, using Eqs. (3)–(5).

For analysis purposes, an average trajectory was calculated from the 20 individual repetitions to each target. To combine trajectories of different durations, the time axis of each trajectory was uniformly scaled to the mean movement duration, τ_{Avg}. That is, the time samples for an individual movement t of duration τ were scaled by a gain factor $\gamma = \tau_{Avg}/\tau$ to create a new set of time samples $t_s = t\gamma$. Each x and y component of the trajectory was then independently resampled using spline interpolation to a common time axis consisting of 200 evenly sampled points. Finally, the mean and SE were calculated separately for each time point of the x and y components.

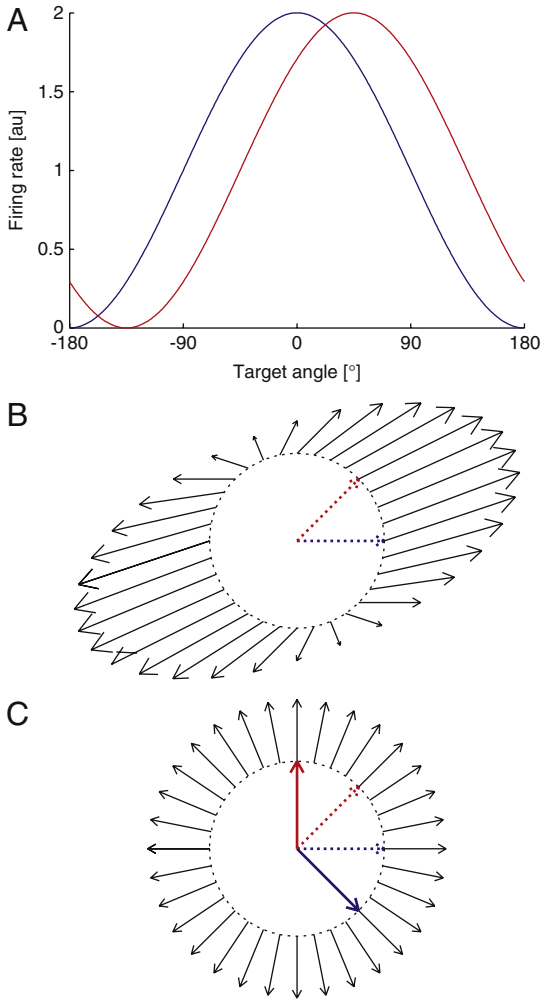

Fig. 1. Bias in the PVA. (A) Tuning functions of two simulated neurons, one with a preferred direction of 0° (blue), the other with a preferred direction of 45° (red). (B) The black lines denote the direction of actual cursor movement computed with the PVA (the vectors $\vec{v}$ from Eq. (4)), for desired movement directions located around the unit circle. Dotted lines denote the preferred directions of the neurons driving the PVA (same neurons as in (A)). Note that for a desired movement direction of 90°, the cursor moves both upwards and to the right. (C) Same as B, but with cursor movements computed with the OLE. Dashed colored lines denote the preferred directions of the simulated neurons; solid colored lines denote the corresponding preferred directions used for OLE decoding. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2.6. Simulation control modes

To simulate each trial, we first chose a direction in which to aim, referred to below as $\vec{d}_{in}$. We then generated spikes from the simulated neurons according to their tuning curves and the aiming direction, and used those firing rates and the decoder parameters to compute the resultant movement of the cursor. The only difference between the open-loop and closed-loop simulations was the method by which we choose the direction in which to aim.

Open-loop control was simulated by assuming that the subject aimed directly at the target, regardless of the calculated direction of cursor movement. That is, a direction vector pointing from the center of the workspace to the presented target was used to drive the firing rates of all the simulated cells and was maintained until the cursor either exited the ring of targets or a timeout of 10 s expired.

Closed-loop control was simulated by assuming that the subject could learn the relationship between aiming direction and associated cursor movement (which may not match the intended movement), and could use this knowledge to plan trajectories. We assumed that the goal was to make the movements as accurate and straight as possible under the current decoding parameters, so we chose aiming directions that would result in cursor movements that pointed directly toward the target, on average. To calculate these aiming directions, it is helpful to realize that since both the PVA and the OLE are linear methods, the entire process transforming the aiming direction $\vec{d}_{in}$ into the cursor movement $\vec{d}_{out}$ can be carried out as a matrix multiplication:

$$\vec{d}_{out} = \mathbf{PVM}\,\vec{d}_{in}, \tag{10}$$

where the population vector matrix, **PVM**, is given as

$$\mathbf{PVM} = \frac{k_s n_D}{N}\mathbf{P^D P}. \tag{11}$$

Essentially, multiplying the aiming direction, $\vec{d}_{in}$, by the true preferred direction matrix **P** transforms it into a set of firing rates for the population of simulated neurons. Multiplying these firing rates by the decoding preferred direction matrix $\mathbf{P^D}$ transforms them into the expected cursor movement, $\vec{d}_{out}$, under certain assumptions. A derivation of this equation is provided in the Appendix. Thus, for the closed-loop simulations, when target direction $\vec{t}$ was presented, we drove the simulated neurons with an aiming direction, $\vec{d}_{in}$, of $\mathbf{PVM}^{-1}\,\vec{t}$ (normalized), until the cursor exited the ring of targets or the 10 s timeout expired.

In short, the difference between open and closed-loop control is that in open-loop control, the subject aims directly for the targets, whereas in closed-loop control, the subject uses his knowledge of cursor error to aim in the direction that should move the cursor towards the target. Note that we did not simulate on-line correction of trajectories. This would have required a much more involved simulation, including assumptions about how errors are translated into movement updates and the lag at which neurons respond. Since these processes are not well understood, we avoided them. Instead, these simulations are meant to mimic updates of the subject's internal model (Wolpert, Ghahramani, & Jordan, 1995) of the cursor dynamics and the results that would occur under blind application of that internal model.

