A high-performance neural prosthesis enabled by control algorithm design

Vikash Gilja[1,2,13], Paul Nuyujukian[3,4,13], Cindy A Chestek[2,5], John P Cunningham[5,6], Byron M Yu[5,7–10], Joline M Fan[3], Mark M Churchland[5,7], Matthew T Kaufman[7], Jonathan C Kao[5], Stephen I Ryu[5,11] & Krishna V Shenoy[2,3,5,7,12]

Neural prostheses translate neural activity from the brain into control signals for guiding prosthetic devices, such as computer cursors and robotic limbs, and thus offer individuals with disabilities greater interaction with the world. However, relatively low performance remains a critical barrier to successful clinical translation; current neural prostheses are considerably slower, with less accurate control, than the native arm. Here we present a new control algorithm, the recalibrated feedback intention–trained Kalman filter (ReFIT-KF) that incorporates assumptions about the nature of closed-loop neural prosthetic control. When tested in rhesus monkeys implanted with motor cortical electrode arrays, the ReFIT-KF algorithm outperformed existing neural prosthetic algorithms in all measured domains and halved target acquisition time. This control algorithm permits sustained, uninterrupted use for hours and generalizes to more challenging tasks without retraining. Using this algorithm, we demonstrate repeatable high performance for years after implantation in two monkeys, thereby increasing the clinical viability of neural prostheses.

Neural prostheses have recently shown considerable promise in proof-of-concept animal experiments[1–9] and in human clinical trials[10–13] for partially restoring motor output in paralyzed individuals. Studies in this field primarily focus on adapting insights and methods from the basic neuroscience of cortical motor control to this engineering context. A critical example of this is the use of motor cortex tuning models, which describe the relationship between single unit firing rates and arm-movement kinematics, to define a mapping for neural control of a computer cursor in a closed loop (for example, refs. 1–3). When such a neural prosthesis is introduced to a monkey, performance can improve over days through learning[3]. In addition to controlling computer cursors, these systems have successfully driven robotic end effectors[4]. Neural prosthesis studies have incorporated additional concepts from motor neuroscience, demonstrating the potential to augment system performance by modeling neural activity related to movement preparation[5] and proprioceptive feedback[8]. Recent work also suggests that when the recorded neural population and control algorithm are held constant, neural prosthetic performance increases over time as a stable neural output map is formed, and multiple mappings, once learned, can be retained and retrieved across different control contexts[7]. Despite these new insights and additional algorithm advances (for example, ref. 12), system performance on simple cursor-control tasks remains low relative to the performance of native arm control, presenting a critical barrier to clinical translation[14].

To improve the performance of neural prostheses, we focused on a systems engineering approach. Building on existing methods in the field, we developed two key innovations that alter the modeling assumptions made by these algorithms and the methods by which these algorithms are trained. In addition, we chose signal-conditioning methods, which transform recorded neural signals into control algorithm input, to improve system stability and performance[15,16]. As demonstrated in closed-loop neural control experiments, these methods resulted in high performance across multiple cursor-control tasks.

RESULTS

Performance overview

We trained monkeys to acquire targets with a cursor controlled by either native arm movement or neural activity. We developed an algorithm, ReFIT-KF, that led to substantially higher-performance neural prosthetic control. **Figure 1** compares cursor movements for three different modalities: native arm control, ReFIT-KF and a velocity Kalman filter (Velocity-KF), which is state of the art for current neural prostheses (for example, refs. 11–13). Monkeys were required to move the computer cursor to a visual target and hold the cursor within a demand box for 500 ms to successfully complete a trial and receive a liquid reward. In this center-out-and-back task, targets alternated between a central location and eight peripheral locations. During online neural control sessions, the monkey's contralateral arm was not restrained and movement continued. However, the physical movement was not stereotyped and would often attenuate or even stop

[1]Department of Computer Science, Stanford University, Stanford, California, USA. [2]Stanford Institute for Neuro-Innovation and Translational Neuroscience, Stanford University, Stanford, California, USA. [3]Department of Bioengineering, Stanford University, Stanford, California, USA. [4]School of Medicine, Stanford University, Stanford, California, USA. [5]Department of Electrical Engineering, Stanford University, Stanford, California, USA. [6]Department of Engineering, University of Cambridge, Cambridge, UK. [7]Neurosciences Program, Stanford University, Stanford, California, USA. [8]Gatsby Computational Neuroscience Unit, University College London, London, UK. [9]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. [10]Department of Biomedical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. [11]Department of Neurosurgery, Palo Alto Medical Foundation, Palo Alto, California, USA. [12]Department of Neurobiology, Stanford University, Stanford, California, USA. [13]These authors contributed equally to this work. Correspondence should be addressed to V.G. (gilja@stanford.edu).

Received 23 May; accepted 18 October; published online 18 November 2012; doi:10.1038/nn.3265

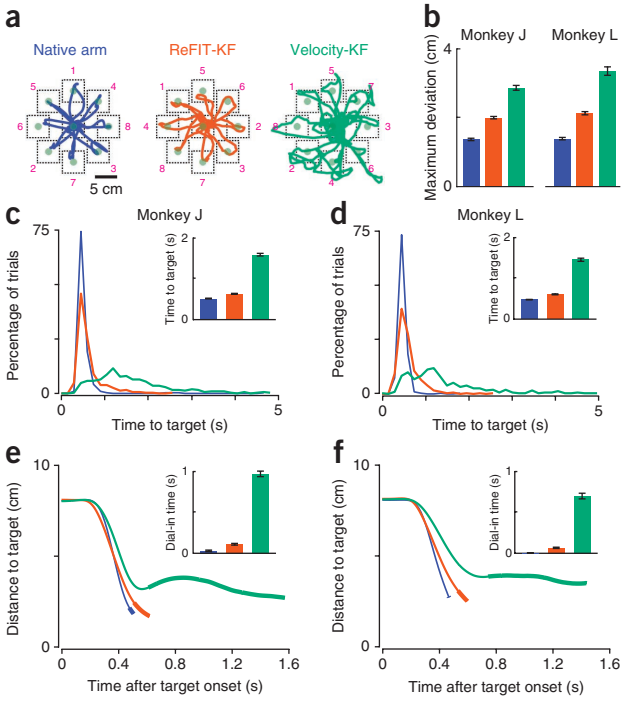


Figure 1 Cursor control with native arm, ReFIT-KF and Velocity-KF. (**a**) Representative traces of cursor path during center-out-and-back reaches by monkey J. Dotted lines (not visible to the monkey) are the demand boxes for the eight peripheral targets and the central target, shown as translucent green circles. Targets alternated between the center and the peripheral in the sequence indicated by the numbers shown. Traces were continuous for the duration of all 16 center-out-and-back movements, representing 15.27 s, 16.87 s and 32.23 s of native arm, ReFIT-KF and Velocity-KF reaching, respectively. (**b**) Maximum deviation from a straight-line path to the target on each successful trial (mean ± s.e.m.). (**c**,**d**) Time to target for successful trials for monkeys J and L. Insets show the time to target (mean ± s.e.m.). (**e**,**f**) Mean distance to the target as a function of time. Insets, mean ± s.e.m. of the dial-in time, or the time required to finally settle on the demand box, after first acquired, to successfully hold for 500 ms. Hold time is not included in the dial-in time. Thickened portions of line graphs also indicate dial-in time, beginning at the mean time of first target acquisition and ending at mean trial duration minus 500 ms. These data are from experiments (designated by monkey identifier letter, year, month and day) J-2010-10-27, J-2010-10-28, J-2010-10-29, J-2010-11-02, L-2010-10-27, L-2010-10-28, L-2010-10-29 and L-2010-11-02. Native arm control is shown in blue, ReFIT-KF control in orange and Velocity-KF control in green. All plots, except the cursor-path traces, were constructed from successful center-out trials from four experimental days for each monkey on which all three control methods were tested. They are computed from 644 native arm, 659 ReFIT-KF, and 619 Velocity-KF trials for monkey J and 632 native arm, 632 ReFIT-KF, and 545 Velocity-KF trials for monkey L.

during some neural control sessions while retaining performance. In additional control experiments, both arms of the monkey were restrained, and we observed little or no arm movement with similar neural control performance (**Table 1**).

The ReFIT-KF algorithm outperformed the Velocity-KF algorithm by several measures. First, cursor movements with ReFIT-KF control were straighter (**Fig. 1a**,**b** and **Supplementary Fig. 1**), producing less movement away from a straight line to the target. Cursor movements produced using the ReFIT-KF were qualitatively similar to native arm movements (**Fig. 1a**, **Supplementary Figs. 1**–**2** and **Supplementary Videos 1** and **2**). Second, these movements were also completed faster. ReFIT-KF cursor-control performance, as measured by the time to successfully acquire the target (**Fig. 1c**,**d**), was 75–85% of native arm control performance and at least twice Velocity-KF control performance (**Supplementary Modeling**). In addition to lower mean time to target, the variance was substantially smaller, which is important as this signifies greater movement consistency and fewer potentially frustrating long trials. Finally, critical to achieving this lower time to target, ReFIT-KF–controlled cursor movements stopped better. The ability to stop is a critical differentiator between the three control modes. The Velocity-KF–controlled cursor took only modestly longer to first acquire the target compared to native arm and ReFIT-KF control, but often significantly overshot the target, requiring additional time and multiple passes to stably acquire and hold the target. This overshoot-correction time dominated the overall time to successful target acquisition for Velocity-KF control and is captured by the metric 'dial-in time', which is the average time required to make the final target acquisition after having first reached the target (**Fig. 1c**–**f**). Both native arm and ReFIT-KF control allowed more precise stopping as compared to that with Velocity-KF control (**Fig. 1e**,**f**).

In all trials in eight experimental sessions with two monkeys, when given 4 s to acquire targets, ReFIT-KF achieved a success rate of >99%, whereas Velocity-KF had a success rate of 95%. We chose a task difficulty to achieve a high success rate for all three control modalities on the first experimental day (**Supplementary Table 1**). When we increased task difficulty, the success rate with Velocity-KF can drop relative to the success rate with ReFIT-KF, and similarly ReFIT-KF success rates and performance can drop relative to those with native arm control (see below).

Experiments across days and years demonstrated consistent high performance of ReFIT-KF control (**Fig. 2**). Performance was stable as measured by throughput (**Supplementary Modeling**) on 280 individual experimental days. We collected these data over 29 months for monkey L and over 16 months for monkey J, spanning 0.4–4.4 years after array implantation. To explore the possibility that performance changed with time, we computed least-squares linear fits on these performance data for each monkey. The slopes of both regression lines are positive, suggesting that performance was stable over the time period of the study and providing evidence consistent with the hypothesis that intracortical microelectrode arrays may permit years of high-performance neural control[15] (Online Methods and **Supplementary Table 2**).

Generalization and robustness

We also tested additional behavioral tasks to assess generalization of the ReFIT-KF control algorithm. We fit the ReFIT-KF algorithm with center-out-and-back reaches, as before, and then tested the algorithm with a pinball task in which targets could appear at any location in the two-dimensional workspace. Monkeys were again required to move the cursor to the target and hold it for 500 ms to successfully complete a trial. Monkey L continuously acquired targets for over 90 min

Table 1 Performance of ReFIT-KF–based control with observation-based algorithm training

Experiment	Target center distance (mm)	Window size (mm)	Acquisition time (s)	Index of difficulty (bits)	Throughput (bits s^{-1})	Success rate (%)
L-2010-08-10	80	40	0.89	1.32	1.49	94
L-2010-08-11	80	40	0.89	1.32	1.48	95
L-2010-08-12	80	40	0.82	1.32	1.59	94
J-2010-08-10	80	40	0.76	1.32	1.74	97
J-2010-08-16	80	40	0.82	1.32	1.60	97
J-2010-08-17	80	40	0.76	1.32	1.73	98

Throughput values, which normalize for task difficulty, are similar to values with ReFIT-KF trials for experimental sessions with arm-based algorithm training (**Fig. 2** and **Supplementary Modeling**), suggesting equivalent control performance.

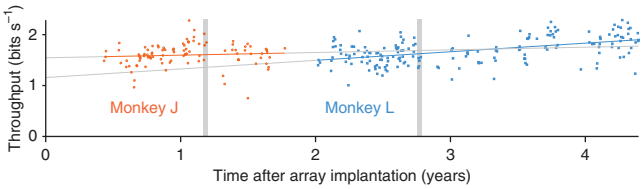


Figure 2 Performance of ReFIT-KF control across four years. Performance was measured by the Fitts's law metric (**Supplementary Modeling**). Data from monkey J and monkey L are shown as 98 orange circles and 182 cyan squares, respectively. Each point plots the performance of the ReFIT-KF algorithm trained on that experimental day. The eight filled data points (four for each monkey) within the time period indicated in gray are calculated from the same data sets used to generate **Figure 1**. Linear regression lines for data for monkey J (orange) and monkey L (cyan) are shown. For all data sets shown, the trial success rate was >90%. Additional details for these data are summarized in **Supplementary Table 2** and **Supplementary Figure 9**.

during two pinball-reaching sessions (**Fig. 3**), one with native arm control and one with the ReFIT-KF control (**Supplementary Videos 3** and **4**). Given 2 s to acquire a target in each trial, both sessions had success rates >98%. Across the whole session, the mean time to target for ReFIT-KF control was 72% as fast as that for native arm control (ReFIT-KF, 710 ms ± 317 ms and native arm, 519 ms ± 196 ms; mean ± s.d.). We observed comparable performance for monkey J (**Supplementary Video 5**). Performance with ReFIT-KF was not only high (over two-thirds as fast as the natural arm, with comparable acquisition-time distributions; **Fig. 3a**), but was also sustained without intervention (**Fig. 3b**). Sustained performance was typical of ReFIT-KF control sessions, whereas Velocity-KF control sessions with the same task parameters had much lower success rates (<40%), and the monkeys could not be motivated to acquire targets for more than 30 min.

To further test ReFIT-KF control, we trained monkey J to avoid visually defined obstacles that appeared in the direct path of the target (**Fig. 4** and **Supplementary Video 6**; maze task[17–19]). The monkey reached from a central starting target to either a left or right peripheral target. In some trials, a barrier appeared along with the peripheral target. To successfully complete a trial, the monkey had to use the cursor to acquire and hold the peripheral target for 500 ms without hitting the barrier. This task was difficult, but the monkey successfully acquired and held the target in 77% of trials with his native arm (**Fig. 4a**) and on 75% of trials with ReFIT-KF control (**Fig. 4b**). Under ReFIT-KF control, mean time to target for this task was 74% as fast as with native arm control (ReFIT-KF, 1,253 ms ± 588 ms and native arm, 932 ms ± 709 ms; mean ± s.d.). With Velocity-KF control, the monkey could not complete the task, and quickly became frustrated and disengaged. As in previous tasks, ReFIT-KF was fit with center-out-and-back movements and was used without modification for the maze task, demonstrating generalization across behavioral contexts.

ReFIT-KF: two innovations for closed-loop neural control

We achieved the described cursor-control performance by redesigning the Velocity-KF algorithm from a closed-loop control perspective (**Supplementary Modeling**). The prosthetic device constitutes a new physical plant with different dynamic properties than the native arm (**Fig. 5a**). The subject controls this new plant by modulating measured neural signals (y_t), which are then decoded into a velocity (v_t) by the control algorithm. This velocity is used to update the cursor on screen, which affects neural signals in subsequent time steps. This closed-loop control perspective suggests two design innovations that both contribute to the described performance (**Supplementary Figs. 3–4**). The first innovation is a modification of neural prosthetic model-fitting methodology. The second innovation is an alteration of the control algorithm.

First innovation

The first design innovation was to fit the neural prosthesis against estimates of intended velocity. Previous algorithms[1,3,5,10] implicitly assume that the subject uses the same control strategy to move the native arm and the prosthetic cursor. As these control strategies may be quite different, we, in the vein of past studies[2,4,6,9,12,20–22], evaluated methods that attempt to better capture the subject's strategy during prosthetic control. Ideally, the control algorithm would be fit to the subject's intended cursor velocity during closed-loop neural control. As we lack explicit access to the monkey's intentions, we hypothesized that the monkey wished to move directly toward the target; this resembles movements made by the native arm and is a good strategy for acquiring rewards.

We used a two-stage optimization procedure (**Fig. 5b**) to fit the neural prosthetic model to these estimates of intended velocity during online neural control. In stage 1, the monkey controls the cursor using his arm. An initial model is fit using arm trajectories and simultaneously recorded neural signals. The monkey then controls the neural prosthesis with this initial model. In stage 2, neurally controlled cursor kinematics and neural signals are recorded and used to fit a new model with an estimate of intended cursor velocity. By starting with cursor velocities collected during the previous online control session (recorded cursor kinematics), these estimates are calculated for model fitting using two transforms. First, the velocities are rotated toward the target to generate

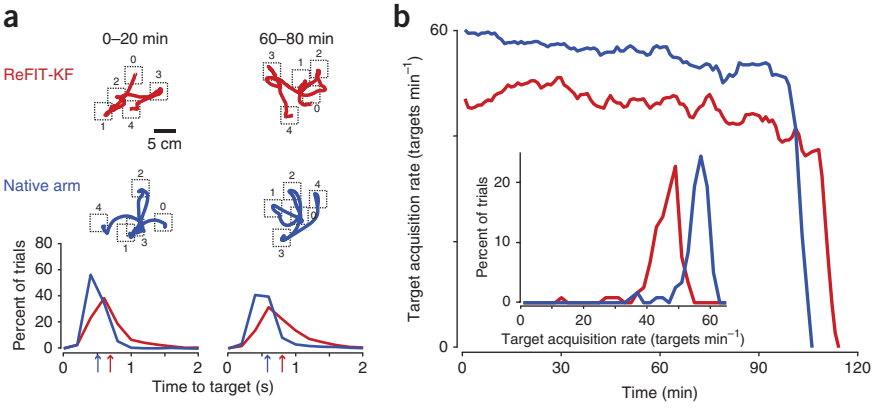


Figure 3 Performance comparison of native arm versus ReFIT-KF for the pinball task. (**a**) For two 20-min segments (columns), shown are randomly selected cursor traces from four consecutive target acquisitions (top; target-demand boxes are shown as dotted lines and target sequence is indicated from 0 to 4). In normalized histograms of time to target for successful trials (bottom), arrows below the plots indicate average time. (**b**) Target acquisition rate per minute throughout the sessions; the sharp rate drop indicates when the monkey lost interest in the task. Inset, acquisition rate across the sessions. The native arm and ReFIT-KF sessions (L-2010-04-01 and L-2010-04-12) were on two separate days, within 11 d of each other, when the monkey demonstrated a high degree of motivation. Native arm control is shown in blue and ReFIT-KF control in red. In this task, each target location was selected from a uniform distribution spanning the workspace.

