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)
Contents

1 Supplementary Figures 4

Supplementary Figure 1: Path Quality Measures for Three Control Modes . . . . 5
Supplementary Figure 2: Average Velocity vs. Time for Three Control Modes . . 6
Supplementary Figure 3: Innovation Contribution Breakdown (Monkey L) . . . . 7
Supplementary Figure 4: Innovation Contribution Breakdown (Monkey J) . . . . 8
Supplementary Figure 5: Contribution Breakdown for Components of Innovation 1 9
Supplementary Figure 6: Performance Comparisons to Position/Velocity-KF Based 10
Control
Supplementary Figure 7: Average Velocity vs. Time for Four Control Modes . . 11
Supplementary Figure 8: Example Single Trial Velocities for Four Control Modes 12
Supplementary Figure 9: Correlation Between Performance and Waveform Amplitude 13

2 Supplementary Tables 14

Supplementary Table 1: Success Rates for Center-Out-and-Back Task . . . . . . 15
Supplementary Table 2: Summary of ReFIT-KF Performance Across Years . . . . 16

3 Supplementary Modeling 17

3.1 Online Performance Measurement & Comparison . . . . . . . . . . . . . . . . . . . 18
3.1.1 Fitts’ Law Performance Metric . . . . . . . . . . . . . . . . . . . . . . . . . . 18
3.1.2 Comparison to Other Studies . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
3.2 Algorithm Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
| Section                                                                 | Page |
|------------------------------------------------------------------------|------|
| 3.2.1 Kalman filter based control algorithm                           | 24   |
| 3.2.2 Innovation 1: Model Fitting                                     | 28   |
| 3.2.3 Innovation 2: Filter Design                                     | 32   |
| 3.3 Offline Analyses of Decoder Performance                           | 36   |
| 3.3.1 Open Loop Trajectory Analysis of Innovation 2                   | 36   |
| 3.3.2 Observation Model Based Analysis                                | 40   |
| References                                                             | 47   |
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 ± standard deviation) of the native arm, ReFIT-KF, and Velocity-KF are 40.1 ± 10.8 (35.5 ± 9.5), 29.6 ± 5.7 (27.0 ± 8.2), 35.7 ± 7.3 (28.6 ± 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 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 × (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 × (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 (R² 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  | 99.7%      | 100%     | 97.8%       |
| (Monkey J)  |            |          |             |
| 2010-10-28  | 100%       | 100%     | 99.0%       |
| (Monkey J)  |            |          |             |
| 2010-10-29  | 100%       | 100%     | 99.3%       |
| (Monkey J)  |            |          |             |
| 2010-11-02  | 99.3%      | 100%     | 99.7%       |
| (Monkey J)  |            |          |             |
| 2010-10-27  | 100%       | 100%     | 92.3%       |
| (Monkey L)  |            |          |             |
| 2010-10-28  | 100%       | 100%     | 67.5%       |
| (Monkey L)  |            |          |             |
| 2010-10-29  | 100%       | 100%     | 99.4%       |
| (Monkey L)  |            |          |             |
| 2010-11-02  | 100%       | 100%     | 94.7%       |
| (Monkey L)  |            |          |             |

**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:

\[
\text{Index Of Difficulty} = \log_2 \frac{\text{Distance} + \text{Window}}{\text{Window}} \quad (3.1)
\]

\[
\text{Throughput} = \frac{\text{Index Of Difficulty}}{\text{Acquire Time}} \quad (3.2)
\]

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)               | 120 (pixels)    | 2.47              | 1.17                       | 0.47           |
| 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                          | 32               | 0.81               | 1.66                       | 2.05           |
| Chase et al., 2009* [10]     | 79                          | 36               | 0.63               | 1.43                       | 2.27           |
| 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:

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

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 cm² 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   | 2 MWAs       | 128                     | Bilateral primary motor cortex                       |
| Kim et al., 2008       | MEA          | 96                      | Primary motor cortex                                |
| Taylor et al., 2002    | MWA          | 64                      | Contralateral primary motor cortex                   |
| Suminski 2010          | MEA          | 96                      | Contralateral primary motor cortex                   |
| Fraser et al., 2009    | MEA          | 96                      | Contralateral primary motor cortex                   |
| Chase et al., 2009     | MEA          | 96                      | Contralateral primary motor cortex                   |
| Mulliken et al., 2008  | 2 MWAs       | 64                      | Posterior parietal cortex                           |
| Serruya et al., 2002   | 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 pros-
theses, 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 imple-
mented 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 ($x_t = [pos_{vert}^t, pos_{horiz}^t, \dot{pos}_{vert}^t, \dot{pos}_{horiz}^t, 1]^T$). 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
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.

\[
\begin{align*}
    x_t &= A x_{t-1} + w_t \\
    y_t &= C x_t + q_t
\end{align*}
\]

(3.4) \hspace{1cm} (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:

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:
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:

\[
A = X_2X_1^T(X_1X_1^T)^{-1} \quad (3.8)
\]

\[
W = \frac{1}{D-1}(X_2 - AX_1)(X_2 - AX_1)^T \quad (3.9)
\]

\( X_1 \) 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:

\[
A = \begin{pmatrix}
1 & 0 & dt & 0 & 0 \\
0 & 1 & 0 & dt & 0 \\
0 & 0 & a_{vel_{horiz},vel_{horiz}} & a_{vel_{horiz},vel_{vert}} & 0 \\
0 & 0 & a_{vel_{vert},vel_{horiz}} & a_{vel_{vert},vel_{vert}} & 0 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix} \quad (3.10)
\]

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:
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.

![Graphical representation of position/velocity Kalman filter](image)

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 used for 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 \textit{a priori} estimate of $x_t$:

$$\hat{x}_{t|t-1} = A \hat{x}_{t-1}$$  \hspace{1cm} (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 \textit{a priori} covariance (or uncertainty) of $\hat{x}_{t|t-1}$:
\[ \Sigma_{t|t-1} = A \Sigma_{t-1} A^T + W \] (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, \(S_t\):

\[ \tilde{y}_t = y_t - C \hat{x}_{t|t-1} \] (3.14)
\[ S_t = C \Sigma_{t|t-1} C^T + Q \] (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 \(\tilde{y}_t\):

\[ K_t = \Sigma_{t|t-1} C^T S_t^{-1} \] (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:
\[ \hat{x}_t = \hat{x}_{t|t-1} + K_t \tilde{y}_t \]  
(3.17)

\[ \Sigma_t = (I - K_tC)\Sigma_{t|t-1} \]  
(3.18)

The Kalman gain transforms the measurement residual into the kinematic space. Since \textit{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 \textit{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 \textit{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 \textit{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 \textit{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 \cite{24} (see also \cite{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.](image)

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:

\[
\Sigma_{t|t-1} = A \Sigma_{t} A^T + W \quad \text{where} \quad \Sigma_{t|t-1} = \begin{bmatrix}
\Sigma_{p,p}^{t|t-1} & \Sigma_{p,v}^{t|t-1} & 0 \\
\Sigma_{v,p}^{t|t-1} & \Sigma_{v,v}^{t|t-1} & 0 \\
0 & 0 & 0
\end{bmatrix}
\] (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-matrices 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:

\[
\Sigma_{t|t-1} = \begin{bmatrix}
0 & 0 & 0 \\
0 & \Sigma_{v,v}^{t|t-1} & 0 \\
0 & 0 & 0
\end{bmatrix}.
\] (3.20)

We are zeroing out all \textit{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.
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
strategy during this epoch is clear. The first row of metrics is reconstruction error relative to native arm trajectory and mean hold time speed\textsuperscript{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

\textsuperscript{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.

Offline Kalman Steady State Filter Decoding

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. 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 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.

---

3We 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]):
\[ K = (C^TQ^{-1}C)^{-1}C^TQ^{-1} \]  
\[ C = YX^T(XX^T)^{-1} \]  
\[ Q = \frac{1}{D}(Y - CX)(Y - CX)^T \]

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:

\[ \hat{x}_t^v = K^v(y_t - CFx_t) \]

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 $C x_t$ 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.
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.
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.