3. Results

When the preferred directions of the recorded neurons are not uniformly distributed, PVA decoding is biased. To see this, consider a case where only two cosine-tuned neurons are used to control the cursor: one with a preferred direction of 0°, the other with a preferred direction of 45° (Fig. 1(A)). If a movement is attempted toward a target located at 0°, both neurons will have firing rates above their baseline activity, and so will have positive normalized rates and contribute positive weights to the population vector. The resulting cursor movement will be in a direction between the preferred directions of the two neurons, not towards the aimed-at target. Fig. 1(B) shows the population vectors that result from attempted movements to targets spaced around the unit circle, where the tail of each arrow starts at the aiming point. Not only do the population vectors show a consistent bias towards the axis formed by summing the constituent preferred directions, the size of the population vectors also vary as a function of aiming direction. Since the size of the vector correlates with cursor speed, this will cause a speed asymmetry across the workspace.

The OLE corrects for this source of bias by decoding with a different set of preferred directions than the ones defined by the tuning curves. When these are calculated according to Eq. (8), the cursor movements that result from attempted movements to targets spaced around the unit circle are both in the same direction as the attempted movement and are of a uniform length (Fig. 1(C)).

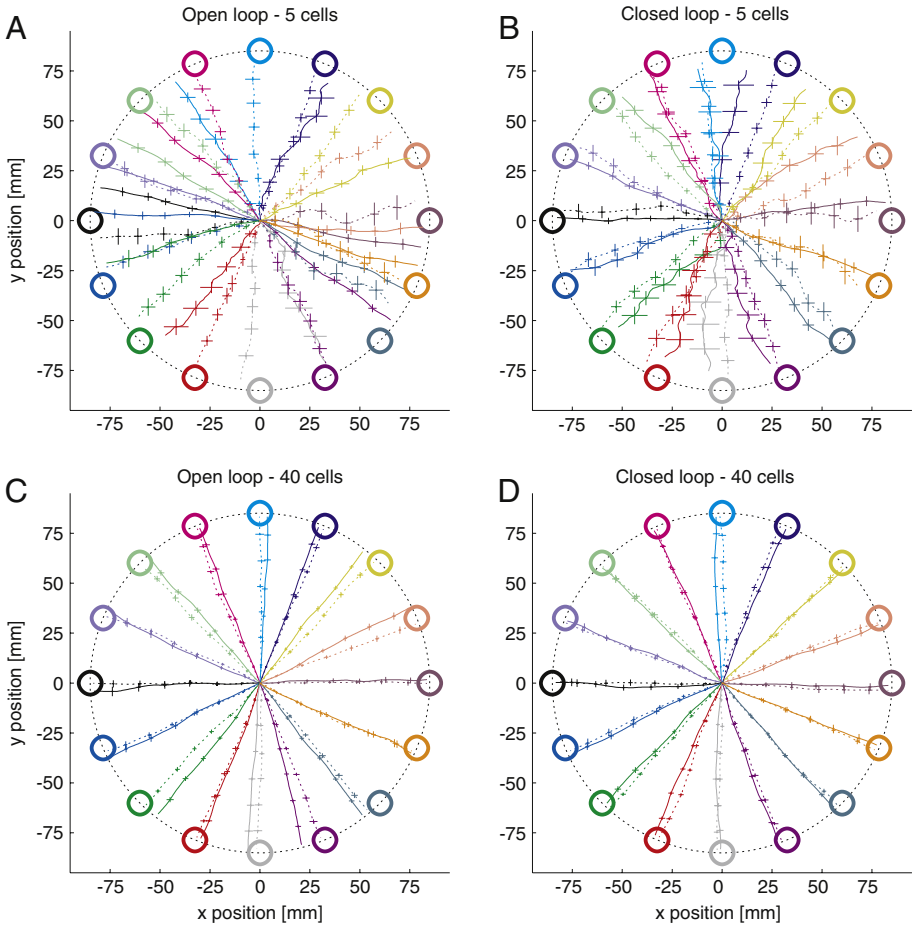

Fig. 2. Simulation results. Mean (±1 SE) of the trajectories to each of the 16 targets. PVA trajectories are shown as solid lines, minimal OLE trajectories as dashed lines. (A) Open-loop control using 5 simulated neurons. (B) Closed-loop control with the same 5 cells. (C) Open-loop control with 40 neurons. (D) Closed-loop control with those 40 cells.

As a result of the bias inherent in the PVA decoder, simulations of open-loop control show the OLE to be far more accurate than the PVA. Fig. 2(A), (C) show two example simulations, comparing the average trajectories under PVA control (solid lines) to the average trajectories under minimal OLE control (dashed lines) using either 5 (Fig. 2(A)) or 40 (Fig. 2(C)) simulated neurons. While the PVA trajectories consistently miss the targets, the OLE trajectories typically end near the targets. This situation is rectified under simulations of closed-loop control, where it is assumed that the subject has learned the mapping between intended movement direction and resultant cursor movement, and re-aims accordingly (Fig. 2(B), (D)). Under these conditions, the PVA and OLE are equally accurate.

The results of all simulations are shown in Fig. 3, plotted as a function of the number of simulated neurons. The data are heavily averaged, to give an indication of the differences in decoding methods and control modes independent of the individual variation in tuning parameters from experiment to experiment. Each point represents the average of 50 simulated experiments, with a new draw of random tuning curves each time; each experiment represents the average across the 16 targets; and each target represents the average of the 20 individual repetitions to that target.

