35th Annual International Conference of the IEEE EMBS
Osaka, Japan, 3 - 7 July, 2013

A stabilized dual Kalman filter for adaptive tracking of brain-computer interface decoding parameters

Yin Zhang[1] and Steve M. Chase[2]

Abstract— **Neural prosthetics are a promising technology for alleviating paralysis by actuating devices directly from the intention to move. Typical implementations of these devices require a calibration session to define decoding parameters that map recorded neural activity into movement of the device. However, a major factor limiting the clinical deployment of this technology is stability: with fixed decoding parameters, control of the prosthetic device has been shown to degrade over time. Here we apply a dual estimation procedure to adaptively capture changes in decoding parameters. In simulation, we find that our stabilized dual Kalman filter can run autonomously for hundreds of thousands of trials with little change in performance. Further, when we apply our algorithm off-line to estimate arm trajectories from neural data recorded over five consecutive days, we find that it outperforms a static Kalman filter, even when it is re-calibrated at the beginning of each day.**

I. Introduction

Brain-computer interfaces (BCI) can restore movement to those who are paralyzed by providing behavioral output directly from the intention to move, bypassing defective neural transmission and muscle activation [7], [11], [13]. Most BCIs require a calibration session to compute decoding parameters that define how recorded neural activity will translate into movement of the device. Calibration is necessary to build models of how neural firing rates are modulated by desired movement. However, these calibrations are not particularly stable: decoding parameters estimated in one session will often not apply even on the next day [2]. This instability could be due to electrode drift, changes in background noise, or changes in the neural tuning curves themselves [10]. Regardless of its source, the daily re-training of the decoding algorithm necessary to achieve optimal performance is a major factor limiting the clinical utility of this technology.

Some approaches have previously been proposed to combat BCI decoder instability. In [8], Bayesian regression self-training methods were used to update parameters based on estimates from an unscented Kalman filter. In [12], a kernelized auto-regressive moving average was employed to change the decoder over time. Here we introduce an alternative method, based on a dual-Kalman filter, as a simple extension to commonly used BCI decoders. Dual-estimation can be sensitive to self-training, which appears to also be a problem in our simulations. However, we find that two simple heuristics can be implemented to substantially improve the estimation stability. We find in both simulation and off-line analysis that our stabilized dual Kalman filter remains robust to parameter drift over long time scales.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and SPAWAR System Center Pacific (SSC Pacific) under Contract No. N66001-12-C-4027. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA) and SPAWAR System Center Pacific (SSC Pacific).

We thank Dr. Andrew Schwartz of the University of Pittsburgh for access to the off-line trajectory reconstruction data.

[1]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213

[2]Department of Biomedical Engineering, and Center for the Neural Basis of Cognition, Carnegie Mellon University, Pittsburgh, PA 15213.

II. Decoding Model

A. Kalman Filter: Fixed Decoding Model

Typical BCI decoders are based on the Kalman filter, due to its rigorous theoretical framework and easy implementation [14]. In this framework, limb state is treated as an unobserved vector that evolves over time. We define the state vector at time t, denoted as $\boldsymbol{x}_t$, as the desired end-point velocity of the effector augmented by 1, i.e., $\boldsymbol{x}_t = \left(\boldsymbol{v}_t^T, 1\right)^T$. The state transition from one time-step to the next is assumed to follow a random walk model (Eq. 1). The neural firing rate for neuron i at time t, denoted as $y_t^{(i)}$, is treated as an observation of this state vector (Eq. 2):

$$\text{Evolution:} \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{\omega}_t,\ \boldsymbol{\omega}_t \sim \mathcal{N}(0, W) \qquad (1)$$

$$\text{Observation:} \quad y_t^{(i)} = \boldsymbol{\beta}^{(i)T}\boldsymbol{x}_t + \epsilon_t^{(i)},\ \epsilon_t^{(i)} \sim \mathcal{N}(0, \sigma_\epsilon^2) \qquad (2)$$

where $\boldsymbol{\beta}^{(i)}$ are the linear tuning parameters corresponding to neuron i, and $\boldsymbol{\omega}_t$ and $\epsilon_t^{(i)}$ are zero mean Gaussian noise with covariance matrix W and variance σ_ϵ^2, respectively.