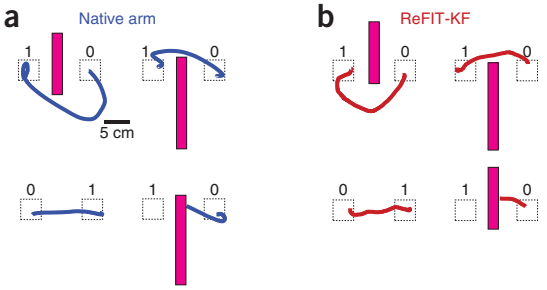


Figure 4 Performance comparison of native arm versus ReFIT-KF for the obstacle-avoidance task. (**a**,**b**) In this task the monkey had to move the cursor from the initial target (labeled 0) to the final target (labeled 1; demand box shown as dotted line) without hitting the magenta-colored barrier. One representative cursor trace is shown from each of the four principal observed movement types: curve under, curve over, straight (no barrier) and collision into barrier. These data are from experiment J-2010-03-09. Native arm control (**a**) is shown in blue and ReFIT-KF control in red (**b**).

estimated intended velocities. Second, if the cursor is on target, the monkey's best strategy is to keep the cursor still to satisfy the hold-time requirement. Thus, in the training set, we assumed that the monkey's intention during these hold periods was to maintain the cursor position by commanding zero velocity. This zero-velocity assumption applied to the fitting of model parameters improved online performance without changing the control algorithm (**Supplementary Modeling** and **Supplementary Fig. 5**). These estimated intentions and corresponding neural data were used to fit the ReFIT-KF control algorithm. Note that we applied the intention estimation only to training data: during online control the neural prosthesis has no knowledge of the task goal or placement of targets (unlike, for example, in refs. 5,23,24).

The aforementioned training protocol used arm-controlled reaches as training data. In a paralyzed individual, it is not possible to record arm kinematics for this step. Instead, this training step could rely on the individual imagining a set of instructed movements. To test this possible strategy, we trained the initial algorithm based on visual-cue observation[8,12], removing the requirement for arm control in step one. During these trials the monkey watched a computer-controlled training cursor that automatically moved to targets. The initial model was fit using automated training cursor trajectories and simultaneously recorded neural activity, without using measured arm movement (Online Methods). **Table 1** summarizes ReFIT-KF performance for three experimental sessions from each monkey in which stage 1 of ReFIT-KF model training was based on observation data instead of arm movements. The performance, as measured by Fitts's law[25] (**Supplementary Modeling**), for these sessions was similar to that attained for the native arm control–initiated sessions described (**Figs. 1** and **2**).

Second innovation

The second design innovation builds on the observation that neural activity is correlated with both the velocity and the position of the cursor. Most existing neural prostheses model a relationship between neural activity and either velocity[2,4] or position[1,10]. A clinical trial in humans has shown that neural prostheses modeling velocity have better performance than those modeling position[11,12]. However, if the control algorithm models only the velocity relationship, then position-based changes in firing will confound decoded velocities (**Supplementary Modeling**). To mitigate this effect, we explicitly modeled velocity as the user's intention and cursor position as an additional variable that affects neural output. This modification allows the user to control velocity with measured neural signals while accounting for the influence of cursor position. We explicitly assumed that the current cursor position, determined by integrating the previous velocity output, is encoded in the neural activity along with the monkey's current intended velocity output. Thus, the expected contribution of position to neural activity is removed, enabling more accurate estimation of intended velocity (**Fig. 1** and **Supplementary Modeling**).

DISCUSSION

Other studies have noted the potential change in plant-and-control strategy and have addressed it by iteratively refining parameters during experiments with neural prosthetics[2,4,6,9,12,20,21]. This approach recognizes that control strategies, and therefore model parameters, are best measured and understood during closed-loop neural prosthetic experiments. However, randomizing initial parameters[2,4,21] may create a control algorithm that never attains the best possible performance, just as optimization problems can easily become trapped in local optima (**Supplementary Modeling**). Although, if the recorded neural population and the neural control mapping are held constant, the consequences of the plant mismatch can be overcome through learning. Such learning has been demonstrated with neural control mappings built to reconstruct arm kinematics, as well as with a neural control mapping in which neuron identities have been shuffled, so the decoder output was no longer predictive of native arm kinematics[7].

The focus of our study was to obtain high control performance in a single session by improving the neural control algorithm and optimizing its parameters. Although the neural prosthesis constitutes a new plant with different properties than the native arm, the motor cortices are involved in native arm control (for example, ref. 26). Therefore, we hypothesized that initializing a model with the relationship between neural activity and natural arm movement would allow the second stage of our training method to achieve greater optimization. Previous studies[4,21] have relied on manipulating the control task to refine the neural decoder, such as by providing assisted control. In those studies, an automated correct answer had been mixed with the output of the neural prosthesis.

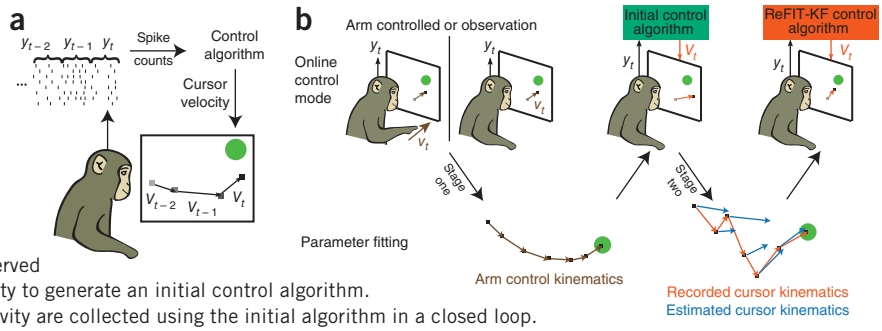


Figure 5 Illustrations of the online neural control paradigm and the ReFIT-KF training methodology. (**a**) The input to the control algorithm at time *i* is a vector of spike counts, y_t, from implanted electrodes. y_t is translated into a velocity output, v_t to drive the cursor. (**b**) ReFIT-KF is trained in two stages. Initially, cursor kinematics and neural activity are collected during arm control or during an observation phase in which cursor movement is automated. These arm movement kinematics or observed cursor kinematics are regressed against neural activity to generate an initial control algorithm. Then, a new set of cursor kinematics and neural activity are collected using the initial algorithm in a closed loop. The kinematics collected during neural control (red vectors) are used to estimate intention by rotating the velocities toward the goal (blue vectors). This estimate of intended kinematics is regressed against neural activity to generate and run ReFIT-KF.

Over successive iterations the weight of automated control had been decreased by experimenter's intuition until only neural activity drove the control. Our approach is different, as the control task remained constant throughout a neural prosthetic experiment session and only the training data were manipulated between the first and second sessions.

In a study with quadriplegic humans[11,12], the neural prosthesis initially had been trained with visual-cue observation, which is similar to the control experiments described above. The study also uses a second neural prosthetic training session to account for differences during online control. Unlike in ReFIT-KF training, in that study both the neural cursor and the automated training cursor were on screen during the second session. The neural cursor was presented to provide feedback so that the participant could attempt to alter their neural output to better follow the training cursor. After this second session, the neural prosthesis was fit with the training cursor kinematics. Thus, the underlying assumption is that the training cursor kinematics capture the intended kinematics during online control, whereas ReFIT-KF fitting assumes that intended kinematics are best inferred by the output of the neurally controlled cursor and knowledge of the task goals.

In studies with adaptive decoders[6,9], the kinematics of the neurally controlled cursor are continuously used to refine neural prosthetic parameters, also allowing them to account for differences when switching to online control. However, they take different approaches to estimating intended kinematics for retraining. One approach is to use decoder parameters noncausally, via smoothing, to estimate intended kinematics for retaining without task or target information[9]. This method has been shown to slow performance declines in one monkey over 29 d when using static spike sorting. However, unlike with ReFIT-KF model fitting, initial performance had not been surpassed, perhaps because without incorporating task goals, the method is subject to inaccuracies present in the initial model fit. In another adaptive study[6], target information had been incorporated in the kinematics used for retraining. Their algorithm was retrained with a weighted average of decoded trajectory positions and the target position for each trial as an estimate of intended position. In contrast, ReFIT-KF estimates intended velocities based on intuitive rules applied to cursor position, decoded velocity and target position.

ReFIT-KF explicitly treats position and velocity differently. The resulting neural prosthesis assumes that the monkey controls velocity and not position, providing performance gains over a position and velocity Kalman filter that does not make this distinction (**Supplementary Figs. 6–8** and **Supplementary Modeling**). We structured the model assuming that velocity intentions evolve smoothly and that the influence of position is based on the monkey's internal model of the cursor. Furthermore, we assumed that the control algorithm output and the monkey's internal belief about cursor position agree. In reality, there is some mismatch between the control algorithm's position estimation and the monkey's internal belief because of inaccuracies in assessing visual information. There are likely spatial and temporal components to this inaccuracy that are not modeled. The spatial aspect is an inexact assessment of the last seen location, and the temporal aspect is due to visual latency. The spatial aspect could be modeled as fixed position uncertainty. To fully account for the temporal aspect, one could attempt to algorithmically model the monkey's internal model of cursor dynamics since the last known position of the cursor. In this work, we chose to start with a simpler model, assuming that this estimation, which is local in time, is exact. It is possible that augmenting the algorithm to account for the mismatch between the temporally local forward model and our dynamics model could improve control performance. Such work could also lead to improvements in the intention-estimation methods used for model training. It is important to note that there may be other explanations for the presence of position information in neural output. For example, this information could be intended cursor position instead of an internal model estimate of current cursor position. In support of the internal model hypothesis, a recent study suggests that a forward model of cursor position is used during closed-loop control[27]. However, additional study of the role of position information in the neural activity during online control is necessary and could aid in the development of future control algorithms.

In experimental sessions, ReFIT-KF performance was stable until the monkey appeared to lose interest in the task (for example, drop in target acquisition rate; **Fig. 3b**). This rapid drop-off is consistent with native arm control session performance and is presumably the analog of when a hypothetical human user is finished using their neural prosthesis. It is expected that performance will drift over time[14], and methods for continuous adaptation of neural-control algorithm parameters may be necessary. In a previous study[9], information from the output of the control algorithm has been used with a Bayesian approach to adapt parameters throughout sessions to sustain performance. If task goals were known throughout neural prosthesis use, the intention estimates defined in this study could be used in conjunction with these parameter-adaptation methods. It may be possible to estimate these task goals based on features of the neural prosthetic output. For example, if a click or target selection signal is simultaneously decoded[12], indicating user intended target selection, intended cursor velocities could be estimated for moments before target selection. Additionally, in future work, it will be important to assess how multiday learning[7] affects the performance and robustness when control algorithm parameters are set as described in this work, based on estimated movement intention, versus existing methods for parameter initialization. Adapting the methods of this work to enable multiday learning and plasticity, such as by providing a consistent controller day over day, may well lead to even better performance over time.

Long-duration, continuous, high-performance operation is central to successful translation of neural prostheses to human patients[14]. The above performance depended on three specific design choices used by both Velocity-KF and ReFIT-KF, in addition to the two key innovations defining ReFIT-KF. First, we did not use spike sorting. The goal of spike sorting is to separate single channels composed of action potentials from many neurons into multiple channels of spiking activity from individual neurons. This standard practice can yield more decodable kinematic information per electrode but requires tracking each sorted action potential shape over days, which has recently been shown to be extremely difficult for many electrode channels[7,28,29]. To reduce signal instabilities that can result from imperfect spike sorting and neuron tracking, we counted the number of threshold crossings per electrode instead of spike sorting (Online Methods)[15,21]. Second, the results reported here were acquired from arrays 19–53 months (monkey L) and 4–21 months (monkey J) after neurosurgical implantation[15,28]. The number of highly distinguishable single neurons on an electrode array tends to decrease over time. Yet remaining multiunit activity often has neural prosthesis–relevant tuning. By using these older array implants, which had relatively few clearly distinguishable single units, we confirmed that threshold crossing–based activity, together with the ReFIT-KF, provides high performance for months and years after array implantation (**Fig. 2**, **Supplementary Table 2** and **Supplementary Fig. 9**). Finally, we used a single, relatively short, 50-ms neural integration time window with no additional temporal lag, unlike some neural prosthetic designs that explicitly incorporate neural data with longer histories and additional lags (for example, multiple 100-ms time bins[7] and multiple 50-ms time bins with history as far back as 1 s (refs. 8,10). This choice was based on experiments with humans using an online

prosthetic simulator[16] and on subsequent neural control experiments with monkeys. Both indicated that shorter time bins are preferable owing to reduced closed-loop feedback time.

This study demonstrated the utility of an online control perspective for the development of neural-control algorithms. Although performance advances must ultimately be verified online, this perspective can be applied in offline simulation studies to examine algorithm-design decisions (**Supplementary Modeling**). However, as with any simulation study, the applicability of the results is subject to both the limitations of the simulation platform and the design decisions made in developing the simulation[16].

The sustained performance and robustness of these ReFIT-KF neural prosthetic experiments demonstrate the potential to provide functional restoration for patients with a limited ability to move and act upon the world because of neurological injury and disease. Although descending pathways are compromised, the motor cortex may be largely intact, enabling this class of technology[10–12,30]. In recent years, brain-interface technologies using a variety of signal sources, such as the intracortical arrays described here, electroencephalography[31] and electrocorticography[32], have been developed. The neural prostheses research community continues to create options for individuals with disabilities and to assess relative risk and benefit[33]. Here we investigated the principled design of closed-loop neural control algorithms, resulting in the development of the ReFIT-KF and demonstrations of a substantial advance in performance and robustness. This algorithm, closed-loop control perspective and system-design methodology may be applied to other neural prosthetic domains with the potential to considerably increase benefit and the clinical viability of prostheses.

METHODS

Methods and any associated references are available in the online version of the paper.

Note: Supplementary information is available in the online version of the paper.

ACKNOWLEDGMENTS

We thank M. Mazariegos, J. Aguayo, W. Kalkus, S. Kang, E. Morgan and C. Sherman for surgical assistance and veterinary care, D. Haven and B. Oskotsky for information technology support, S. Eisensee, B. Davis and E. Castaneda for administrative assistance, P. Ortega for mathematical insight, and S. Stavisky for data-collection assistance. This work was supported by a US National Defense Science and Engineering Graduate Fellowship (V.G.), National Science Foundation Graduate Research Fellowships (V.G., C.A.C., J.M.F., M.T.K. and J.C.K.), Stanford Medical Scholars Program, Howard Hughes Medical Institute Medical Research Fellows Program, Paul and Daisy Soros Fellowship, Stanford Medical Scientist Training Program (P.N.), Stanford Graduate Fellowship (C.A.C., J.P.C. and J.M.F.), Gatsby Charitable Foundation (B.M.Y.), a Helen Hay Whitney postdoctoral fellowship (M.M.C.), Burroughs Welcome Fund Career Awards in the Biomedical Sciences (M.M.C. and K.V.S.), the Christopher and Dana Reeve Paralysis Foundation (S.I.R. and K.V.S.), Defense Advanced Research Projects Agency Revolutionizing Prosthetics 2009 N66001-06-C-8005, Reorganization and Plasticity to Accelerate Injury Recovery N66001-10-C-2010, US National Institutes of Health, National Institute of Neurological Disorders and Stroke Collaborative Research in Computational Neuroscience grant R01-NS054283, and National Institutes of Health Directors Pioneer Award 1DP1OD006409 (K.V.S.).

AUTHOR CONTRIBUTIONS

V.G. and P.N. were responsible for infrastructure development, animal training, and data collection and analysis. V.G. was responsible for algorithm design, supplementary modeling and writing of the paper. P.N. and B.M.Y. participated in algorithm design. C.A.C. participated in animal training, infrastructure development and data analysis. J.P.C. participated in algorithm design and infrastructure development. J.M.F. participated in data collection and analysis. M.M.C. and M.T.K. provided initial animal training for the obstacle avoidance task. J.C.K. participated in data collection. S.I.R. was responsible for surgical implantation. K.V.S. was involved in all aspects of the study.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

Published online at http://www.nature.com/doifinder/10.1038/nn.3265.
Reprints and permissions information is available online at http://www.nature.com/reprints/index.html.

1. Serruya, M.D., Hatsopoulos, N.G., Paninski, L., Fellows, M.R. & Donoghue, J.P. Instant neural control of a movement signal. *Nature* **416**, 141–142 (2002).
2. Taylor, D.M., Tillery, S.I. & Schwartz, A.B. Direct cortical control of 3D neuroprosthetic devices. *Science* **296**, 1829–1832 (2002).
3. Carmena, J.M. *et al.* Learning to control a brain-machine interface for reaching and grasping by primates. *PLoS Biol.* **1**, E42 (2003).
4. Velliste, M., Perel, S., Spalding, M.C., Whitford, A.S. & Schwartz, A.B. Cortical control of a prosthetic arm for self-feeding. *Nature* **453**, 1098–1101 (2008).
5. Mulliken, G.H., Musallam, S. & Andersen, R.A. Decoding trajectories from posterior parietal cortex ensembles. *J. Neurosci.* **28**, 12913–12926 (2008).
6. Shpigelman, L., Lalazar, H. & Vaadia, E. Kernel-arma for hand tracking and brain-machine interfacing during 3D motor control. *Adv. Neural Info. Proc. Sys.* **21**, 1489–1496 (2009).
7. Ganguly, K. & Carmena, J.M. Emergence of a stable cortical map for neuroprosthetic control. *PLoS Biol.* **7**, e1000153 (2009).
8. Suminski, A.J., Tkach, D.C., Fagg, A.H. & Hatsopoulos, N.G. Incorporating feedback from multiple sensory modalities enhances brain-machine interface control. *J. Neurosci.* **30**, 16777–16787 (2010).
9. Li, Z., O'Doherty, J.E., Lebedev, M.A. & Nicolelis, M.A. Adaptive decoding for brain-machine interfaces through bayesian parameter updates. *Neural Comput.* **23**, 3162–3204 (2011).
10. Hochberg, L.R. *et al.* Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature* **442**, 164–171 (2006).
11. Kim, S.P., Simeral, J.D., Hochberg, L.R., Donoghue, J.P. & Black, M.J. Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. *J. Neural Eng.* **5**, 455–476 (2008).
12. Kim, S.P. *et al.* Point-and-click cursor control with an intracortical neural interface system in humans with tetraplegia. *IEEE Trans. Neural Syst. Rehabil. Eng.* **19**, 193–203 (2011).
13. Hochberg, L.R. *et al.* Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature* **485**, 372–375 (2012).
14. Judy, J. Neural interfaces for upper-limb prosthesis control: opportunities to improve long-term reliability. *IEEE Pulse* **3**, 57–60 (2012).
15. Chestek, C.A. *et al.* Long-term stability of neural prosthetic control signals from silicon cortical arrays in rhesus macaque motor cortex. *J. Neural Eng.* **8**, 045005 (2011).
16. Cunningham, J.P. *et al.* A closed-loop human simulator for investigating the role of feedback-control in brain-machine interfaces. *J. Neurophysiol.* **105**, 1932–1949 (2011).
17. Churchland, M.M., Cunningham, J.P., Kaufman, M.T., Ryu, S.I. & Shenoy, K.V. Cortical preparatory activity: representation of movement or first cog in a dynamical machine? *Neuron* **68**, 387–400 (2010).
18. Kaufman, M.T. *et al.* Roles of monkey premotor neuron classes in movement preparation and execution. *J. Neurophysiol.* **104**, 799–810 (2010).
19. Sadtler, P.T., Ryu, S.I., Yu, B.M., & Batista, A.P. High-performance neural prosthetic control along instructed paths. *Conf. Proc. IEEE Eng. Med. Biol. Soc.* **2011**, 601–604 (2011).
20. Gage, G.J., Ludwig, K.A., Otto, K.J., Ionides, E.L. & Kipke, D.R. Naive coadaptive cortical control. *J. Neural Eng.* **2**, 52–63 (2005).
21. Fraser, G.W., Chase, S.M., Whitford, A. & Schwartz, A.B. Control of a brain-computer interface without spike sorting. *J. Neural Eng.* **6**, 055004 (2009).
22. Chase, S.M., Schwartz, A.B., & Kass, R.E. Bias, optimal linear estimation and the differences between open-loop simulation and closed-loop performance of spiking-based brain-computer interface algorithms. *Neural Networks* **22**, 1203–1213 (2009).
23. Srinivasan, L., Eden, U.T., Mitter, S.K. & Brown, E.N. General-purpose filter design for neural prosthetic devices. *J. Neurophysiol.* **98**, 2456–2475 (2007).
24. Yu, B.M. *et al.* Mixture of trajectory models for neural decoding of goal-directed movements. *J. Neurophysiol.* **97**, 3763–3780 (2007).
25. Card, S.K., English, W.K. & Burr, B.J. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics* **21**, 601–613 (1978).
26. Kalaska, J.F. From intention to action: motor cortex and the control of reaching movements. *Adv. Exp. Med. Biol.* **629**, 139–178 (2009).
27. Golub, M.D., Yu, B.M. & Chase, S.M. Internal models engaged by brain-computer interface control. *Conf. Proc. IEEE Eng. Med. Biol. Soc.* **2012**, 1327–1330 (2012).
28. Chestek, C.A. *et al.* Neural prosthetic systems: current problems and future directions. *Conf. Proc. IEEE Eng. Med. Biol. Soc.* **2009**, 3369–3375 (2009).
29. Santhanam, G. *et al.* HermesB: a continuous neural recording system for freely behaving primates. *IEEE Trans. Biomed. Eng.* **54**, 2037–2050 (2007).
30. Hochberg, L.R. Turning thought into action. *N. Engl. J. Med.* **359**, 1175–1177 (2008).
31. McFarland, D.J., Sarnacki, W.A. & Wolpaw, J.R. Electroencephalographic (EEG) control of three-dimensional movement. *J. Neural Eng.* **7**, 036007 (2010).
32. Schalk, G. *et al.* Two-dimensional movement control using electrocorticographic signals in humans. *J. Neural Eng.* **5**, 75–84 (2008).
33. Gilja, V. *et al.* Challenges and opportunities for next-generation intra-cortically based neural prostheses. *IEEE Trans. Biomed. Eng.* **58**, 1891–1899 (2011).

