J Neurophysiol 100: 3445–3457, 2008.
First published October 1, 2008; doi:10.1152/jn.90833.2008.

Toward Optimal Target Placement for Neural Prosthetic Devices

John P. Cunningham,[1] Byron M. Yu,[1,4,5] Vikash Gilja,[2] Stephen I. Ryu,[3] and Krishna V. Shenoy[1,4]
[1]*Departments of Electrical Engineering,* [2]*Computer Science, and* [3]*Neurosurgery, and* [4]*Neurosciences Program, Stanford University, Stanford, California; and* [5]*Gatsby Computational Neuroscience Unit University College London, London, United Kingdom*

Submitted 30 July 2008; accepted in final form 19 September 2008

Cunningham JP, Yu BM, Gilja V, Ryu SI, Shenoy KV. Toward optimal target placement for neural prosthetic devices. *J Neurophysiol* 100: 3445–3457, 2008. First published October 1, 2008; doi:10.1152/jn.90833.2008. Neural prosthetic systems have been designed to estimate continuous reach trajectories (motor prostheses) and to predict discrete reach targets (communication prostheses). In the latter case, reach targets are typically decoded from neural spiking activity during an instructed delay period before the reach begins. Such systems use targets placed in radially symmetric geometries independent of the tuning properties of the neurons available. Here we seek to automate the target placement process and increase decode accuracy in communication prostheses by selecting target locations based on the neural population at hand. Motor prostheses that incorporate intended target information could also benefit from this consideration. We present an optimal target placement algorithm that approximately maximizes decode accuracy with respect to target locations. In simulated neural spiking data fit from two monkeys, the optimal target placement algorithm yielded statistically significant improvements up to 8 and 9% for two and sixteen targets, respectively. For four and eight targets, gains were more modest, as the target layouts found by the algorithm closely resembled the canonical layouts. We trained a monkey in this paradigm and tested the algorithm with experimental neural data to confirm some of the results found in simulation. In all, the algorithm can serve not only to create new target layouts that outperform canonical layouts, but it can also confirm or help select among multiple canonical layouts. The optimal target placement algorithm developed here is the first algorithm of its kind, and it should both improve decode accuracy and help automate target placement for neural prostheses.

INTRODUCTION

Most neural prostheses (motor prostheses) decode neural activity into commands which guide a smoothly moving on-screen cursor or robotic arm (Carmena et al. 2003; Hochberg et al. 2006; Serruya et al. 2002; Srinivasan et al. 2007; Taylor et al. 2002; Velliste et al. 2008). Some neural prostheses (communication prostheses) estimate just the intended reach target. These communications prostheses could allow severely disabled patients to communicate messages or perform simple tasks by making a series of discrete choices such as selecting keys on a keyboard (Hatsopoulos et al. 2004; Musallam et al. 2004; Santhanam et al. 2006; Shenoy et al. 2003). Motor prostheses can also incorporate neural information about the reach target into their models (Kemere et al. 2004; Srinivasan et al. 2007; Yu et al. 2007). For communication prostheses or motor prostheses with discrete reach targets, it is critical to decode the intended target accurately. There is a great deal of interest in improving the decode performance of these prosthetic systems, as increased performance will enhance usability and therefore clinical viability. There are many factors which should be considered for improving prosthetic performance, including decoding algorithms (Brockwell et al. 2004; Brown et al. 1998; Georgopoulos et al. 1986; Wu et al. 2004, 2006), incorporating multiple signal modalities [e.g., electroencephalography (EEG), ECoG, LFP, and spiking activity], improving recording technology, and improving design of prosthetic end effectors, be that a robotic arm or computer cursor (Lebedev and Nicolelis 2006; Schwartz 2004). Here we address the problem of target placement in a communication prosthetic system (or a motor prosthesis using reach target information) that uses intracortical neural spiking activity.

In the behavioral paradigm employed in communication prosthesis studies, a monkey is trained to make center-out, delayed reaches to one of a discrete number of visual targets presented on a frontoparallel screen (Fig. 1). Using neural spiking activity recorded from dorsal premotor (PMd) cortex before the onset of movement, during the *instructed delay period,* maximum likelihood (ML) decoding algorithms can predict the intended reach target with high speed and accuracy (Santhanam et al. 2006). Because a neural prosthesis often consists of a keyboard or some other user interface, the key or target layout can be physically configured as the system designer sees fit. These prostheses (Hochberg et al. 2006; Kennedy and Bakay 1998; Kennedy et al. 2000; Musallam et al. 2004; Santhanam et al. 2006; Wolpaw and McFarland 2004) commonly place a number of targets (typically 2–16) evenly spaced around one or two rings, the radius of which is determined by the subject's maximum reach extent (Fig. 1) (see also Fig. 2*B* of Santhanam et al. 2006 and Fig. 5*C* of Hochberg et al. 2006). This canonical target layout, known as the *ring topology,* reflects the observation that neural activity is more strongly modulated by reach direction than reach extent (Churchland et al. 2006a; Fu et al. 1993; Messier and Kalaska 2000; Moran and Schwartz 1999b; Riehle and Requin 1989). Ad hoc attempts at improving decode performance by altering target configurations were made previously (see target configurations in Fig. 2*B* of Santhanam et al. 2006). However, if we understand the tuning properties of the particular neurons from which we are recording, we can quantitatively exploit this prior knowledge to place targets in a configuration that will yield lower decode error. Thus our goal here is both to increase decode accuracy by placing targets optimally, and to do so in an automated fashion.

Address for reprint requests and other correspondence: K. Shenoy, CISX 319, 330 Serra Mall, Stanford University, Stanford, CA 94305-4075 (E-mail: shenoy@stanford.edu).

The costs of publication of this article were defrayed in part by the payment of page charges. The article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. Section 1734 solely to indicate this fact.

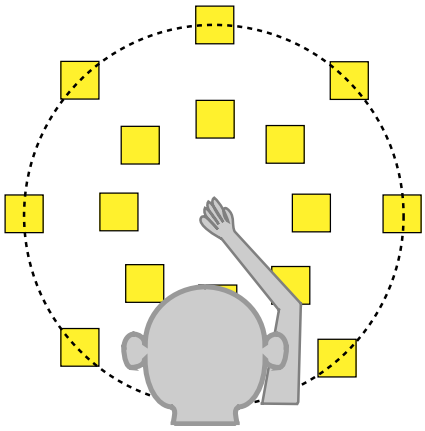

FIG. 1. A communication prosthesis paradigm with sixteen targets. Yellow squares show placement of targets in a canonical ring topology, evenly spaced around 2 rings of 8 targets each. Dotted line indicates the workspace bound.

Problem intuition

To motivate our approach, we provide here an illustration of the target placement problem. We first consider a hypothetical case where we record from only one neuron, and further we suppose that this neuron's firing rate is cosine tuned with a rightwards preferred direction (Georgopoulos et al. 1982). We show this case in Fig. 2*A,* where we represent this neuron with an arrow pointing right (the preferred direction). Let the length of the arrow correspond to the depth of tuning. As in Fig. 1, we have a dotted line corresponding to the workspace bound, which may be the monkey's reach extent or the extent of the visual field (targets must be placed within this workspace). Given this one neuron, where should we place two targets, T1 and T2, to maximize our decode accuracy? In Fig. 2*A,* we show two possible target configurations. In the *left subpanel,* we place the targets T1 and T2 at the far right and at the far left of the workspace (targets shown in black). In this configuration, a reach to target T1 will elicit maximal neural spiking activity, whereas a reach to T2 will elicit minimal activity, thereby maximizing our decode accuracy (the neural responses are most distinguishable). In contrast, we consider the *right subpanel* of Fig. 2*A,* where we have placed the targets (shown in gray) at the top and bottom of the workspace. While this configuration is geometrically similar to the *left subpanel* (the targets are maximally separated on the workspace, to exploit distance tuning), we can see that this cosine-tuned neuron will fire at the same rate (on average) to both targets, and the decoder will perform at chance accuracy. Thus we see that target placement is important, and it should also consider the neural population at hand. In other words, symmetric geometries alone are inadequate.

In Fig. 2*B,* we add one neuron with identical tuning strength but different preferred direction (shown in blue). In the *left subpanel,* with the neurons preferring left and right (blue and red arrows), we intuit again that the horizontal target layout (T1 and T2 in black) will have optimal decode accuracy, and the vertical target layout (T1 and T2 in gray) will perform at chance. However, if we instead record from the two neurons shown in the *right subpanel* of *B* (where the blue neuron has an upward preferred direction), the placement problem becomes more complicated. It seems both the black and gray pairs of targets will decode reasonably, but is there a better configuration? Perhaps, by a symmetry argument, the optimal layout is a pair of diagonally oriented targets [white targets marked (?)], but this intuition cannot be verified without simulation or experimental testing.

Let us complicate the situation further. In Fig. 2*C,* we add a third neuron (green arrow). In the *left subpanel,* we again see

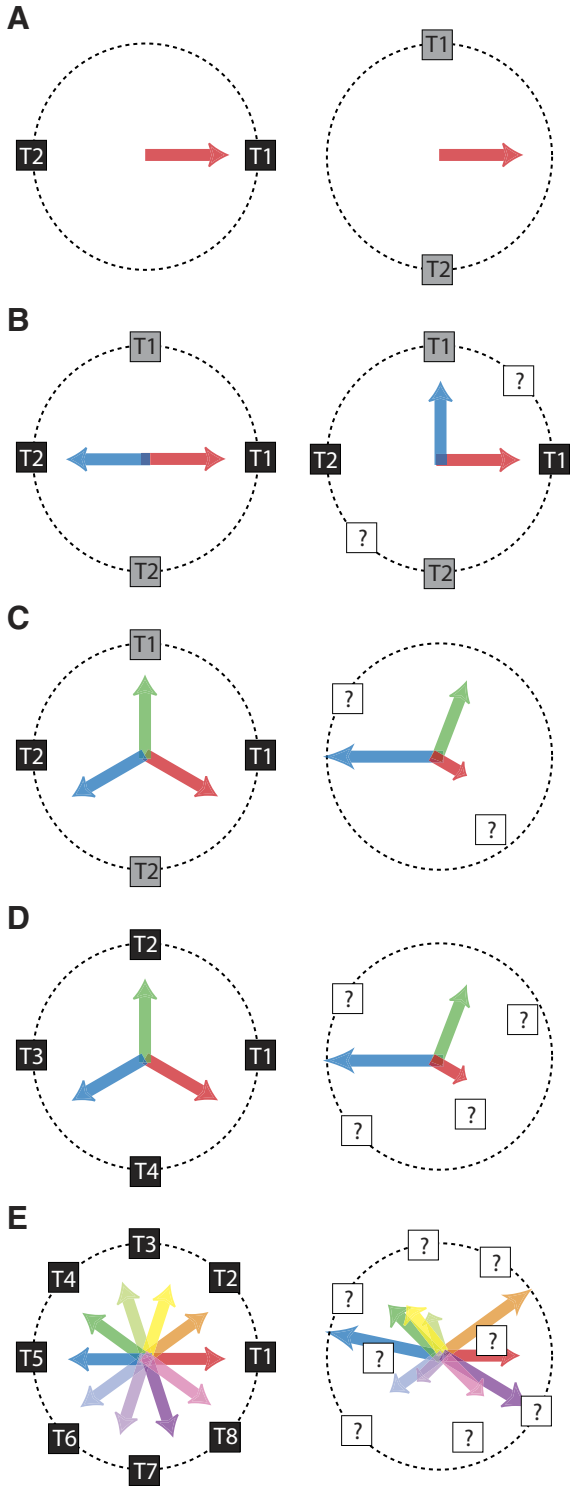

FIG. 2. Intuition of optimal target placement problem, where we consider progressively (*A–E*) more neurons (each neuron's tuning direction and strength being represented by 1 arrow) and targets (black, gray, or white squares). See INTRODUCTION (*Problem intuition*) for a full description.

