J. Neural Eng. **4** (2007) 336–347 doi:10.1088/1741-2560/4/3/018

Free-paced high-performance brain–computer interfaces

Neil Achtman[1,5], Afsheen Afshar[1,2,5], Gopal Santhanam[1], Byron M Yu[1], Stephen I Ryu[1,3] and Krishna V Shenoy[1,4]

[1] Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA
[2] Medical Scientist Training Program, School of Medicine, Stanford University, Stanford, CA 94305, USA
[3] Department of Neurosurgery, Stanford University, Stanford, CA 94305, USA
[4] Neurosciences Program, School of Medicine, Stanford University, Stanford, CA 94305, USA

E-mail: shenoy@stanford.edu

Received 9 June 2007
Accepted for publication 1 August 2007
Published 22 August 2007
Online at stacks.iop.org/JNE/4/336

Abstract
Neural prostheses aim to improve the quality of life of severely disabled patients by translating neural activity into control signals for guiding prosthetic devices or computer cursors. We recently demonstrated that plan activity from premotor cortex, which specifies the endpoint of the upcoming arm movement, can be used to swiftly and accurately guide computer cursors to the desired target locations. However, these systems currently require additional, non-neural information to specify when plan activity is present. We report here the design and performance of state estimator algorithms for automatically detecting the presence of plan activity using neural activity alone. Prosthesis performance was nearly as good when state estimation was used as when perfect plan timing information was provided separately (∼5 percentage points lower, when using 200 ms of plan activity). These results strongly suggest that a completely neurally-driven high-performance brain–computer interface is possible.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

Each year millions of people suffer from neurological injuries and disease, resulting in the permanent loss of motor functions. Although most central nervous system impairments still do not have effective treatments, electronic medical systems that interface with the nervous system (termed neural prostheses) have started to fill some of these treatment gaps.

One emerging and promising class of neural prosthesis aims to provide control of paralyzed upper limbs and computer cursors. Figure 1 illustrates the basic operating principle behind these prostheses (Fetz 1999, Nicolelis 2001, Donoghue 2002, Schwartz 2004, Scott 2006). Neural activity from various brain regions, recorded using permanently-implanted arrays of electrodes, is electronically processed to create control signals for enacting the desired movement. After determining how each neuron responds before and during a movement, estimation (decoding) algorithms can infer the desired movement from only the neural activity. Several groups have recently demonstrated that monkeys (Serruya *et al* 2002, Taylor *et al* 2002, Carmena *et al* 2003, Musallam *et al* 2004, Santhanam *et al* 2006) and humans (Kennedy and Bakay 1998, Kennedy *et al* 2000, Hochberg *et al* 2006) can learn to move computer cursors and robotic arms to various target locations simply by activating neural populations that participate in natural arm movements. But even these compelling proof-of-concept laboratory demonstration systems fall short of exhibiting the full level of control needed for many everyday behaviors.

One critical control problem that must be addressed before neural prostheses become clinically viable is the estimation of the cognitive state (Shenoy *et al* 2003). This state estimation involves determining, from neural data alone, when the

[5] These authors contributed equally to this work.

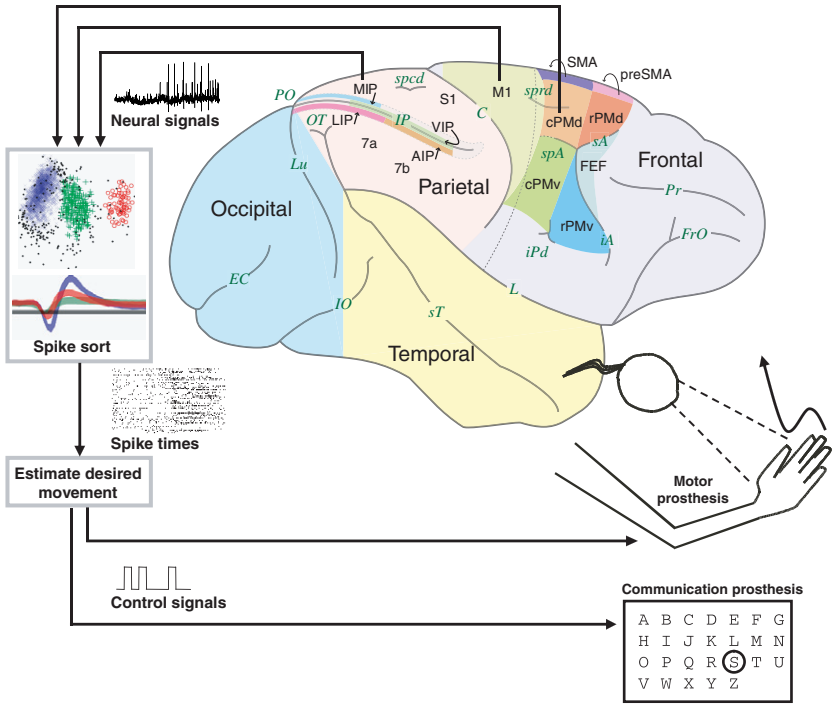

Figure 1. Concept sketch of cortically-controlled prostheses. Several cortical areas in rhesus monkeys, and in homologous areas in humans, participate in the preparation and execution of natural arm movements. Areas include the medial intraparietal area (MIP)/parietal reach region (PRR) with largely plan activity prior to the movement onset (Shenoy *et al* 2003, Musallam *et al* 2004), the dorsal aspect of premotor cortex (PMd) with both plan and movement activities (Santhanam *et al* 2006), and motor cortex (M1) with largely movement activity around the time of the movement (Taylor *et al* 2002, Serruya *et al* 2002, Carmena *et al* 2003, Hochberg *et al* 2006). Neural prostheses measure electrical neural signals (action and local field potentials) using arrays of chronically-implanted electrodes, extract action potential times for each neuron on every electrode and, finally, estimate the desired arm movement and generate control signals for guiding prosthetic devices. Motor prostheses use these signals to generate a continuous movement of an effector device while communication prostheses select from a discrete set of targets.

prosthesis should move. Several brain regions are active well before, or even without, upcoming arm movements while other regions are active primarily during arm movements (figure 1). Neural activity present well before movements would normally begin is termed 'delay' or 'plan' activity, since it is typically measured during a delay period separating target presentation from 'go' cue presentation (e.g., Churchland *et al* 2006a, 2006b, 2006c, Churchland and Shenoy 2007, Crammond and Kalaska 2000). Regardless of whether delay or peri-movement activity is used to control prostheses, it is essential to determine when plan activity is present or when peri-movement activity is present as this is the time, and the only time, that a prosthesis should move.

Prosthetic systems which translate peri-movement activity into moment-by-moment movement commands currently do not perform cognitive state estimation (Serruya *et al* 2002, Taylor *et al* 2002, Carmena *et al* 2003, Hochberg *et al* 2006). As a result, these systems must be manually turned on when the prosthesis is to be operated and turned off at other times so as to avoid unwanted movement. Moreover, when the prosthetic limb or computer cursor is to remain at rest, neural noise continues to drive the prosthesis causing unwanted motion. Similarly, prosthetic systems which translate plan activity into endpoint control signals (Musallam *et al* 2004, Santhanam *et al* 2006) rely on an external non-neural source of information to specify when plan activity is present, as they too do not currently perform state estimation. In both cases, clinically viable systems which operate autonomously for days and weeks must incorporate some form of neurally-driven state estimation.