The open-loop PVA simulation behaves the worst in terms of angular error (Fig. 3(A)). However, there are also slight, though statistically significant, differences between the open-loop OLE and the closed-loop OLE in terms of angular error, where the open-loop error is always greater than the closed-loop error. For 5 simulated cells, the error difference between open and closed-loop control for OLE is $0.55° \pm 0.25°$ (SE) ($p = .03$, t-test); for 160 cells, it is $0.18° \pm 0.05°$, ($p = 10^{-4}$). These performance differences stem from a misestimation of the true tuning parameters during the calibration session. If the tuning parameters are misestimated, the OLE decoding parameters will not be correct, which can lead to a decoder that has a small amount of bias in it. This bias can be compensated for in closed-loop control. The differences in angular error between the OLE and PVA decoders under closed-loop control were not statistically significant.

The error improvement of the PVA under closed-loop control does not come without a price. Movements made under closed-loop PVA control are consistently slower than the movements with any other decoder or control scheme (Fig. 3(B)). Also, the closed-loop control scheme does not correct for the speed asymmetry exemplified by the change in arrow length as a function of target angle in Fig. 1(B). When the difference between the times required to hit targets in the slowest and fastest directions is computed (Fig. 3(D)), the PVA-decoding asymmetry is always larger than the OLE-decoding asymmetry. The speed asymmetry is paralleled by a change in the variance of the trajectories as a function of target angle. Although the standard deviation of the average trajectory under the two decoders is roughly equivalent when averaged across the workspace (less than 1 mm difference under all conditions, Fig. 3(C)), the difference between the standard deviations of the maximally and minimally variable trajectories is larger with PVA decoding than with OLE decoding (Fig. 3(E)).

That speed asymmetry still exists under simulated closed-loop control is a consequence of the method by which we generated closed-loop trajectories. We assumed that while the subject could

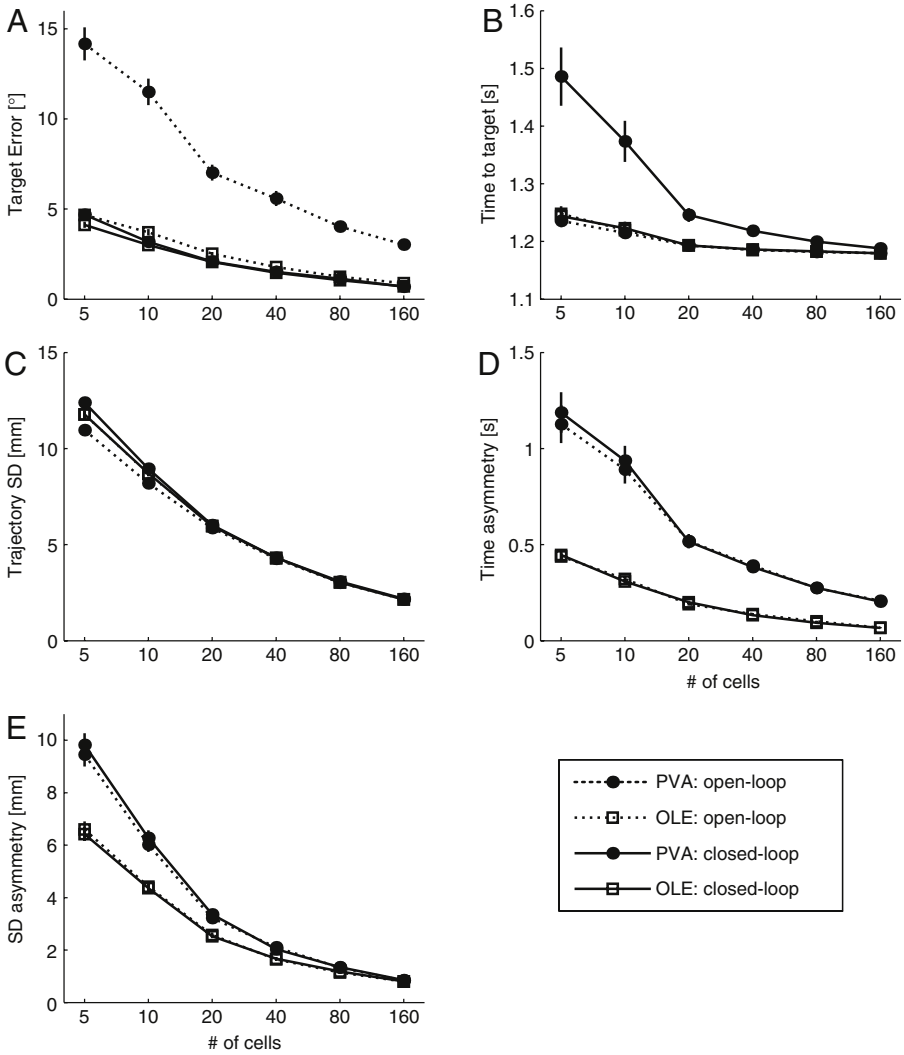

Fig. 3. A comparison of the simulation performance under different decoders and control modes. (A) Average target error as a function of the number of simulated neurons. Error is calculated as the angular difference between the target direction vector and the vector from the origin to the point at which the trajectory exits the ring of targets. (B) Average time to target acquisition. (C) Average trajectory standard deviation. (D) The average time asymmetry, defined as the average difference between the maximum and minimum time to acquire a target. (E) Difference between the maximum SD of a target's average trajectory and the minimum SD. All error bars denote the SE over the 50 different simulated experiments.

modulate his intended movement direction to compensate for errors in movement direction, he had no control over his intended movement speed. Thus, every intended movement was a unit vector pointing in a certain direction.