that if these neurons had symmetric preferred directions of even tuning strength, either the pair of black targets or gray targets should decode well. In the *right subpanel,* however, we now change the various tuning strengths (as represented by the length of the arrows) and allow the preferred directions to be less regular. In this case, our intuition breaks down. It is unclear where to put a pair of targets to maximize decode accuracy. This loss of intuition worsens in *D,* where we now consider the same neurons, but instead consider the problem of placing four targets (T1–T4) not the two target cases in *A–C.* Again, in the *left subpanel,* ideal neurons should perhaps suggest a symmetrical layout as are often used in experiments. A more realistic neural population, shown at *right,* significantly increases the difficulty of the target placement problem.

Finally, in Fig. 2*E,* we show a case of placing eight targets when recording from 10 neurons. At *left,* an idealized, symmetric neural population might imply a symmetric target configuration. However, the more realistic neural population (*right*) makes impossible any reasonable guesses about target placements. In prosthetic systems with more targets and more neurons (as in Hatsopoulos et al. 2004; Musallam et al. 2004; Santhanam et al. 2006; Shenoy et al. 2003), the problem of target placement only gets more difficult.

One might consider a few strategies for optimal target placement. First, as is convention, one might lay out targets in symmetrical geometric patterns. Indeed, we see in Fig. 2*A* why this strategy can fail. Thus the characteristics of the neural population should be considered. One might then imagine a brute-force approach, choosing some two-dimensional grid (or three dimensional in the most general case) of possible target locations and then picking the best choices among all target configurations on that grid. Each possible configuration has a decode accuracy that must be found via simulating many reach trials, which takes a reasonable amount of computational effort (depending on the number of targets and the number of simulated trials). Even with a coarse grid of 16 or 32 possible target locations, choosing a layout of 8 targets and simulating decode accuracy would be computationally intractable: there are over 10^5 ($_{16}C_8$, the number of combinations of 8 distinct items chosen from 16 possible items, i.e., 16!/8!8!) and 10^8 ($_{32}C_8$ or 32!/24!8!) choices for these layouts with grids of size 16 and 32, respectively. These difficulties with initial approaches led us to consider the problem from a communications theory perspective.

To our knowledge, this problem has not yet been investigated. We present the optimal target placement algorithm (OTP), which uses Kullback-Leibler divergence to provide a constellation of optimal target placements. We describe the method and then compare the decode performance of the optimal constellation with canonical ring topologies, using both simulated and experimental neural data.

An introduction to this algorithm has been published in preliminary form (Cunningham et al. 2006).

METHODS

Overview

We want to construct an algorithm that places reach targets such that they are maximally distinguishable (to achieve optimal performance) in terms of the neural signals we record. To do so, we must first define a model that relates the reach target position to neural spiking during the delay period. We then consider a rule for decoding a particular target, given an observation of spike data. These steps are detailed in *Spike count model and decoding* in the following text. This rule implies a decode error (our measure of prosthetic performance) that is a function of the target locations. Ideally we could then minimize decode error by moving reach targets appropriately. This general problem is intractable. However, by making standard, reasonable approximations to put this error function (a function of the target locations) into a solvable form, we can optimize the function to produce a set of target placements that approximately minimizes decode error. These steps are detailed in *Optimal target placement algorithm* in the following text. Finally, we test this method with data from two monkeys trained to perform reaches to canonically placed targets. For both monkeys, we fit a neural population (using real reaches) and evaluate decode performance on simulated neural data generated from canonically placed and optimally placed targets (hereafter, simulated data). The second monkey also performed real reaches to both canonically and optimally placed targets, and we compare decode accuracy (hereafter, experimental data). These steps are detailed in *Reach task and neural recordings* and *Evaluating decode performance* in the following text.

Spike count model and decoding

We must first consider how target position is reflected in neural spiking. As described in the preceding text, we present a reach target on the screen during an instructed delay period. We call this time period Δ (e.g., $\Delta = 200$ ms, the window beginning 150 ms after target presentation and before the subject is given a movement cue) (as used in Santhanam et al. 2006). We collect spike counts from K neural units, and the frequency of each unit's spiking (that is, the number of spikes) is indicative of the intended reach target to an extent that allows target location to be predicted (Hatsopoulos et al. 2004; Musallam et al. 2004; Santhanam et al. 2006; Shenoy et al. 2003). We choose a simple firing rate model (as in Smith and Brown 2003; Yu et al. 2007) and a simple spiking model (as in (Yu et al. 2007; Zhang et al. 1998). We will later discuss more advanced models, but even these basic models help to simply illustrate the conceptual advance that this method offers.

Let us consider M reach targets placed on a screen as in Fig. 1 (where $M = 16$). We define each target by its Cartesian position on the screen $x_m \in \Re^2$ (for all M targets $m \in \{1, \ldots, M\}$). We define the center of the screen as the origin, but the optimal target placement algorithm will be invariant to that choice. We call the collection of all M targets a *constellation* of targets $\chi \in \Re^{2M}$, that is $\chi = [x_1^T, \ldots, x_M^T]^T$.

Having defined the target constellation, we must define a model that maps target position to neural spiking. Let us assume we record from K neural units. Then we map position x_m to a neural firing rate for the kth neural unit (as in Smith and Brown 2003; Yu et al. 2007) using

$$f_k(x_m) = e^{c_k^T x_m + d_k} \quad (1)$$

where d_k specifies a baseline firing rate, and c_k specifies both the preferred direction (Georgopoulos et al. 1982) and the depth of tuning modulation for unit k. The linear mapping $c_k^T x_m + d_k$ implies a cosine tuning model (Georgopoulos et al. 1982; Moran and Schwartz 1999a,b). We group these parameters c_k, d_k (over all K neural units) into $C \in \Re^{2\times K}$ (the matrix with columns c_k) and $d \in \Re^{K\times 1}$ (the vector of elements d_k). Thus $f_k(x_m)$ calculates the delay period firing rate underlying the spiking of unit k, when the target m is presented at position x_m.

To relate this firing rate to spike counts, we use a simple Poisson count model (Yu et al. 2007; Zhang et al. 1998). Specifically we assume the delay period spiking activity for one neural unit, when conditioned on the target m (at position x_m), is independent of other neural units and of its own spiking history. The probability of all observed spike counts y (the vector $y \in \Re^{K\times 1}$ is a vector of nonnegative integer spike counts), during the delay period Δ, is then

$$p(y \mid m) = \prod_{k=1}^{K} \text{Poisson}(y_k; f_k(x_m)\Delta) = \prod_{k=1}^{K} \frac{(f_k(x_m)\Delta)^{y_k}}{y_k!} e^{-(f_k(x_m)\Delta)} \quad (2)$$

According to this model, on a given trial, the presented target m^* at position x_{m^*} is chosen by the experimenter, where $m^* \in \{1, \ldots, M\}$. The observed spike counts y, conditioned on m^*, are assumed to be distributed according to *Eq. 2*. We record y and want to decode the identity of the presented target m^* from among the M possible choices. We note that we only consider spike counts from the delay period Δ, during which we assume the reach target is fixed and the firing rate (*Eq. 1*) is constant. Thus all decodes are made from that time period alone (and accordingly, error rates and all other values are calculated during that fixed window). To decode a reach target, we use maximum a posteriori (MAP) decoding (Zhang et al. 1998)

$$\hat{m} = \underset{m}{\text{argmax}}\, p(m \mid y) \quad (3)$$

$$= \underset{m}{\text{argmax}} \frac{p(y \mid m)p(m)}{p(y)} \quad (4)$$

$$= \underset{m}{\text{argmax}}\, p(y \mid m) \quad (5)$$

where $\hat{m}$ is the index of the estimated reach target (at position $x_{\hat{m}}$). *Equation 3* states that we choose the decoded target as the most likely target, given the neural data. *Eq. 4* is obtained using Bayes' rule; *Eq. 5* is a result of all reach directions being equally likely (since in our experiments all targets are presented an equal number of times)[1] and $p(y)$ not being dependent on m. Thus the decode rule (*Eq. 3*) reduces to a maximum likelihood (ML) estimator (*Eq. 5*) (Papoulis and Pillai 2002). If the assumptions of the model are satisfied (i.e., Poisson spiking statistics, cosine tuning, etc.), this decode rule will minimize the total error probability (at a given target constellation χ) (Cover and Thomas 1991)

$$P_{\text{error}} = \sum_{m=1}^{M} P(\{\hat{m} \neq m\} \mid \{m^* = m\}) \quad (6)$$

In words, *Eq. 6* is the probability that, when any target m is presented (the presented target $m^* = m$), some other target is erroneously decoded (the decoded target $\hat{m} \neq m$). Thus the goal of our optimal target placement algorithm is to choose the constellation χ that will minimize the total probability of decode error of *Eq. 6*.