While the Kalman filter has successfully been used to achieve proficient closed-loop BCI control [6], [14], the implicit assumption that the linear model parameters are fixed over time may limit its performance. A common problem in BCI decoding is that neurons are unstable. Micro-movements of the electrode array relative to the brain can cause the amplitude of recorded neurons to change dramatically [5]. Furthermore, there is evidence that neural tuning curves themselves may change over time, slowly altering the relationship between firing rate and intended movement [1], [4], [10]. Traditional Kalman filters will not compensate for this variation.

B. Parameter Tracking Algorithm

To compensate for neural tuning changes, we propose an adaptive decoding model with parameter tracking. In this model, we extract the parameter vector $\boldsymbol{\beta}^{(i)}$ for every neuron i and treat it as a state vector. If we also use a linear dynamical system to model this state vector, then the history of firing rates of neuron i and estimated limb states can be

used to track the evolution of $\boldsymbol{\beta}^{(i)}$. Therefore, for n neurons, the change of their tuning functions over time can be tracked by a set of n Kalman filters running in parallel [9].

We rewrite the Kalman filter equations as follows:

$$\text{Evolution:} \quad \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{\omega}_t,\ \boldsymbol{\omega}_t \sim \mathcal{N}(0, W) \tag{3}$$

$$\text{Observation:} \quad y_t^{(i)} = \boldsymbol{\beta}_\tau^{(i)T} \boldsymbol{x}_t + \epsilon_t^{(i)},\ \epsilon_t^{(i)} \sim \mathcal{N}(0, \sigma_\epsilon^2) \tag{4}$$

In contrast to Eq. 2, in Eq. 4 the linear model parameters $\boldsymbol{\beta}_\tau^{(i)T}$ are now time-varying, as indexed by the subscript τ. We use t and τ to indicate different time scales for limb state change and tuning parameter change.

For parameter evolution, the history of firing rates and the estimated velocity during the last T steps can be used to track the evolution of the parameter vector [9],

$$\text{Evolution:} \quad \boldsymbol{\beta}_{\tau+1}^{(i)} = \boldsymbol{\beta}_\tau^{(i)} + \boldsymbol{\nu}_\tau^{(i)},\ \boldsymbol{\nu}_\tau^{(i)} \sim \mathcal{N}(0, \Pi^{(i)}) \tag{5}$$

$$\text{Observation:} \quad \boldsymbol{y}_{t|T}^{(i)} = \hat{X}_{t|T}^T \boldsymbol{\beta}_\tau^{(i)} + \boldsymbol{\epsilon}_{t|T}^{(i)},\ \boldsymbol{\epsilon}_{t|T}^{(i)} \sim \mathcal{N}(0, \sigma_\epsilon^2 I) \tag{6}$$

where $\boldsymbol{y}_{t|T}^{(i)}$ represents the recorded firing rates during the previous T steps, i.e., $\boldsymbol{y}_{t|T}^{(i)} = \left(y_{t-T+1}^{(i)}, \ldots, y_t^{(i)}\right)^T$, $\hat{X}_{t|T}$ represents the estimated limb state during the previous T steps, i.e., $\hat{X}_{t|T} = \left(\hat{\boldsymbol{x}}_{t-T+1}, \ldots, \hat{\boldsymbol{x}}_t\right)$ and the noise $\boldsymbol{\epsilon}_{\tau|T}^{(i)} = \left(\epsilon_{\tau-T+1}^{(i)}, \ldots, \epsilon_\tau^{(i)}\right)^T$. We assume that parameter change is quite slow compared to the state change, so parameters are updated only every T steps, and are fixed between updates.

C. Heuristics for stabilizing the dual Kalman filter

In simulation, we have observed that tracking parameters and limb state without any regularization tends to lead to long-term parameter drift. To further constrain the tuning parameters, we propose two heuristics for regularization:

1) Baseline firing rate estimation: For a sufficiently long window of time, the average firing rate of each neuron will converge to its baseline firing rate $r_\tau^{(i)}$. This is because the long-term average velocity of the prosthetic limb must be zero. Thus,

$$\bar{y}_{t|T}^{(i)} = \boldsymbol{b}_\tau^{(i)T} \bar{\boldsymbol{x}}_{t|T} + \bar{\varepsilon}_{t|T}^{(i)} \approx r_\tau^{(i)} \tag{7}$$

where $\bar{y}_{t|T}^{(i)}$ is the average fire rate between time $t-T+1$ and t. To implement this in our parameter tracking algorithm, when the decoding parameters are updated we take the average firing rate over the previous T timesteps as the new baseline firing rate.