The importance of state estimation in plan-activity-based prostheses, as well as preliminary estimator designs and offline simulations based on individually-recorded parietal reach region/medial intraparietal (PRR/MIP) neurons, was described previously (Shenoy *et al* 2003). We report here the design of a more advanced state-machine-based state estimator for use in plan-activity-based prostheses (Musallam *et al* 2004, Santhanam *et al* 2006) and performance simulations based on simultaneously-recorded dorsal premotor (PMd) and motor (M1) cortical neurons. Performance is quantified by operating a state estimator and a maximum-likelihood-based target estimator in tandem and calculating how well the correct (1 of 8) target in a prosthetic task could be predicted. We recently reported that it is possible to swiftly and accurately predict the desired target in a plan-activity-based brain–computer interface (BCI) task if external information regarding the plan period timing was provided (Santhanam *et al* 2006). We report here, using a similar task, that it is possible to perform nearly as well when a state estimator replaces this external timing information.

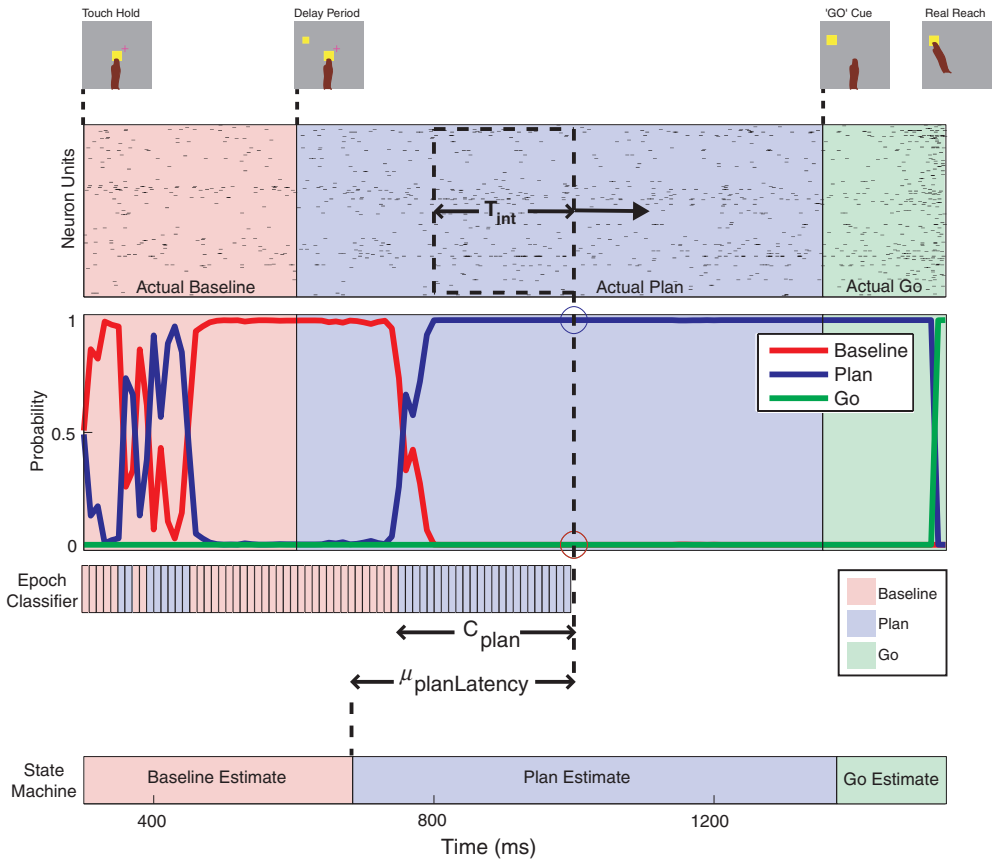

Figure 2. Overview of the state estimator operating on a single trial. Neural activity is recorded as the monkey performs a delayed center-out reach task. Time 400 refers to 400 ms after the beginning of the trial. The epoch classifier calculates the baseline, plan and go probabilities (red, blue and green lines) at each time step, using neural activity over the integration window specified by T_{int}. From these probabilities it forms a state classification of either *Baseline*, *Plan* or *Go*, represented by differently colored rectangles in the figure. The state machine declares a **Baseline** → **Plan** transition after the *Plan* classification has been received C_{Plan} times in a row. The state machine then estimates the start of the **Plan** period by subtracting the average plan detection latency, $\mu_{\text{PlanLatency}}$. This figure displays the moment at which the **Baseline** → **Plan** transition has been detected.

2. Methods

Animal protocols were approved by the Stanford University Institutional Animal Care and Use Committee. Our basic surgical, behavioral training, and experimental data collection methods have been described previously (Santhanam *et al* 2006, Churchland *et al* 2006c, Hatsopoulos *et al* 2004) and are only briefly described below.

2.1. BCI operation without state estimation

Communication prostheses aim to allow a patient to select from a set of discrete 'keys'. To investigate how quickly and accurately a communication prosthesis can perform when driven by cortical plan activity, but *without* attempting to estimate when *plan* activity was present, we recently conducted a series of experiments and computational simulations (Santhanam *et al* 2006). Cognitive state estimation was not considered in this recent study in order to first focus on fundamental, neurobiologically dictated performance limits. With these limits identified, we can now design complementary state estimators which use trial timing parameters identified in Santhanam *et al* (2006) yet maintain overall prosthetic performance.

We trained monkeys to fixate and touch central targets, and plan to reach a visual target that could appear at one of several (2, 4, 8 or 16) different locations (see the top of figure 2 for a depiction of the task). Meanwhile, we collected action potentials from all neurons recorded with a 96-electrode array (typically 100–200 neural units) and used the number of action potentials during an integration period (T_{int}) to predict where the monkey was planning to reach. If our prediction, made using maximum-likelihood (ML) techniques and Gaussian or Poisson neural response models, matched the true target location we displayed a circle around the target, played an auditory tone, and provided a liquid reward to indicate a successful trial. In this way, we were able to assess how fast selections could be made and how often the selections were correct. Importantly, since the computer-controlled experimental apparatus turned on the visual target at a known time (time 0), T_{int} could be arbitrarily positioned. Thus, activity during known plan periods could be accessed. In the present paper, we seek to position the neural integration window based on neurally-derived estimates of when plan activity is present.