Optimal target placement algorithm

This general problem of minimizing total error probability of *Eq. 6* (over the constellation χ), well known in communications literature (see e.g., Proakis and Salehi 1994), is often analytically intractable (i.e., there is no closed form solution, which will be required so we can calculate how changes in target position effect decode error). Indeed, minimizing *Eq. 6*[2] is similarly difficult in our case. As a result, it is common to instead minimize the worst pairwise error probability (we denote pairwise probabilities P_{pair}) (Gockenbach and Kearsley 1999). Pairwise error probability is simpler to calculate than total error probability because pairwise error does not consider the influence of other targets. For example, a pair of targets might have a certain error rate in isolation, but that may change with the presence of a third target, because the correct target can now be mistaken for this third target as well. Minimizing the worst pairwise error probability is equivalent to minimizing an upper bound to *Eq. 6*. That is, instead of considering all the targets jointly, we consider all pairs of targets. We then select the least distinguishable ("worst") pair of targets (that is, the pair with the highest error rate when trying to decode which of these two targets is the intended reach goal), and we will try to minimize this error rate (make these two targets more distinguishable). Doing this procedure jointly across all pairs of targets should yield a lower global decode error (*Eq. 6*). Mathematically, we define the solution to this problem χ_{otp} (the optimal constellation) as

$$\chi_{\text{otp}} = \underset{\chi}{\text{argmin}}\left(\max_{m' \neq m} P_{\text{pair}}(\{\hat{m} = m'\} \mid \{m^* = m\})\right) \quad (7)$$

The inner expression $P_{\text{pair}}(\cdot)$ is the pairwise probability of error between two targets m (the correct, presented target m^*) and m' (the erroneously decoded target $\hat{m}$). For M targets, there are $M(M-1)$ such probabilities of error (all target pairs). The maximum of these probabilities of error is the worst pair in that it has highest decode error. Finally, the outermost expression [argmin$(\cdot)$] finds the constellation χ (the collection of target positions), which minimizes this worst pairwise error. Thus *Eq. 7* provides a constellation of targets that minimizes the worst pairwise error over all targets.

To calculate the probability of decode error between any pair of targets, we must consider the spiking noise introduced by the Poisson output distributions (*Eq. 2*). Owing to the noisy Poisson model, particular spike counts will erroneously decode a target m' when in fact the presented target was m. There is no closed-form expression for the probability of decode error between two Poisson noise distributions (Verdu 1986). Kullback-Leibler (KL) divergence is often used as a close proxy to pairwise error probability (Gockenbach and Kearsley 1999; Johnson and Orsak 1993; Johnson et al. 2001). KL divergence measures how different two probability distributions are. Pairwise error probability also measures how different two distributions are in that it quantifies how often a draw from one distribution will be incorrectly classified as having been drawn from another distribution. The use of KL as a proxy to error probability is intuitively sound, and our simulations have shown that increasing KL divergence (making the two distributions more different) corresponds well to decreasing error probability. KL is commonly used when error probability is not closed form (Gockenbach and Kearsley 1999; Johnson and Orsak 1993; Johnson et al. 2001) with the understanding that making distributions more distinguishable (increasing the KL divergence) will generally reduce probability of error also. The relationship between KL and error probability can be motivated mathematically by returning to the two-target case of the ML decode rule (*Eq. 5*) and writing it as

$$\hat{m} = \begin{cases} m & \text{if } \dfrac{p(y \mid m)}{p(y \mid m')} \geq 1 \\ m' & \text{otherwise,} \end{cases} \quad (8)$$

(i.e., the decoder predicts m if $p(y|m)$ is larger than $p(y|m')$, and m' otherwise). Assuming a trial has reach target m presented, we want to maximize the likelihood ratio in *Eq. 8* over all possible instances of y. Doing so will provide the maximum distinguishability between the distributions [$p(y|m)$ and $p(y|m')$] and will minimize the chance that the likelihood ratio will be <1 (which implies an error). We can equivalently maximize the logarithm of this likelihood ratio, and, taking the expectation to consider all possible y, we have the KL divergence

$$\text{KL}(x_m \,||\, x_{m'}) = E_{y|m}\left[\log \frac{p(y \mid m)}{p(y \mid m')}\right] \quad (9)$$

[1] If the targets were not presented with equal frequency (for example, in a keyboard application, one might know that certain targets/keys will be used more often than others), then *Eq. 5* would still have $p(m)$ (MAP estimation). The OTP algorithm can be extended to incorporate this change; see *Future work* in DISCUSSION.

[2] This upper bound can be seen by expanding each term in the sum of *Eq. 6* using the union of events bound (Boole's inequality), such that $\sum_{m=1}^{M} P[(\hat{m} \neq m) | (m^* = m)] \leq \sum_{m=1}^{M} \sum_{m' \neq m} P_{\text{pair}}[(\hat{m} = m') | (m^* = m)]$. This sum of pairwise errors is upper bounded by $M(M-1)$ times the worst pair, and thus minimizing the worst pair is equivalent to minimizing an upper bound on total error probability.

where the expectation is taken with respect to y given m. We write KL as a function of the target positions x_m and $x_{m'}$ to emphasize that it calculates our proxy to error probability *in terms of the target positions*. Thus KL in *Eq. 9* is a measure of distinguishability between targets m and m'; to minimize the probability of decode error, we want to maximize *Eq. 9* by changing target locations x_m and $x_{m'}$. Under the Poisson output distribution, we introduced in *Eq. 2*, KL divergence can be calculated exactly (substitute *Eq. 2* into *Eq. 9*; see Appendix A for details)

$$\mathrm{KL}(x_m \,\|\, x_{m'}) = \Delta \sum_{k=1}^{K} \left(f_k(x_{m'}) - f_k(x_m) + f_k(x_m) \log \frac{f_k(x_m)}{f_k(x_{m'})} \right). \quad (10)$$

Note that this form is not constrained by the form of the firing rate $f_k(x_m)$ in *Eq. 1*, allowing OTP to easily generalize for other, more complex firing rate models (on the other hand, changing the spiking model—*Eq. 2*—will change the form of the KL; see *Future work* in DISCUSSION).

In summary, we have replaced the analytically intractable probability of error between two Poisson distributions with the tractable form of *Eq. 10*. We have done so with the understanding that finding a pair of target positions $(x_m, x_{m'})$ that maximizes the KL divergence from x_m to $x_{m'}$ is nearly equivalent to finding that which minimizes the probability of decoding m' when m was presented. Mathematically, we write

$$\underset{x_m, x_{m'}}{\operatorname{argmax}}\, \mathrm{KL}(x_m \,\|\, x_{m'}) \approx \underset{x_m, x_{m'}}{\operatorname{argmin}}\, P_{\mathrm{pair}}(\{\hat{m} = m'\} \,|\, \{m^* = m\}) \quad (11)$$

For the Poisson distributions used here, as we have noted, our simulations show that the relationship between error probability and KL divergence is nearly monotonic. Thus we believe maximizing KL divergence is a valuable proxy to minimizing probability of error in this problem. One might also consider using the Chernoff bound (Cover and Thomas 1991), which proves an upper bound on error probability with respect to KL divergence in hypothesis testing. However, this bound has been found to be loose (Johnson et al. 2001). Although not a provable bound (upper or lower) on error probability, KL divergence does provide a very close proxy; further supporting arguments can be found in (Johnson and Orsak 1993; Johnson et al. 2001).

Having made this approximation, we can return to our problem of interest, namely finding the optimal target placement χ_{otp} as in *Eq. 7*. Using KL divergence, *Eq. 7* becomes

$$\chi_{\mathrm{otp}} = \underset{\chi}{\operatorname{argmax}} \left(\min_{m \neq m'} \mathrm{KL}(x_m \,\|\, x_{m'}) \right) \quad (12)$$

Note that the two targets with the smallest KL divergence (the inner expression in *Eq. 12*) are the least distinguishable and thus should have the highest probability of error (the worst pair as in *Eq. 7*). Accordingly, improving the worst pair in *Eq. 7* (minimizing the maximum error probability) is the same as improving the worst pair in *Eq. 12* (maximizing the minimum KL divergence). An algorithm solving this problem will push the target positions x_m as far apart as possible from each other in terms of KL divergence. We impose a workspace limitation such as how far a subject's arm can reach, the extent of the subject's visual field, or the bounds imposed by the prosthesis (such as a computer screen). We capture this limitation with a constraint γ on the Euclidean distance of x_m from the center of the workspace screen (other constraints, such as a rectalinear workspace, could be readily included as well). With this constraint, our optimal target placement χ_{otp} is the solution to

$$\underset{\chi}{\text{maximize}} \left(\min_{m \neq m'} \mathrm{KL}(x_m \,\|\, x_{m'}) \right)$$

$$\text{subject to } \| x_m \| \leq \gamma \;\; \forall\, m = 1 \ldots M \quad (13)$$

We call the algorithm that solves *Eq. 13* the optimal target placement algorithm. We applied sequential quadratic programming (SQP) (Boggs and Tolle 1996; Gockenbach and Kearsley 1999) to solve *Eq. 13*. It is important to note here that SQP is an established technology for optimizing nonlinear, constrained objectives such as *Eq. 13*. For example, the MATLAB (The MathWorks, Natick, MA) function fmincon (nonlinear, constrained optimization solver) uses SQP. SQP finds an optimum to this problem (*Eq. 13* is not convex in χ), and this optimum depends on the choice of seed constellation χ_0. To find the global optimum, we solved the SQP multiple times (8–32, depending on the number of targets in the constellation) starting at randomly chosen χ_0. After these iterations, a "best" optima (best in terms of having the largest objective, i.e., the minimum worst pair KL divergence, as in *Eq. 12*) typically appeared several times, giving confidence that we had indeed found the global optimum. This solution was designated the optimal constellation χ_{otp}. We include notes on our use of SQP in APPENDIX B.