2) Velocity Normalization: The second heuristic we implement to stabilize and improve the parameter tracking algorithm is to normalize the estimated velocity before the parameter update. Essentially, we assume that the subject's velocity over a sufficiently long time window will follow a stable distribution. To implement the velocity normalization, the median of the absolute value of the estimated velocity during T steps, denoted as $median(|\hat{X}_{t|T}|)$, is required to be the median of the absolute value of the velocity recorded from training data, denoted as $\boldsymbol{z}$. Therefore, instead of using $\hat{X}_{t|T}$ in Eq. 6, we use $C\hat{X}_{t|T}$ where C is a diagonal matrix with diagonal $\left(z_1/median(|\hat{X}_{1,t|T}|), z_2/median(|\hat{X}_{2,t|T}|), 1\right)$. Here z_1 is the element of the first dimension of $\boldsymbol{z}$ and $\hat{X}_{1,t|T}$ is a vector corresponding to the first dimension of $\hat{X}_{t|T}$.

III. Results

A. Simulation

To test the efficacy of our parameter tracking algorithm, we designed the following 100,000 trial simulation.

1) Kinematics: Our simulated subject repeatedly draws a Lissajous curve in 2D. The velocity of the trajectory is given by $\boldsymbol{v}(t) = \left(s_1 f_1 \sin(f_1 t),\ s_2 f_2 \cos(f_2 t)\right)$, where t is the time (in seconds), and $s_1 = 0.2$m, $s_2 = 0.1$m, $f_1 = 2/3\pi$ Hz, $f_2 = 2\pi$ Hz.

Each trial is one complete cycle of the curve. We set the time step $\Delta t = 3$ms, so each trial consists of 100 steps. Parameters are updated every 20 trials (described below), comprising one session. We run 5,000 sessions in total.

2) Encoding Model: 100 neurons are simulated. The initial parameters for neuron i include:

- the preferred direction, $\theta_0^{(i)}$, randomly sampled from $[0, 2\pi)$;
- the baseline firing rate, $r_0^{(i)}$, randomly sampled from $[5, 10]$Hz;
- the modular depth, $m_0^{(i)}$, randomly sampled from $[4, 8]$m/s^{-1}.

We use a vector $\boldsymbol{b}_0^{(i)}$ to represent the initial parameters associated with neuron i:

$$\boldsymbol{b}_0^{(i)} = \begin{pmatrix} m_0^{(i)} \cos\left(\theta_0^{(i)}\right) \\ m_0^{(i)} \sin\left(\theta_0^{(i)}\right) \\ r_0^{(i)} \end{pmatrix} \tag{8}$$

The encoding model is assumed to be linear-Gaussian in our simulation. That is, the firing rate of neuron i at time t is a linear function of the state $\boldsymbol{x}_t$

$$y_t^{(i)} = \boldsymbol{b}_\tau^{(i)T} \boldsymbol{x}_t + \varepsilon_t^{(i)}, \quad \varepsilon_t^{(i)} \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right) \tag{9}$$

where $\varepsilon_t^{(i)}$ represents the spiking noise.

3) Unstable Neurons: To model instability in the neural tuning curves, the parameter vector's dynamics are simulated as

$$\boldsymbol{b}_{\tau+1}^{(i)} = \boldsymbol{b}_\tau^{(i)} + \boldsymbol{v}_\tau^{(i)}, \quad \boldsymbol{v}_\tau^{(i)} \sim \mathcal{N}\left(\boldsymbol{0}, V^{(i)}\right) \tag{10}$$

As stated before, the parameter change is quite slow compared to the velocity change. So in our simulation, the parameter changes every session (20 trials or 2,000 timesteps).

4) Decoding Algorithms: We compare the following decoding algorithms

- Static Kalman Filter (KF): The traditional Kalman filter with the assumption that the linear model parameters are fixed over time.
- Ground-truth Kalman Filter (KF$_{GT}$): In simulation we always know the exact values of the parameters. Therefore, we also run the ideal Kalman filter where at each step the tuning parameters $\boldsymbol{b}_\tau^{(i)}$ are set to be the ground truth. This method has the best decoding performance that can be achieved in the presence of unknown observation noise. We use this as a benchmark on which to test the performance of our parameter tracking algorithm.