ONLINE METHODS

Surgical procedures and behavioral experiments. All procedures and experiments were approved by the Stanford University Institutional Animal Care and Use Committee (IACUC). Experiments were conducted with adult male rhesus macaques (L and J), implanted with 96-electrode Utah Microelectrode arrays (Blackrock Microsystems) using standard neurosurgical techniques[34]. Data were collected 19–53 months and 4–21 months after implantation for monkeys L and J, respectively. Electrode arrays were implanted in the dorsal aspect of premotor cortex (PMd) and primary motor cortex (M1), as estimated visually from local anatomical landmarks.

The monkeys were trained to make point-to-point reaches in a two-dimensional plane with a virtual cursor controlled by the contralateral arm or by a neural decoder[16]. The virtual cursor and targets were presented in a three-dimensional environment (MusculoSkeletal Modeling Software, Medical Device Development Facility, University of Southern California). Hand-position data were measured at 60 Hz with an infrared reflective bead–tracking system (Polaris, Northern Digital). Behavioral control and neural decode were run on separate PCs using the Simulink/xPC platform (Mathworks) with communication latencies of less than 3 ms. This system enabled millisecond timing precision for all computations. Neural data were initially processed by the Cerebus recording system (Blackrock Microsystems) and were available to the behavioral control system within 5 ms ± 1 ms. Visual presentation was provided via two LCD monitors with refresh rates at 120 Hz, yielding frame updates of 7 ms ± 4 ms. Two mirrors visually fused the displays into a single three-dimensional percept for the user, creating a Wheatstone stereograph (see figure 2 in ref. 16).

Central results were replicated on multiple days in each monkey, using a within-day A-B-A block structure trial design to highlight algorithmic impact and thereby quantify performance and robustness (**Supplementary Figs. 3**–**4**).

Center-out-and-back task configurations. Training sets for fitting the neural control algorithm were collected using the same center-out-and-back task shown in **Figure 1a**. Targets were either uniformly placed at an 8-cm radius or at a 12-cm radius. For some native arm control sessions, the top target was at 14 cm and the upper right and upper left targets were at 13 cm from center. Training sets were typically composed of about 500 (peripheral and central) target acquisitions. All of the test sets shown in **Figure 1** were collected using a standardized target configuration, with eight peripheral targets uniformly placed 8 cm away from the central target with either 5-cm or 6-cm acceptance windows.

Signal acquisition and conditioning. Neural signals were acquired from an implanted 96-channel Utah Microelectrode Array (Blackrock Microsystem) using the Cerebus Recording System (Blackrock Microsystems). An analog band-pass filter with a 0.3 Hz to 7.5 kHz pass-band wass applied to each channel. Channels were sampled at 30,000 samples s^{-1} and are filtered with a 250 Hz to 7.5 kHz digital band-pass filter. A threshold detector was applied to each band-passed channel. The threshold value was set automatically as −4.5 times the measured root mean squared value of the channel. When the signal value was less than threshold, a spike event registered for that channel. The number of spike events was counted in nonoverlapping temporal bins (typically 50 ms). The counts for each channel were the inputs to the control algorithm.

Quantifying performance across months. The same center-out-and-back task was run on 280 sessions across monkeys L and J, spanning at least 16 months for each monkey. Although additional experiments (using different control algorithms and behavioral tasks) may have been run on these experimental days, at least 200 trials of center-out-and-back with the ReFIT-KF control algorithm were tested. On most experimental days, the task difficulty was greater than that shown in **Figure 1** and **Supplementary Table 1**. For the experiments documented in **Figure 1**, the task difficulty was selected so that the monkey could successfully complete the task with the lower quality of control afforded by the Velocity-KF algorithm.

The Fitts' law calculation was used to provide a metric that normalizes across task difficulty. For reference, monkey L was implanted on 22 January 2008 and monkey J was implanted on 24 August 2009. Data for monkey L were collected on 182 sessions over 29 months (from 24 months to 53 months after implantation). Data for monkey J were collected on 98 sessions over 16 months (from 5 months to 21 months after implantation). Each open square and circle in **Figure 2** corresponds to a single experimental day on which the index of difficulty was 1.32 (4-cm targets 8 cm from center) and throughput was calculated from at least 40 trials of center-out to either a vertical or horizontal target. All experiments from the time spans indicated that match these criteria were included, except for days on which other experiments may have impacted animal behavior. Regression lines were fit for data from each monkey using least-square regression, and *P* values were calculated using an ANOVA for linear regression models.

Observation-based model training. As paralyzed users of neural prostheses cannot generate overt arm movements, an observation based algorithm training methodology can be used, as in previous animal studies[8] and clinical trials[10]. We tested the ReFIT-KF algorithm with observation-based training, replacing the native arm movement stage of algorithm training with an observation stage (**Fig. 5b**).

Observation-based decode models were built with both of the monkey's arms comfortably restrained along his side. A previously recorded arm-controlled experimental block of 500 center-out-and-back trials was shown to the monkey while in this posture. The kinematics of this recording were derived from an arm-controlled session for monkey L. To help keep the monkey engaged in the task, he was rewarded when the computer-controlled cursor acquired and held the target for 500 ms.

Under this experimental context, the neural data recorded during these observation sessions and the previously recorded cursor kinematics served as the training data to build the initial decode model. This resulting model was then run online and used as training data to build the ReFIT-KF decoder. Little to no arm movement was visually noted during both observational blocks and decoding blocks.

Performance of ReFIT-KF–based control during these sessions, as measured by the Fitts' law metric, was roughly equivalent to performance in sessions that initially trained from arm-movement data.

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

doi:10.1038/nn.3265

SUPPLEMENTARY MATERIAL

Technical Report:
A High-Performance Neural Prosthesis Enabled by Control Algorithm Design

Vikash Gilja*, Paul Nuyujukian*, Cindy A. Chestek, John P. Cunningham, Byron M. Yu, Joline M. Fan, Mark M. Churchland, Matthew T. Kaufman, Jonathan C. Kao, Stephen I. Ryu, Krishna V. Shenoy

Correspondence should be addressed to V.G. (gilja@stanford.edu)

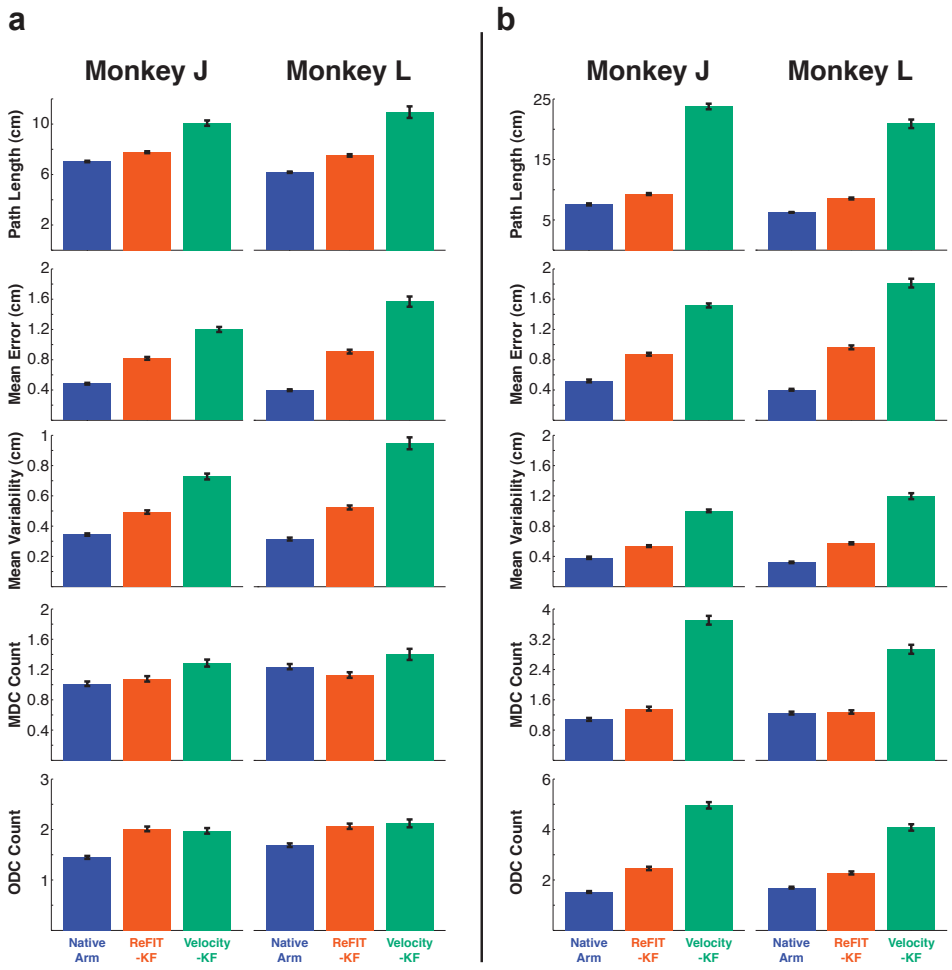
Nature Neuroscience: doi:10.1038/nn.3265

Contents

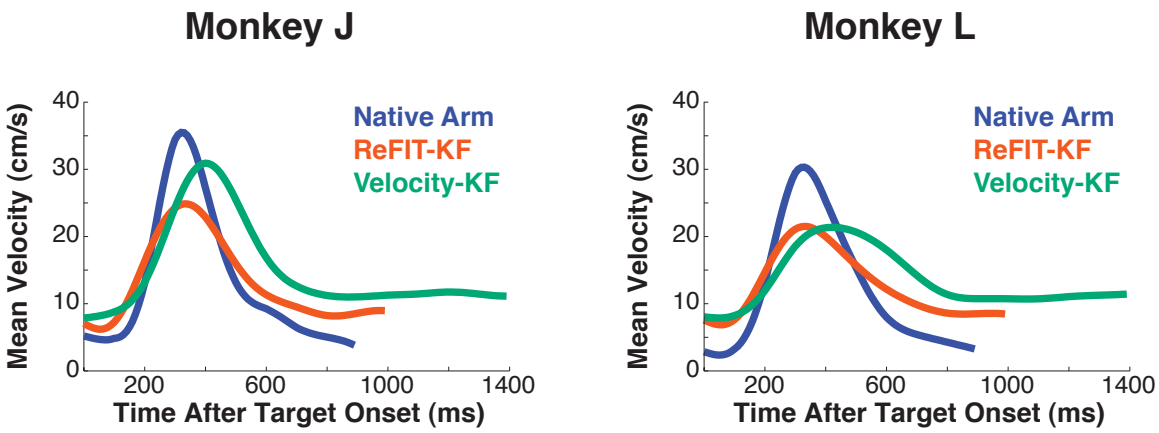
Nature Neuroscience: doi:10.1038/nn.3265

Nature Neuroscience: doi:10.1038/nn.3265

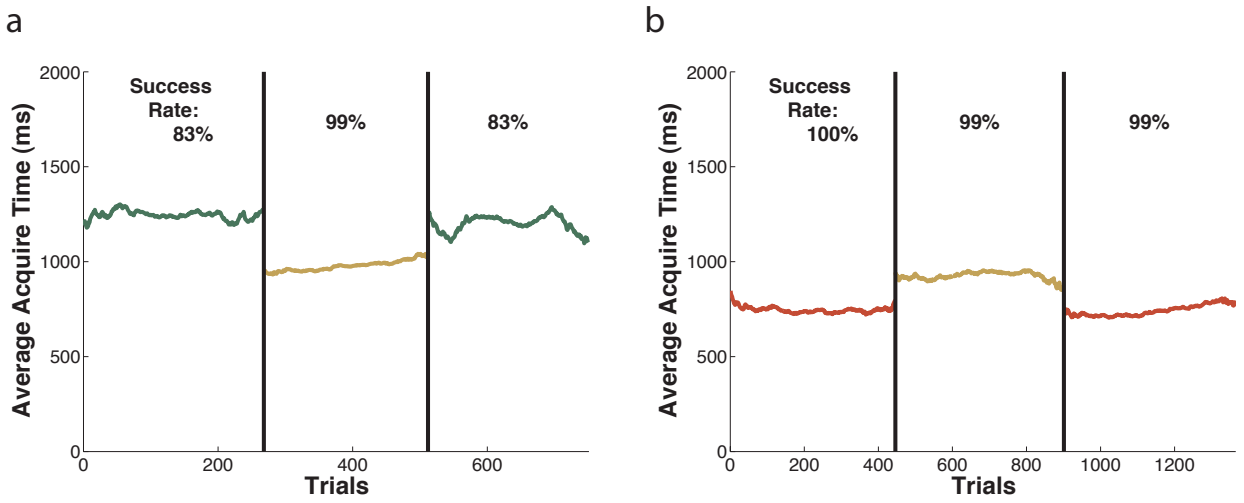
Supplementary Figures



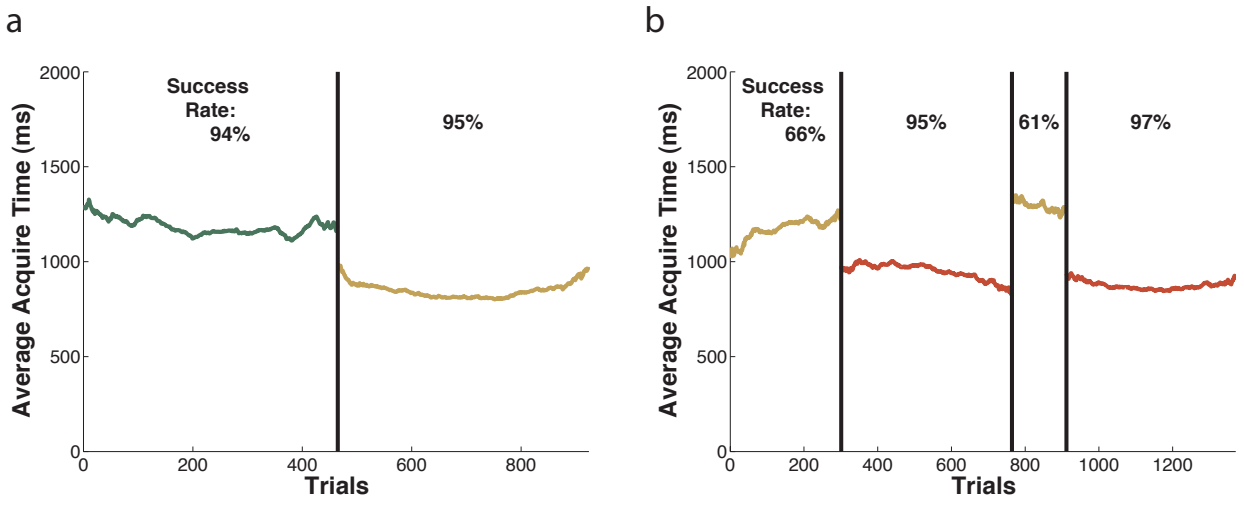
Supplementary Figure 1: Path Quality Measures for Three Control Modes. Five additional measures of path quality were computed for the data presented in Fig 1 of the main text (maximum deviation from a straight line path, calculated from target onset to first target acquire is shown in that figure). (a) Calculated from target onset until first target acquisition. (b) Calculated from target onset until the last target entry before target successfully held. These measures comprise all of those used in two studies [1, 2] and lower values indicate higher control quality. Path length is defined as the integrated cursor displacement throughout the trial. The remaining measures rely on the definition of a movement axis, defined by the direct line path from the cursor position at the start of the trial to the target position. Mean error is the integrated distance from that axis and mean variability is the standard deviation of this distance. Movement direction change (MDC) count is the number of times the cursor velocity in the movement axis reversed signs. Orthogonal direction change (ODC) count is the number of times the cursor velocity orthogonal to the movement axis reversed signs. Native arm control performs best with respect to all measures. When computing until the time of first target acquire, ReFIT-KF outperforms Velocity-KF on all measures except for ODC count. If we include the dial-in time in the computation, ReFIT-KF outperforms on all measures, with a wider performance gap.