Reach task and neural recordings

Animal protocols were approved by the Stanford University Institutional Animal Care and Use Committee. We trained two adult male monkeys (*Macaca mulatta, monkeys H* and *L*) to perform delayed center- out reaches for juice rewards. As illustrated in Fig. 3, visual targets were back-projected onto a frontoparallel screen 30 cm in front of the monkey. The monkey touched a central target and fixated his eyes on a crosshair adjacent to the central target. After a center hold period of 300–500 ms for *monkey L* and 400–600 ms for *monkey H,* a pseudorandomly chosen target was presented at one of the target locations. For the canonical reach data sets, the 16 targets were placed

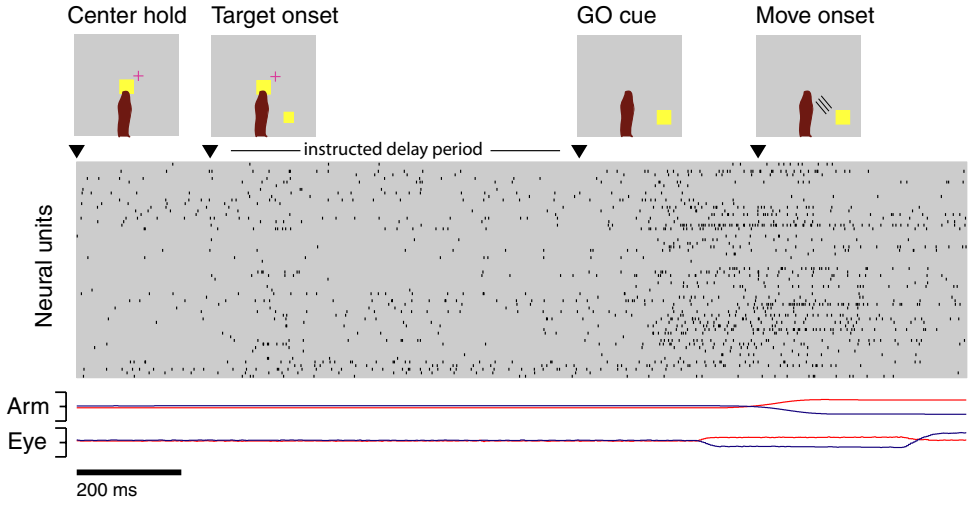

FIG. 3. Task timeline (*top*), simultaneously recorded spike trains (*middle*), and arm and eye position traces (*bottom*) are shown for a single trial. Red and blue lines correspond to horizontal and vertical position, respectively. The range of movement for the arm and eye position (on the screen) is ±15 cm from the center target. Neural unit activity and physical behavior were taken from trial *H20041106.1*.

in two rings of 8, as shown in Fig. 1, of radius 7 and 12 cm for *monkey H* (4 and 8 cm for *monkey L*). After a pseudorandomly chosen instructed delay period (*monkey H*: uniformly distributed between 200 and 500 ms; *monkey L*: exponentially distributed with a mean of 750, 850, or 950 ms, shifted to be no less than 50, 100, or 150 ms), the "go" cue (signaled by both the enlargement of the target and the disappearance of the central target) was given, and the monkey reached to the target. After a hold time of 200 or 300 ms at the reach target (depending on the experimental day), the monkey received a liquid reward. The next trial started 100–400 ms later (depending on the experimental day). Eye fixation at the crosshair was enforced throughout the delay period. Reaction times (defined as the time between the go cue and movement onset) were enforced to be >80 or 100 ms and <400 or 425 ms (depending on the experimental day).

During experiments, the monkey sat in a custom chair (Crist Instruments, Hagerstown, MD) with the head braced. The presentation of the visual targets was controlled using the Tempo software package (Reflective Computing, St. Louis, MO). A custom photo-detector recorded the timing of the video frames with 1-ms resolution. The position of the hand was measured in three dimensions using the Polaris optical tracking system (Northern Digital, Waterloo, Ontario, Canada; 60 Hz, 0.3- mm accuracy), whereby a passive marker taped to the monkey's fingertip reflected infrared light back to the position sensor. Eye position was tracked using an overhead infrared camera (Iscan, Burlington, MA; 240 Hz, estimated accuracy of 1°).

A 96-channel silicon electrode array (Cyberkinetics, Foxborough, MA) was implanted straddling PMd and motor (M1) cortex (left hemisphere for both *monkeys H* and *L*), as estimated visually from local landmarks, contralateral to the reaching arm. Surgical procedures have been described previously (Churchland et al. 2006b; Hatsopoulos et al. 2004; Santhanam et al. 2006). Spike sorting was performed off-line using techniques described in detail elsewhere (Sahani 1999; Santhanam et al. 2004; Zumsteg et al. 2005). Briefly, neural signals were monitored on each channel during a 2-min period at the start of each recording session while the monkey performed the behavioral task. Data were high-pass filtered, and a threshold level of three times the RMS voltage was established for each channel. The portions of the signals that did not exceed threshold were used to characterize the noise on each channel. During experiments, snippets of the voltage waveform containing threshold crossings (0.3-ms precrossing to 1.3-ms postcrossing) were saved with 30-kHz sampling. After each experiment, the snippets were clustered as follows. First, they were noise-whitened using the noise estimate made at the start of the experiment. Second, the snippets were trough-aligned and projected into a four-dimensional space using a modified principal components analysis. Next, unsupervised techniques determined the optimal number and locations of the clusters in the principal components space. Events assigned to each cluster are considered spikes for a given neural unit.

Figure 3 shows the delayed reach task timeline along with neural and behavioral data for a single trial with a lower-right reach target. We refer to the time between reach target onset and the go cue as the delay period. It is the neural activity during this delay period that will be used to predict the reach target.

The monkeys were trained over several months, and multiple data sets of the same behavioral task were collected. Each data set was collected in one day's recording session. For monkey H, all reaches were made to canonically placed targets. For monkey L, each data set was split into two segments, the first comprising reaches to a canonical target topology, and the second to an OTP constellation. After collecting 700–2,000 trials of the canonical topology, the task was stopped. Units isolated by the spike sorting method were fit to the cosine tuning model of *Eq. 1*. We counted spikes for the 200-ms period that started 150 ms after target onset (a 200-ms integration window, i.e., $T_{skip} = 150$ ms and $T_{int} = 200$ ms, in the terminology of Santhanam et al. 2006). We fit C, d from the neural data by maximizing the data likelihood (*Eq. 2*) taken across all trials. This fitting problem is convex (in C, d) and can be readily solved using Newton's Method (Boyd and Vandenberghe 2004) (glmfit in MATLAB will also readily solve this problem). This population of neural fits was then given to the OTP algorithm, which then generated an optimal target topology. This entire OTP process generally took <10 min on 2006-era workstations (Linux Fedora Core 4 with 64 bit, 2.2- to 2.4-GHz AMD processors and 2–4 GB of RAM) running MATLAB (R14). For *monkey H*, this neural population fitting was done off-line to provide neural tuning data for OTP simulation. For *monkey L*, the task was begun again with reaches to the newly configured OTP target topology, typically for 700–1,500 more trials. For both the canonical and OTP trials, we only analyzed successful trials (where the monkey obeyed the hold times, reached to the target with a proper reaction time, etc.) that had delay periods long enough to allow T_{skip} and T_{int} as just described. This screening typically left 300–800 valid OTP and 300–800 valid canonical trials for analysis (we used equal numbers of OTP and canonical trials so performance comparisons could be meaningfully made). This segmentation allows us to analyze and compare decode performance from the canonical and optimal target topologies.

Evaluating decode performance in experimental data

Collecting experimental data allows us to verify the performance improvements we see in simulation. In *Spike count model and decoding* (preceding text), we introduced a maximum likelihood decoder that (when the model assumptions are satisfied) minimizes the probability of decode error (*Eq. 6*). For simulated trials, we know the neural parameters C and d, and thus we calculate the firing rate $f_k(x_m)$ exactly for any target position x_m (in simulation, the data fit the model of *Eqs. 1* and *2* exactly). In this case, we use the ML decoding rule of *Eq. 5* directly. However, in experimental reach trials, we do not have access to the neural parameters C and d, and thus we do not have $f_k(x_m)$. Instead, for each target x_m, we must fit an estimate $\hat{f}_k(x_m)$. Because each unit y_k is modeled as Poisson (conditioned on the target x_m), y_k has expected value of $f_k(x_m)\Delta$. With a set of training trials to a particular target, our estimate $\hat{f}_k(x_m)$ is the empirical mean (normalized by) of those training trials (a ML estimator of $f_k(x_m)$ (Papoulis and Pillai 2002)).

In an experimental data set, we have J blocks of trials, where a block consists of one trial to each of the M reach targets. We define the neural data collected during the delay period of each trial as $y^{(j,m)}$ for a block $j \in \{1, \ldots, J\}$ and a target $m \in \{1, \ldots, M\}$. To decode a single trial, we use J-fold cross validation (Duda et al. 2001). For a given block j of reach trials, we exclude the block as a test data set and use all other $(J - 1)$ blocks as the training set to train the decoder (i.e., estimate $\hat{f}_k(x_m)$ for all $k \in \{1, \ldots, K\}$ and $m \in \{1, \ldots, M\}$). With these parameter estimates, we can again use the ML decoder of *Eq. 5* as in Santhanam et al. (2006), Shenoy et al. (2003), Yu et al. (2004), and Musallam et al. (2004). This J-fold cross validation is repeated across all blocks of trials and produces a total decode performance for a given data set.

We note that $\hat{f}_k(x_m)$ does not in general equal $f_k(x_m)$ because the empirical mean over the training trials we collected will not be exact (even if the firing rate model holds). This factor may degrade decoder performance in experimental data, but such performance reductions should be seen equally for OTP and canonical topologies. In this study, we are only interested in how the different target constellations compare in decode accuracy, not their absolute values. We confirmed in simulation that using the empirical mean resulted in similar performance reductions across both topologies, thereby suggesting that OTP is no more susceptible to this source of error than is the canonical topology. The accuracy of the target decoder also varies with the duration and placement of the time window in which spikes are counted and the spike count model $P(y|x_m)$ that is used (Hatsopoulos et al. 2004; Santhanam et al. 2006). Optimizing these aspects of the target decoder (which we again expect to affect performance equally

across topologies) is beyond the scope of this work and is treated in detail in Santhanam et al. (2006).

RESULTS

As we saw in Fig. 2, for small numbers of targets and neural units, we can make a reasonable prediction about where the optimal targets should be placed, even without the use of an optimization algorithm. In the simplest case, we seek to place two targets optimally with only one neural unit (Fig. 2*A*). Given the preferred direction of the unit c_1, the targets should be placed as far apart as possible (on the circular workspace bound, as firing rate is typically modulated by reach target distance) along the axis defined by c_1. In this configuration, the presentation of one target elicits maximal firing, while the other target gives minimal firing. Indeed, our SQP approach to optimal target placement yields this result. Extending beyond this trivial case, the utility of OTP becomes apparent when looking at larger neural populations and larger numbers of targets.

With a population of neural units that are fairly uniform in their preferred directions and tuning strength, we imagine that the placement of four or eight targets will reduce effectively to a geometric problem, and placing the targets evenly around a ring will produce a near optimal result. We will validate this intuition in the following text. When the number of targets grows larger, intuition breaks down: for example, with 16 targets, should they be placed evenly around the circular workspace bound? Should they be placed in two rings; if so, how many targets in each ring? OTP gives answers to these questions. Two examples are shown in Fig. 4, where OTP returns a constellation (blue circles) with 11 targets spaced roughly evenly around the circular workspace bound and with 5 targets placed elsewhere in the workspace for *monkey H* (in *A*). For *monkey L* (*B*), OTP finds a constellation with 10 targets on the workspace bound and 6 placed on the workspace interior. Despite the complexity of this problem, there is some intuition to be gleaned from the constellations discovered by the OTP algorithm; see DISCUSSION (*Intuition gained from OTP*).