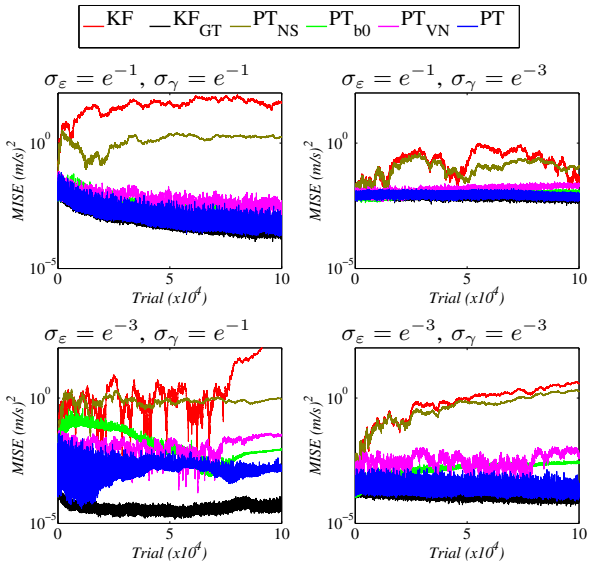

Fig. 1. **Simulation results**: MISE (in log scale) of each of the different methods under various observation noise levels, σ_ε, and parameter noise levels, σ_γ.

- Parameter Tracker without stabilization (PT_{NS}): The parameter tracking algorithm without either stabilization heuristics.
- Parameter Tracker with baseline stabilization (PT_{b_0}): The parameter tracking algorithm using only the baseline firing rate estimation as the stabilization heuristic.
- Parameter Tracker with velocity normalization (PT_{VN}): The parameter tracking algorithm using only the velocity normalization as the stabilization heuristic.
- Stabilized Parameter Tracker (PT): The parameter tracking algorithm using both heuristics for stabilization.

5) Parameter Setting: All algorithms are initialized at time zero with the ground truth parameters, $\boldsymbol{\beta}^{(i)} = \boldsymbol{b}_0^{(i)}$. The state-evolution covariance matrix W is estimated from the actual kinematics. We take T to be 2000 time steps. We quantify performance with the mean integrated squared error (MISE) between the decoded velocity and the actual velocity.

We ran several simulations with varying amounts of noise. The standard deviation of the observation noise, σ_ε, took on values of e^{-1} and e^{-3} Hz. The parameter evolution noise was set to $V^{(i)} = \sigma_\gamma^2 I$, where σ_γ took on values of e^{-1} and e^{-3} (units of Hz(m/s)$^{-1}$ for the first two dimensions and Hz for the last dimension).

6) Simulation Results: The MISE results are shown in Fig. 1. KF_{GT} always has the lowest MISE and the performance does not change much as the experiment runs. KF has very low MISE at the beginning, but diverges very fast since it assumes the parameters are fixed. The performance of PT is close to KF_{GT} and demonstrates that PT can track the parameter evolution quite well. From the results we can see the parameter tracker without stabilization doesn't perform very well due to the over-fitting problem. However, the performance improves substantially with either heuristic. An interesting phenomenon is that the MISE of KF_{GT} decays over time under the noise setting $\sigma_\varepsilon = e^{-1}$, $\sigma_\gamma = e^{-1}$. This is because under the random-walk model, the magnitude of the tuning parameters can grow over time, effectively increasing the signal to noise ratio. We did not attempt to constrain this parameter in our simulations.

The reconstructed velocity of different methods of the 100,000th trial is shown in Fig. 2. KF_{GT} represents the best decoding performance we can get with this level of observation noise. The performance of PB at the 100,000th trial is similar to KF_{GT}, while the performance of KF is quite biased.

B. Offline Trajectory Reconstruction

We also demonstrate the efficacy of our approach by reconstructing off-line the trajectories of actual arm reaches over 5 consecutive days using simultaneously recorded neural data.