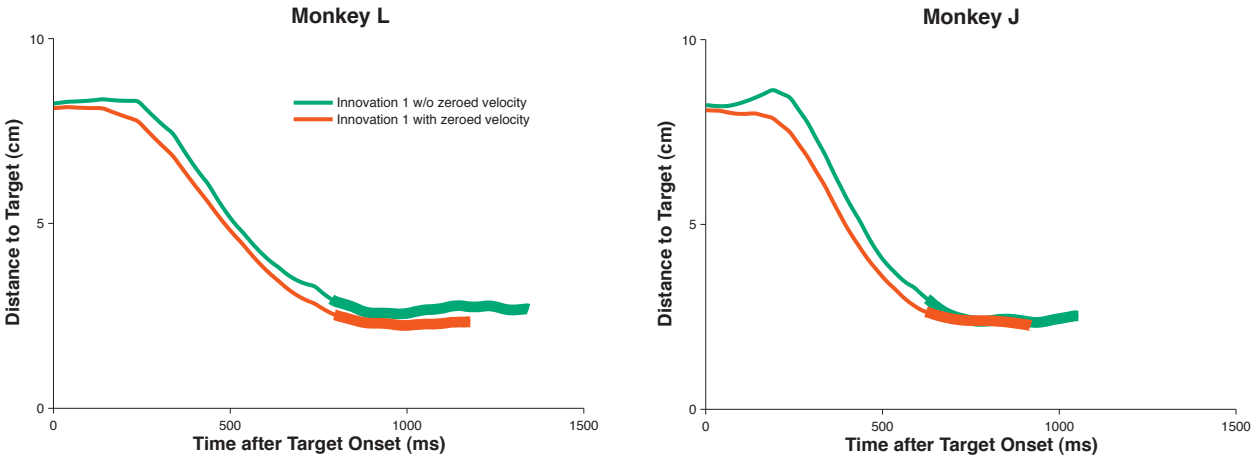
Supplementary Figure 2: Average Velocity vs. Time for Three Control Modes. Average cursor velocity is plotted as a function of elapsed time from target onset. Velocity is calculated at 100 ms intervals and interpolated to a 10 ms sampling interval. Each control type is plotted up to a time point for which at least 300 trials are in the dataset. These data are from the same trials used to generate Fig. 1 of the main text. Note that the average ReFIT-KF profile is more local in time than Velocity-KF. Both neural control modes have lower peaks in this average profile than native arm control. The peak velocities (mean $\pm$ standard deviation) of the native arm, ReFIT-KF, and Velocity-KF are 40.1 $\pm$ 10.8 (35.5 $\pm$ 9.5), 29.6 $\pm$ 5.7 (27.0 $\pm$ 8.2), 35.7 $\pm$ 7.3 (28.6 $\pm$ 7.2) cm/s for monkey J (monkey L). Although the peak velocities were higher for Velocity-KF than ReFIT-KF, ReFIT-KF control resulted in faster target acquisitions than Velocity-KF control. This is likely due to more precise control with ReFIT-KF.



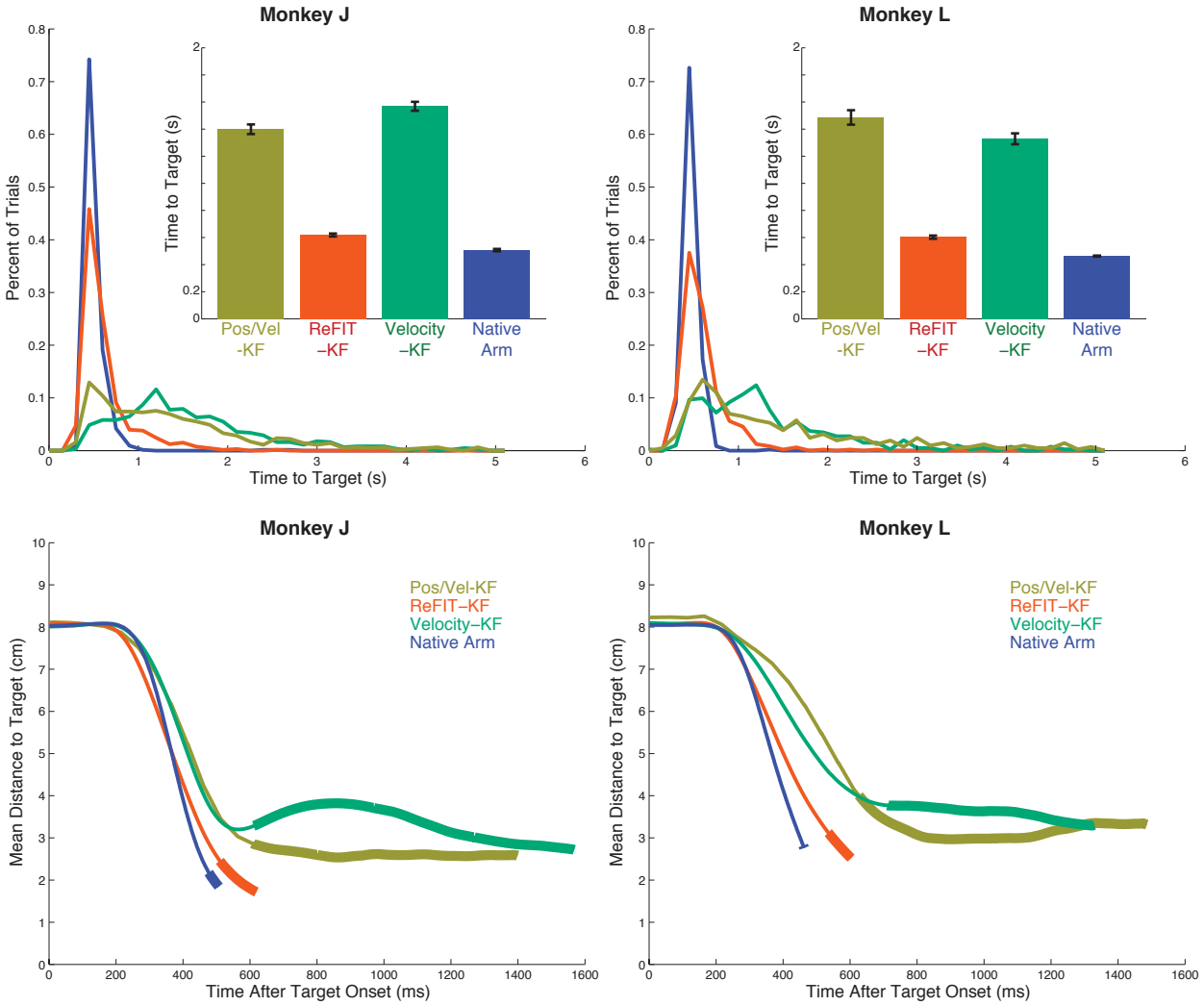
Supplementary Figure 3: Innovation Contribution Breakdown (Monkey L). Here we show the relative contributions to performance that each innovation makes. We tested algorithms in succession, switching between them on the same day against identical trial conditions. Observed differences in performance between trial blocks while holding both behavioral task and neural recording conditions constant can be attributed primarily to differences in the control algorithm. (a) shows Monkey L's performance with Velocity-KF (green) compared against the Kalman filter with only the first innovation (yellow) (L-2010-01-13) and (b) shows the Velocity-KF with the first innovation (yellow) compared against the ReFIT-KF (both innovations, red) (L-2010-08-19). The task conditions for these trial blocks were a randomized center-out and back chain of 8 targets, with a demand box of 5cm for (a) and 4cm for (b), allowing up to 3 seconds to acquire the target. Note that each innovation reduced the average acquire time. In some instances, the overall success rate also increased. Given that these tasks have a maximum acquire time (typically 2.5 seconds), improved performance is marked by higher success rates and/or lower acquire times.



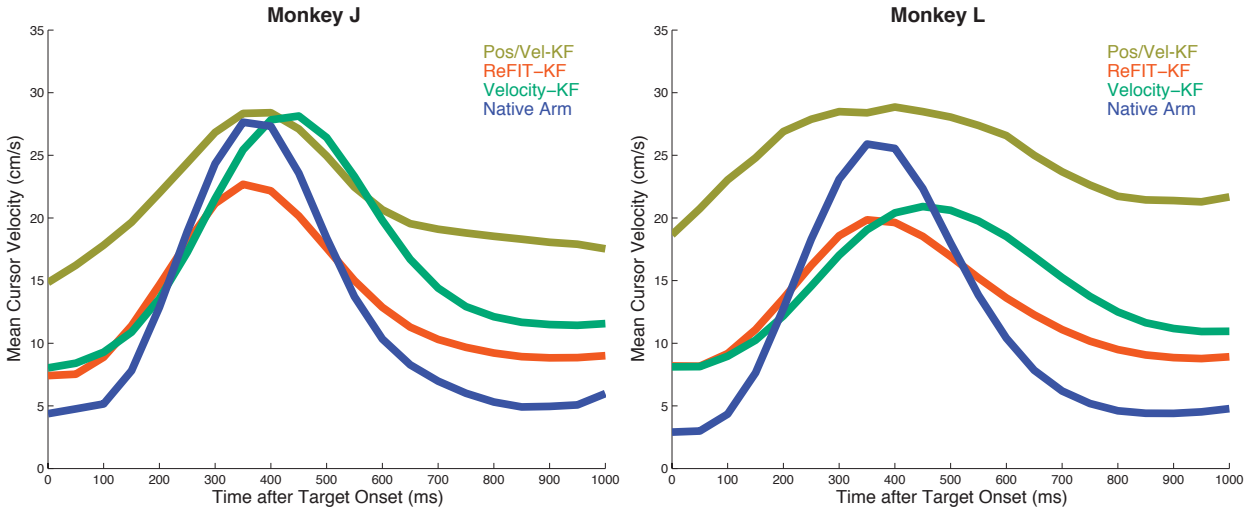
Supplementary Figure 4: Innovation Contribution Breakdown (Monkey J). Here we show the relative contributions to performance that each innovation makes. We tested algorithms in succession, switching between them on the same day against identical trial conditions. Observed differences in performance between trial blocks while holding both behavioral task and neural recording conditions constant can be attributed primarily to differences in the control algorithm. (a) shows Monkey J's performance with Velocity-KF (green) compared against the Kalman filter with only the first innovation (yellow) (J-2010-08-20) and (b) shows the Velocity-KF with the first innovation (yellow) compared against the ReFIT-KF (both innovations, red) (J-2010-01-20). The task conditions for these trial blocks were a randomized center-out and back chain of 8 targets, with a demand box of 5cm for (a) and 4cm for (b), allowing up to 3 seconds to acquire the target. Note that each innovation reduced the average acquire time. In some instances, the overall success rate also increased. Given that these tasks have a maximum acquire time (typically 2.5 seconds), improved performance is marked by higher success rates and/or lower acquire times.



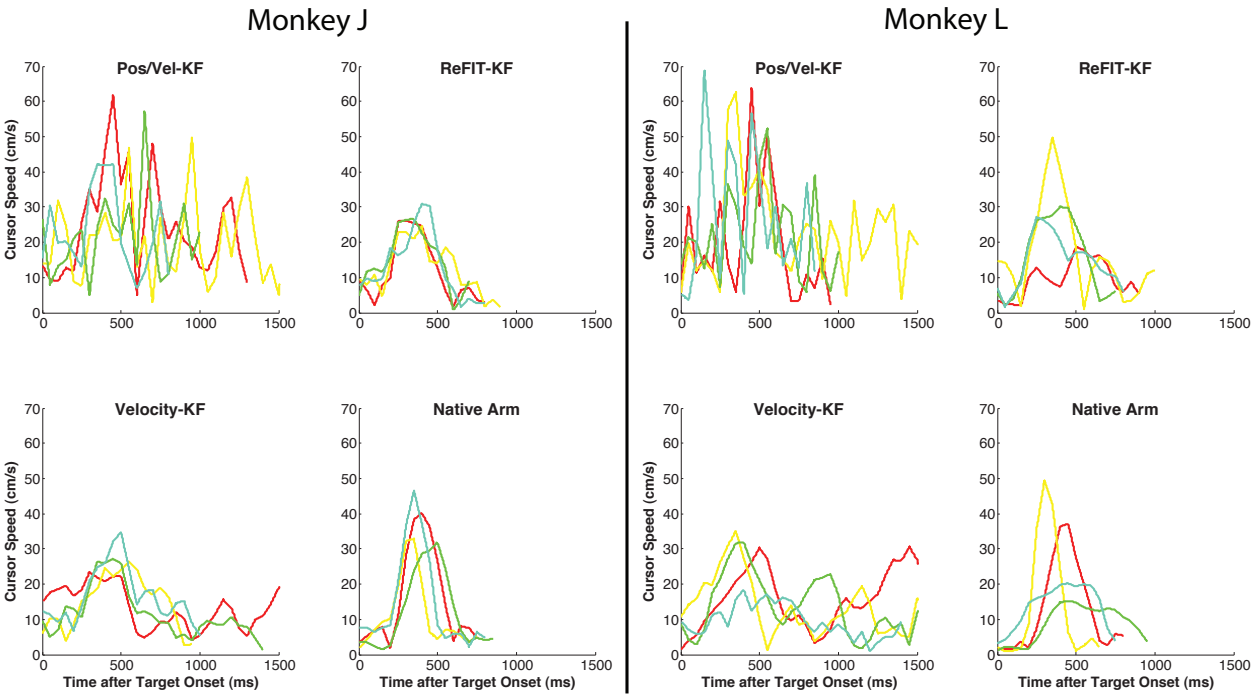
Supplementary Figure 5: Contribution Breakdown for Components of Innovation 1. We can further dissect innovation 1 into two procedures applied to the training data. The first is the rotation of the velocity vectors towards the target and the second is zeroing velocity when the cursor is on target. The figure shows the effect of zeroing velocity for both monkeys (L-2010-01-27 & J-2010-01-26). These plots follow the convention of Figure 1c of the main text. The initial thin line is the mean distance to the target as the cursor approaches the target, the thick line is the mean distance after the monkey has initially acquired the target and is attempting to "dial-in" and stop on the target. Zeroing velocity has little effect on the time taken to initially acquire the target, but substantially decreases the time required to stop on the target. Note, innovation 2 has not been applied in these online sessions.



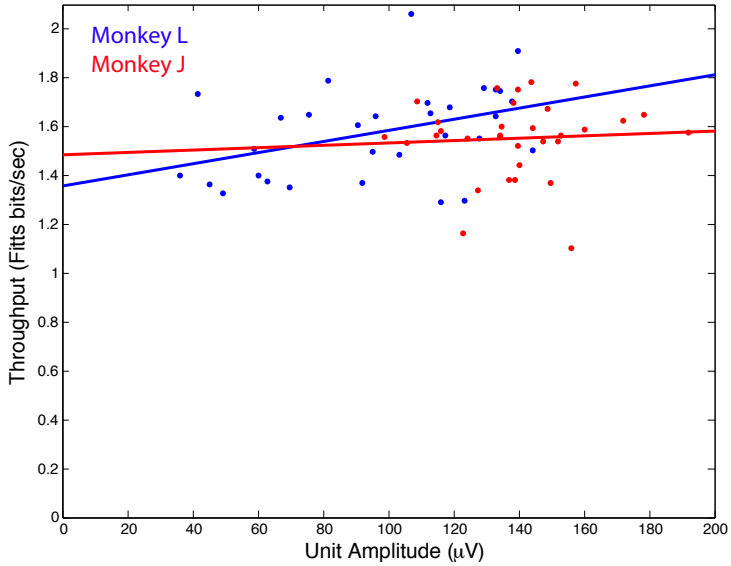
Supplementary Figure 6: Performance Comparisons to Position/Velocity-KF Based Control. The top row is histograms of time to target for successful trials are shown as line graphs. The inset bar graphs plot the time to target (mean ± SE). The bottom row is mean distance to the target as a function of time. Thickened portion of the plotted lines indicate dial-in time, beginning at the mean time of first target acquire, and ending at mean final target acquisition time. These data are from four experiment sessions with monkey J (J-2010-10-27, J-2010-10-28, J-2010-10-29, J-2010-11-02) and three experiment sessions with monkey L (L-2010-10-28, L-2010-10-29, L-2010-11-02). For each of these sessions, data were collected for all four control modes. The task parameters were identical to those used in the experiments presented in Fig. 1 of the main text, facilitating direct comparison to the data presented in the main text. We note that performance was comparable for both the Velocity-KF and the Pos/Vel-KF on the center-out-and-back task with respect to both acquisition and dial-in time. On the pinball task, Fig. 3 of the main text, Pos/Vel-KF performance was low, with a success rate of 42%. As with the Velocity-KF it was difficult to keep the monkey engaged in the task.



Supplementary Figure 7: Average Velocity vs. Time for Four Control Modes. Average cursor velocity is plotted as a function of elapsed time from target onset. Velocity is calculated at 100 ms intervals and interpolated to a 10 ms sampling interval. Each control type is plotted up to a time point for which at least 300 trials are in the dataset. These data are from experiments J-2010-10-27, J-2010-10-28, J-2010-10-29, J-2010-11-02, L-2010-10-28, L-2010-10-29, and L-2010-11-02. Note that the average ReFIT-KF profile is more local in time than Pos/Vel-KF. The peak velocities (mean ± standard deviation) of the native arm, ReFIT-KF, Velocity-KF, and Pos/Vel-KF are 38.4 ± 10.0 (35.5 ± 9.2), 29.2 ± 5.6 (27.0 ± 8.4), 35.7 ± 7.3 (28.8 ± 7.2), 49.1 ± 10.5 (56.9 ± 14.9) cm/s for monkey J (monkey L). Although the peak velocities were higher for Pos/Vel-KF and Velocity-KF than ReFIT-KF, ReFIT-KF control resulted in faster target acquisitions.



Supplementary Figure 8: Example Single Trial Velocities for Four Control Modes. Trials were selected randomly from experiments J-2010-10-27, J-2010-10-28, J-2010-10-29, J-2010-11-02, L-2010-10-28, L-2010-10-29, and L-2010-11-02. Velocity was calculated in 50 ms intervals and traces were color coded to correspond to a different reach direction (left, right, top, bottom).



Supplementary Figure 9: Correlation Between Performance and Waveform Amplitude. Scatter plot of average action potential amplitudes from our previous study [3] versus online performance in this study. Each point in the scatter plot represents one experimental session. We previously published a study that analyzes datasets from a total of 382 days across four electrode arrays implanted in three different monkeys [3]. The results suggest that decoding performance, when using threshold crossings, is not strongly correlated with measures of signal quality, including action potential amplitude. Two of the implants analyzed were used in this study and a subset of the experiments shown in Figure 2 of the main text correspond to data analyzed in this prior longevity study of offline performance. For monkey L, 31 experimental sessions were tested in both studies and are plotted. These sessions were run 2.02 to 2.61 years post implantation. The linear regression of these data is Throughput $= 1.35 + 0.0023 \times$ (Unit Amplitude in μV). The slope of the regression and the intercept is statistically significant from zero ($p<0.026$). For monkey J, 32 experimental sessions were tested in both studies and are plotted. These sessions were run 0.43 to 0.87 years post implantation. The linear regression of these data is Throughput $= 1.49 + 0.00045 \times$ (Unit Amplitude in μV). The slope of the regression is not statistically significant from zero ($p>0.74$). Consistent with [3], these data suggest that a correlation between peak waveform amplitude and online performance is present, but that this correlation is weak (R^2 values are 0.16 and 0.0036 for monkey L and monkey J, respectively). Although a correlation was present for monkey L in the period analyzed, decline was not present across the years of online performance measured in this study.

Supplementary Tables

Dataset	Native arm	ReFIT-KF	Velocity-KF
2010-10-27 (Monkey J)	99.7%	100%	97.8%
2010-10-28 (Monkey J)	100%	100%	99.0%
2010-10-29 (Monkey J)	100%	100%	99.3%
2010-11-02 (Monkey J)	99.3%	100%	99.7%
2010-10-27 (Monkey L)	100%	100%	92.3%
2010-10-28 (Monkey L)	100%	100%	67.5%
2010-10-29 (Monkey L)	100%	100%	99.4%
2010-11-02 (Monkey L)	100%	100%	94.7%

Supplementary Table 1: Success Rates for Center-Out-and-Back Task. Calculated for all datasets used to generate Figure 1 of the main text.