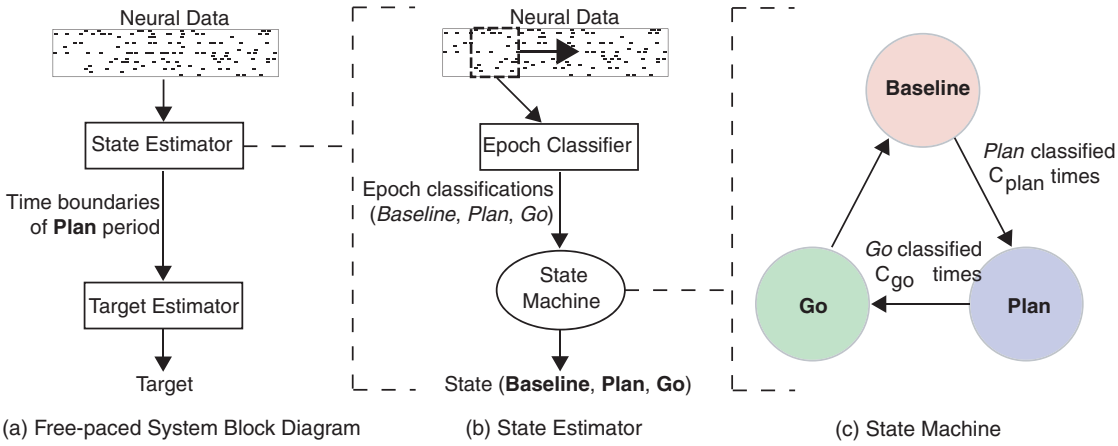

Figure 3. Free-paced system architecture. (a) Block diagram illustrating information flow. Note that epoch classifications are in *italics* and states are in **bold**. The state estimator uses neural activity to estimate the time boundaries of the **Plan** period. The target estimator then uses this **Plan** period estimate to decode the target using the same techniques as in fixed-paced systems (e.g. (Santhanam *et al* (2006)). (b) The state estimator contains an epoch classifier, which provides instantaneous estimates of the activity period, and a state machine, which uses the set of past classifications to estimate the activity period transition times. (c) The state machine determines transitions between **Baseline**, **Plan**, and **Go**. The **Baseline** → **Plan** transition occurs when the epoch classifier decodes *Plan* C_{Plan} times in a row, while the **Plan** → **Go** transition occurs when the epoch classifier decodes *Go* C_{Go} times in a row. Note that these constraints are an improvement upon the algorithm in Shenoy *et al* (2003).

While it is possible to predict the desired target location using neural activity from a variety of different windows (Shenoy *et al* 2003, Musallam *et al* 2004), we swept the temporal location and length of T_{int} in order to see how these parameters affected accuracy (Santhanam *et al* 2006). We found that a 200 ms window that started 150 ms after the target presentation was the shortest window that resulted in near asymptotic accuracy (∼90%).

These prior results suggest that it is possible to position computer cursors quickly and accurately enough to be potentially clinically viable. However, and to reiterate, these recently published results rely on knowing when the reach target was specified—and thus when plan activity is present—and we seek here to replace this external timing information with a strictly neurally-derived estimate.

2.2. The behavioral task and neural recordings

Two adult male rhesus monkeys (G and H) performed a delayed center-out reach task (see top of figure 2). Trials begin by touching a yellow square and fixating a magenta cross, both located near the center of a fronto-parallel screen. After a brief random hold period (200–400 ms), a target cue appeared at one of eight possible locations. Following a variable delay period (200–1000 ms), the central touch and fixation targets disappear; this is the 'go' cue indicating that the reach may begin. The movement ends with the monkey acquiring the target and holding for 200 ms. We recorded neural activity from a 96-channel electrode (Cyberkinetics Neurotechnology Inc.) implanted in the arm representation of PMd (see supplementary materials by Santhanam *et al* (2006) for the exact placement) and automatically spike-sorted data using methods described previously by Santhanam *et al* (2004, 2006).

2.3. The free-paced system: state estimator and target estimator

In addition to replacing external timing information, state estimators, which are capable of identifying *when* plan activity is present, are critical components of 'free-paced' systems. In contrast to the system we recently reported (Santhanam *et al* 2006), a free-paced system can wait indefinitely until plan activity is detected and then move a prosthetic cursor to the desired target. Similarly, such a system could wait indefinitely for peri-movement activity to appear and then guide a prosthetic arm to an object. There are three activity periods within each trial that must be identified in order to achieve free pacing:

(1) **Baseline**, before the target is shown (figure 2, the touch hold period).
(2) **Plan**, after the target is shown but before the 'go' cue is given (figure 2, the delay period).
(3) **Go**, from the 'go' cue until the acquisition of the target (figure 2, the go period and real reach).

To identify these three activity periods we used the free-paced system shown in figure 3(a), which performs state estimation and target estimation separately. The state estimator determines the start and end times of the **Baseline**, **Plan**, and **Go** periods. Given the start and end times of the estimated **Plan** period, the target estimator uses neural activity from within this interval to provide an estimate of the reach target.

2.4. The state estimator

The state estimator, shown in figure 3(b), consists of an epoch classifier and state machine. The epoch classifier uses neural activity within a sliding time window of fixed length to provide a classification (*Baseline*, *Plan*, or *Go*) at each time step. The state machine (figure 3(c)) then uses all past

classifications to estimate the times of transition between the **Baseline**, **Plan**, and **Go** states. It is important to distinguish between the epoch classifications (in *italics*) and the states (in **bold**). The classifications are instantaneous estimates of the activity period using only a fixed period of time. The state machine uses multiple classifications in time, as well as training information from the reach trials, to form estimates of the activity period transition times. These time estimates can then be used for the target estimation.

2.4.1. The state estimator stage 1: epoch classifier. To perform epoch classification, we first built a statistical model of the neuron spike rates in each of the three activity periods. We denote the time of the target appearance as T_{Target}, and the time of the 'go' cue as T_{Cue}. Using the neural activity from a set of reach tasks, we modeled the spike rate distribution by target using data from the following windows:

(1) **Baseline**: $T_{\text{Target}} + [-150, 50]$ ms.
(2) **Plan**: $T_{\text{Target}} + [50, 250]$ ms.
(3) **Go**: $T_{\text{Cue}} + [50, 250]$ ms.

In forming the statistical models for **Plan** and **Go**, we begin using spike rate data following an offset of 50 ms from the time of target presentation and time of go cue presentation. This offset reflects the approximate time necessary for visual information to be transduced at the retina, processed by several subcortical and cortical areas, and reach PMd (Santhanam *et al* 2006). For each of the eight potential reach targets, we determined the mean spike rate and variance in each of the three activity periods: one representing baseline; eight representing plan activity for each possible reach target; eight representing go activity for each possible reach target. These parameters characterized the response distributions and were used to later classify targets using either Poisson or Gaussian models. Note that since plan and go activity is tuned to the reach direction, using one model for all plan or go period activity would not perform well. This is because using one model would necessitate averaging the neural responses across targets. Thus, if one were not to consider each target separately when learning a model of plan (or go) period activity, one would have a more difficult time differentiating plan from baseline (or plan from go) activity.

Provided with this statistical model, the epoch classifier used a sliding window, moving 10 ms at a time, that measured the neural spike rates over an integration window, T_{int}. Since 200 ms of neural activity is sufficient for good target estimation accuracy (Santhanam *et al* 2006) we chose $T_{\text{int}} = 200$ ms. The 200 ms window provided enough neural data to overcome noise in the measured spike rates, yet was also short enough to respond to activity period transitions within a reasonable period of time. The epoch classifier used maximum-likelihood decoding (Yu *et al* 2004, Shenoy *et al* 2003) to calculate the probabilities of the 17 sub-classifications.

The activity period corresponding to the sub-classification with the highest probability (which is the epoch classification) was sent to the state machine; information about the target itself was discarded. Figure 2 illustrates how an epoch classification is obtained from the neural activity at a given time step. Note that at each moment in time, the activity period with the highest probability is the corresponding epoch classification at that time.

2.4.2. The state estimator stage 2: state machine. The two-stage state estimator was developed to solve two problems that arose when using the epoch classifier alone to determine activity period transition times. The first problem is the latency between the appearance of the target and the first *Plan* classification. If the time of the first *Plan* classification were assumed to be the start of the **Plan** activity period, then a significant amount of plan activity would be ignored by the target estimator. The second stage of the state estimator applies a correction factor to compensate for this latency.