These closed-loop simulation results match the on-line performance of our experimental subject well. Fig. 4 shows an example of the center-out trajectories from an experiment switching between PVA decoding and variance-only OLE decoding (recall that in the variance-only OLE implementation, decoding parameters are calculated according to Eq. (9), where $\boldsymbol{\Sigma}$ is the diagonal matrix whose entries are the noise variances of the cells). During PVA decoding (Fig. 4(A)), the trajectories to each target are fairly straight and well separated; there is little indication of the bias that we would expect from open-loop simulations. This bias exists though; when the firing rates that generated the PVA trajectories are decoded off-line with the variance-only OLE decoder, the off-line trajectories are consistently skewed toward the axis running at 45° (Fig. 4(C)). The opposite effect was observed when the monkey was using the variance-only OLE decoder with the same set of recorded cells: movements made on-line with this decoder were typically straight (Fig. 4(B)), while movements decoded off-line with the PVA decoder are biased towards the axis running at 135° (Fig. 4(D)). We suspect that under OLE control, the subject is aiming for the targets, and the PVA is decoding these trajectories in a biased manner, clustering them toward the axis at 45°. Under PVA control, the subject is re-aiming to compensate for this bias, which requires aiming toward points clustered around the axis at 135°. It should be noted that the success rate for this task was 100%. Using visual feedback, the monkey was able to rapidly switch between the two control modes and compensate for the bias imposed by the decoder.

Although the subject compensated for the visuo-motor distortion caused by the PVA decoder, he did not compensate for the speed asymmetry. Fig. 6(A) shows the time to target acquisition plotted as a function of target angle for both decoders. The movement times achieved with the PVA decoder have a bimodal distribution, which would be expected from the speed asymmetry profiles shown in Fig. 1(B). The OLE decoded trajectories do not exhibit this bimodal distribution, although they do show large variation with target direction. This is probably due to cursor drift, which could be caused by misestimating the baseline firing rates of the recorded neurons.

One surprising finding from this experiment was that the OLE trajectories are slightly more variable than the corresponding PVA trajectories (the lines in Fig. 4(C) are more dispersed than those