Simulated data results

Having shown a few examples of optimal target placements, we now turn to systematic performance comparisons of OTP versus canonical ring topologies. We randomly drew a set of K units from one of two collected data sets: one from *monkey H* (*H20041119*) and one from *monkey L* (*L20061030*). We ran OTP to find the constellation χ_{otp}, and then we generated simulated spike counts for 1,000 trials to each of M optimally placed targets according to *Eq. 2 (i.e.,* $M \times 1{,}000$ total trials). We then computed decode accuracy using the method described in the preceding text in *Evaluating decode accuracy*. We also simulated 1,000 trials to each target of the canonical ring topology (again, $M \times 1000$ total trials).[3] This whole procedure was repeated 100 times (10 times for the 16 target case, due to computational limitations) for each K.

These results are shown for *monkey H* in Fig. 5 for 2, 4, 8, and 16 targets and similarly for *monkey L* in Fig. 6. In Fig. 5*A,* for two targets, OTP provides up to 8% improvement in decode accuracy (from 71 to 79% with $K = 2$, for example). OTP provides similar results for *monkey L,* raising performance 6% (from 71 to 77% with $K = 4$, for example). As K grows and decode accuracy saturates to the performance ceiling of 100%, we expect the canonical topology to approach the performance

[3] To get a true average performance for the canonical topology, we rotated the ring topology across trials to prevent any possible bias in the results. For example, if we chose a vertical canonical layout in the two target case (as in Fig. 2*A, right*) and these particular neural populations had more tuning strength in the horizontal axis, then canonical layouts would be artificially punished in decode performance (so too, canonical performance could be artificially inflated if we instead chose the layout of Fig. 2*A, left*). Rotating the canonical targets ensures a fair comparison between decode performances.

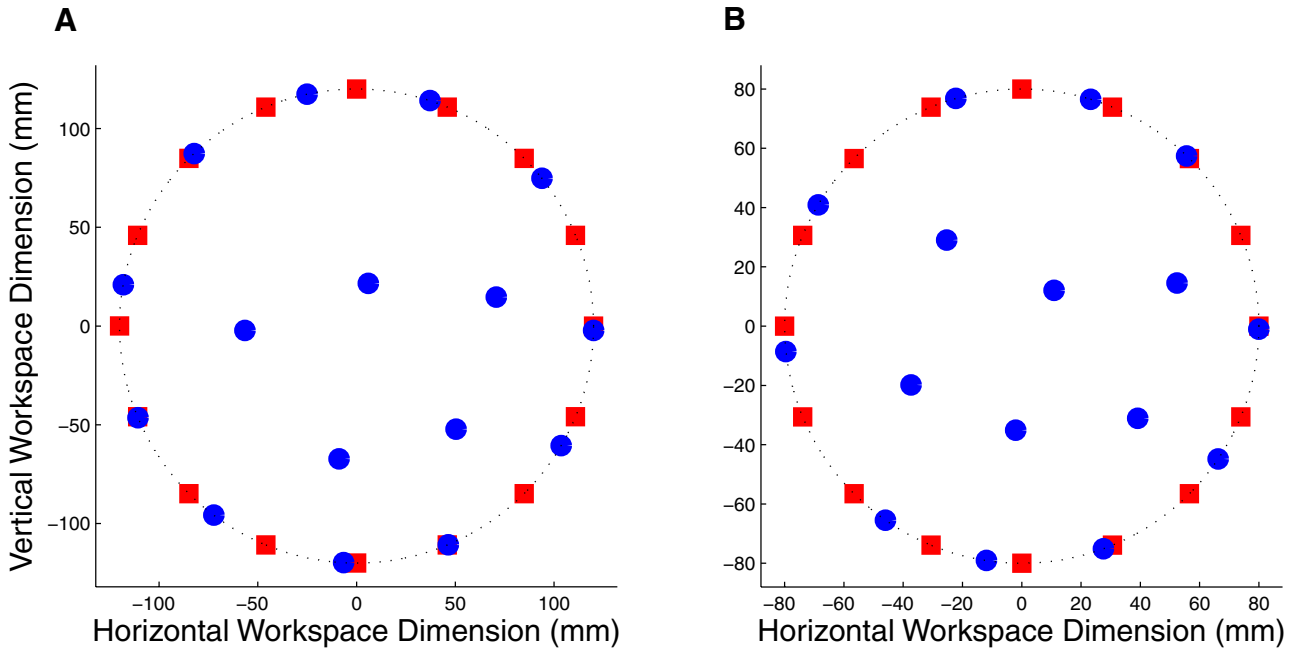

FIG. 4. Sixteen target placement examples from *data set H20041119* (*A*) and *data set L20061030* (*B*). Blue circles: optimal target placement algorithm (OTP) solution; red squares: a canonical ring topology. Workspace bound shown as a dotted line ($\gamma = 120$ mm in *A,* 80 mm in *B*).

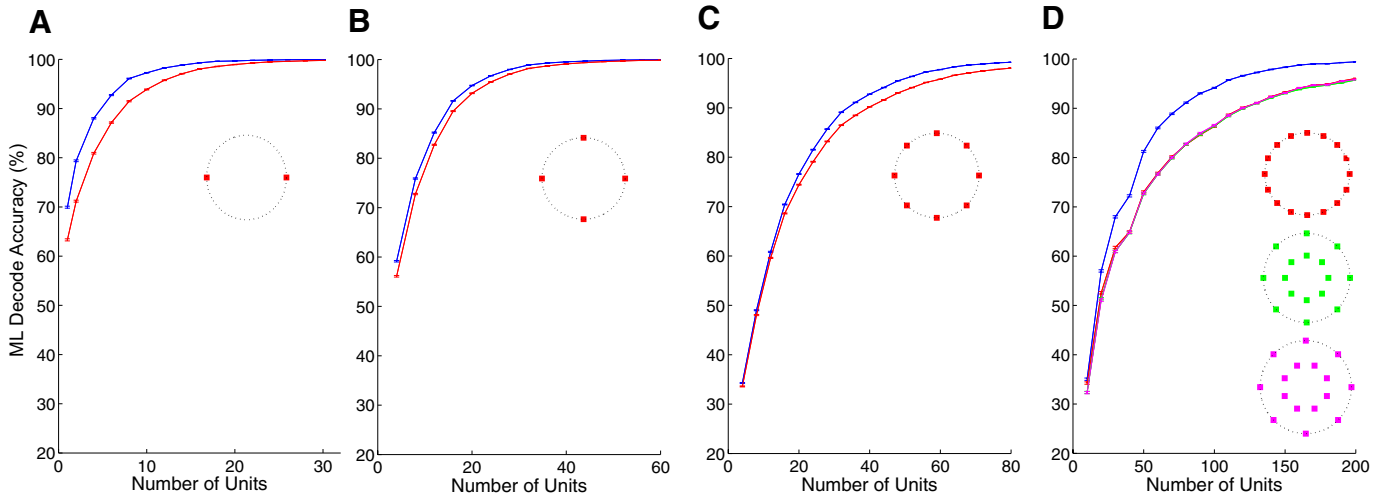

FIG. 5. Comparison of performance in simulated data: optimal target placements vs. ring topologies. *Monkey H* (*H20041119*). *A*: 2 targets. *B*: 4 targets. *C*: 8 targets. *D*: 16 targets. Blue and red lines show performance under OTP and a single ring topology, respectively. In *D,* green and magenta lines show performance under the aligned and staggered double ring topologies, respectively (red, green, and magenta curves are highly overlapped). Error bars (vanishingly small, due to hundreds of thousands of simulation trials) based on a binomial distribution with 95% confidence level (see Zar 1999). *Insets*: different ring topologies tested.

of OTP, and indeed we see this effect. In both the four- and eight-target cases, there is less improvement above the ring topology for both *monkeys H* and *L* with performance improvements ranging from 0 to 3% (*monkey H*) and 0 to 1% (*monkey L*). This is not surprising: the OTP layouts closely resemble canonical ring topologies. For example, a four- or eight-target OTP layout is often just a rotated version of a canonical layout, which does not look much different from a canonical layout (i.e., a rotated version of the black targets Fig. 2, *D* and *E,* appears quite similar to an unrotated constellation). Contrast this to a two target case, where a rotation of a pair of targets can look significantly different (i.e., in Fig. 2*A,* the black and gray pairs of targets are quite different). Thus we do not expect a substantial performance difference in the four- and eight-target cases.

At larger target constellations, we can again see substantial improvements offered by OTP. Figure 5*D* illustrates the performance of the optimal configuration in the 16-target case with *monkey H* and similarly in Fig. 6*D* for *monkey L.* We compare OTP to three canonical ring topologies: 16 targets evenly spaced around the workspace bound, two radially aligned rings of 8 targets each, and two radially staggered rings of 8 targets each. In our experience, most OTP constellations seen in the 16-target case for both monkeys (for different values of *K* and different sets of units drawn at random) place 4–6 interior targets and 10–12 on the workspace bound; examples are shown in Fig. 4. See DISCUSSION (*Intuition gained from OTP*) for comments about why these constellations are sensible results of the algorithm. Over a range from 50 to 100 units, OTP target topologies yield 8–9% average improvement