Second, the amount of neural activity used in each classification is fixed by the choice of the integration window size. If lower error rates were desired in estimating the activity period, a larger window would be needed. However, since a given statistical model for an activity period assumes a particular neural window size, one would need to create a separate statistical model for each new window size desired. With a dual-stage approach, the second stage can incorporate a varying amount of neural activity by taking into account multiple classifications. Note that this does not require a separate statistical model for each potential value of T_{int}. Thus, larger amounts of neural data are considered with less computational cost and an easier implementation.

To overcome these problems, we employed the state machine shown in figure 3, which uses training data and the collection of past classifications to assign one of the three activity periods to the system. To allow the system to trade higher latency for a lower error rate, we specified that the state machine would only transition to the **Plan** state after the *Plan* classification had been received C_{Plan} consecutive times. The time at which C_{Plan} consecutive *Plan* classifications are received is denoted by $\hat{T}_{\text{PlanDetect}}$. Increasing C_{Plan} will decrease the probability of a premature **Baseline** $\rightarrow$ **Plan** transition, but will increase the delay in detecting that transition. For example, in figure 2, the epoch classifier provides a *Plan* classification at 450 ms, though the target has not appeared yet. By setting C_{Plan} high enough, the system will not transition to **Plan** at this time. A similar consistency rule is applied to the **Plan** $\rightarrow$ **Go** transition, with C_{Go} *Go* classifications required. With C_{Plan} and C_{Go} set independently, one can trade off the error rate for latency differently for the **Baseline** $\rightarrow$ **Plan** and **Plan** $\rightarrow$ **Go** transitions. These are critical differences from the implementation described in Shenoy *et al* (2003), which did not look for consecutive classifications at this stage.

The state machine also contains the average plan and go detection latencies, $\mu_{\text{PlanLatency}} = E[\hat{T}_{\text{PlanDetect}} - T_{\text{Target}}]$ and $\mu_{\text{GoLatency}} = E[\hat{T}_{\text{GoDetect}} - T_{\text{Cue}}]$, which are learned through training trials. Once a given test trial reaches $\hat{T}_{\text{PlanDetect}}$, the average plan detection latency is subtracted in order to obtain the estimate of the target appearance time, $\hat{T}_{\text{Target}}$. $\hat{T}_{\text{Target}}$ is the start of the estimated **Plan** period that is used by the target estimator. Similarly, $\hat{T}_{\text{GoDetect}}$ and the average go detection latency are used to obtain an estimate of the go cue time, $\hat{T}_{\text{Cue}}$. $\hat{T}_{\text{Cue}}$ is the end of the estimated **Plan** period.

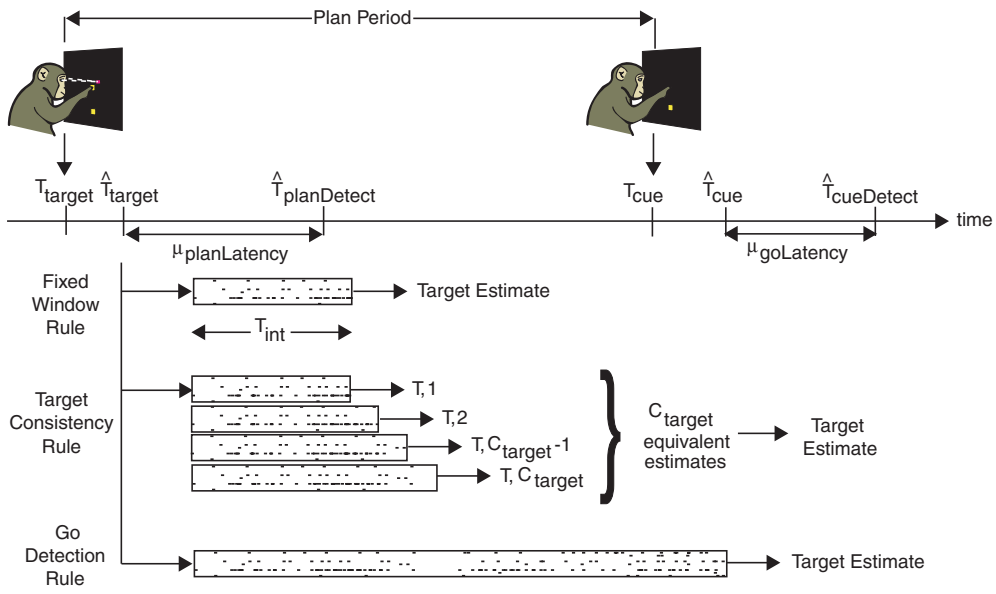

Figure 4. Timing diagram for the three transition rules. After the state estimator provides an estimate of the target appearance time, $\hat{T}_{\text{Target}}$, and the 'go' cue time, $\hat{T}_{\text{Cue}}$, the target estimator decodes the target using the window specified by the transition rule. The fixed window rule uses a fixed length integration window. The target consistency rule uses a steadily expanding window, providing a series of target estimates until the same target is decoded C_{Target} consecutive times. The go detection rule uses a window covering the entire **Plan** period.

2.5. The target estimator

For target estimation we built a different statistical model which was optimized for decoding reach target locations, as opposed to detecting changes in the activity period. We made two changes: (1) we ignored **Baseline**, since it provided no information about the target and (2) for **Plan** and **Go**, we used data within [150, 350] ms after the start of each period instead of [50, 250] ms as before. This further delay in the time window reflects the additional time required for plan activity to fully form and stabilize in PMd, as discussed above in association with our recent fixed-pace results (termed T_{skip} in Santhanam *et al* (2006); and normalized variance reduction time in Churchland *et al* (2006c)). As with the state detection model, we measured the mean and variance of the neuron spike counts for each target during both **Plan** and **Go** periods.

Equipped with these statistical response models and the estimate of the **Plan** period onset time ($\hat{T}_{\text{Target}}$) we then examined three target estimation rules for decoding the desired target. These estimation rules, versions of which were proposed in Shenoy *et al* (2003) and also used in epoch classification, are illustrated in figure 4. These methods use differing amounts of **Plan** neural activity and demonstrate the tradeoffs between speed and accuracy when estimating the desired target. Noted below are the differences between our implementation and that in Shenoy *et al* (2003). The rationale behind these choices is discussed in section 2.6.

2.5.1. The fixed window rule. The target estimator uses a fixed time window, $\hat{T}_{\text{Target}}$ + [150, 350] ms, to calculate the neuron spike count. Since only 200 ms of data is used, this rule is most sensitive to errors in predicting the target appearance time. This is because if the predicted appearance time differs from the actual appearance time, then the target estimator will use spike rate data from outside the time range used in building the target prediction model. Note that this differs from the time transition rule in Shenoy *et al* (2003) since it uses 200 ms of activity instead of 500 ms.

2.5.2. The target consistency rule. The target estimator uses an expanding window, starting with the $\hat{T}_{\text{Target}}$ + [150, 350] ms window of the fixed window rule and expanding by 10 ms at a time. The target estimator decodes the target at each step, only assigning a final estimate when the same target has been decoded C_{Target} consecutive times. By enforcing this consistency requirement, the system can be made more immune to variance in the neuron spike rates and errors in the predicted target appearance time. However, a target may not be declared at all if the system fails to decode any target C_{Target} consecutive times. This rule also introduces a higher latency than the fixed window rule. The target consistency rule may be useful in some human systems where failing to move a prosthetic device is preferable to moving the device in an unintended manner (e.g., wheelchair commands when near automobile traffic).