	Monkey L	Monkey J
Number of sessions	182	98
Avg. success rate (% ± s.d.)	96.6±4.3	95.7±4.4
Avg. throughput (bits/s ± s.d.)	1.69±0.26	1.60±0.22
Linear regression intercept (bits/s)	1.16	1.55
Linear regression slope (bits/s/year)	0.17	0.052
Linear regression slope p-value	<0.001	>0.43

Supplementary Table 2: Summary of ReFIT-KF Performance Across Years.

Supplementary Modeling

3.1 Online Performance Measurement & Comparison

3.1.1 Fitts' Law Performance Metric

Since reach distance and target diameters vary across experiments, we apply the Fitts Law derived index of difficulty to provide a summary statistic for comparisons within this study and across studies. This metric has been suggested as a method for standardized assessment of neural prostheses [4]. Briefly, index of difficulty provides a metric in bits based on target size and distance. From this metric we can calculate the throughput as Fitts bits/sec based upon target selection rate. This calculated bits/sec has been shown for a variety of computer input devices to be invariant over a larger range of target sizes and distances [5].

Here we measure index of difficulty and throughput as:

$$\text{Index Of Difficulty} = log_2 \frac{\text{Distance} + \text{Window}}{\text{Window}} \tag{3.1}$$

$$\text{Throughput} = \frac{\text{Index Of Difficulty}}{\text{Acquire Time}} \tag{3.2}$$

Note that here we use a one dimensional index of difficulty metric. Although a few two-dimensional derived Fitts metrics exist in the literature [6], none have been standardized or universally accepted. ISO 9241-9 details performance requirements for non-keyboard input devices and utilizes the one-dimensional Fitts calculation as the measure of throughput as we have in this study. Furthermore, given that some of the neural prosthetic tasks used in the papers compared below do not require dwelling on target or an explicit click, such measures are not valid. Success is essentially marked in such tasks by crossing the target boundary, as in a standard one-dimensional Fitts' law task.

3.1.2 Comparison to Other Studies

Study	**Target center distance (mm)**	**Window size (mm)**	**Acquire time (sec)**	**Index of difficulty (bits)**	**Fitts bits/sec**
Native Arm (Monkey L)	80	60	0.48	0.87	1.82
Native Arm (Monkey J)	80	50	0.54	1.07	1.98
ReFIT-KF (Monkey L)	80	60	0.59	0.87	**1.48**
ReFIT-KF (Monkey J)	80	50	0.59	1.07	**1.81**
Velocity-KF (Monkey L)	80	60	1.36	0.87	0.64
Velocity-KF (Monkey J)	80	50	1.56	1.07	0.69
Ganguly et al., 2009 [7]	70	15	2.5	2.37	0.95
Kim et al., 2008 [1]	210 (pixels) 255,300 (pixels)	120 (pixels) 40	2.47 5.51	1.17 2.89	0.47 0.52
Taylor et al., 2002 [8]	87	70	1.5	0.80	0.53

Table 3.1: Performance comparison between studies employing center-out target acquisition tasks with hold times greater than 250 ms and a free running neural control algorithm (no assistance, such as automatic cursor recentering). Data from the current study (first 6 rows) are from experiments J-2010-10-27, J-2010-10-28, J-2010-10-29, J-2010-11-02, L-2010-10-27, L-2010-10-28, L-2010-10-29, and L-2010-11-02.

Each neural prosthetic study presented in Table 3.1 uses a variant of the basic target acquisition task (i.e., unless noted, the center-out task and not the pinball or obstacle avoidance task).

The data from Figure 1 of the main text are summarized in Table 3.1. The target sizes were larger (and consequently the index of difficulty lower) than those typically tested with the ReFIT-KF algorithm. Target sizes were selected to permit a high success rate with Velocity-KF control, to allow a direct comparison of acquire times between these control modes. All three control modes were tested on each of the eight experimental days analyzed

to generate the data in the table. Data presented in Figure 2 and Table 1 of the main text show Fitts' Law performance for smaller targets. Although the index of difficulty is higher for these examples, the throughput is comparable, which is expected as the Fitts' Law throughput metric is meant to normalize these task differences.

In this study, all behavioral tasks described in the main text require the neural cursor to acquire the target and stay within the demand box for 500 ms. This, in turn, requires a tradeoff between the swiftness of cursor movement and stopping ability. In studies with behavioral tasks in which no hold-time is required or enforced, there is no such trade off, and cursors could be made to move rapidly, hitting the target without the ability to stop or hold on the target location. Stopping ability is a critical differentiator between the three control modes presented in this study. The dial-in time metric in Figure 1c of the main text demonstrates this by measuring the difference between the time it takes to get out to the target and the time it takes to make the final target acquisition before holding for 500 ms. Both native arm and ReFIT-KF cursor control require much shorter dial-in times than Velocity-KF control, and also achieve more precise cursor stops and holds at the target location.

Study	**Target center distance (mm)**	**Window size (mm)**	**Acquire time (sec)**	**Index of difficulty (bits)**	**Fitts bits/sec**
Current Study*	60	20	0.60	1.81	3.01
Suminski 2010 [2]#	55	15	1.1	2.06	1.87
Fraser et al., 2009 [9]*	85 79	32 36	0.81 0.63	1.66 1.43	2.05 2.27
Chase et al., 2009* [10]	85	32	1.04	1.66	1.60
Mulliken et al., 2008 [11]	12.85° (visual angle)	9° (visual angle)	0.88	0.95	1.08
Serruya et al., 2002 [12]#	7.8° (visual angle)	2.4° (visual angle)	0.90	1.91	2.12

Table 3.2: Performance comparisons between studies on target acquisition tasks with hold time shorter than 250 ms. Studies marked with * use automatic recentering of the cursor. All tasks are center out, except for those marked with #, these are pinball tasks. For the pinball tasks, the average distance between successive targets was approximated based on the specified workspace size.

Another important subtlety is that we used a free-running neural cursor which is initialized once and is then controlled by solely neural activity. Some studies in the literature

reinitialize the neural cursor to the center of the screen at the beginning of every trial, possibly simplifying control. Noting the importance of this feature and the hold time requirement, we have constructed Table 3.2. Table 3.1 studies have a task design that matches the current study and Table 3.2 studies have recentering and/or no hold time requirement. Also, we performed a study using ReFIT-KF with automatic recentering and no hold period (not discussed in the main text) to aid in comparison (Table 3.2, top row).

For Tables 3.1 and 3.2, we list the distance to the center of the target, as is typically done in neural prostheses papers. To calculate the distance to target we use:

$$\text{Distance} = (\text{Target center distance}) - \frac{\text{Window}}{2} \tag{3.3}$$

3.1.2.1 Individual Study Behavioral Task Details

Although we make an effort here to compare results across studies, it should be emphasized that a precise quantitative comparison between studies is not possible. The studies in tables 3.1 and 3.2 have many differences that we cannot account for, including: laboratory setups, research subjects (possibly different species), implantation location, implant technology, surgical techniques, and prior behavioral training. These tables are a best effort at comparison, presented to provide intuition, but should be viewed with the above limitations in mind. Not all studies included explicitly list the statistics necessary for the Fitts calculation, so we describe how these data were estimated:

Ganguly et al., 2009 [7]: The cursor radius, target distance, and target radius are specified in the main text. We assume that the cursor radius does not affect the task difficulty, that the center of the cursor must be on target. In multiple sections of the main text a 2.5 s acquire time is mentioned.

Kim et al., 2008 [1]: The methods section defines two tasks with different levels of difficulty. In the results section, there are two tables that specify movement times for each task, respectively. From each table we take the lowest mean movement time. We subtract 500 ms from this time, as it includes a 500 ms hold period.

Taylor et al., 2002 [8]: Target distance and size for closed loop neural control are defined in the supplement. The time to target was the best reported mean time for peripheral target acquisition in table 2 of the main text. It is important to note that this is a 3D task, which increases the task difficulty in a manner that is not captured by the Fitts calculation defined

in this section. Thus, the calculated performance is likely an underestimate relative to the other studies.

Suminski 2010 [2]: The methods section of the main text describes a 12 cm x 6 cm workspace and 2.25 cm^2 square targets (so 15 mm x 15 mm). They specify that subsequent targets were selected with a uniform distribution. We simulated such target selection with the constraint that subsequent targets must be at least 3 cm apart (so that they do not overlap), and took the mean distance between subsequent targets in this simulation, 55 mm, as the target distance. It is important to note that this pinball task is more difficult than the center-out task.

Fraser et al., 2009 [9]: The task parameters are defined in the methods section of the main text. Acquire times are from the table in the results section.

Chase et al., 2009* [10]: The mean acquire time, cursor radius, and target radius are from personal correspondence with a study author (S. Chase). This correspondence mentioned that the cursor radius and target radius were both 16 mm. The cursor and target had to touch for acquisition (the center of the cursor was not required to be on target as in most of the other studies listed). Thus the effective window size is 32 mm as listed in the table. The target distance was specified in the methods section of the study.

Mulliken et al., 2008 [11]: The study lists a range of distances to target with respect to visual angle, 11°-14.7°; we use the middle of this range, 12.85°. A target size of 9° for brain control is specified in the methods section. In the results section, they mention that with subject training time to target dropped to a median (not mean) of 883 ms.

Serruya et al., 2002 [12]: The study specifies a 14° x 14° workspace with targets appearing at random within this space. A 2.4° window size was assumed by measuring a 1.2° cursor and target radius from figure 1. Based on the description of this figure, it was assumed that target and cursor had to touch, not necessarily overlap, to acquire a target, so cursor and target radii were summed to estimate window size. Figure 1 also plots the median (not mean) acquire time, we estimate this median as 0.9 s. It is important to note that based on the data shown, this median is less than the mean, and so the Fitts score for this study is likely overestimated.

3.1.2.2 Individual Study Neural Implant Descriptions

The studies listed in the tables above use either microwire arrays (MWA) or the Utah microelectrode array (MEA) with channel counts from 64-128. Table 3.3 summarizes the

recording technologies used in these studies. It is important to note that the relationship between channel count and performance is nonlinear, with performance saturating as channel count increases [13, 14]. Additionally, each study uses different methods for channel inclusion and threshold detection and/or spike sorting. As shown in [13], when adding units in order based on a measure of informativeness, maximum performance is achieved with a subset of units. Thus, there is no simple way to normalize performance to account for implant differences.

Study	**Implant type**	**Potential channel count**	**Implant location**
Current Study	MEA	96	Contralateral primary motor and premotor cortex
Ganguly et al., 2009 [7]	2 MWAs	128	Bilateral primary motor cortex
Kim et al., 2008 [1]	MEA	96	Primary motor cortex
Taylor et al., 2002 [8]	MWA	64	Contralateral primary motor cortex
Suminski 2010 [2]	MEA	96	Contralateral primary motor cortex
Fraser et al., 2009 [9]	MEA	96	Contralateral primary motor cortex
Chase et al., 2009 [10]	MEA	96	Contralateral primary motor cortex
Mulliken et al., 2008 [11]	2 MWAs	64	Posterior parietal cortex
Serruya et al., 2002 [12]	MEA	96	Contralateral primary motor cortex

Table 3.3: Summary of implant technologies and locations for the studies listed in Tables 3.1 and 3.2.

3.2 Algorithm Design

In this section we describe the ReFIT-KF control algorithm design. First we discuss the basic Kalman filter algorithm that has been used in previous work for neural decoding. The remainder of the section describes the two algorithm innovations to the Kalman filter that comprise ReFIT-KF and the rationale behind them.

3.2.1 Kalman filter based control algorithm

Many control algorithms, or continuous decoding methods, have been studied for neural prosthetics applications. There are three methods commonly applied online: the population vector (e.g., [8]), the optimal linear filter (e.g., [15, 16]), and the Kalman filter (e.g [1]). The population vector was first suggested by Georgopolous et. al. as a method for decoding intended movement direction [17]. The population vector, as implemented for neural prostheses, can be seen as a special case of a linear filter [10]. In turn, the linear filter can be viewed as a special case of the more general Kalman filter [18]. The Kalman filter, as implemented in their work and in this study, will converge to a recursive linear filter over time. Given this similarity and the effectiveness of the Kalman filter online and in simulation, we chose to base this work on the Kalman filter.

Since its initial description [19] as a method for recursive linear filtering, the Kalman filter has been applied in many engineering disciplines, including aerospace, radio communications, robotics, and computer vision. The basic intended application of this filter is to track the state of a dynamical system throughout time using noisy measurements. Although we have a model of how dynamics evolve through time, the underlying system may not be deterministic. If we know the state of the system perfectly at time t, our dynamical model only gives us an estimate of the system state at time $t+1$. We can use the measurements (or observations) of the system to refine our estimate, and the Kalman filter provides the method by which these sources of information are fused over time. The filter can be presented from a dynamical Bayesian network (DBN) perspective, and is considered to be one of the simplest DBNs.

A graphical model of the basic DBN representation of the Kalman filter is shown in Figure 3.1. For neural prosthetic applications, the system state vector x_t is commonly used to represent the kinematic state. In this study, the state vector represents position and velocity of the cursor ($x_t = [pos_t^{vert}, pos_t^{horiz}, vel_t^{vert}, vel_t^{horiz}, 1]^T$). The constant 1 is added to the vector to allow observations to have a fixed offset (i.e., baseline firing rate). y_t is the measured neural signal, which is binned spike counts. The choice of bin width can affect

Nature Neuroscience: doi:10.1038/nn.3265

the quality of prosthetic control: assuming local stationarity, long bin widths can provide a more accurate picture of neural state but with poorer time resolution. Thus, there is an implicit tradeoff between how quickly the prosthesis can change state and how accurately those states are estimated. Typical bin widths used in studies range from 10 ms to 300 ms. Through online study (see [20] for details), we find that shorter bin widths result in better performance. The results discussed in this study use 50 ms bin widths.

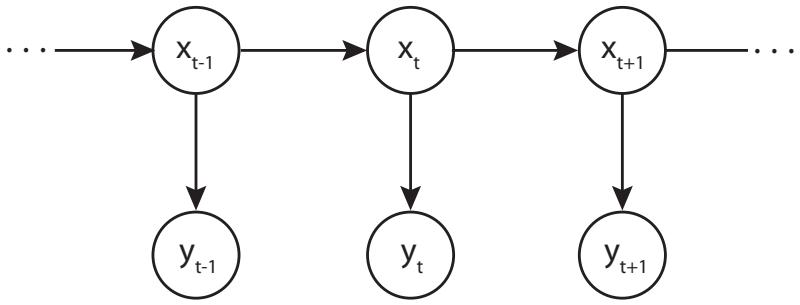


Figure 3.1: **A graphical model representing the assumptions of a Kalman filter.** x_t and y_t are the system state and measurement at time t, respectively.

When applying the standard Kalman filter, the system is modeled with linear dynamics, a linear relationship between kinematic state and neural observations, and Gaussian distributed noise and uncertainty:

$$x_t = Ax_{t-1} + w_t \tag{3.4}$$

$$y_t = Cx_t + q_t \tag{3.5}$$

where $A \in \mathbb{R}^{p \times p}$ and $C \in \mathbb{R}^{k \times p}$ represent the state and observation matrices, and w and q are additive, Gaussian noise sources, defined as $w_t \sim \mathcal{N}(0, W)$ and $q_t \sim \mathcal{N}(0, Q)$. A is the linear transformation from previous kinematic state, or dynamics, and C is mapping from kinematic state to expected observation. This formulation allows for very fast inference (decoding) of kinematics from neural activity and the parameters $\theta = \{A, C, W, Q\}$ can be quickly learned from training data with a closed form solution. The observation model of the Kalman filter, C and Q , is fit by regressing neural activity on observed arm kinematics:

$$C = YX^T(XX^T)^{-1} \tag{3.6}$$
$$Q = \frac{1}{D}(Y - CX)(Y - CX)^T \tag{3.7}$$

where X and Y are the matrices formed by tiling the D total data points x_t and y_t. For the Kalman filter, we also assume that the dynamics of observed arm kinematics match the desired neural cursor kinematics, and so the parameters of the dynamics or trajectory model, A and W, are fit from observed arm kinematics:

$$A = X_2X_1^T(X_1X_1^T)^{-1} \tag{3.8}$$
$$W = \frac{1}{D-1}(X_2 - AX_1)(X_2 - AX_1)^T \tag{3.9}$$

X_1 is all columns of X except for the last column and X_2 is all columns of X except for the first column, introducing a one time-step shift between the two matrices.

In practice we constrain the form of the A and W matrices to obey simple physical kinematics; integrated velocity perfectly explains position:

$$A = \begin{pmatrix} 1 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & dt & 0 \\ 0 & 0 & a_{vel_{horiz},vel_{horiz}} & a_{vel_{horiz},vel_{vert}} & 0 \\ 0 & 0 & a_{vel_{vert},vel_{horiz}} & a_{vel_{vert},vel_{vert}} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

After fitting with either set of kinematics, $a_{vel_{vert},vel_{horiz}}$ and $a_{vel_{horiz},vel_{vert}}$ are typically close to 0 and $a_{vel_{horiz},vel_{horiz}}$ and $a_{vel_{vert},vel_{vert}}$ are less than 1. The resulting model introduces damped velocity dynamics. Therefore, given no neural measurements, we expect a cursor in motion to smoothly slow down. We also constrain the W matrix, so that for the dynamics model, integrated velocity fully explains position:

$$W = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{vel_{horiz},vel_{horiz}} & w_{vel_{horiz},vel_{vert}} & 0 \\ 0 & 0 & w_{vel_{vert},vel_{horiz}} & w_{vel_{vert},vel_{vert}} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.11}$$

If we fit the full C matrix, the resulting filter is a position/velocity Kalman filter (neural firing simultaneously describes position and velocity). If we constrain the position terms to be 0, the resulting filter is a velocity only Kalman filter (neural firing describes only velocity).

Figure 3.2 is a graphical representation of the position/velocity Kalman filter. Note that it differs from the standard Kalman filter presented in Figure 3.1 in two ways. The first is that x_t has been split into two components, p_t for position variables and v_t for velocity variables. The second is that, position variables do not have any direct influence on velocity variables. This representation explicitly states that position does not influence velocity as is also dictated by the described constraints on the A matrix.

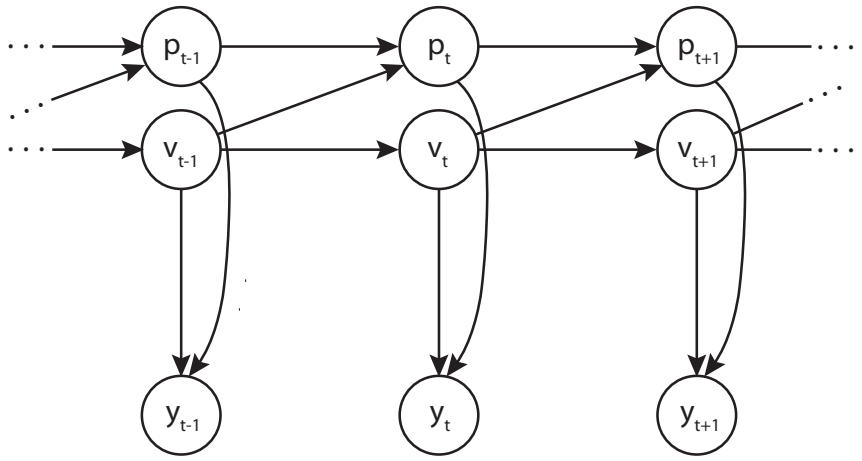


Figure 3.2: A graphical representation of the position/velocity Kalman filter used for neural control.