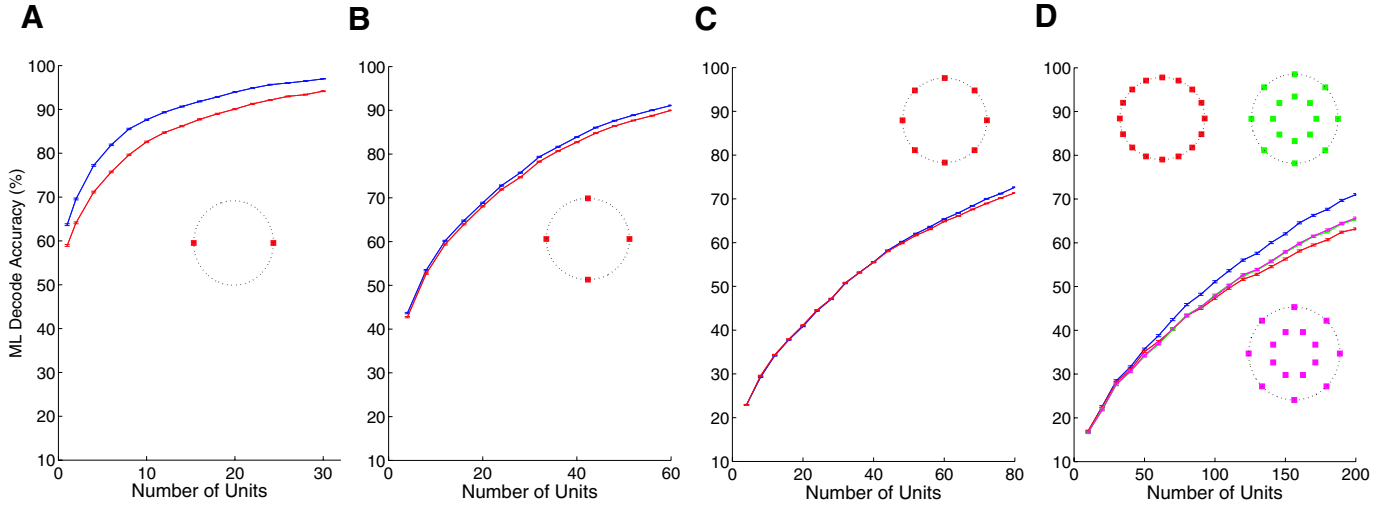

FIG. 6. Comparison of performance in simulated data: optimal target placements vs. ring topologies. *Monkey L* (*L20061030*). *A*: 2 targets. *B*: 4 targets. *C*: 8 targets. *D*: 16 targets. Blue and red lines show performance under OTP and a single ring topology, respectively. In *D,* green and magenta lines show performance under the aligned and staggered double ring topologies, respectively (red, green, and magenta curves are highly overlapped). Error bars (vanishingly small, due to hundreds of thousands of simulation trials) based on a binomial distribution with 95% confidence level (see Zar 1999). *Insets*: different ring topologies tested.

over the canonical ring topologies in *monkey H*.[4] For *monkey L*, more units are required to see substantial performance gains, achieving 4–5% improvement for 140–200 neural units.[4] We discuss this difference between *monkeys H* and *L* in DISCUSSION (*Comparing results from two monkeys*). Again, in the 16-target case, we see that the OTP and canonical layouts perform comparably when either we have very many neural units (performances saturate) or when we have very few neural units (there is insufficient neural information, and thus many constellations will be indistinguishable in terms of performance). These findings should ideally all be confirmed with real experimental data.

Experimental data results

Having developed some expectation of the improvements offered by OTP in simulation, we wanted to verify the algorithm in real experiments, in at least some regime of the data studied in simulation. Across the four different constellation sizes tested above, creating Figs. 5 and 6 required 100 full data sets (each with 1,000 trials per condition) for each choice of neural population size. This implies tens of thousands of experimental days to replicate this result in experimental data; clearly this is infeasible. Instead, we tested this algorithm with four full day data sets in *monkey L,* with the goal of providing some evidence that the proposed algorithm offers improvements in a real experimental setting. Although many more experiments are needed to fully validate the simulation results, finding similar improvements in these experimental results should give confidence that, over a broader range of conditions, the method could well perform as predicted by simulation.

As described in METHODS, *monkey L* first performed many reaches to sixteen canonically placed targets, and a subset of these reaches were used to fit an OTP constellation. In each of the four data sets (*L20061106–L20061122*), we used roughly 40 neural units (sorted by our automatic spike sorter), regardless of unit quality (single unit, multi unit, or "noise" unit (Wahnoun et al. 2006) or tuning depth. The results for these data sets are in Table 1. Note that, as Table 1 comes from a sixteen target experiment with *monkey L,* Table 1 is comparable to Fig. 6*D*. Looking at Fig. 6*D* at 35–45 neural units (the *x* axis), simulation suggests OTP should realize 0–2% decode improvement, and we see in Table 1 that we achieved 3.6%, so the results are comparable. Factors such as array lifetime and the quality of unit tuning resulted in these experimental days having low decode performance, regardless of the topology used. We see that in each data set OTP improved our decode accuracy. We want to ask, for the data we collected, if OTP shows a statistically significant improvement over canonical placements in terms of decode performance. Using a binomial significance test with 95% confidence level (Zar 1999), we see across all our data that indeed OTP does statistically outperform canonical placements (Table 1), confirming what we saw in simulation to be a meaningful improvement.[5] This improvement is captured in the final column of Table 1, where we see in this case that the decode performance is raised from 13.4 to 17.0% on >2,000 trials. The purpose of these data are to validate experimentally that the OTP algorithm is giving us the improvements we anticipate. Although the absolute decode accuracy is low, we nonetheless see that there is a meaningful improvement in decode accuracy. Further, we see that in no case (of the 4 full data sets) is there a reduction in decode accuracy. Thus our experimental results serve to verify that OTP is indeed making good use of the neural population available to find a nontrivial improvement to decode performance.

DISCUSSION

We have shown, for communication prosthetic systems using spiking activity, that reach target decode accuracy can be improved by optimally placing the reach targets. We have introduced this general problem, and we have created a first-of-its-kind algorithm that finds an improved target constellation by approximating an intractable problem with a tractable form. For four and eight targets, OTP layouts closely resembled canonical layouts, thus validating the canonical topology used in Santhanam et al. (2006). Also we realized substantial decode performance improvements in simulation for 2- and 16-target configurations across a wide range of unit counts. Our experiments in real data (Table 1) confirm the expected improvement offered by OTP, at least in the limited regime tested, indicating that target placement is a valuable consideration in the design of neural prostheses.

[4] To put these results into the context of a few other prostheses studies, Santhanam et al. (2006) reported recording 80–130 units in a typical session, and Hatsopoulos et al. (2004) reported recording 32–143 units in a typical session.

[5] Note that doing statistical tests on the data by day (or by any subdivision) will reduce the statistical power (fewer trials) of the data and may lead to inconsistent results (some data sets significant; others not). Thus the total number of trials should be used to show that OTP does indeed outperform canonical topologies. One might also want to know whether or not, by using OTP on a given experimental day, the performance will be improved over canonical placements. Our results indicate that indeed an improvement will be seen, based on the 2,000 trials we collected. Note also that, had we run this experiment on a stronger array with more units, we would expect statistically significant effects with much smaller numbers of trials, since the magnitude of the performance improvement would be greater (cf. Fig 5*D,* at 50–150 units, or Fig 6*D,* at 140–200 units).

TABLE 1. *Decode performance in experimental data for canonical and OTP methods on monkey L*

	L20061106	L200611113	L20061117	L20061122	**Total**
Neural units (*K*)*	46	41	43	35	**NA**
Number of targets (*M*)	16	16	16	16	**16**
Total OTP reach trials†	324	528	720	544	**2116**
Canonical decode performance, %	13.0	15.3	14.2	10.8	**13.4**
OTP decode Performance, %	15.9	15.5	20.6	14.5	**17.0**
Decode improvement, %	2.9	0.2	6.5	3.7	**3.6**

*Units include all automatic spike sort isolations used to fit the optimal target placement constellation. This includes all units regardless of tuning strength or modulation significance. †We compared equal numbers of OTP and canonical reach trials.

Intuition gained from OTP

In Fig. 2 and *Problem intuition,* we saw that intuition for how to place targets quickly breaks down when faced with many neurons and many targets. The OTP algorithm addresses this difficulty, and indeed it gives us reasonable solutions that outperform canonical layouts. Besides the performance improvements, is there anything to be learned from the results of this algorithm? We have noted that ring topologies have classically been chosen based on the observation that neural activity is more strongly modulated by reach direction than reach distance (Churchland et al. 2006a; Fu et al. 1993; Messier and Kalaska 2000; Moran and Schwartz 1999b; Riehle and Requin 1989). If direction was essentially the only source of discriminability, a single 16-target ring would presumably be optimal. If the opposite was true, perhaps a line of targets at various distances from the origin would be chosen. When placing 16 targets with OTP, we typically see 4–6 targets on the interior and 10–12 on the workspace bound. The performance improvements seen in these OTP results (Figs. 5 and 6 and Table 1) support the mixtures of tuning reported in previous studies (Churchland et al. 2006a; Fu et al. 1993; Messier and Kalaska 2000; Moran and Schwartz 1999b; Riehle and Requin 1989). However, it is important to note that this observation depends on the choice of tuning model (here the cosine model of *Eq. 1*). More tuning functions should be tried (recall that OTP is general to the choice of tuning function; see *Eq. 10*) before this claim can be formalized.

Approximations in OTP algorithm

Across the range of unit and target counts tested in simulation, OTP outperforms each canonical ring topology, with performance gains of up to 9%. The minimum improvement was in all cases 0%, in the case of very many neural units, where performances saturate to 100%; or very few neural units, where noise dominates and many different layouts are indistinguishable. We speculate that this logical simulation result would also hold in experimental data, but future experiments should confirm more points on these performance curves. The results shown here are subject to three approximations, which we summarize here: in *Eq. 7*, we solve a minimax problem (minimizing an upper bound) instead of a total probability of error problem; we use KL divergence as a proxy for pairwise error probability in *Eq. 11*; and we optimize a nonconvex problem in *Eq. 13* via a sequence of local quadratic approximations (SQP). Each of these approximations is necessary to put the problem in a tractable form and enables us to address this previously unanswerable question. Although the impact of these approximations has yet to be fully characterized, their use allows us to achieve performance gains (cf. Figs. 5 and 6) that would not otherwise be possible.

It is important also to note that, even when we bundle these approximations together (as we do in the OTP algorithm), we still get consistent improvements in decode accuracy versus canonical target placements. It is possible in theory for OTP to under perform canonical layouts, if, for example, one of the approximations was highly inappropriate. Interestingly, OTP never underperforms canonical layouts in either the simulated data or the experimental data. We anticipate performance improvements beyond the 9% shown here, by using better approximations and improved algorithmic techniques.

Comparing results from two monkeys

Comparing the simulation results for *monkeys H* and *L,* there is an apparent difference in the decode accuracies. We found in our experiments that *monkey H* had significantly better tuned delay period activity than did *monkey L.* Factors such as electrode array lifetime (Polikov et al. 2005), array positioning in M1/PMd (Crammond and Kalaska 2000), and behavioral training could all contribute to these differences. The net result in *monkey L* is that a given number of neural units did not decode as well as in *monkey H.* Hence the performance curves in *monkey L* saturate less quickly than in *monkey H.* It is encouraging nonetheless to see that OTP has similar performance effects at similar regions of the performance curves for both monkeys regardless of the performance scaling introduced by different strengths of neural populations.