1) Experimental Details: Briefly, a Rhesus macaque was trained to sit in a primate chair and make center-out and out-center reaches to 26 targets presented in 3D space. Hand position samples were tracked at 30Hz in 3D space using an Optotrak recording system. Spike trains from 53 neurons were recorded with a Utah microelectrode array and tracked over the 5 days of the experiment. Firing rates were computed in 100ms sliding windows sampled at 30Hz. Full experimental details can be found in [3]. We used a fixed lag of three timesteps between neural activity and predicted kinematics for all neurons.

2) Reconstruction Methods: We compare the ability of each of the algorithms described above to reconstruct the actual trajectories made by the subject over the 5 consecutive days. However, instead of using a ground truth Kalman filter, as done in simulation, we compare the decoding results to a static Kalman filter calibrated only on day 1 (KF), as well as to a static Kalman filter recalibrated using the first session at the beginning of each day (KF_{SD}).

3) Parameter Estimation: Data recorded from day 1, session 1 are used for training to learn the parameters, including $\boldsymbol{\beta}^{(i)}$ for Kalman filter, $\boldsymbol{\beta}_0^{(i)}$ for parameter tracker, W and σ_ϵ. To learn the covariance matrix Π for parameter evolution we first notice that there are two kinds of parameter evolutions, one between sessions, denoted as $\Pi_{session}$, and one between days, denoted as Π_{day}. We use the data from day 1 to learn $\Pi_{session}$ and the data from all 5 days to learn Π_{day}. We assume the covariance matrixes of parameter evolution are the same for each neuron.

4) Results: The results are shown in Fig. 3. On day 1, each of the algorithms performs similarly, as expected. However, on subsequent days the static Kalman filter returns progressively worse trajectory reconstructions. In contrast, the parameter-tracking algorithms perform well on all days, and even outperform a static Kalman filter calibrated on the trials at the beginning of each day.

IV. Conclusions

Current brain-computer interface systems require daily re-calibration of decoding parameters to optimize performance. These re-calibration sessions can be lengthy, and may limit the clinical utility of neural prosthetic systems. In order

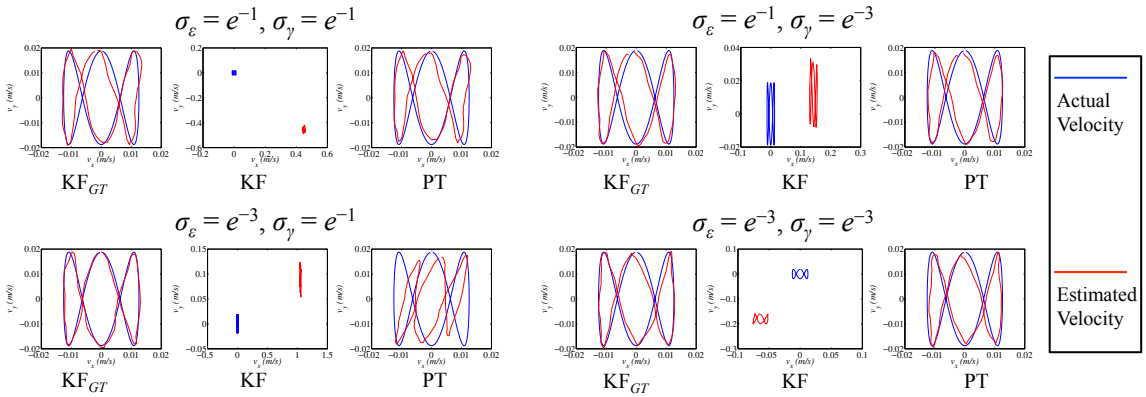

Fig. 2. **Simulation results**: Reconstructed velocity of the 100,000th trial of the simulation resulting from three of the algorithms under various amounts of observation and parameter noise. Note the static KF (middle panels) tends to show an offset, indicative of strongly biased decoding. In contrast, the stabilized dual Kalman filter (right-hand panels) performs nearly as well as the ground-truth Kalman filter (left-hand panels) under most noise contexts.