One further source of error arises from the nonstationarity of the spike rate statistics throughout the **Plan** period. To limit the errors due to the nonstationarity, we expanded our statistical model to include the spike rate statistics for windows from T_{Target} + [150, 350 + 50k] ms, where k is an integer. As our target estimation window increased, we used the window model which provided the closest time fit in order to decode the target.

Note that this is different from the time consistency rule in Shenoy *et al* (2003) since different amounts of neural activity are used (this rule uses a range of window sizes while the time consistency rule always uses 500 ms). In addition, Shenoy *et al* (2003) did not build multiple neural models as we did.

2.5.3. The go detection rule. The target estimator waits for the **Plan** → **Go** transition, then uses the entire **Plan** period to estimate the target. A window of [$\hat{T}_{\text{Target}}$ + 150, $\hat{T}_{\text{Cue}}$ + 50] ms is used to estimate the target. Since this rule ideally incorporates data from the entire **Plan** period, it can provide better accuracy than the (shorter) fixed window rule if the **Plan** period is significantly longer than the 200 ms window. However, longer **Plan** periods also lead to increased latency. There are also errors arising from the need to estimate two transition times (T_{Target} and T_{Cue}) instead of just one and from the nonstationarity of the spike rate statistics. As with the target consistency rule, we used the closest-fit window from the expanded statistical model in order to offset errors from nonstationarity. There still remains the possibility of no target being decoded if the **Plan** → **Go** transition is not detected. Nonetheless, the rule may serve as a useful benchmark in suggesting an upper bound on detection accuracy in free-paced systems.

This differs from the go transition rule in Shenoy *et al* (2003) since the entire plan period is being used for decoding, as opposed to only the last 500 ms in Shenoy *et al* (2003).

2.6. Summary of advances

As noted in the subsections above, there are several key differences between our internally-paced system and that in Shenoy *et al* (2003). These are summarized below.

First, there are fundamental differences in the datasets that were used. We recorded from a different area of the cortex, PMd/M1 as opposed to PRR/MIP. It is not immediately clear that this algorithm would be successful using data from a different area of the cortex. In addition, we recorded our neurons simultaneously instead of using single-electrode recordings. This is important for a few reasons. We have a small handful of single units that are of as good quality as single-electrode recordings, but most of the recordings are multi-unit. In addition, it is theoretically possible that there would be such strong correlations across neurons on a given single trial that the decoding performance would not increase relative to single-electrode recordings when using multiple electrodes simultaneously. This is because recording from additional neurons might not give us any useful information about the motor plan. Thus, the success of our algorithm shows that it is possible to build a free-paced prosthesis using signals from a chronically-implanted array. Finally, we were able to use our simultaneous recordings to demonstrate our system's feasibility in real time (see section 2.7).

Second, we used improved algorithms. Given our results in Santhanam *et al* (2006), we were able to optimize the parameters used, whereas those used in Shenoy *et al* (2003) were chosen in an ad hoc fashion. Specifically, the size of neural windows used was based on the demonstration in Santhanam *et al* (2006) that a 200 ms window was the shortest needed for near optimal decoding accuracy. Note that we assume that such a window was also good enough for epoch classification. In addition, we required our epoch classifier to have a certain number of consecutive decodes

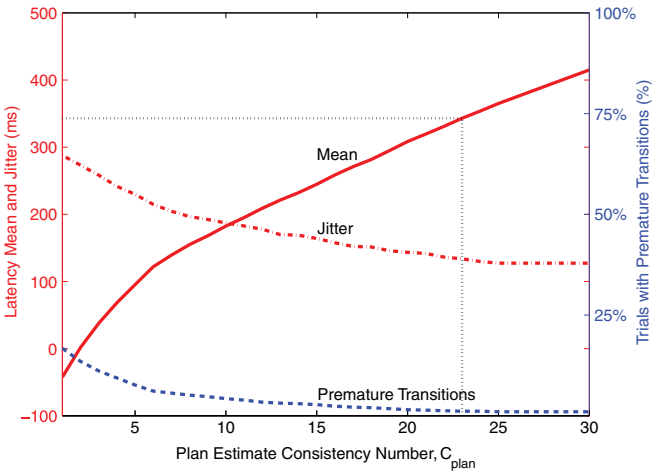

Figure 5. Mean and jitter of the **Plan** detection latency, $\hat{T}_{\text{PlanDetect}} - T_{\text{Target}}$, for dataset H20041118. The number of false (premature) state transitions denotes cases where the onset of plan activity was estimated to have occurred before the actual appearance of the target.

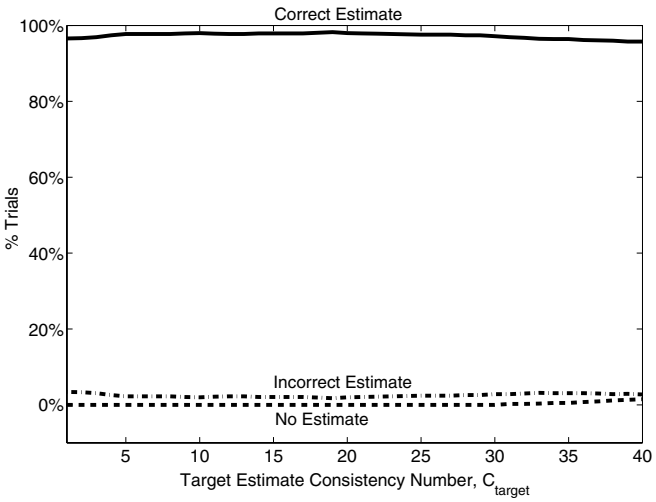

Figure 6. Target estimation accuracy versus latency (via effect of C_{Target}) for the target consistency transition rule for dataset H20041118. In the target consistency rule, a target estimate is provided only after that target has been decoded C_{Target} consecutive times. As C_{Target} increases, there is initially an increase in estimation accuracy. However, for larger values of C_{Target}, there are cases in which no target estimate is provided. The target estimation latency also increases approximately linearly with C_{Target} (not shown).

before transitioning from one state to the next; this restriction was not present at all in Shenoy *et al* (2003). By not enforcing a certain number of consecutive classifications, their method was susceptible to being affected by noise when transitioning out of the baseline state. Most importantly, we optimized these parameters (see figures 5 and 6) in order to achieve the best performance.

2.7. Real-time implementation

The causal design of our free-paced algorithm lends itself to a real-time implementation. However, it is unclear how much

computational delay the real-time implementation would introduce in addition to the latency caused by the algorithm itself (see section 3 for discussion of algorithmic latency and jitter). This is a delay in association with state and target estimation as well as the length of time required to render the appropriate icon to the monkey (termed $T_{\text{dec+rend}}$ in Santhanam *et al* (2006)).

In order to show that it is possible to create an implementation that does not result in an unacceptable additional delay, we extended our experimental platform to allow for real-time state and target estimation. This required performing real-time spike sorting and neural data collection as described in Santhanam *et al* (2006). In addition, two other computers were used: one to extract the relevant window of neural activity at sub-millisecond resolution (30 kHz sampling with Pentium 4, 2.8 GHz); another to calculate all of the relevant state probabilities, execute the target estimation rule, and report a decoded target when appropriate (AMD Athlon dual-core, 2.2 GHz). All of the machines that handled raw neural data from the amplifier were running real-time Linux, which allowed for the real-time hardware control necessary for the precise timing required; the machines that dealt with stimuli presentation were running Windows and DOS. Our implementation fetched 50 ms blocks of neural data at a time to allow for latencies involved in packaging the spiking data into Matlab structures. The 200 ms sliding neural window was then moved every 10 ms as described above.