3.2.2 Innovation 1: Model Fitting

Many existing proof-of-concept neural prosthetics control algorithms are initially designed, tested, and fit offline using data collected without the neural prosthesis in the loop (e.g., [15, 16, 18]). The data are fit against real, observed (i.e., previously recorded arm-movements replayed on a screen that can be observed [21]), or imagined movement (i.e., while viewing automated movement of a cursor) [2, 16]. For example, at the beginning of the session, the movement of the cursor is controlled by the native limb as illustrated in Figure 3.3a. During this task, the kinematics of arm movements (x_t) and neural activity (y_t) are recorded. These data are used to develop the mathematical model used for neural control. The underlying assumption is that observations of neural signals during arm control provide a good estimate of signal characteristics while under brain control (Figure 3.3c).

Kalman filter parameters found to explain arm kinematics from neural observations can be for used brain control. The hypothetical plot in Figure 3.3b shows the relationship between parameter settings and reconstruction quality or control performance suggested by this perspective. Imagine we were to systematically sweep one of the Kalman filter parameters and measure the filter's effectiveness[1]. For arm kinematic reconstruction quality, this is a measure of correspondence between observed and reconstructed arm movements, which can be fully quantified and understood offline. For neural prosthetic control performance, we wish to measure the user's ability to complete task goals during online control. The offline perspective assumes that both applications have the same optimal parameters and so the offline and online measures share the same global maximum, as shown by the black arrow.

It could be that these two maxima are not necessarily aligned, such as in the hypothetical plot in Figure 3.3d. More concretely, a model designed for offline reconstructions may not necessarily translate to a good online controller. Thus, we pursued a different approach and start with the observation that the system can also be fit online, while the user is getting real-time feedback, as in Figure 3.3c. Such a strategy regresses neural activity against the kinematics of the neural cursor (Figure 3.3c), and has been employed previously [1, 22]. One strategy is to randomly seed decoder parameters and to provide assistive control during the training procedure [22]. In this assistive control scheme, the prosthetic output is driven by a mixture of decoder output and task relevant movements, such as precomputed trajectories directly to the target. At each iterative refinement the decoder's contribution is increased, until the prosthesis is fully driven by the decoder. This scheme works well in practice, especially when easing monkeys into performing the task, but the space of possible decode

[1]The Kalman filter parameter space is high dimensional, so a full parameter sweep is not practical, nor something we could easily visualize. Thus, in this hypothetical case we imagine sweeping a single parameter

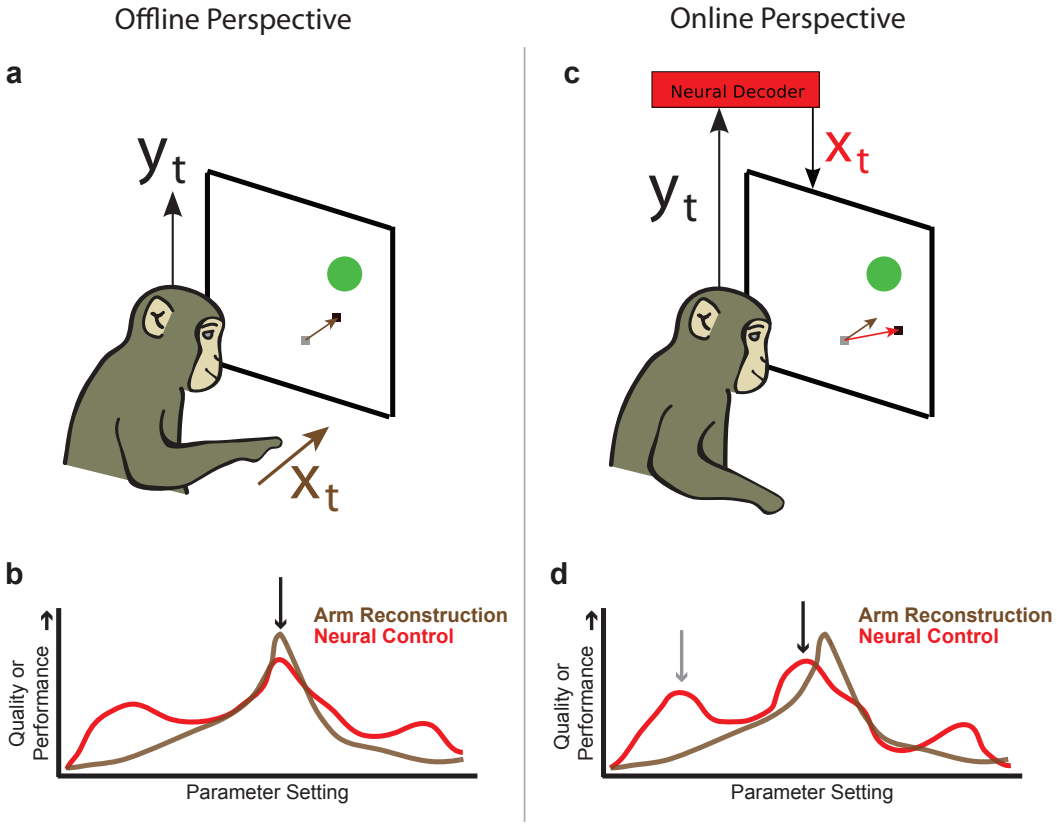


Figure 3.3: **A comparison of the offline and online perspectives for neural prosthetic design.** (a) is a depiction of a monkey controlling a cursor with native arm movements; neural data y_t and arm kinematics x_t are collected to fit the parameters of a neural prosthesis. (b) is a hypothetical plot of parameter setting vs. quality/performance of the resulting system by such a fitting procedure, given the assumptions of the offline perspective, essentially that the maximum for arm reconstructions and neural prosthetic control occur with the same parameter settings. (c) is a depiction of a monkey controlling a cursor via neural control. x_t is no longer arm kinematics; data are collected under closed loop neural control and x_t is derived from the kinematics of the neural cursor. This model fitting procedure assumes that ideal parameter settings for a neural prosthesis and arm reconstructions may vary, as indicated in hypothetical plot (d).

parameters is vast and in principle it is possible to get stuck in local maxima (such as the one pointed to by the gray arrow in Figure 3.3d). When randomly seeding, if the random seed is close to a suboptimal local maximum, performance may be limited as the system is likely to end up in a mode near these initial parameter settings.

Since prosthetic systems typically aim to record from arm related motor areas, it is possible that the global maximum is close to the parameter set fit by the offline perspective (the black arrow in Figure 3.3d). Thus, instead of a random seed, the decoder can be seeded with this reasonable choice of parameters. Previous reports have employed this approach by having the prosthetic user observe movements to establish an initial model fit [1, 2, 22] . Then iterative training procedures fit the model with either the kinematics of an observed [1, 2] or controlled [22] cursor. The kinematics of the observed cursor are subject to the same limitations as arm kinematics: since the control algorithm is not in the feedback loop during this initial observation stage, this model fitting procedure is still fundamentally an offline approach. Regressing against the kinematics of the controlled cursor is, therefore, perhaps a step in the right direction, since it regresses against measurements of the neural control signals during online control. However, this approach will tend to carry forward aspects of model misfit acquired during the initial seeding of decoder parameters. As a simple example, imagine an initial decoder (see Figure 5(b) in main text) that rotates the user's desired velocity by 90 degrees. All measured movements of this cursor will retain this bias and when we refit the prosthesis this bias will remain.

To address the presumed limitations described above, we propose and test a new method for training neural prosthetic parameters, which is the first innovation described in the main manuscript. Initially, the neural prosthetic system is fit from neural data and cursor data, where the cursor moves along with the native arm. Next, the monkey is placed in online brain control mode with this "offline perspective" control algorithm. Training data are collected during brain control and are transformed to estimate the user's intended control command. The details of this transformation are summarized in Figure 3.4. The kinematics of the neurally driven cursor at each time-step may not be the best estimate of the user's intentions. The monkey generates intentions by applying knowledge of the task goal, in this case "acquire the green target," to the current state of the cursor. We make the simple assumption that the monkey intends to generate a velocity oriented towards the target at every time-step, since this is the most direct path to the goal and should lead to most rapid trial completion and reward. Thus, for model training purposes only, we rotate the velocity vector of the neural cursor (in red) to orient towards the goal, resulting in a new set of "intention-based" kinematics (in cyan). Additionally, when the cursor is on target,

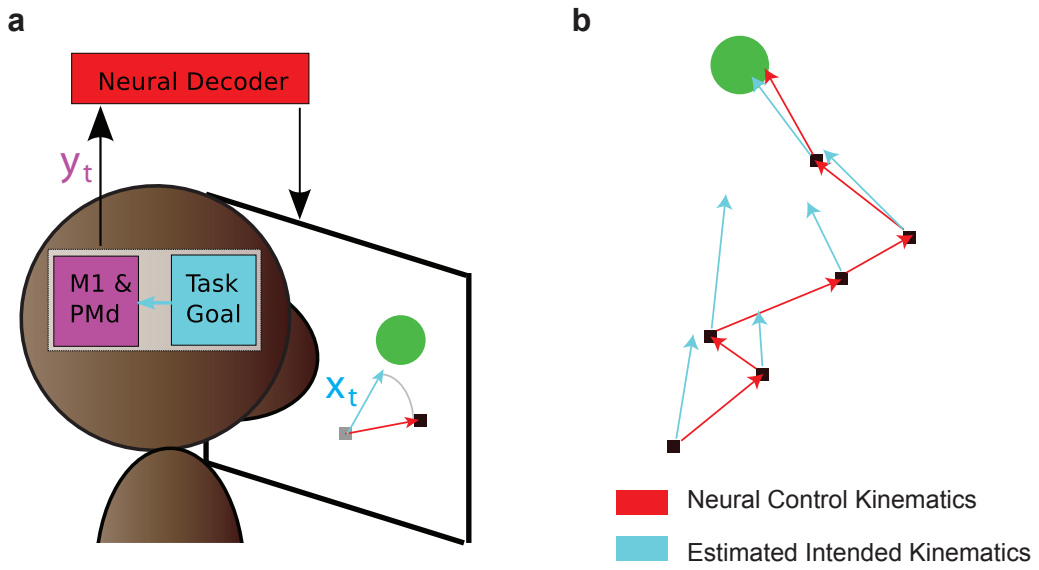


Figure 3.4: **Generating an "intention-based" kinematic training set.** In (a) the user is engaged in online control with a neural cursor. During each moment in the session, the neural decoder drives the cursor with a velocity, shown as a red vector. We assume that the monkey intended the cursor to generate a velocity towards the target in that moment, so following data collection we rotate this vector to generate an estimate of intended velocity, shown as a blue vector. Note that this blue vector is not rendered on the screen as part of the experiment; it is drawn here just to aid in explanation. This new set of kinematics is the training set used to train the ReFIT-KF control algorithm. (b) is an example of this transformation applied to successive cursor updates.

we assume that the user wishes to instruct zero velocity. We believe that this new set of kinematics are a better estimate of the user's intention than the original neural cursor kinematics. Importantly, after refitting the model in this way, the resulting decoder can be used with neural data alone, in the exact same manner as the decoder trained with arm or neural cursor kinematics. A similar manipulation to training data was used in a rat study to adapt one dimensional neural controller over time [23]. The study shows how this approach can be used to continuously update the control algorithm as the user is engaged in the task.

3.2.3 Innovation 2: Filter Design

Existing work typically decodes either position (e.g., [12, 16]) or velocity (e.g., [22]). In a comparison of position and velocity decoders, tetraplegic patients demonstrated a higher performance control with velocity decoders than with position decoders [1] . We find that when position decoding is removed, decoded velocities tend to be less stable. Colloquially put, the cursor appears to get caught in "force fields" resulting in "orbiting" around the target and getting "stuck" in parts of the workspace. This is not surprising, given that firing rates in the recorded brain areas are correlated to cursor position.

Imagine a hypothetical prosthesis that decodes from a single neuron. This neuron fires more vigorously when leftward velocities are instructed and also happens to fire more when the cursor is positioned on the left side of the workspace. If our decoder translates this firing rate into velocities, without any knowledge of positional effects, every time the cursor enters the left side of the screen positive feedback will accelerate the cursor to the left. Positive feedback results because the firing rate increase due to leftward position is mapped to leftward velocity by the decoder, thereby driving the cursor faster to the left than the user intends. By accounting for position, some of the increased firing can be explained by the position of the cursor, and this feedback effect can be mitigated.

However, the way in which the position/velocity Kalman filter (Figure 3.2) models the relationship between position and velocity leads to undesired high frequency jitter in the cursor position. The dynamics model described in the previous section is physically based, acting like an object moving in a gravity free 2-D space with damped velocity, so we may expect cursor position to evolve smoothly. However, the Kalman filter translates velocity uncertainty into position uncertainty at subsequent time-steps. To understand why this occurs, we examine how the filter is run online. At time t we have a previous estimate of the kinematic state, $\hat{x}_{t-1}$ and a new neural observation, y_t. The first step in each filter iteration is to apply the dynamics model, estimating $x_t = [p_t, v_t]$ with all neural observations up to time $t-1$. This is the *a priori* estimate of x_t:

$$\hat{x}_{t|t-1} = A\hat{x}_{t-1} \tag{3.12}$$

The matrix A is the trajectory model, describing how the kinematic state is expected to evolve given no additional information. The model also estimates the *a priori* covariance (or uncertainty) of $\hat{x}_{t|t-1}$:

Nature Neuroscience: doi:10.1038/nn.3265

$$\Sigma_{t|t-1} = A\Sigma_{t-1}A^T + W \tag{3.13}$$

W is a covariance matrix that is the uncertainty introduced by the trajectory model update. We know that the W adds no uncertainty to *a priori* position, given its structure as defined in equation 3.11. However, $A\Sigma_{t-1}A^T$ translates previous velocity uncertainty into current position uncertainty. This makes sense: if we do not know the previous velocity with certainty, we do not know the integrated velocity with certainty and so our position estimate may have error. Thus, in practice, there is uncertainty in the *a priori* estimate of every kinematic variable. To see how this uncertainty in position translates to jitter in the decode, we can continue to step through the algorithm. Once we update the *a priori* estimate, we must incorporate the information acquired from the neural observation. The model has an expected neural output given $\hat{x}_{t|t-1}$, and this output may not match y_t. This error is the measurement residual, $\tilde{y}_t$, and also has a corresponding covariance (or uncertainty) estimate, S_t:

$$\tilde{y}_t = y_t - C\hat{x}_{t|t-1} \tag{3.14}$$

$$S_t = C\Sigma_{t|t-1}C^T + Q \tag{3.15}$$

If this residual is nonzero (which is almost always true in practice), then the measurement and $\hat{x}_{t|t-1}$ do not agree and we must decide how much weight this observation residual has relative to $\hat{x}_{t|t-1}$. This weight is based on how much uncertainty is present in the kinematics suggested by the *a priori* estimate of x_t versus the kinematics suggested by $\tilde{y}_t$:

$$K_t = \Sigma_{t|t-1}C^T S_t^{-1} \tag{3.16}$$

Finally, we can use K_t, called the Kalman gain, to find the estimate of x_t that incorporates all of the neural observations up to time t, this is called the *a posteriori* estimate. We calculate the *a posteriori* estimate for x_t and its covariance:

$$\hat{x}_t = \hat{x}_{t|t-1} + K_t\tilde{y}_t \quad (3.17)$$

$$\Sigma_t = (I - K_tC)\Sigma_{t|t-1} \quad (3.18)$$

The Kalman gain transforms the measurement residual into the kinematic space. Since *a priori* estimates of both position and velocity kinematics have uncertainty and neural measurements have information about position and velocity, the Kalman gain will translate neural measurements into updated *a posteriori* estimates of both position and velocity. For offline trajectory reconstruction, this makes sense, as this coupled position/velocity uncertainty exists throughout time. However, these assumptions breakdown in the online setting, and substantially limit performance.

We must distinguish online and offline use of the Kalman filter. In the online setting, the user is presented with the *a posteriori* estimate of cursor kinematics at every time-step. If we believe that the user sees and internalizes the presentation of the cursor on the screen at each time-step, then the way in which we model *a posteriori* covariance no longer makes sense, as the user accepts the presented position as the current position state. By presenting the decode to the user, we are creating a *causal intervention*, that explicitly sets the value of the kinematic variable. This operation is defined by probability theory and is well described by causal calculus [24] (see also [25, 26]).

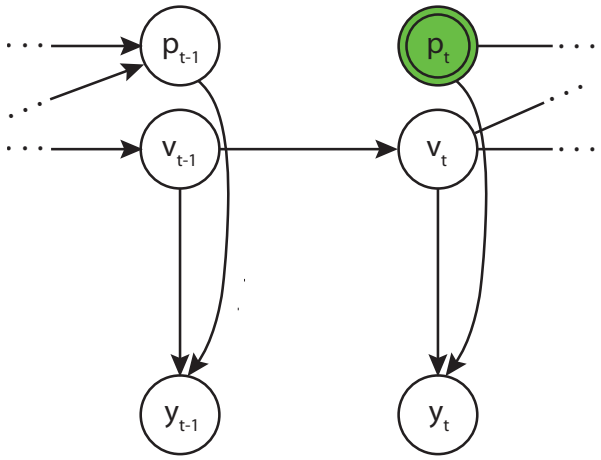


Figure 3.5: **The position/velocity Kalman filter modeling position feedback through causal intervention.** The intervention is indicated by the double circle shown in green.

As a first step to modify the filter to incorporate this feedback, we presume that the

user internalizes the filter's estimate of cursor position, $\hat{p}_t$, with complete certainty at time t. Accordingly, p_t is explicitly set to $\hat{p}_t$, with no uncertainty. We are assuming that the user knows the previous cursor position via feedback and that his forward model is exact. This is shown graphically in Figure 3.5, where the intervened variable is in green (adding another circle is standard notation for causal interventions, see [26]). Note also that the arrows coming into p_t have been removed, to indicate that p_t has been externally set and uncertainty is not propagated.

The result of this intervention is to remove uncertainty in p_t. All parameter fitting methods described in previous sections remain unchanged. To implement this position feedback filter, only a small change in the online operation of the standard filter is necessary. All steps outlined above are the same except for equation 3.13. Previously, we had:

$$\Sigma_{t|t-1} = A\Sigma_t A^T + W \quad \text{where} \quad \Sigma_{t|t-1} = \begin{bmatrix} \Sigma^{p,p}_{t|t-1} & \Sigma^{p,v}_{t|t-1} & 0 \\ \Sigma^{v,p}_{t|t-1} & \Sigma^{v,v}_{t|t-1} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.19}$$

where each block of the matrix $\Sigma_{t|t-1}$ represents the uncertainty propagated from previous kinematic estimates (position to position, position to velocity, and so on). Each one of these sub-matricies of $\Sigma_{t|t-1}$ is 2x2, representing horizontal and vertical components of each kinematic type. The bottom row and right column of zeros encodes the fact that the bias or constant offset term of x_t, the last element of the state vector, is known with certainty. Since we have intervened and set p_t with feedback, this matrix becomes:

$$\Sigma_{t|t-1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & 0 \\ \mathbf{0} & \Sigma^{v,v}_{t|t-1} & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.20}$$

We are zeroing out all *a priori* position uncertainty, as we are explicitly assuming that the monkey and the control algorithm have matching beliefs about the position of the cursor at time t. Otherwise, this filter is run in the same manner as the standard Kalman filter. This modified Kalman filter is used online and, together with Innovation 1 described above, comprise the ReFIT-KF control algorithm.