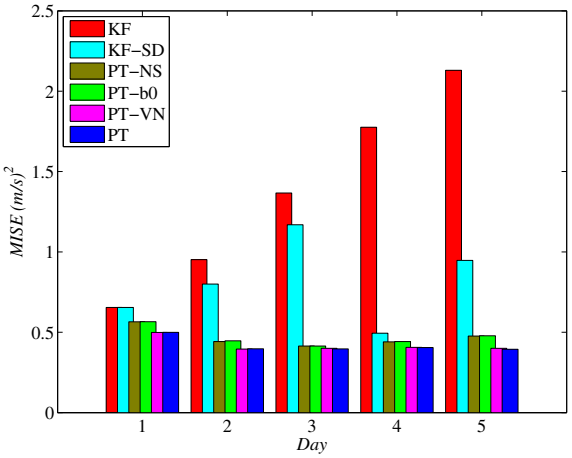

Fig. 3. **Off-line trajectory reconstruction results**: MISE of different methods on the offline data.

to eliminate the need of calibration, we implement a dual Kalman filter to track both the limb states and the parameters, augmented with two stabilizing heuristics: baseline firing rate estimating and velocity normalization. Our stabilized dual Kalman filter performs well in both simulation and in estimating arm movement trajectories off-line.

Here we focused on the instability of the neural tuning parameters while assuming fixed variance. Testing how much improvement may be gained by also tracking the variance remains the subject of future work. We also assumed that the number of neurons we tracked remained fixed. In practice, neurons may drop out or come in to the recording over time. This complicates the implementation of parameter tracking, because the dimensionality of the tuning matrix will change over time [8]. One way to simply capture this behavior would be to always track a stable "maximum" number of neurons. Those that correspond to blank signals would return tuning coefficients of zero. If the residual variance assigned to those units is constrained to be non-zero, they should not affect the decoding solution. As neurons come in to the recording, the algorithm would start to measure their tuning. Implementation of this extension remains a subject of future work.

References

[1] S. M. Chase, R. E. Kass, and A. B. Schwartz. Behavioral and neural correlates of visuomotor adaptation observed through a brain-computer interface in primary motor cortex. *Journal of Neurophysiology*, 108(2):624–644, 2012.

[2] C. A. Chestek, J. P. Cunningham, V. Gilja, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy. Neural prosthetic systems: current problems and future directions. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3369–3375, 2009.

[3] G. W. Fraser and A. B. Schwartz. Recording from the same neurons chronically in motor cortex. *Journal of Neurophysiology*, 107(7):1970–1978, 2012.

[4] K. Ganguly and J. M. Carmena. Emergence of a stable cortical map for neuroprosthetic control. *PLoS Biol*, 7(7):e1000153, 2009.

[5] V. Gilja, C. Chestek, I. Diester, J. Henderson, K. Deisseroth, and K. Shenoy. Challenges and opportunities for next-generation intracortically based neural prostheses. *IEEE Transactions on Biomedical Engineering*, 58(7):1891–1899, 2011.

[6] V. Gilja, P. Nuyujukian, C. A. Chestek, J. P. Cunningham, B. M. Yu, J. M. Fan, M. M. Churchland, M. T. Kaufman, J. C. Kao, S. I. Ryu, and K. V. Shenoy. A high-performance neural prosthesis enabled by control algorithm design. *Nature Neuroscience*, 15(12):1752–1757, 2012.

[7] 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*, 442:164–171, 2006.

[8] Z. Li, J. E. O'Doherty, M. A. Lebedev, and M. A. L. Nicolelis. Adaptive decoding for Brain-Machine interfaces through bayesian parameter updates. *Neural Computation*, 23:3162–3204, 2011.

[9] L. Nelson. The simultaneous on-line estimation of parameters and states in linear systems. *IEEE Transactions on Automatic Control*, 21:94–98, 1976.

[10] U. Rokni, A. G. Richardson, E. Bizzi, and H. S. Seung. Motor learning with unstable neural representations. *Neuron*, 54(4):653–666, 2007.

[11] A. B. Schwartz, X. T. Cui, D. J. Weber, and D. W. Moran. Brain-controlled interfaces: movement restoration with neural prosthetics. *Neuron*, 52:205–220, 2006.

[12] L. Shpigelman, H. Lalazar, and E. Vaadia. Kernel-ARMA for hand tracking and Brain-machine interfacing during 3d motor control. In *Advances in Neural Information Processing Systems*, pages 1489–1496. 2009.

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

[14] W. Wu, M. J. Black, Y. Gao, E. Bienenstock, M. Serruya, A. Shaikhouni, and J. P. Donoghue. Neural decoding of cursor motion using a Kalman filter. In *Advances in Neural Information Processing Systems*, pages 133–140. 2003.