Parameters were chosen such that the mean delay of the algorithm was less than 350 ms (see section 3), and the fixed window rule was used for our real-time target estimation. The algorithm would therefore always wait until 350 ms after the estimated target presentation time to perform the target decode. Thus, the only additional implementational delay was due to the time to perform the last ML target decode and render the target. As stated in Santhanam *et al* (2006), this is only approximately 40 ms.

3. Results

After learning the statistical models, we ran the free-paced system offline on three datasets: one for monkey G (G20040508) and two for monkey H (H20041118 and H20041217). From each dataset, we used 400 trials—50 trials for each of the eight targets—to build the statistical models and to choose a value of C_{Plan}, C_{Go} and C_{Target}. We then chose a separate 800 trials—100 for each of the targets—to run through the state and target estimators to determine the target prediction accuracy and latency.

3.1. The state estimator

For each dataset, we first characterized the plan and go detection latency of the state estimator so that we could choose an appropriate C_{Plan} and C_{Go}. Figure 5 displays the performance curves for **Plan** onset detection for H20041118, as calculated by the state machine. While varying the consistency number, C_{Plan}, we calculated the mean ($\mu_{\text{PlanLatency}}$) and jitter (standard deviation) of the **Plan** detection latency, using both Poisson and Gaussian neural response-model statistics. The plot only shows the results from Poisson statistics, since Poisson statistics performed better during target estimation for these datasets. Figure 5 also shows the percentage of false state transitions, defined as the cases where the time the algorithm determined the target was presented before its actual appearance. The average detection latency is calculated using all trials, including those where there are false state transitions.

In our recent fixed-paced experiments, where plan activity (T_{int}) was used starting 150 ms after T_{Target} until a few tens or hundreds of milliseconds later, accuracy was observed to saturate for T_{int} times around 200 ms or slightly less (Santhanam *et al* 2006). In other words, by approximately 350 ms (T_{skip} of 150 ms plus T_{int} around 200 ms) after target appearance the system should decode the desired target and move the prosthetic device accordingly.

Therefore, as long as we chose C_{Plan} such that $\mu_{\text{PlanLatency}}$ $\sim$350 ms, the average latency of the free-paced system would be quite similar to that of the fixed-paced system which yielded high-performance values. For the data displayed here, we chose $C_{\text{Plan}} = 23$. At this point, the jitter is near its minimum while $\mu_{\text{PlanLatency}}$ is still approximately within the 350 ms limit (343 ms in this case). For greater values of C_{Plan}, the number of false state transitions decreases only marginally, while $\mu_{\text{PlanLatency}}$ increases and the jitter does not change appreciably. Note that this determination could not have easily been made *a priori* since the relationship between $\mu_{\text{PlanLatency}}$ and C_{Plan} cannot readily be found analytically.

C_{Go}, which is required for the go detection rule, was chosen in a similar way to C_{Plan}. We looked for a choice of C_{Go} which minimized the jitter of difference between the inferred and actual go cue times while also keeping the latency, $\mu_{\text{GoLatency}}$, about 350 ms. Although the 350 ms latency limit was obtained from experiments that decoded based on plan activity alone (Santhanam *et al* 2006), we chose it here as an approximation of a good latency limit for movement activity as well. For this dataset, C_{Go} was chosen to be 21 to result in a $\mu_{\text{GoLatency}}$ of 348 ms.

Note that choosing a C_{Plan} and C_{Go} that result in a mean latency less than 350 ms for the training data does not guarantee that the mean latency for test data will also be $<$350 ms. This is because the statistical models learned on the training data do not necessarily perfectly explain the test data.

3.2. The target estimator

Having inherited the **Plan** activity period start time estimate ($\hat{T}_{\text{Target}}$) from the state estimator, the target estimator now uses one of the target estimation rules to predict the desired target location. For $C_{\text{Plan}} = 23$, we used the estimated target appearance time to obtain the decoded target and target detection time for the three estimation rules. Table 1 displays the percentage of trials decoded correctly, along with the target detection latency average and jitter, sorted by dataset and estimation rule type. Also included is the percentage of trials where no target was detected, due to either an undetected activity period transition or the failure to decode enough

Table 1. Performance summary for the free-paced system.

	Correct target detected (%)			No target detected (%)		
	G20040508	H20041118	H20041217	G20040508	H20041118	H20041217
Fixed window	81.8	82.3	82.3	4.6	0	0
Target consistency	83.0	84.8	84.8	4.6	0	0
Go detection	90.3	88.8	90.2	4.6	0.3	0
Fixed paced	87.3	85.0	84.2	0	0	0
	Latency mean (ms)			Latency jitter (ms)		
Fixed window	358	369	356	162	110	98
Target consistency	500	554	550	177	149	138
Go detection	1232	1148	1197	187	214	198
Fixed paced	350	350	350	0	0	0

consecutive targets for the target consistency rule. Note that premature transitions are considered to be errors (not listed in either column). For comparison purposes, table 1 also contains the decoding accuracy for an offline fixed-paced system run on these exact datasets, in which the target estimator is provided with the exact **Plan** onset time of the target appearance and uses a target estimation window of T_{Target} + [150, 350] ms.

As expected, the target consistency rule provided better accuracy than the fixed window rule across all of the datasets, though only showing slight improvement. The go detection rule provided the best accuracy across all three datasets, since many of the trials had **Plan** periods that were longer than the 200 ms used by the fixed window rule. Our algorithm's performance on the G20040508 and H20041118 datasets was impacted by the failure to detect an activity period transition, with 4.6% of G20040508 trials and 0.3% of H20041118 trials failing to detect **Go**.

Note that for the target consistency rule there was no performance decrease due to failure to decode a target C_{Target} consecutive times. As displayed in figure 6, the values of C_{Target} for which target estimation accuracy is greatest are much less than the values of C_{Target} for which no target estimations are made. This result suggests that the increased latency of this rule in comparison with the fixed window rule is more of a concern than the possibility of failing to estimate a target. The slight decrease in accuracy after reaching a certain C_{Target} arises from the use of **Go** neural activity in the target estimation. The target consistency rule does not check for the **Go** transition and uses the **Plan** statistical models. Therefore, as C_{Target} increases, the trial may enter the **Go** period, where the **Plan** model is not valid, leading to reduced accuracy.

Finally, in comparing the fixed window rule with the fixed-paced system, the free-paced system incurred a loss in accuracy ranging from 1.9 to 5.5% while having a mean latency increase of 6–19 ms, suggesting that a free-paced system could be utilized in place of a fixed-paced system without sacrificing much accuracy or latency.

4. Discussion

Neural prostheses hold considerable promise for dramatically increasing the quality of life of severely disabled patients, but these systems will only become a reality if overall safety and performance are sufficiently high (Scott 2006). Recent research has provided very encouraging proof-of-concept demonstrations (Serruya *et al* 2002, Taylor *et al* 2002, Carmena *et al* 2003, Musallam *et al* 2004), evidence that penetrating electrode arrays can be implanted safely in humans and that well-tuned neural activity persists long after spinal cord injury (Hochberg *et al* 2006), and design principles for achieving performance potentially high enough to outweigh the surgical risk accompanying neurosurgery (Santhanam *et al* 2006). However, several major challenges must be addressed before these cortically-controlled prosthesis are adopted for widespread use. Among the challenges is complementing prosthetic decode algorithms, which spatially guide prostheses, with state estimation algorithms, which must temporally instruct prostheses.