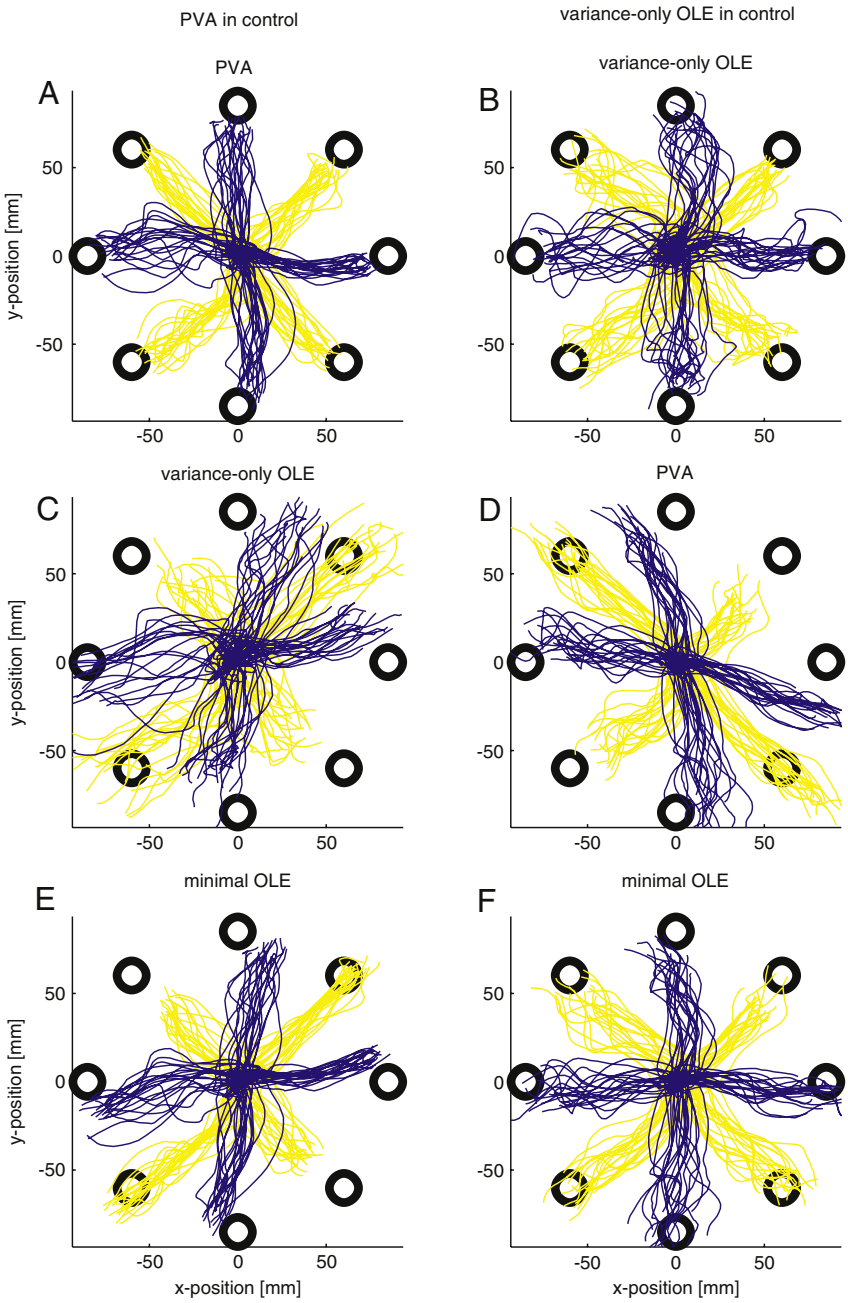

Fig. 4. Experimental results comparing PVA decoding (left hand column) to variance-only OLE decoding (right hand column). The same 30 cells were used with each decoder. Color changes across targets are to aid visibility. (A) Actual cursor trajectories during PVA control. (C), (E) Trajectories that result when the spike trains recorded during the PVA-controlled trajectories in A are decoded off-line with the variance-only OLE decoder (C) or the minimal OLE decoder (E). Essentially, these are the open-loop trajectories that result from OLE decoding of a PVA-controlled system. (B) Actual cursor trajectories during variance-only OLE control. (D), (F) Trajectories that result when the spike trains from the variance-only OLE-controlled trajectories in B are decoded with the PVA decoder (D) or the minimal OLE decoder (F).

in Fig. 4(A), and similarly for Fig. 4(B) relative to Fig. 4(D)). These variance differences are significant: the average SD ($\pm$ SE) of the trajectories during PVA control was 5.5 $\pm$ .35 mm, the average during OLE control was 7.9 $\pm$ .46 mm ($p < .001$, as assessed by a two factor ANOVA based on direction and decoder type). The variance increase appears to be a consequence of the variance-only implementation of the OLE; when the same trajectories were decoded off-line using a minimal OLE decoder, the variance during the OLE session decreased (average SD: 5.8 $\pm$ .29 mm; Fig. 4(E), (F)). To verify that that specific OLE implementation was the cause of the variance differences, we performed another experiment switching between a PVA decoder and a minimal OLE decoder (Fig. 5). Although there is less indication of bias in this example (the lines in Fig. 5(C) and (D) are relatively on target), the main result is that the minimal OLE decoding is, if anything, less variable than the PVA decoding (PVA SD: 5.2 $\pm$ .28 mm; minimal OLE SD: 5.0 $\pm$ .29 mm, $p > .05$). Trajectories that would have resulted from variance-only OLE decoding, however, prove to be quite variable (Fig. 5(E) and (F); during the OLE decoding session, the variance-only OLE SD was 7.8 $\pm$ .45 mm).

To help understand these results, we calculated, from simulations, the average standard deviation of trajectories resulting from different implementations of the OLE decoder as a function of the amount of data collected during the calibration session (Fig. 7). The variance-only OLE decoder performed better than the full OLE decoder, but worse than the minimal OLE decoder until more than

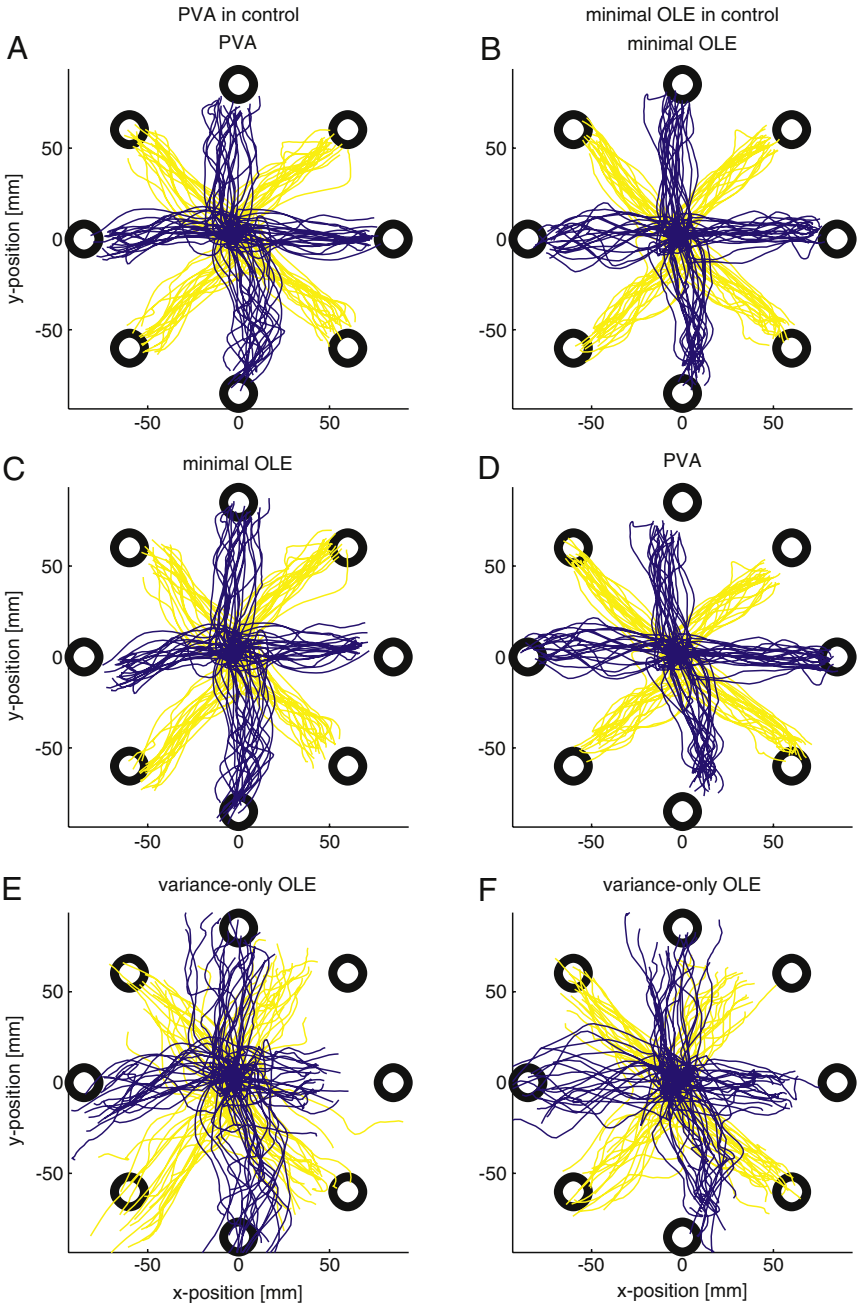

Fig. 5. Same as Fig. 4, for an experiment where PVA decoding was compared to minimal OLE decoding. 33 cells drove the cursor in this experiment.