Comparing simulated results to experimental results

For *monkey L,* comparing Table 1 results to Fig. 6*D,* one sees a difference between the predicted decode performances at given numbers of units and the actual results found in experiments. In simulated data, while the parameters of the firing rate model (*Eq. 1*) were fit to real neural data, the spike counts used to measure performance in Figs. 5 and 6 were generated from the model in *Eq. 2* (simulated data). The performance improvements for real experimental data depend further on how well the spiking model (firing rate—*Eq. 1*—and output distribution—*Eq. 2*) fits the neural data collected, how well the model generalizes to other target locations for which we have no neural data, and a host of other factors (spike sort instabilities, behavioral changes, etc.). These factors can reduce the performance of both the canonical and OTP topologies (e.g., spiking model) or can reduce just the performance of the OTP topology (e.g., generality of the firing rate model).

Regarding the spiking model approximation, in our simulation study, a unit fit with a particular tuning model behaved according to that model, and its spiking was Poisson. In real experiments, these assumptions do not hold for any target constellation. Real units can be untuned to target position and/or tuned to some other behavioral correlate; both possibilities can introduce a punitive source of noise to the decoder with a limited number of training trials. The spiking model assumptions are approximations that can only reduce performance for both topologies. This performance reduction should be equivalent across topologies, and so we focus our results on the performance differences between topologies and not the absolute accuracies of each decoder. Using a different output distribution (e.g., Barbieri et al. 2001; Cunningham et al. 2008; Truccolo et al. 2004) might improve decoding for both OTP and the canonical topology. Nonetheless the simple Poisson choice allows us to readily demonstrate the improvements offered by OTP.

Our experiments also require an assumption about how different target locations modulate neural firing. The cosine tuning model in *Eq. 1* is a simple first approach. The tractability of the OTP algorithm does not, however, rely on this specific firing rate form, so any improved model (e.g., Kauf-

man et al. 2005) can be seamlessly incorporated into OTP (as noted in *Eq. 10*; to be clear, this is the case with the firing rate model of *Eq. 1*, not the spiking model of *Eq. 2*). As tuning models were not the focus of this study, however, we chose a simple firing rate model to show the improvements offered by OTP even in this case. A more accurate firing rate model, as it would improve the ability of OTP to find an optimal constellation, should only increase the OTP performance gains from a canonical topology. Our presentation of OTP is conservative in this regard.

Implementation considerations

When considering implementing OTP in a neural prosthetic system, an investigator may consider some factors regarding usage mode. In our experiments with *monkey L,* we split our experimental days, using the first half for canonical topology reaches and the second half for OTP topology reaches. In a real system, this training time need not be spent daily. Realistically, the extent to which one records the same neural units (with the same tuning properties) dictates how often one needs to reoptimize the target constellation. We recently reported that neural tuning is stable at least during an experimental day and potentially over multiple days (Chestek et al. 2007). If this is the case, the target configuration would need only be trained infrequently and possibly during an off-line period (e.g., while the subject is sleeping). Anecdotally, we find that OTP fits similar constellations across adjacent days, which further supports this possibility. Furthermore, in our experiments, we used neural units isolated by our automatic spike sorting algorithm regardless of the quality of these units. We did this to focus on the difference in decode accuracy from canonical to OTP, but this choice drags down absolute decode accuracy (due to untuned "noise" units) (for example, see Wahnoun et al. 2006). In a real prosthetic system, better sorts and unit isolations may be made and fed into the OTP algorithm. Doing so would likely raise the decode accuracy of both canonical and OTP topologies. Again, this step may be done off-line to improve overall system performance without compromising the availability of the prosthetic system. The OTP algorithm can also be run on data collected from any target constellation, so one could also iteratively run OTP on a previous OTP configuration (there is no need to revert the system to a canonical topology).

Future work

As mentioned earlier in this paper (e.g., *Comparing simulated results to experimental results*), future work should focus on extending OTP beyond the cosine tuning and Poisson spiking models of *Eqs. 1* and *2*. Future work could also incorporate an iterative OTP algorithm that would monitor for new units appearing, old units rolling off, and units changing tuning, all the while updating the target constellation appropriately. Technology is being developed to allow this recording capability (Chestek et al. 2008; Santhanam et al. 2007). The experiments in this paper presented targets with equal frequency, but future experiments should relax this assumption and extend OTP to handle this case. Furthermore, we have shown here an algorithm using spiking activity only. As multiple modalities (LFP, EEG, ECoG, etc.) are incorporated into a prosthetic system, OTP could be extended to place target constellations based on those sources of neural information as well.

APPENDIX A. DERIVATION OF KL DIVERGENCE FOR POISSON NEURONS

We wish to show, for the Poisson spiking distribution $p(\mathrm{y}|m)$ (parameterized by target position x_m, as given in *Eq. 2*), that the KL divergence has the simple closed form of *Eq. 10*. We begin by substituting *Eq. 2* into *Eq. 9*

$$\mathrm{KL}(x_m \mid x_{m'}) = E_{y|m}\left[\log\frac{p(y \mid m)}{p(y \mid m')}\right] \tag{A1}$$

$$= E_{y|m}\left[\log\prod_{k=1}^{K}\frac{(f_k(x_m)\Delta)^{y_k}e^{-(fk(x_m)\Delta)}y_k!}{(f_k(x_{m'})\Delta)^{y_k}e^{-(f_k(x_{m'})\Delta)}y_k!}\right] \tag{A2}$$

$$= E_{y|m}\left[\sum_{k=1}^{K}\left(y_k\log\frac{f_k(x_m)}{f_k(x_{m'})} + \Delta f_k(x_{m'}) - \Delta f_k(x_m)\right)\right] \tag{A3}$$

where the third line follows using standard rules of exponents and logarithms (and canceling redundant terms in both numerator and denominator). Note that all $f_k(\cdot)$ terms are constant with respect to the expectation [that is, $f_k(\cdot)$ does not depend on y because x_m or $x_{m'}$ is given]. Using this fact and the linearity of expectation (bringing out the sum), we can simplify this KL divergence to

$$\mathrm{KL}(x_m \,||\, x_{m'}) = \sum_{k=1}^{K}\left(\Delta f_k(x_{m'}) - \Delta f_k(x_m) + E_{y|m}[y_k]\log\frac{f_k(x_m)}{f_k(x_{m'})}\right). \tag{A4}$$

Finally, we note that $E_{y|m}[y_k] = \Delta f_k(x_m)$, and so we see

$$\mathrm{KL}(x_m \,||\, x_{m'}) = \Delta\sum_{k=1}^{K}\left(f_k(x_{m'}) - f_k(x_m) + f_k(x_m)\log\frac{f_k(x_m)}{f_k(x_{m'})}\right). \tag{A5}$$

which is the form given in *Eq. 10*.

APPENDIX B. NOTES ON SQP

SQP is a method for solving nonlinear constrained nonconvex problems as in *Eq. 13*. The following gives only a brief overview of our implementation; the interested reader is referred to the excellent tutorial (Boggs and Tolle 1996) and the general reference on convex optimization (Boyd and Vandenberghe 2004). We note at first that SQP is a well known general method; the commonly used MATLAB (The MathWorks, Natick, MA) function for constrained optimization *fmincon* uses an SQP implementation (for medium-scale optimization problems). At low number of targets ($M = 2$ or 4), we found this implementation to be effective. With more targets ($M = 16$), this MATLAB implementation had convergence difficulties presumably associated with its numerical estimates of derivatives. Our implementation of this specific SQP problem, which calculates gradients and Hessians (2nd derivatives) in closed form, remains very effective to larger numbers of targets.

To begin, we must pose *Eq. 13* as a standard optimization problem. To solve this *minimax* problem [i.e., minimizing the maximum element of a finite set, here the $M(M-1)$ target pairs], it is common to introduce a *slack* variable t (Boyd and Vandenberghe 2004; Gockenbach and Kearsley 1999)

$$\begin{aligned}&\underset{\chi,t}{\text{maximize}}\ t^2\\ &\text{subject to } \mathrm{KL}(x_m \,||\, x_{m'}) \geq t^2\ \forall\ m \neq m'\\ &\qquad ||\, x_m \,|| \leq \gamma\ \forall\ m = 1 \ldots M\end{aligned} \tag{B1}$$

Maximizing t subject to the KL constraints imposes that t^2 will have the value of the worst pairwise KL divergence. Introducing this slack variable only makes the problem algorithmically tractable; it does not change the result.

Newton's Method minimizes an (unconstrained) objective function by iterating through a series of minimizations of quadratic approximations to the objective function. Similarly, SQP minimizes a constrained objective function by iterating through a series of minimizations of constrained quadratic approximations to the original problem. These approximations are convex quadratic programs (QP) (see Boyd and Vandenberghe 2004 for extensive reading on QP). Each QP locally approximates the Lagrangian of *Eq. B1* at the current estimates of χ and t [Boggs and Tolle (1996) justifies the choice of the Lagrangian instead of the objective itself]. In our algorithm, we solve each QP quickly and accurately using the MATLAB solver *quadprog*. SQP requires a merit function to determine the length of steps that are taken in χ and t; we used backtracking line search with an l_1 merit function. Beyond these particulars of our algorithm, the reader is again referred to (Boggs and Tolle 1996) for many general implementation details and practical considerations.

ACKNOWLEDGMENTS

We thank M. Sahani, G. Santhanam, and G. Gemelos for valuable technical discussions. We thank G. Santhanam and A. Afshar for the neural data used to fit the firing rate model for *monkey H.* We thank M. Risch for veterinary care, D. Haven for technical support, and S. Eisensee for administrative assistance.

GRANTS

This work was supported by the Michael Flynn Stanford Graduate Fellowship to J. P. Cunningham, National Defense Science and Engineering Graduate fellowships to B. M. Yu and V. Gilja, Gatsby Charitable Foundation funding to B. M. Yu, National Science Foundation graduate research fellowships to B. M. Yu and V. Gilja, National Institute of Neurological Disorders and Stroke Grant CRCNS-R01, Christopher and Dana Reeve Foundation funding to S. I. Ryu and K. V. Shenoy, and the following grants to K. V. Shenoy: Burroughs Wellcome Fund Career Award in the Biomedical Sciences, Stanford Center for Integrated Systems, NSF Center for Neuromorphic Systems Engineering at Caltech, Office of Naval Research, Sloan Foundation, and Whitaker Foundation.

REFERENCES

Barbieri R, Quirk MC, Frank LM, Wilson MA, Brown EN. Construction and analysis of non-Poisson stimulus-response models of neural spiking activity. *J Neurosci Methods* 105: 25–37, 2001.