We report here the design and characterization of a relatively straightforward cognitive state estimation system capable of largely preserving overall system performance while enabling the user to freely pace prosthetic movements. Our offline simulations show that neural data can be used to detect transitions between the various activity periods (**Baseline**, **Plan**, and **Go**). These transition-time estimates are accurate enough to allow target estimates that come close in accuracy to the estimates from a fixed-paced system (table 1). This represents an important step toward what must eventually become a theoretically optimal framework for joint state and target estimation.

Increasing the speed and accuracy of free-paced systems will largely rely upon improving the detection of the **Baseline** → **Plan** transition. One method for improving target estimation accuracy is to reduce the jitter in the plan detection latency. This might be achieved by using local field potential (LFP) data in conjunction with the action potentials since low-frequency power is known to be higher during preparatory (plan) periods than during baseline or movement periods (Shenoy *et al* 2003, Santhanam *et al* 2003, Scherberger *et al* 2005). Other methods could provide new algorithms for converting the epoch classifications into estimates of the activity period transition times. For example, rather than requiring C_{Plan} consecutive *Plan* classifications, the state machine could merely require that 90% of the classifications are *Plan* over a given time interval.

We also observed that in many of the cases where the target was incorrectly predicted, the estimated target was adjacent to the correct target. As discussed in Santhanam *et al* (2006), this error pattern contains information beyond what the simple accuracy metric, which does not differentiate systematic errors from random errors, captures. This information can be readily used in prostheses. Importantly, these systematic and useful error patterns which are present in fixed-paced experiments are also present in the current free-paced system, further indicating that the latency and jitter endemic to free-paced systems is not so large as to destroy these patterns.

In addition, not many neurons are required for this system to function properly. First, the majority of neurons that we recorded from were significantly tuned ($p < 0.01$) to the **Baseline** $\rightarrow$ **Plan** and/or **Plan** $\rightarrow$ **Go** transitions (see the insets in figure 7). Further, most of these neurons were also tuned during the **Plan** period, so they would be useful when decoding the target selection as well. As shown in the neuron dropping analysis in figure 7, about 120 neurons are necessary to reach near asymptotic performance for our algorithm. Since most electrode arrays can successfully record from 100 to 200 units, this should not be difficult to achieve with current technology.

As discussed in section 2.7, we have recently implemented this entire free-paced system in real time. Although a detailed discussion of the performance achieved is outside the scope of this paper, this demonstration underscores the point that such a system is indeed realizable. Further, our platform is easily extensible to allow for the integration of other algorithms, such as mixture models that decode trajectories (Yu *et al* 2007) or other free-paced algorithms such as hidden Markov models (Kemere *et al* 2006, 2004).

We believe that this free-paced system (or one similar to it) can be used by patients. It has already been shown that neural activity from paralyzed and diseased patients can be used to control prostheses (Hochberg *et al* 2006, Kennedy *et al* 2000). These prostheses employ algorithms that are dependent upon the fact that individuals neurons are tuned to control signals such as the desired movement direction. It therefore seems likely that our algorithm, which also exploits this tuning of neurons, would be successful in the clinic. In a similar fashion to what was done in Hochberg *et al* (2006), the patient would be asked to make imagined movements to targets displayed on a computer screen. These trials would have known timing and would serve as the training trials for our free-paced system since all epochs would be known. Once sufficient training trials have been recorded, the free-paced system would then be asked to determine on its own when the patient had selected a target and which target had been selected. Note that it is still important to determine the **Baseline** $\rightarrow$ **Plan** transition, since plan-like activity has been observed during the reaction time interval (Crammond and Kalaska 2000, Churchland *et al* 2006c). In addition, teasing apart plan and peri-movement activity would be useful in the clinical setting since studies have noted improved decoding performance when these activities are considered separately (Yu *et al* 2007, Kemere *et al* 2004, 2006). In our clinical implementation, each target could have a corresponding function, such as typing letters on a keyboard or opening a particular application. The patient would be able to stop and think mid-sentence about what to write next while writing an e-mail without fear of the

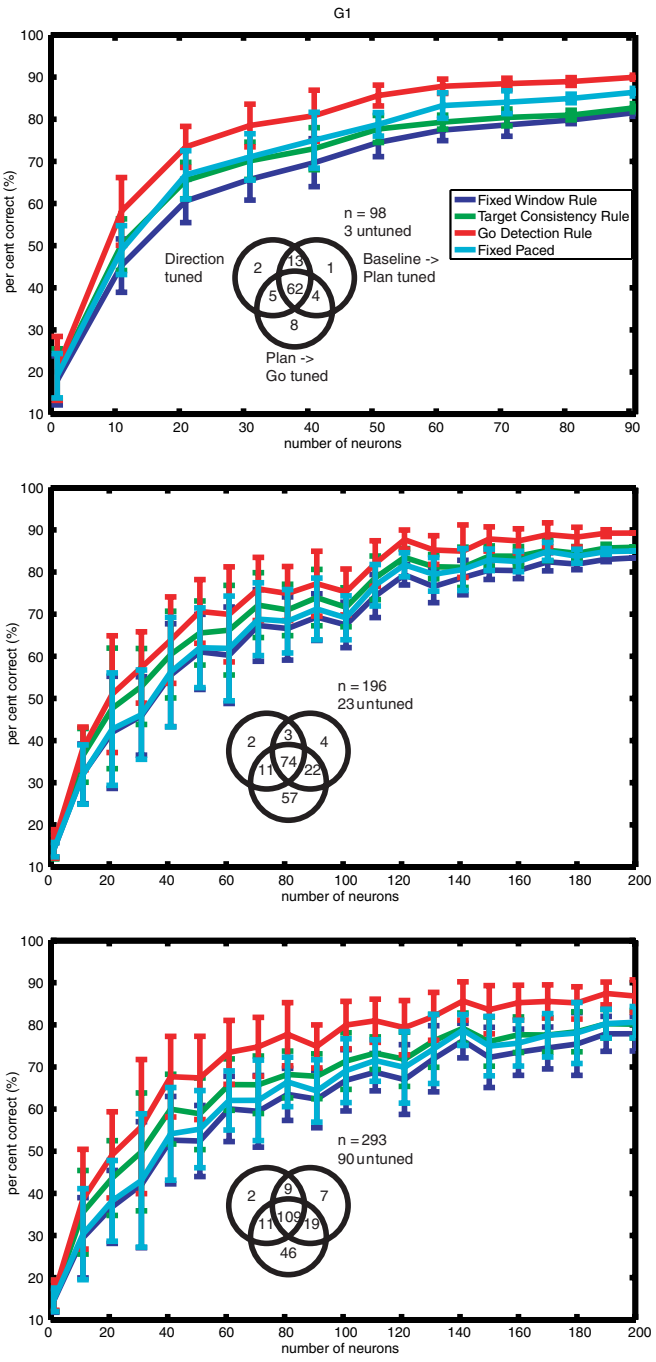

Figure 7. Results from a neuron dropping analysis for all three datasets: G20040508, H20041118, H20041217. For each point in the graph, ten simulations were run by selecting the appropriate number of neurons without replacement for a given run. The mean and standard deviation are plotted. *Insets*: Venn diagrams showing the number of neurons for each dataset that were significantly ($p < 0.01$) tuned to the target direction (top left circle), **Baseline** $\rightarrow$ **Plan** transition (top right circle) and **Plan** $\rightarrow$ **Go** transition (bottom circle). Significance for state transitions was determined by performing a paired *t*-test between firing rates in **Baseline** and each **Plan** state (or between each rates in **Plan** state and its corresponding **Go** state) and looking for those neurons that had a significance of at least $p < 0.01/$(number of pairs) (8 in this case due to eight targets) in at least one of the tests.

decoder continuing to decode possible words, for instance. Contrast this functionality with that offered by a fixed-pace decoder, with which the patient could only type letters when prompted and could not easily stop when he chose to do so. It is also possible that the patient's performance will improve with time as he adapts to the prosthesis (Tillery *et al* 2003).