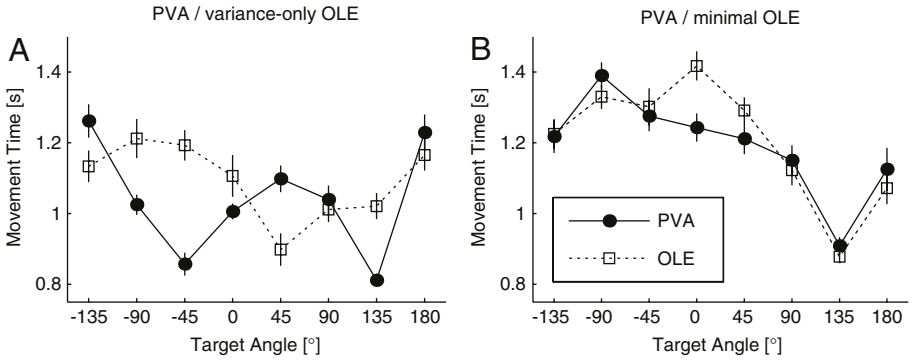

Fig. 6. Time to acquire target plotted as a function of target angle for the experiments of Figs. 4 and 5. (A) PVA compared to the variance-only OLE (corresponding to Fig. 4). (B) PVA compared to the minimal OLE (corresponding to Fig. 5).

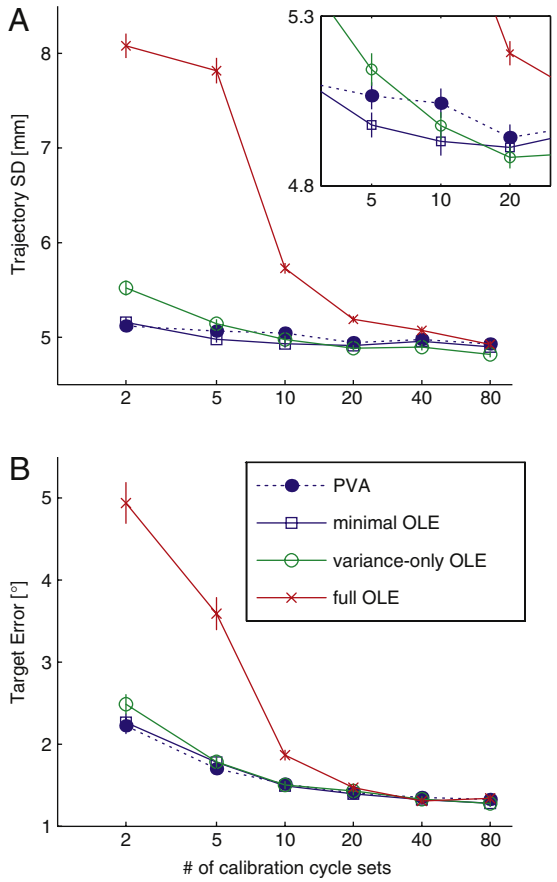

Fig. 7. Decoding performance of the different decoders under closed-loop control, for 30 simulated neurons. (A) Average trajectory SD (± 1 SE) plotted as a function of the amount of data collected during the calibration session, where a cycle set is one presentation of each of 8 targets spaced uniformly around a circle. The inset shows a close up of the given range. (B) Same, for the target angular error.

10 cycle sets of data were collected for the calibration session. As we never used more than 5 cycle sets of calibration data in our experiments, these simulation results match the experimental data well.

The full OLE is theoretically optimal. However, its performance in terms of the average trajectory SD was worse than all of the other decoders until about 80 cycle sets of calibration data were collected. At one target presentation every 2 s, this would take more than 20 min. As an aside, it should be noted that although we attempted to use the full OLE in experiments, the monkey refused to work with that decoder, giving up in frustration after several movement attempts.

4. Discussion

We have demonstrated that a subject using a brain–computer interface can compensate for at least the directional biases that result from the use of the PVA decoder. This result is not particularly surprising. The rotation between the aimed-at direction and the cursor direction induced by the PVA varies smoothly and slowly across the workspace (Fig. 1(B)), so that generalization from one target should not interfere with movements to another (Krakauer et al., 2000; Paz, Nathan, Boraud, & Bergman, 2005). Furthermore, the average angle between the intended and actual directions decreases as a function of the number of cells used in control, averaging less than 10° when using more than 20 neurons with randomly distributed preferred directions (Fig. 3(A), open-loop PVA control).