Nature Neuroscience: doi:10.1038/nn.3265

3.3 Offline Analyses of Decoder Performance

Although, offline decoding results may not be indicative of closed-loop decoding performance [10, 20, 27], they offer a quick and low cost method for exploring and piloting the design of new decoders. Any manipulations made to decoder designs must ultimately be compared in closed loop experiments. In this section we provide an analysis of offline decoding to provide further insight into the design of ReFIT-KF.

3.3.1 Open Loop Trajectory Analysis of Innovation 2

A standard method for assessing the quality of new online decoding methods is to test offline trajectory reconstruction quality of native arm reaches. We can apply this methodology to test innovation 2, which introduces a causal intervention for position estimates. Thus, we test three filters types:

- Velocity Kalman filter
- Position/Velocity Kalman filter
- Velocity Kalman filter with causal position (innovation 2 as described in supplement section 6)

Multiple datasets from each of the two monkeys were used (J-2010-10-27, J-2010-10-28, J-2010-10-29, J-2010-11-02, L-2010-10-27, L-2010-10-28, and L-2010-10-29). For each of these datasets, the three filters were fit to data collected during 150 native arm control trials and were tested against native arm control trials. Neural threshold counts were summed in 50 ms bins for the regression and for the test set. Training set kinematics were calculated in matching 50 ms bins. Training and test sets were from the same day and were partitioned, so no trials were in both the training and test sets. Training and test sets for all three filters were identical, allowing for direct comparison of the resulting metrics. For testing, the Kalman filter is applied to each trial separately, resetting initial kinematics to match the conditions during the online native arm control session and resetting the uncertainty of the prior on kinematics.

In Figure 3.6, we plot metrics for Kalman filter decoded native arm trajectories. All metrics, except mean hold time speed, are calculated from initial cursor movement out to the target, or the time from target onset until the first target acquire, as the monkey's

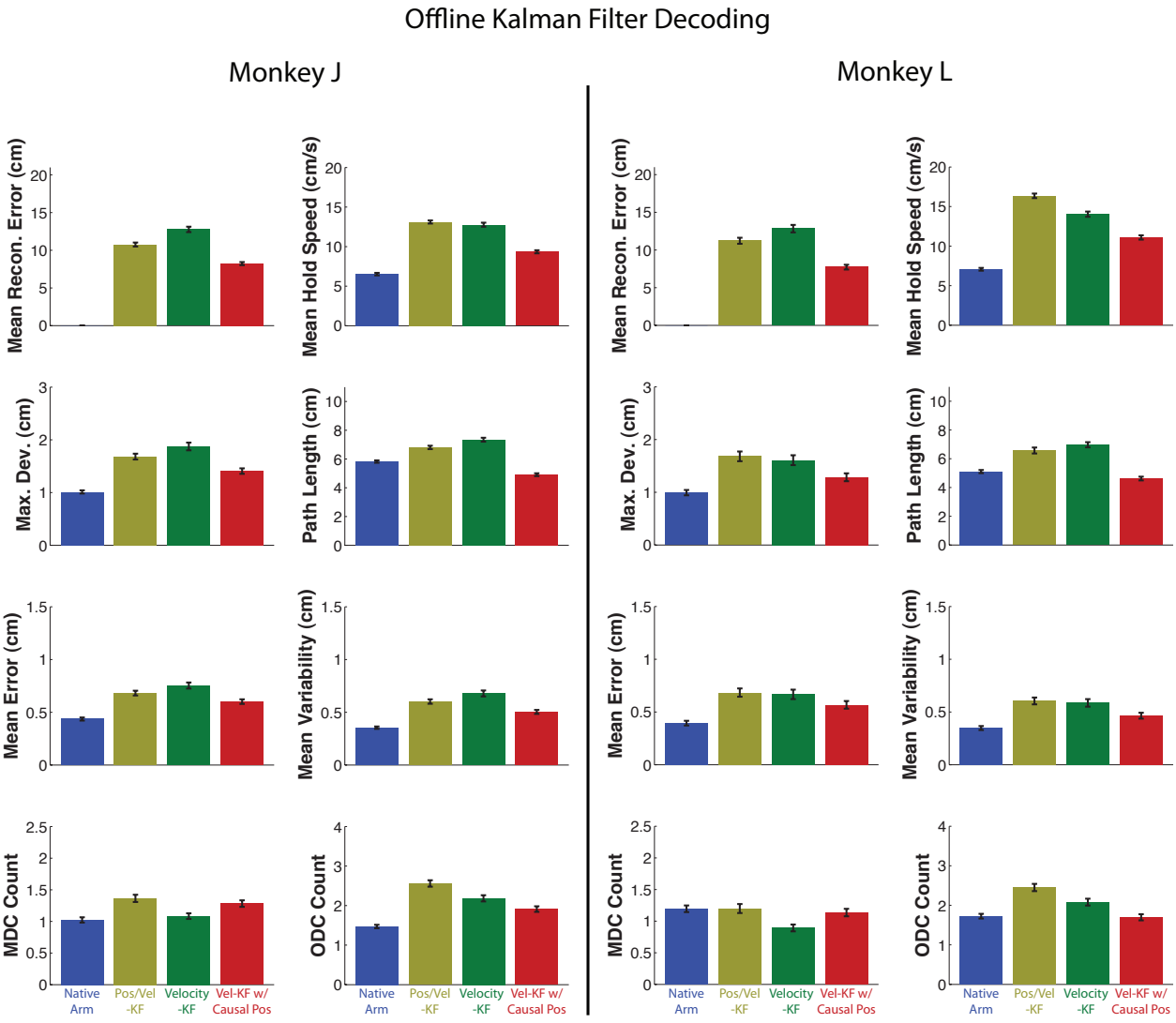


Figure 3.6: **Metrics for Kalman filter decoded native arm trajectories.**

strategy during this epoch is clear. The first row of metrics is reconstruction error relative to native arm trajectory and mean hold time speed[2]. The remaining metrics are identical to those defined in supplement section 12. The inclusion of position information, both in the Pos/Vel-KF and the Vel-KF with causal position, reduces reconstruction error for both monkeys. However, the Pos/Vel-KF increases the jitter of the reconstructed trajectories, note the increase in ODC count and mean hold speed. Vel-KF with causal position outperforms Pos/Vel-KF and Vel-KF on all metrics except MDC.

The increase in jitter for Pos/Vel-KF is expected (as described mathematically in supplement section 6) and is worse during online sessions. For the linear Gaussian model used

[2]Calculated for the last 350 ms of the hold period

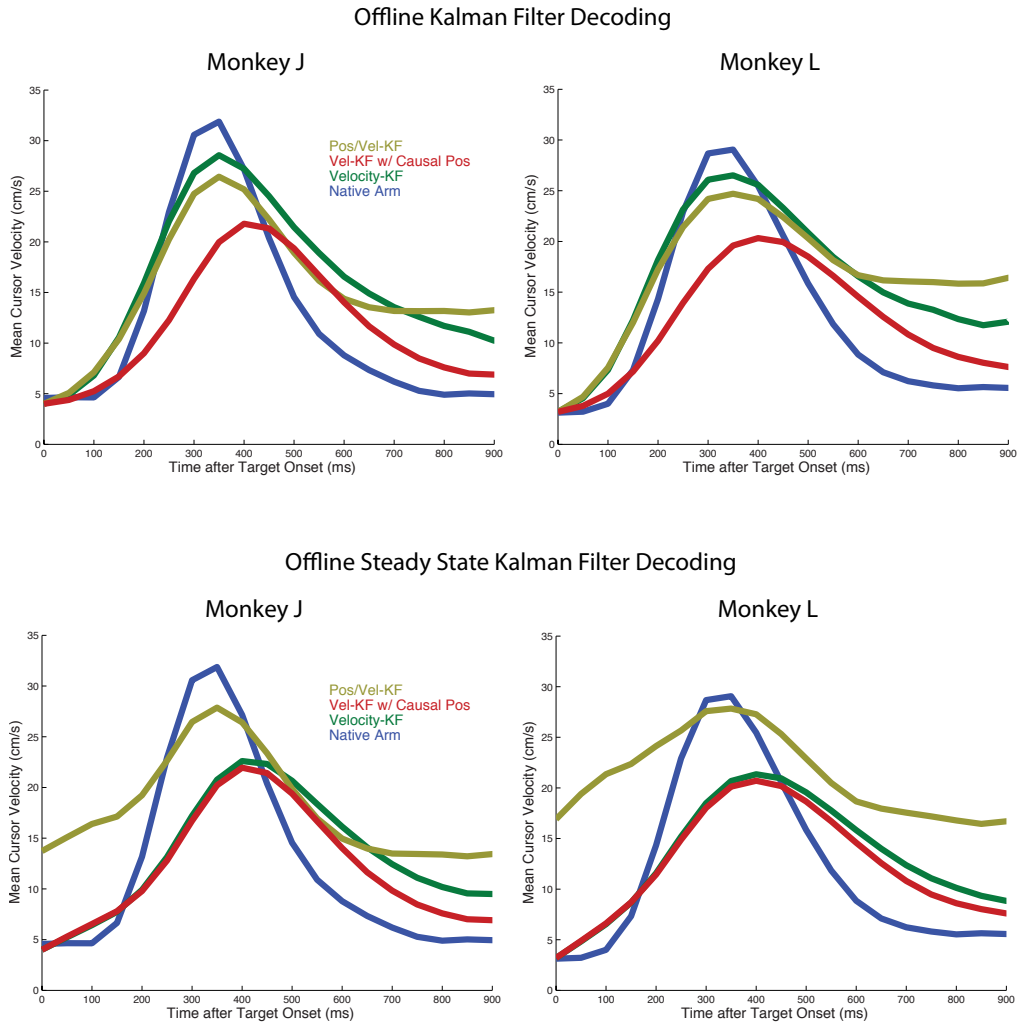


Figure 3.7: **Mean velocity profiles for Kalman filter and steady state Kalman filter decoded native arm trajectories.** All trials are aligned to target onset and are averaged at 50 ms bins. The profiles are smoothed with neighboring bins.

in these studies and described in supplement sections 4-6, position uncertainty starts at zero and increases as the update equation stabilizes. When tested empirically, this stabilization occurs within 10s of iterations or within the first few seconds / trials of online experiments, resulting in a set of steady state equations. We can test the offline performance of each of these three filters in steady state, setting initial kinematics as before, but using the steady state equations to update the trajectory updates at each time step. Offline comparisons made with the steady state versions of all three filter types may be more appropriate for gaining insight into online performance, because the filters stabilize within a few trials and are not reset during the sessions.

If we plot the average velocity profiles, Figure 3.7, for these control modes, and compare the Kalman filter to the steady state Kalman filter, the impact of this difference is apparent. The initial velocity for Pos/Vel-KF is much higher. Metrics for the steady state Kalman filter are plotted in Figure 3.8. Note that the relative performance of Pos/Vel-KF is worse across all metrics. However, the offline steady state Kalman filter analysis also suggest that performance gains in Velocity-KF due to the addition of causal position are smaller (the gap between the green and red bars in Figure 3.8 is smaller than the gap in Figure 3.6). However, it is important to note that comparing offline trajectory reconstruction quality to online performance is not straightforward. Thus, in the next section, we offer an alternative analysis that better correlates with the presented online performance.

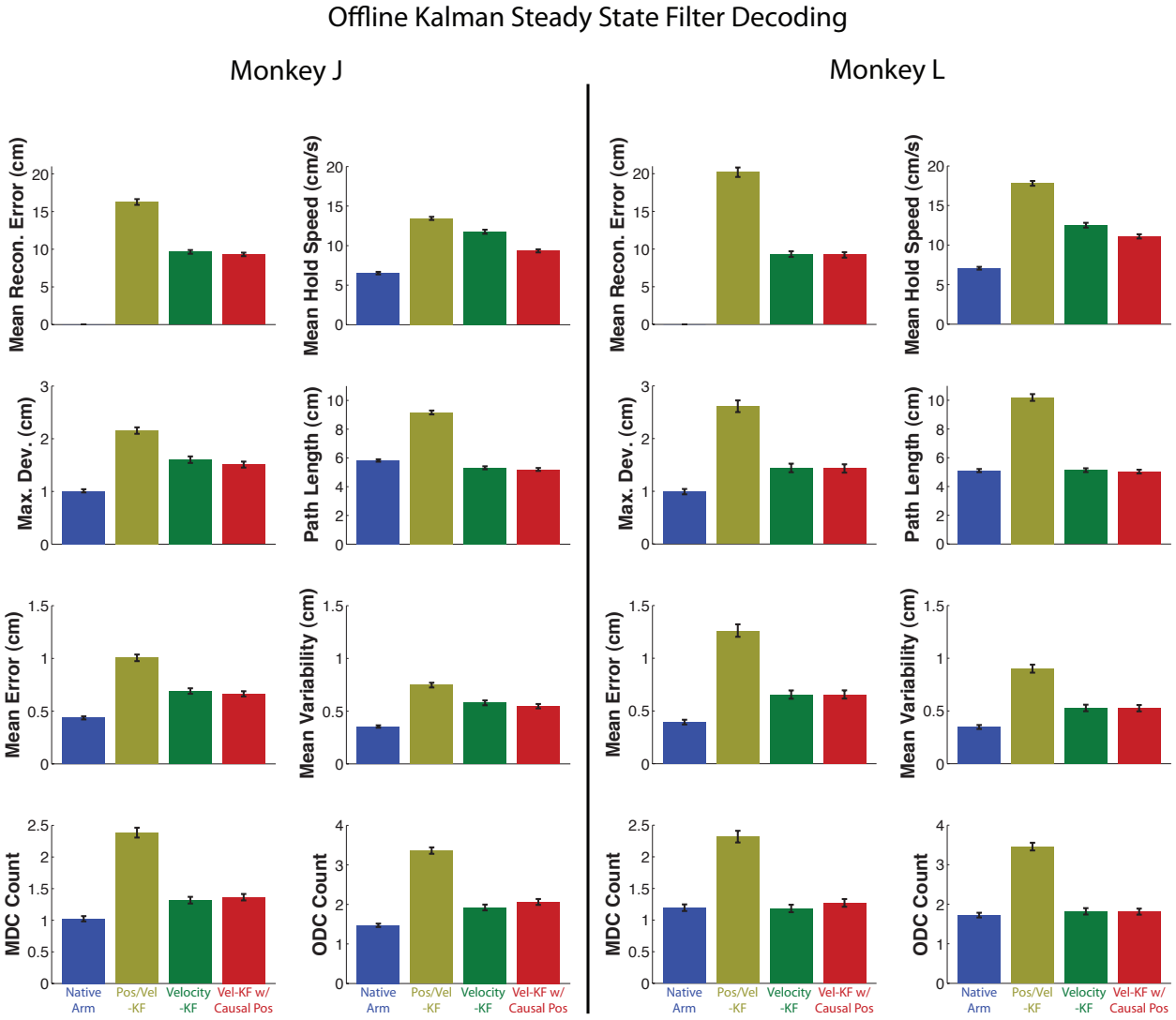


Figure 3.8: **Metrics for steady state Kalman filter decoded native arm trajectories.**

3.3.2 Observation Model Based Analysis

In the previous subsection, both the trajectory and observation models of the Kalman filter were applied to data offline. In this subsection, we focus on the observation model by comparing offline velocity decoding performance for different training set manipulations related to innovation 1 and to different relationships between velocity and position related to innovation 2. In particular, we are interested in the quality of velocity reconstruction for the target acquisition task. We assess this quality using two metrics, one estimating direction decoding accuracy and one measuring relative velocity magnitude between movement and hold epochs.

The metrics in the previous section were focused upon reconstruction of trajectories during native arm control. In this section, we assess the ability of different observation models to decode the velocity of previously recorded online cursor control trials. Thus, new metrics are defined in this section that focus on instantaneous control quality by assessing decoded time bins separately, instead of scoring the quality of entire trajectories.

We analyze offline decoding for two epochs of each trial. The first epoch is from 150 ms after target onset until initial acquisition[3]. During this epoch of the trial, we assume that the monkey is actively attempting to move directly to the target. The second epoch is the final hold period for each trial. During the second epoch, we assume the monkey is attempting to minimize velocity magnitude.

We estimate direction decoding accuracy only during the movement epoch, as the monkey has a clear directional goal. We define angular decoding error as the angle between the decoded velocity and a straight line path from current cursor position to the target. To assess velocity direction, we calculate the angular deviation[28] of this angular decoding error. To assess velocity magnitude, we calculate the ratio of mean speed (or velocity magnitude) across each hold epoch to the mean speed during movement epochs. A lower value indicates better ability to modulate between movement and stopping.

3.3.2.1 Dataset preparation

Datasets from each of the two monkeys were used. For each of these datasets, multiple decoders were tested by fitting models to data collected during either 150 native arm control trials or 75 position/velocity Kalman filter control trials. Half as many neural control trials were used because, on average, neural control trials were twice as long as native arm control

[3]We skip 150 ms from target onset to allow for visual delay.

trials. Using a matched number of trials did not change the reported trends. Neural threshold counts were summed in 50 ms bins for the regression. Training set kinematics were calculated in matching 50 ms bins, by either using the kinematics during the control session or by applying the "intention-based" kinematic transformations defined in supplement section 5. In total, four kinematic transformations were tested:

- Identity Transform: unaltered control session kinematics.
- Vector Rotation: rotate velocity vectors towards the target.
- Magnitude Scaling: scale all velocities during the hold period to zero.
- Vector Rotation & Magnitude Scaling (innovation 1 as described in supplement section 5)

Thus, for each dataset, eight different training sets were tested: four kinematic transformations applied to kinematics from two control methods. The test set was partitioned from the training set and was composed of trials from position/velocity Kalman filter control.

3.3.2.2 Filter building and offline decoding

These training sets were used to fit the observation models of three different filters:

- Velocity Kalman filter
- Position/Velocity Kalman filter
- Velocity Kalman filter with causal position (innovation 2 as described in supplement section 6)

Decodes were calculated as the maximum likelihood estimate of velocity without a kinematics model. To simulate causal position, the cursor position during the control session was used, as is done during online decoding. The maximum likelihood estimator for this linear Gaussian regression problem is often referred to as weighted linear regression and an optimal weighting matrix from neural observations to kinematics has a closed form solution (for a detailed description with respect to brain machine interface, see [10]):

$$K = (C^T Q^{-1} C)^{-1} C^T Q^{-1} \quad (3.21)$$

$$C = Y X^T (X X^T)^{-1} \quad (3.22)$$

$$Q = \frac{1}{D}(Y - CX)(Y - CX)^T \quad (3.23)$$

Note C and Q are fit as described for the Kalman filters used in online experiments. For the velocity Kalman filter, the neural dataset, Y, was regressed against corresponding horizontal and vertical velocities as defined by one of the four kinematic transformations described above. For the Pos/Vel and Causal Pos/Vel Kalman filter, Y was regressed against horizontal and vertical position in addition to the velocities. As before, a column of ones was added to X for all filter regressions to allow for a mean offset in firing rate. Note that K, C, and Q are identical for Pos/Vel and Causal Pos/Vel Kalman filter fits. For each 50 ms bin, intended velocity was estimated:

$$\hat{x}_t^v = K^v(y_t - CFx_t) \quad (3.24)$$

where K^v is the sub-matrix (rows) of K that map to velocity and F is a diagonal matrix with only ones and zeros on the diagonal to select which expected contributions of x_t to subtract off (as Cx_t is the expected firing rate for x_t kinematics). For the Velocity and the Pos/Vel Kalman filters, F is set to only remove the baseline firing rate (the constant one element of x_t). For the Causal Pos/Vel Kalman filter, F is set to also remove the expected contribution of position as presented on screen during the previously recorded online session. Thus, for offline decoding, the Causal Pos/Vel Kalman filter is the only filter that is provided with neural cursor position information from the session. During online sessions, the Pos/Vel Kalman filter does use this information as well, but the filter assumes position information is unreliable and so it is down-weighted.