Note that the average latency of our algorithm (∼350 ms) is about as large as the observed reaction times (200–400 ms). Therefore, it is reasonable to think that one could instruct a human user to merely select keys without the instructed delay present in our task design and maintain our reported performance. Thus, a human user might not detect a difference in speed between having and not having an instructed delay. Importantly, we were able to choose our T_{int} and T_{skip} values that allowed for this speed because of the number of neurons used. If we had recorded from fewer, a larger T_{int} would have been necessary for similar accuracy (see figure 3 in Santhanam *et al* (2006)).

This system encompasses several different parameters and algorithms, and we do not claim to have optimized the system over this entire space. Furthermore, the strategy in choosing the parameters and algorithms will change depending on the relative importance of accuracy and speed in a given application. We present here only one possible implementation of a free-paced system, providing a framework that is adaptable enough to meet the needs of different prosthetic applications.

Future experiments must further explore the real-time implementation of these algorithms and push the online performance of free-paced systems employing joint state and target estimators. While this initial online implementation of the approach reported here appears encouraging, extensive experimentation and characterization remains.

Acknowledgments

We thank M Howard for surgical assistance and veterinary care and S Eisensee for administrative assistance. This work was supported in part by NDSEG Graduate Fellowships (GS, BMY), NSF Graduate Fellowships (GS, BMY), Christopher Reeve Paralysis Foundation (SIR, KVS), Stanford University Bio-X Fellowship (AA), Stanford-NIH Medical Scientist Training Program (AA) and the following awards to KVS: the Burroughs Wellcome Fund Career Award in the Biomedical Sciences, the Center for Integrated Systems at Stanford, the Christopher Reeve Foundation, the NSF Center for Neuromorphic Systems Engineering at Caltech, the Office of Naval Research (Adaptive Neural Systems), the Sloan Foundation and the Whitaker Foundation.

References

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 and Nicolelis M A L 2003 Learning to control a brain–machine interface for reaching and grasping by primates *PLoS Biol.* **1** 193–208

Churchland M M, Afshar A and Shenoy K V 2006a A central source of movement variability *Neuron* **52** 1085–96

Churchland M M, Santhanam G and Shenoy K V 2006b Preparatory activity in premotor and motor cortex reflects the speed of the upcoming reach *J. Neurophysiol.* **96** 3130–46

Churchland M M and Shenoy K V 2007 Delay of movement caused by disruption of cortical preparatory activity *J. Neurophysiol.* **97** 348–59

Churchland M M, Yu B M, Ryu S I, Santhanam G and Shenoy K V 2006c Neural variability in premotor cortex provides a signature of motor preparation *J. Neurosci.* **26** 3697–712

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

Donoghue J P 2002 Connecting cortex to machines: recent advances in brain interfaces *Nature Neurosci.* **5** (Suppl.) 1085–8

Fetz E E 1999 Real-time control of a robotic arm by neuronal ensembles *Nature Neurosci.* **2** 583–4

Hatsopoulos N, Joshi J and O'Leary J G 2004 Decoding continuous and discrete motor behaviors using motor and premotor cortical ensembles *J. Neurophysiol.* **92** 1165–74

Hochberg L R, Serruya M D, Friehs G M, Mukand J A, Saleh M, Caplan A H, Branner A, Chen D, Penn R D and Donoghue J P 2006 Neuronal ensemble control of prosthetic devices by a human with tetraplegia *Nature* **442** 164–71

Kemere C T, Santhanam G, Ryu S I, Yu B M, Meng T H and Shenoy K V 2004 Reconstruction of arm trajectories from plan and peri-movement motor cortical activity *Ann. Neural Prosthesis Program Meeting* (San Diego, CA: Society for Neuroscience)

Kemere C T, Yu B M, Santhanam G, Ryu S I, Afshar A, Meng T H and Shenoy K V 2006 Hidden Markov models for spatial and temporal estimation for prosthetic control *Abstract Viewer/Itinerary Planner* (Atlanta, GA: Society for Neuroscience)

Kennedy P R and Bakay R A E 1998 Restoration of neural output from a paralyzed patient by a direct brain connection *Neuroreport* **9** 1707–11

Kennedy P R, Bakay R A E, Moore M M, Adams K and Goldwaithe J 2000 Direct control of a computer from the human central nervous system *IEEE Trans. Rehabil. Eng.* **8** 198–202

Musallam S, Corneil B D, Greger B, Scherberger H and Andersen R A 2004 Cognitive control signals for neural prosthetics *Science* **305** 258–62

Nicolelis M A L 2001 Actions from thoughts *Nature* **409** 403–7

Santhanam G, Churchland M M, Sahani M and Shenoy K V 2003 Local field potential activity varies with reach distance, direction, and speed in monkey pre-motor cortex *Ann. Neural Prosthesis Program Meeting* (Orlando, FL: Society for Neuroscience)

Santhanam G, Ryu S I, Yu B M, Afshar A and Shenoy K V 2006 A high-performance brain–computer interface *Nature* **442** 195–8

Santhanam G, Sahani M, Ryu S I and Shenoy K V 2004 An extensible infrastructure for fully automated spike sorting during online experiments *Proc. 26th Ann. Int. Conf. IEEE EMBS (San Francisco, CA)* pp 4380–4

Scherberger H, Jarvis M R and Andersen R A 2005 Cortical local field potential encodes movement intentions in posterior parietal cortex *Neuron* **46** 347–54

Schwartz A B 2004 Cortical neural prosthetics *Annu. Rev. Neurosci.* **27** 487–507

Scott S H 2006 Neuroscience: converting thoughts into action *Nature* **442** 141–2

Serruya M D, Hatsopoulos N G, Paninski L, Fellows M R and Donoghue J 2002 Instant neural control of a movement signal *Nature* **416** 141–2

Shenoy K V, Meeker D, Cao S, Kureshi S A, Pesaran B, Mitra P, Buneo C A, Batista A P, Burdick J W and Andersen R A 2003 Neural prosthetic control signals from plan activity *Neuroreport* **14** 591–6

Taylor D M, Tillery S I H and Schwartz A B 2002 Direct cortical control of 3D neuroprosthetic devices *Science* **296** 1829–32

Tillery S I H, Taylor D M and Schwartz A B 2003 Training in cortical control of neuroprosthetic devices improves signal extraction from small neuronal ensembles *Rev. Neurosci.* **14** 107–19

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

Yu B M, Ryu S I, Santhanam G, Churchland M M and Shenoy K V 2004 Improving neural prosthetic system performance by combining plan and peri-movement activity *Proc. 26th Ann. Int. Conf. of the IEEE EMBS (San Francisco, CA)* pp 4516–9