4.1. Off-line simulations must be detailed to capture on-line results

Care must be taken when using off-line results to predict the performance of on-line decoders. Under closed-loop control, the PVA and the OLE are essentially the same in terms of target angular error or average trajectory variability, whereas under open-loop control the OLE is decidedly better (Fig. 3). Of course, the differences in off-line prediction would not have been as severe if we were comparing the performance of two unbiased decoders. Even this situation must be approached with caution, however. First, if parameters are misestimated during the calibration session, even theoretically unbiased decoders can become biased. Second, from the experimental results it is clear that the variance-only OLE, which is theoretically superior to the minimal OLE, gave worse results in terms of trajectory jitter (Figs. 4 and 5). We should note that the performance of the variance-only OLE or full OLE might have been better had we used a different estimator for the covariance matrix in Eq. (9). For example, the approach of Daniels and Kass (2001) has been shown to give more consistent results for small sample sizes than the standard sample variance estimator we used. The point we would like to make, however, is that only careful modeling of the calibration procedure reveals the sensitivity of the decoders to noisy estimates of the extra parameters (Fig. 7).

The calibration process is a complicated issue in brain–computer interfaces. On one hand, it seems obvious that one should gather as much data as possible when trying to make estimates of the true tuning curves of the recorded cells. On the other hand, it is possible that the tuning curves themselves are non-stationary, due to random drift (Rokni, Richardson, Bizzi, & Seung, 2007), postural changes, electrode noise, or simply due to attention or fatigue. Even if the tuning curves themselves are stationary, the subject's intention may not be: it is likely that the learning process begins as soon as movements are attempted (Mazzoni & Krakauer, 2006). Further work will be necessary to understand exactly how the parameter estimation procedure itself affects control.

4.2. Speed asymmetry

One implementation of brain–computer interface devices is as spelling devices for locked-in patients. In these cases, one can take the recorded spike trains and, instead of using them to drive the continuous movement of a cursor, use them to distinguish among several different possible categories (Musallam, Corneil, Greger, Scherberger, & Andersen, 2004; Santhanam, Ryu, Yu, Afshar, & Shenoy, 2006). One could do this with signals from the motor cortex by using a cursor and allowing the center-out target selection to serve as the category choice. In this case, asymmetries in speed or variance across the workspace would translate directly into asymmetries in the ability to select certain categories over others, and would degrade device performance.

The subject in our experiment did not compensate for the speed asymmetry across the workspace that he experienced when working with either the PVA or the OLE controller (Fig. 6). It is difficult to interpret whether or not he could do so, however, as our task did not specifically control for speed, other than providing an upper bound on the overall movement time that was acceptable for success. Most likely, the optimal strategy for the subject to adopt in our task is to hit every target as rapidly as possible in order to get the reward as quickly as possible. In this case, the speed asymmetry might reflect the performance upper bound. Further experiments will need to be conducted to probe whether or not speed bias can be corrected under closed-loop conditions.

4.3. Conclusions

So which of these decoding algorithms is better? One could argue that the minimal implementation of the OLE is the best, since

it is unbiased (given linear neurons) and less prone to estimation error than the other OLE implementations. From the point of view of overall control ability, however, it appears not to matter: our subject compensated for the directional inconsistencies between the two decoders, and while the average speed per target changed, the overall difference between the fastest and slowest target was approximately the same. While it is possible that this subject, who was extensively trained with the PVA decoder, might be more adept at compensating for directional biases than an untrained subject, these results indicate that there is no fundamental limitation on the ability to learn the directional distortions of the PVA. The important point, however, is that the differences between PVA and OLE are not nearly as extreme as open-loop simulation results would indicate. Only simulations that account for the subject's ability to learn can approximate closed-loop experimental results.

One interesting aspect of this work is that it implies that a minimum variance, unbiased estimator may not outperform a biased estimator under closed-loop control, provided that the leftover bias is readily learnable. To take advantage of this, it is necessary to know what kinds of perturbations are learnable and what kinds are not. The full extent to which a subject can learn to modulate its neural activity when given the proper form of feedback is not well understood, and merits further investigation from both a scientific and a practical, device-oriented standpoint. The brain–computer interface promises to be a flexible platform on which such types of learning studies could be done.

Acknowledgements

The authors would like to thank Andrew Whitford and Dr. Meel Velliste for comments on an earlier draft of this manuscript. Support for this work came from NIH grant R01-EB005847, part of the CRCNS program.

Appendix. Derivation of the population vector matrix (Eq. (11))

Inserting Eq. (3) into Eq. (4), we have

$$\vec{v}\,(t) = k_s \frac{n_D}{N} \sum_{i=1}^{N} \frac{f_i\,(t) - b_{0_i}^D}{m_i^D} \vec{p}_i^D. \tag{12}$$

If $b_{0_i} = b_{0_i}^D$ and $m_i = m_i^D$, and we ignore the stochastic nature of $f_i(t)$ by taking the expectation of each side, then we can rewrite the above equation by substituting in Eq. (1):

$$E\left[\vec{v}\,(t)\right] = k_s \frac{n_D}{N} \sum_{i=1}^{N} \left(\vec{p}_i \cdot \vec{d}_{in}\right) \vec{p}_i^D. \tag{13}$$

Equating $E\left[\vec{v}\,(t)\right]$ with $\vec{d}_{out}$, extracting the intended direction $\vec{d}_{in}$ out to the right-hand side, and rewriting in matrix form yields

$$\vec{d}_{out} = \frac{k_s n_D}{N} \begin{bmatrix} \sum_{i=1}^{N} p_{x_i} p_{x_i}^D & \sum_{i=1}^{N} p_{y_i} p_{x_i}^D \\ \sum_{i=1}^{N} p_{x_i} p_{y_i}^D & \sum_{i=1}^{N} p_{y_i} p_{y_i}^D \end{bmatrix} \vec{d}_{in}, \tag{14}$$

or, equivalently, Eqs. (10) and (11):

$$\vec{d}_{out} = \frac{k_s n_D}{N} \mathbf{P^D P} \vec{d}_{in}.$$

As a check to ensure that the OLE is unbiased, we can substitute in Eq. (8) for the OLE decoding parameters,

$$\vec{d}_{out} = \frac{k_s n_D \alpha}{N} \left(\mathbf{B}'\mathbf{B}\right)^{-1} \mathbf{B}'\mathbf{P} \vec{d}_{in},$$

where we recall that $\mathbf{P}$ is the matrix of true preferred directions, and $\mathbf{B}$ is its noisy estimate from the calibration session. If $\mathbf{B} = \mathbf{P}$, $\vec{d}_{out}$ becomes a scaled, undistorted version of $\vec{d}_{in}$.