Training and test sets were always partitioned, so no trials were in both the training and test sets. For each monkey multiple datasets were tested (J-2010-10-27, J-2010-10-28, J-2010-10-29, J-2010-11-02, L-2010-10-27, L-2010-10-28, and L-2010-10-29). The test set was identical for each training set type and filter type, allowing for direct comparison.

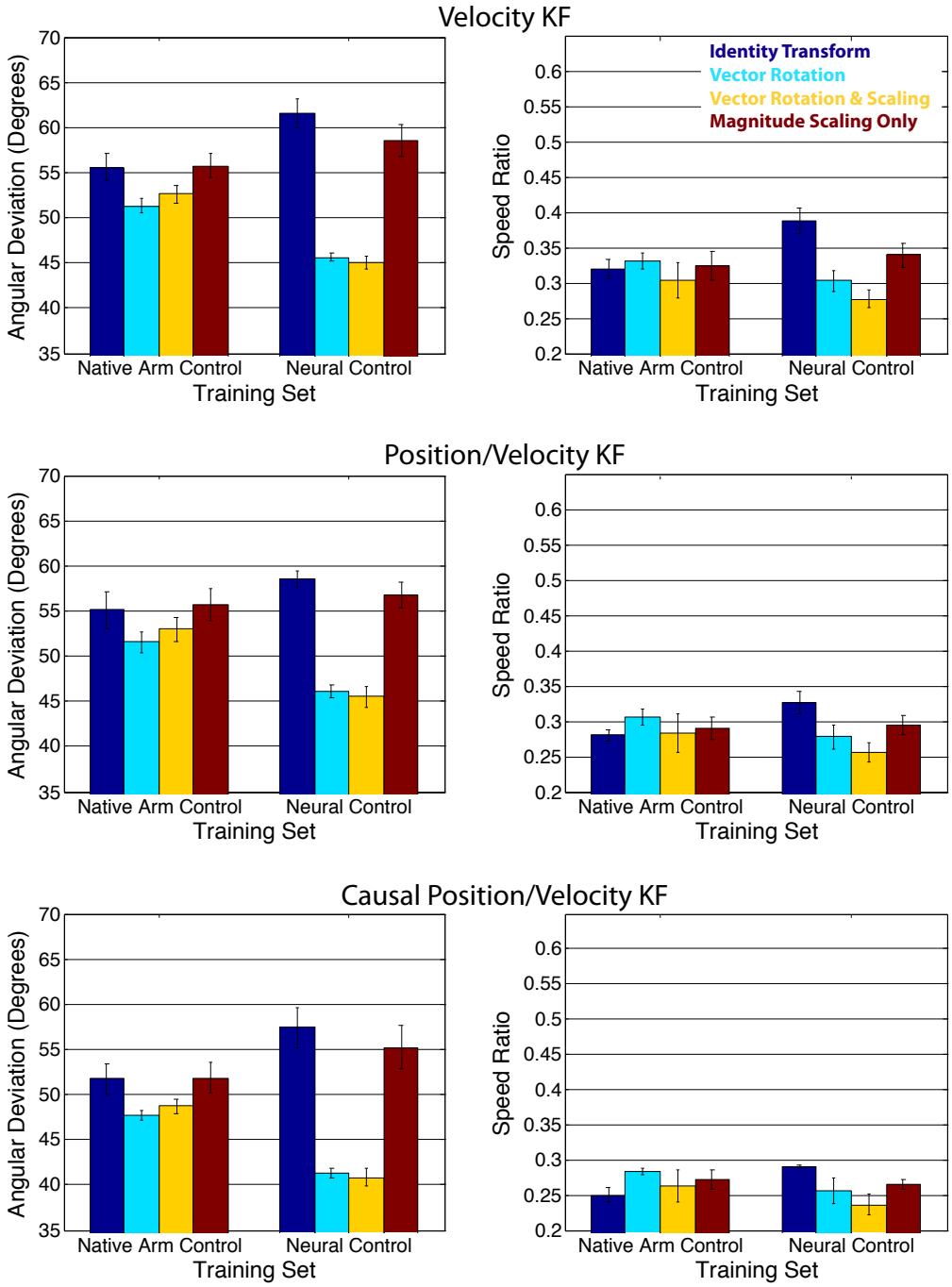


Figure 3.9: Offline decoding metrics for Monkey J for all training set and filter types, error bars indicate standard deviation.

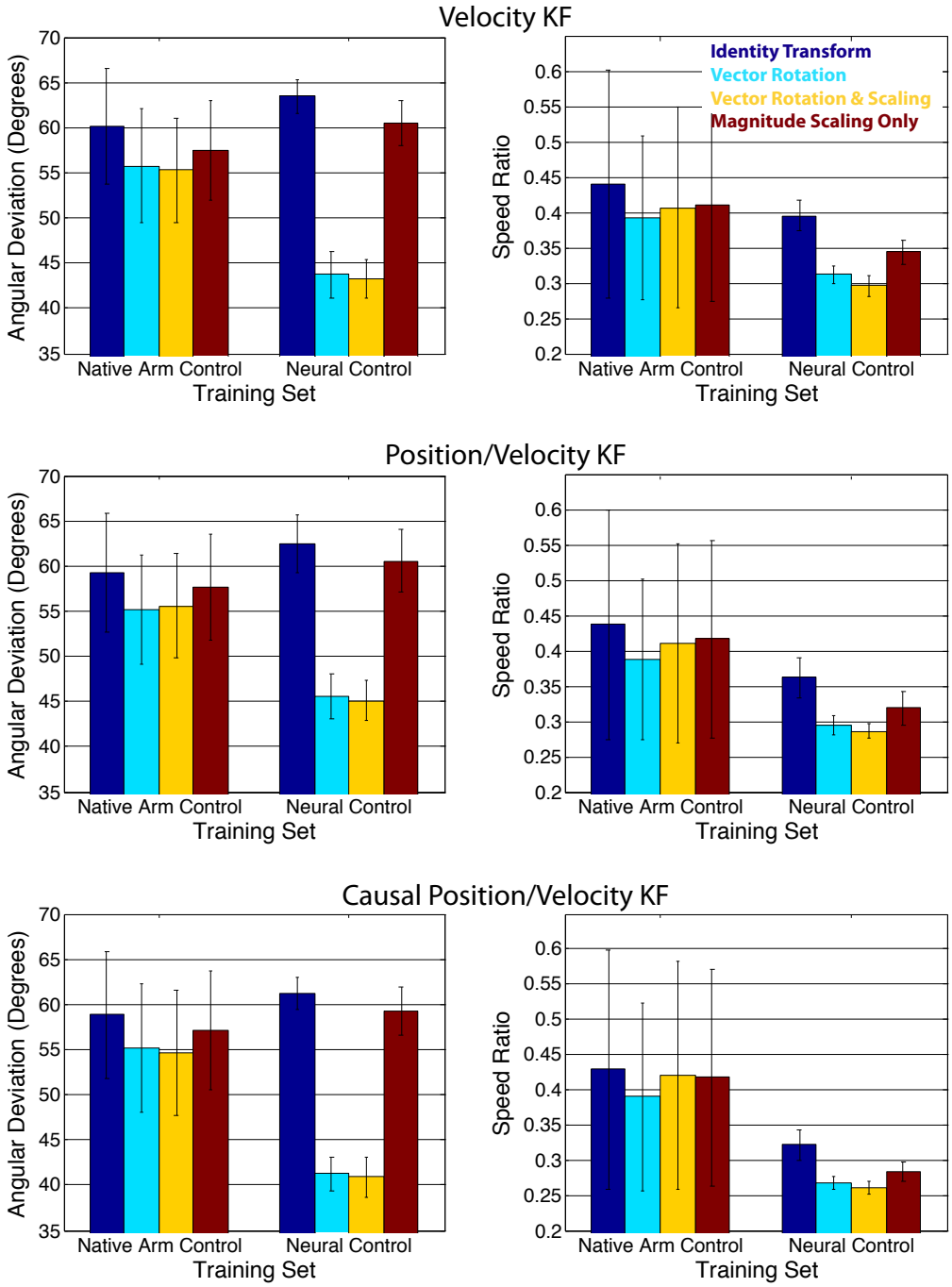


Figure 3.10: Offline decoding metrics for Monkey L for all training set and filter types, error bars indicate standard deviation.

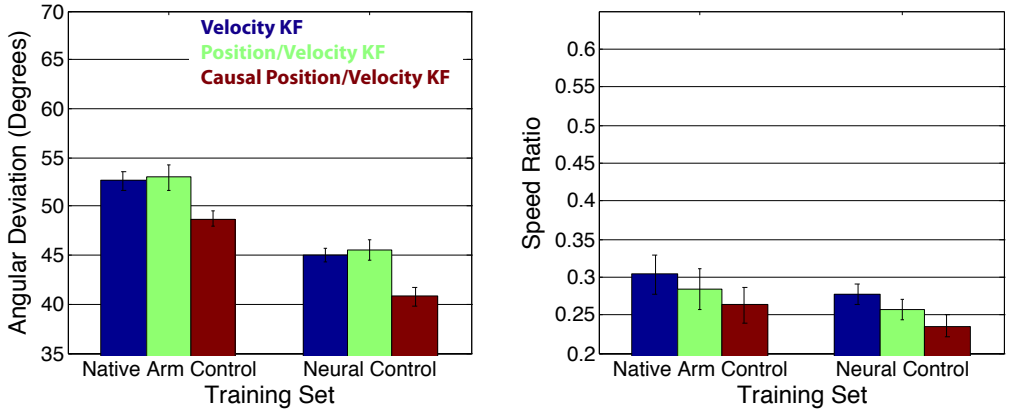


Figure 3.11: Offline decoding metrics for Monkey J comparison across filter types with vector ration and magnitude scaling training sets, error bars indicate standard deviation.

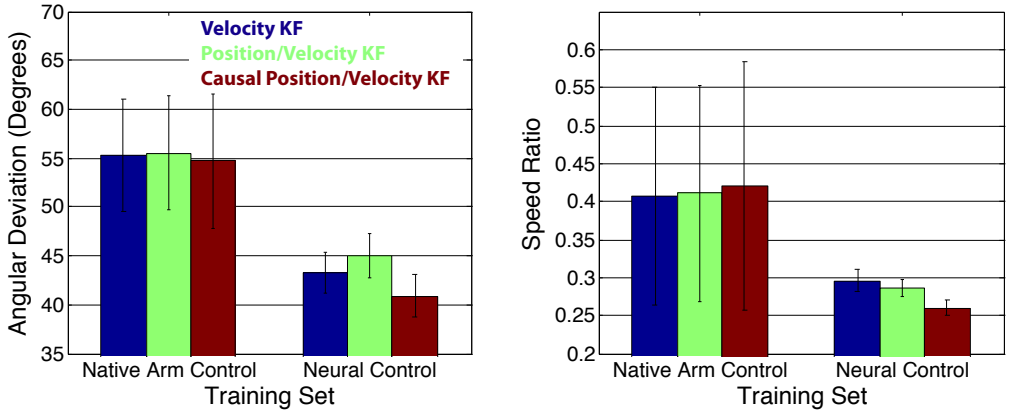


Figure 3.12: Offline decoding metrics for Monkey L comparison across filter types with vector ration and magnitude scaling training sets, error bars indicate standard deviation.

3.3.2.3 Results

Figures 3.9 and 3.10 plot the two offline metrics for both monkeys across all 24 training set conditions. Note that for both measures lower values indicate better performance. Major trends are similar for both monkeys and are consistent with online experiments. Specifically, vector rotation applied to neural control kinematics improves performance and the addition of the magnitude scaling transform results in further improvement. The causal position/velocity Kalman filter performance is better than both the velocity Kalman filter and the position/velocity Kalman filter by these metrics as well. Figures 3.11 and 3.12 compare performance of the three filters when training from kinematics transformed with vector rotation and magnitude scaling (these data are plotted in Figures 3.9 and 3.10, but are reprinted

to facilitate comparison). For monkey L, the standard deviation of both metrics when using native arm control trained decoders was relatively large. This is consistent with the day to day online performance of native arm control trained filters for monkey L, which tended to vary more than neural cursor control trained filters.

Nature Neuroscience: doi:10.1038/nn.3265

References

[1] S. P. Kim, J. D. Simeral, L. R. Hochberg, J. P. Donoghue, and M. J. Black, "Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia," Journal of Neural Engineering, vol. 5, pp. 455–476, Dec 2008.

[2] A. J. Suminski, D. C. Tkach, A. H. Fagg, and N. G. Hatsopoulos, "Incorporating feedback from multiple sensory modalities enhances brain-machine interface control," Journal of Neuroscience, vol. 30, pp. 16777–16787, Dec 2010.

[3] C. A. Chestek, V. Gilja, P. Nuyujukian, J. D. Foster, J. M. Fan, M. T. Kaufman, M. M. Churchland, R.-A. Z, J. P. Cunningham, S. I. Ryu, and K. V. Shenoy, "Long-term stability of neural prosthetic control signals from silicon cortical arrays in rhesus macaque motor cortex," Journal of Neural Engineering, vol. 8, p. 045005, Aug 2011.

[4] J. P. Donoghue, J. D. Simeral, S.-P. Kim, G. M. Friehs, L. R. Hochberg, and M. J. Black, "Toward standardized assessment of pointing devices for brain-computer interfaces," Society for Neuroscience (abstract), 2007.

[5] S. K. Card, W. K. English, and B. J. Burr, "Evaluation of Mouse, Rate-Controlled Isometric Joystick, Step Keys, and Text Keys for Text Selection on a CRT," Ergonomics, vol. 21, no. 8, pp. 601–613, 1978.

[6] I. S. MacKenzie and W. Buxton, "Extending fitts' law to two-dimensional tasks," in Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '92, (New York, NY, USA), pp. 219–226, ACM, 1992.

[7] K. Ganguly and J. M. Carmena, "Emergence of a stable cortical map for neuroprosthetic control," PLoS Biology, vol. 7, Jul 2009.

[8] D. M. Taylor, S. I. Tillery, and A. B. Schwartz, "Direct cortical control of 3d neuroprosthetic devices," Science, vol. 296, pp. 1829–1832, Jun 2002.

[9] G. W. Fraser, S. M. Chase, A. Whitford, and A. B. Schwartz, "Control of a brain-computer interface without spike sorting," Journal of Neural Engineering, vol. 6, pp. 055004–055004, Oct 2009.

[10] S. M. Chase, A. B. Schwartz, and R. E. Kass, "Bias, optimal linear estimation, and the differences between open-loop simulation and closed-loop performance of spiking-based brain-computer interface algorithms," Neural Networks, vol. 22, pp. 1203–1213, Nov 2009.

[11] G. H. Mulliken, S. Musallam, and R. A. Andersen, "Decoding trajectories from posterior parietal cortex ensembles," Journal of Neuroscience, vol. 28, pp. 12913–12926, Nov 2008.

[12] M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue, "Instant neural control of a movement signal," Nature, vol. 416, pp. 141–142, Mar 2002.

[13] J. C. Sanchez, J. M. Carmena, M. A. Lebedev, M. A. L. Nicolelis, J. G. Harris, and J. C. Principe, "Ascertaining the importance of neurons to develop better brain-machine interfaces.," IEEE Transactions on Biomedical Engineering, vol. 51, no. 6, pp. 943–953, 2004.

[14] R. Wahnoun, J. He, and S. I. Tillery, "Selection and parameterization of cortical neurons for neuroprosthetic control," Journal of Neural Engineering, vol. 3, no. 2, p. 162, 2006.

[15] J. M. Carmena, M. A. Lebedev, R. E. Crist, J. E. O'Doherty, D. M. Santucci, D. F. Dimitrov, P. G. Patil, C. S. Henriquez, and M. A. Nicolelis, "Learning to control a brain-machine interface for reaching and grasping by primates," PLoS Biology, vol. 1, Nov 2003.

[16] L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan, A. Branner, D. Chen, R. D. Penn, and J. P. Donoghue, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," Nature, vol. 442, pp. 164–171, Jul 2006.

[17] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner, "Neuronal population coding of movement direction," Science, vol. 233, pp. 1416–1419, Sep 1986.

[18] W. Wu, Y. Gao, E. Bienenstock, J. P. Donoghue, and M. J. Black, "Bayesian population decoding of motor cortical activity using a kalman filter," Neural Computation, vol. 18, pp. 80–118, Jan 2006.

[19] R. E. Kalman, "A new approach to linear filtering and prediction problems," Journal of Basic Engineering, vol. 82, pp. 35–45, 1960.

[20] J. P. Cunningham, P. Nuyujukian, V. Gilja, C. A. Chestek, S. I. Ryu, and K. V. Shenoy, "A closed-loop human simulator for investigating the role of feedback-control in brain-machine interfaces," Journal of Neurophysiology, vol. 105, pp. 1932–1949, Oct 2011.

[21] D. Tkach, J. Reimer, and N. G. Hatsopoulos, "Observation-based learning for brain-machine interfaces," Current Opinion in Neurobiology, vol. 18, pp. 589–594, Dec 2008.

[22] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, "Cortical control of a prosthetic arm for self-feeding," Nature, vol. 453, pp. 1098–1101, Jun 2008.

[23] G. J. Gage, K. A. Ludwig, K. J. Otto, E. L. Ionides, and D. R. Kipke, "Naive coadaptive cortical control," Journal of Neural Engineering, vol. 2, no. 2, p. 52, 2005.

[24] J. Pearl, Causality. Cambridge, UK: Cambridge University Press, 2000.

[25] P. A. Ortega and D. A. Braun, "A bayesian rule for adaptive control based on causal interventions," Proceedings of the Third Conference on Artificial General Intelligence, pp. 121–126, 2010.

[26] D. Koller and N. Friedman, Probabilistic Graphical Models. Cambridge, UK: MIT Press, 2009.

[27] S. Koyama, S. M. Chase, A. S. Whitford, M. Velliste, A. B. Schwartz, and R. E. Kass, "Comparison of brain-computer interface decoding algorithms in open-loop and closed-loop control.," Journal of Computational Neuroscience, vol. 29, pp. 73–87, Aug. 2010.

[28] P. Berens, "Circstat: A matlab toolbox for circular statistics," Journal of Statistical Software, vol. 31, 2009.