Boggs PT, Tolle JW. Sequential quadratic programming. *Acta Num* 4: 1–52, 1996.

Boyd SP, Vandenberghe L. *Convex Optimization.* Cambridge, UK: Cambridge Univ. Press., 2004.

Brockwell AE, Rojas AL, Kass RE. Recursive Bayesian decoding of motor cortical signals by particle filtering. *J Neurophysiol* 91: 1899–1907, 2004.

Brown EN, Frank LM, Tang D, Quirk MC, Wilson MA. A statistical paradigm for neural spike train decoding applied to position prediction from the ensemble firing patterns of rat hippocampal place cells. *J Neurosci* 18: 7411–7425, 1998.

Carmena JM, Lebedev MA, Crist RE, O'Doherty JE, Santucci DM, Dimitrov DF, Patil PG, Henriquez CS, Nicolelis MAL. Learning to control a brain-machine interface for reaching and grasping by primates. *PLoS Biol* 1: 193–208, 2003.

Chestek, C, Gilja, V, Nuyujukian, P, Ryu, S, Kier, R, Solzbacher, F, Harrison, R, Shenoy, KV. HermesC: Rf wireless low-power neural recording system for freely behaving primates. *Proc IEEE ISCAS* 1752–1755, 2008.

Chestek CA, Batista AP, Santhanam G, Yu BM, Afshar A, Cunningham JP, Gilja V, Ryu SI, Churchland MM, Shenoy KV. Single-neuron stability during repeated reaching in macaque premotor cortex. *J Neurosci* 27: 10742–10750, 2007.

Churchland MM, Santhanam G, Shenoy KV. Preparatory activity in premotor and motor cortex reflects the speed of the upcoming reach. *J Neurophysiol* 96: 3130–3146, 2006a.

Churchland MM, Yu BM, Ryu SI, Santhanam G, Shenoy KV. Neural variability in premotor cortex provides a signature of motor preparation. *J Neurosci* 26: 3697–3712, 2006b.

Cover TM, Thomas JA. *Elements of Information Theory.* New York: Wiley, 1991.

Crammond DJ, Kalaska JF. Prior information in motor and premotor cortex: activity during the delay period and effect on pre-movement activity. *J Neurophysiol* 84: 986–1005, 2000.

Cunningham, JP, Yu, BM, Shenoy, KV. Optimal target placement for neural communication prostheses. *Proc of the 28th Annual Intl Conf of the IEEE,* New York. *EMBS 2006,* p. 2912–2915.

Cunningham, JP, Yu, BM, Shenoy, KV, Sahani, M. Inferring neural firing rates from spike trains using Gaussian processes. In: *Advances in Neural Information Processing Systems 20*, edited by Platt J, Koller D, Singer Y, and Roweis S. Cambridge, MA: MIT Press, 2008.

Duda, RO, Hart, PE, Stork, DG. *Pattern Classification.* New York: Wiley, 2001.

Fu QG, Suarez JI, Ebner TJ. Neuronal specification of direction and distance during reaching movements in the superior precentral premotor area and primary motor cortex of monkeys. *J Neurophysiol* 70: 2097–2116, 1993.

Georgopoulos AP, Kalaska JF, Caminiti R, Massey JT. On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. *J Neurosci* 2: 1527–1537, 1982.

Georgopoulos AP, Schwartz AB, Kettner RE. Neuronal population coding of movement direction. *Science* 233: 1416–1419, 1986.

Gockenbach MS, Kearsley AJ. Optimal signal sets for non-Gaussian detectors. *SIAM J Optimization* 9: 316–326, 1999.

Hatsopoulos N, Joshi J, O'Leary JG. Decoding continuous and discrete motor behaviors using motor and premotor cortical ensembles. *J Neurophysiol* 92: 1165–1174, 2004.

Hochberg LR, Serruya MD, Friehs GM, Mukand JA, Saleh M, Caplan AH, Branner A, Chen D, Penn RD, Donoghue JP. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature* 442: 164–171, 2006.

Johnson DH, Gruner CM, Baggerly K, Seshagiri C. Information-theoretic analysis of neural coding. *J Comput Neurosci* 10: 47–69, 2001.

Johnson DH, Orsak GC. Relation of signal set choice to the performance of optimal non-Gaussian detectors. *IEEE Trans Commun* 41: 1319–1328, 1993.

Kaufman CG, Ventura V, Kass RE. Spline-based non-parametric regression for periodic functions and its application to directional tuning of neurons. *Stat Med* 24: 2255–2265, 2005.

Kemere C, Shenoy KV, Meng TH. Model-based neural decoding of reaching movements: a maximum likelihood approach. *IEEE Trans Biomed Eng* 51: 925–932, 2004.

Kennedy PR, Bakay RAE. Restoration of neural output from a paralyzed patient by a direct brain connection. *Neuroreport* 9: 1707–1711, 1998.

Kennedy PR, Bakay RAE, Moore MM, Adams K, Goldwaithe J. Direct control of a computer from the human central nervous system. *IEEE Trans Neural Syst Rehabil Eng* 8: 198–202, 2000.

Lebedev MA, Nicolelis MAL. Brain-machine interfaces: past, present, and future. *Trends Neurosci* 29: 536–546, 2006.

Messier J, Kalaska JF. Covariation of primate dorsal premotor cell activity with direction and amplitude during a memorized-delay reaching task. *J Neurophysiol* 84: 152–165, 2000.

Moran DW, Schwartz AB. Motor cortical activity during drawing movements: Population representation during spiral tracing. *J Neurophysiol* 82: 2693–2704, 1999a.

Moran DW, Schwartz AB. Motor cortical representation of speed and direction during reaching. *J Neurophysiol* 82: 2676–2692, 1999b.

Musallam S, Corneil BD, Greger B, Scherberger H, Andersen RA. Cognitive control signals for neural prosthetics. *Science* 305: 258–262, 2004.

Papoulis, A, Pillai, SU. *Probability, Random Variables, and Stochastic Processes.* Boston, MA: McGraw Hill, 2002.

Polikov VS, Tresco PA, Reichert WM. Response of brain tissue to chronically implanted neural electrodes. *J Neurosci Methods* 148: 1–18, 2005.

Proakis JG, Salehi M. *Communication Systems Engineering.* Englewood Cliffs, NJ: Prentice-Hall, 1994.

Riehle A, Requin J. Monkey primary motor and premotor cortex: single-cell activity related to prior information about direction and extent of an intended movement. *J Neurophysiol* 61: 534–549, 1989.

Sahani, M. *Latent Variable Models for Neural Data Analysis* (PhD thesis). Pasadena, CA: California Institute of Technology, 1999.

Santhanam G, Linderman MD, Gilja V, Afshar A, Ryu SI, Meng TH, Shenoy KV. HermesB: a continuous neural recording system for freely behaving primates. *IEEE Trans Bio Med Eng* 17: 609–618, 2007.

Santhanam G, Ryu SI, Yu BM, Afshar A, Shenoy KV. A high-performance brain- computer interface. *Nature* 442: 195–198, 2006.

Santhanam, G, Sahani, M, Ryu, SI, Shenoy, KV. An extensible infrastructure for fully automated spike sorting during online experiments. *Proc 26th Annual Conf IEEE,* San Francisco, CA. *EMBS* 2004, p. 4380–4384.

Schwartz AB. Cortical neural prosthetics. *Annu Rev Neurosci,* 27: 487–507, 2004.

Serruya MD, Hatsopoulos NG, Paninski L, Fellows MR, Donoghue JP. Instant neural control of a movement signal. *Nature* 416: 141–142, 2002.

Shenoy KV, Meeker D, Cao S, Kureshi SA, Pesaran B, Mitra P, Buneo CA, Batista AP, Burdick JW, Andersen RA. Neural prosthetic control signals from plan activity. *Neuroreport* 14: 591–596, 2003.

Smith A, Brown E. Estimating a state-space model from point process observations. *Neural Comput* 15: 965–991, 2003.

Srinivasan L, Eden UT, Mitter SJ, Brown EN. General purpose filter design for neural prosthetic systems. *J Neurophysiol* 98: 2456–2475, 2007.

Taylor DM, Tillery SIH, Schwartz AB. Direct cortical control of 3D neuroprosthetic devices. *Science* 296: 1829–1832, 2002.

Truccolo W, Eden UT, Fellows MR, Donoghue JP, Brown EN. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *J Neurophysiol* 93: 1074–1089, 2004.

Velliste, M, Perel, S, Spalding, M, Whitford, A, and Schwartz, A. Cortical control of a prosthetic arm for self-feeding. *Nature* 453; 1098–1101, 2008.

Verdu S. Asymptotic error probability of binary hypothesis testing for Poisson point-process observations. *IEEE Trans Inform Theory* 32: 113–115, 1986.

Wahnoun R, He J, Tillery SIH. Selection and parameterization of cortical neurons for neuroprosthetic control. *J Neural Eng* 3: 162–171, 2006.

Wolpaw JR, McFarland DJ. Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans. *Proc Natl Acad Sci USA* 101: 17849–17854, 2004.

Wu W, Black MJ, Mumford D, Gao Y, Bienenstock E, Donoghue JP. Modeling and decoding motor cortical activity using a switching Kalman filter. *IEEE Trans Biomed Eng* 51: 933–942, 2004.

Wu W, Gao Y, Bienenstock E, Donoghue JP, Black MJ. Bayesian population decoding of motor cortical activity using a Kalman filter. *Neural Comput* 18: 80–118, 2006.

Yu BM, Kemere C, Santhanam G, Afshar A, Ryu SI, Meng TH, Sahani M, Shenoy KV. Mixture of trajectory models for neural decoding of goal-directed movements. *J Neurophysiol* 97: 3763–3780, 2007.

Yu, BM, Ryu, SI, Santhanam, G, Churchland, MM, Shenoy, KV. Improving neural prosthetic system performance by combining plan and peri-movement activity. *Proc 26th Annual Conf IEEE,* San Francisco, CA. *EMBS* 2004, p. 4516–4519.

Zar J. *Biostatistical Analysis.* Englewood Cliffs, NJ: Prentice Hall, 1999.

Zhang K, Ginzburg I, McNaughton BL, Sejnowski TJ. Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells. *J Neurophysiol* 79: 1017–1044, 1998.

Zumsteg ZS, Kemere C, O'Driscoll S, Santhanam G, Ahmed RE, Shenoy KV, Meng TH. Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems. *IEEE Trans Neural Syst Rehabil Eng* 13: 272–279, 2005.