References

Bock, O., Abeele, S., & Eversheim, U. (2003). Human adaptation to rotated vision: Interplay of a continuous and a discrete process. *Experimental Brain Research*, *152*, 528–532.

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.

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. (2003). Learning to control a brain–machine interface for reaching and grasping by primates. *PLoS Biology*, *1*, E42.

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

Cunningham, H. A. (1989). Aiming error under transformed spatial mappings suggests a structure for visual-motor maps. *Journal of Experimental Psychology: Human Perception and Performance*, *15*, 493–506.

Daniels, M. J., & Kass, R. E. (2001). Shrinkage estimators for covariance matrices. *Biometrics*, *57*, 1173–1184.

Flash, T., & Gurevich, I. (1991). Arm stiffness and movement adaptation to external loads. *Proceedings of IEEE Engineering in Medical and Biology Society Conference*, *13*, 885–886.

Fukushi, T., & Ashe, J. (2003). Adaptation of arm trajectory during continuous drawing movements in different dynamic environments. *Experimental Brain Research*, *148*, 95–104.

Georgopoulos, A. P., Kalaska, J. F., Caminiti, R., & Massey, J. T. (1982). On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neuroscience*, *2*, 1527–1537.

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

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

Ghahramani, Z., Wolpert, D. M., & Jordan, M. I. (1996). Generalization to local remappings of the visuomotor coordinate transformation. *Journal of Neuroscience*, *16*, 7085–7096.

Held, R., & Freedman, S. J. (1963). Plasticity in Human Sensorimotor Control. *Science*, *142*, 455–462.

Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., Branner, A., Chen, D., Penn, R. D., & Donoghue, J. P. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, *442*, 164–171.

Hwang, E. J., Donchin, O., Smith, M. A., & Shadmehr, R. (2003). A gain-field encoding of limb position and velocity in the internal model of arm dynamics. *PLoS Biology*, *1*, E25.

Kass, R. E., Ventura, V., & Brown, E. N. (2005). Statistical issues in the analysis of neuronal data. *Journal of Neurophysiology*, *94*, 8–25.

Kemere, C., Shenoy, K. V., & Meng, T. H. (2004). Model-based neural decoding of reaching movements: A maximum likelihood approach. *IEEE Transactions in Biomedecal Engineering*, *51*, 925–932.

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

Kim, S. P., Simeral, J. D., Hochberg, L. R., Donoghue, J. P., & Black, M. J. (2008). Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. *Journal of Neural Engineering*, *5*, 455–476.

Krakauer, J. W., Pine, Z. M., Ghilardi, M. F., & Ghez, C. (2000). Learning of visuomotor transformations for vectorial planning of reaching trajectories. *Journal of Neuroscience*, *20*, 8916–8924.

Kulkarni, J. E., & Paninski, L. (2008). State-space decoding of goal-directed movements. *IEEE Signal Processing Magazine*, *25*, 78–86.

Lackner, J. R., & Dizio, P. (1994). Rapid adaptation to Coriolis force perturbations of arm trajectory. *Journal of Neurophysiology*, *72*, 299–313.

Mazzoni, P., & Krakauer, J. W. (2006). An implicit plan overrides an explicit strategy during visuomotor adaptation. *Journal of Neuroscience*, *26*, 3642–3645.

Mulliken, G. H., Musallam, S., & Andersen, R. A. (2008). Decoding trajectories from posterior parietal cortex ensembles. *Journal of Neuroscience*, *28*, 12913–12926.

Musallam, S., Corneil, B. D., Greger, B., Scherberger, H., & Andersen, R. A. (2004). Cognitive control signals for neural prosthetics. *Science*, *305*, 258–262.

Paz, R., Nathan, C., Boraud, T., Bergman, H., & Vaadia, E. (2005). Acquisition and generalization of visuomotor transformations by nonhuman primates. *Experimental Brain Research*, *161*, 209–219.

Rokni, U., Richardson, A. G., Bizzi, E., & Seung, H. S. (2007). Motor learning with unstable neural representations. *Neuron*, *54*, 653–666.

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

Santhanam, G., Ryu, S. I., Yu, B. M., Afshar, A., & Shenoy, K. V. (2006). A high-performance brain–computer interface. *Nature*, *442*, 195–198.

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

Shadmehr, R., & Mussa-Ivaldi, F. A. (1994). Adaptive representation of dynamics during learning of a motor task. *Journal of Neuroscience*, *14*, 3208–3224.

Srinivasan, L., Eden, U. T., Willsky, A. S., & Brown, E. N. (2006). A state-space analysis for reconstruction of goal-directed movements using neural signals. *Neural Computation*, *18*, 2465–2494.

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

Truccolo, W., Friehs, G. M., Donoghue, J. P., & Hochberg, L. R. (2008). Primary motor cortex tuning to intended movement kinematics in humans with tetraplegia. *Journal of Neuroscience*, *28*, 1163–1178.

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.

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

Wolpert, D. M., Ghahramani, Z., & Jordan, M. I. (1995). An internal model for sensorimotor integration. *Science*, *269*, 1880–1882.

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.

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.