Breaking the fixed-arrival-time restriction in reaching movements of neural prosthetic devices

Lakshminarayan Srinivasan and Marco da Silva

***Abstract*— We routinely generate reaching arm movements to function independently. For paralyzed users of upper-extremity neural prosthetic devices, flexible, high-performance reaching algorithms will be critical to restoring quality-of-life. Previously, algorithms called real-time reach state equations (RSE) were developed to integrate the user's plan and execution-related neural activity to drive reaching movements to arbitrary targets. Preliminary validation under restricted conditions suggested that RSE might yield dramatic performance improvements. Unfortunately, real-world applications of RSE have been impeded because the RSE assume a fixed, known arrival time. Recent animal-based prototypes attempted to break the fixed-arrival time assumption by proposing a Standard Model (SM) that instead restricted the user's movements to a fixed, known set of targets. Here, we leverage General Purpose Filter Design (GPFD) to break both of these critical restrictions, freeing the paralyzed user to make reaching movements to arbitrary target sets with various arrival times and definitive stopping. In silico validation predicts that the new approach, GPFD-RSE, outperforms the SM while offering greater flexibility. We demonstrate GPFD-RSE against SM in the simulated control of an overactuated 3-dimensional virtual robotic arm with a real-time inverse kinematics engine.**

***Index Terms*— reach state equation, neural prosthetic, BMI**

I. Introduction

TRANSLATING our intentions into purposeful behavior requires the coordinated interaction between sensorimotor neural activity and muscle contraction. Trauma and disease destroy neural circuitry in the brain and spinal cord, disrupting the intuitive correspondence between our intentions and our resulting actions. Stroke, Parkinson's disease, spinal cord injury, multiple sclerosis, and amyotrophic lateral sclerosis are a few examples. In an emerging therapeutic strategy, neural prosthetic devices attempt to restore an intuitive correspondence between intentions and actions. These devices include an algorithm that defines the interface and a hardware substrate (cellular, metallic, semiconductor, etc.) that implements these algorithms. The standard algorithm first produces an estimate of the patient's intention, and then drives the prosthesis to achieve that estimate [2]. The prosthesis itself may be a physical object or a virtual environment.

The control of target-directed reaching movements represents a critical need for prosthetic users because it facilitates routine tasks like picking up a telephone or reaching for a glass of water. Neural activity from areas of the frontal and parietal cortices can be filtered to estimate the target state and the arm's path to the target [9-12]. Bayesian principles have been used to combine both plan and execution activity for even greater decoding precision [1, 4, 13-16]. In these algorithms, a reach model defines the statistical relationship between path and target of the intended movement.

The RSE represents over five years of refinements in reach models. Initially, recursive, real-time-implementable algorithms using stochastic state-space models [1, 3, 13] replaced batch-processed trajectory estimation on canonical reach trajectory models [14, 15]. Second, generative models of reaching movements [1, 3, 13] were developed to support reaches to arbitrary target locations without training data from those targets [7, 14, 15]. Third, discrete target location priors were incorporated into reach estimates [2, 7]. Fourth, targets were allowed to switch mid-reach [2].

Despite these advances, the RSE has been limited to reaches with fixed arrival time. In contrast, natural reaches can be slow or fast, depending on the context of a movement. You reach quickly to stop a child from dashing into the busy street, but more slowly to pick up a hot cup of coffee. The tempo of a reach shapes the lilt of a pianist's phrase, or the texture of a brush stroke. A neural prosthetic device with fixed-arrival-time RSE would be inadequate in these real-life activities.

Until now, the fixed arrival time assumption has prevented animal and human prototypes from directly leveraging the powerful advances embodied in RSE [17-24, 29]. To circumvent fixed arrival times, recent experimental demonstrations [7, 8] instead applied Standard Models (SM), which trade one critical restriction for another by assuming a fixed set of target locations [7] or a representative training database [8] to accommodate variable arrival times.

Can a single neural prosthetic algorithm support *both* arbitrary target locations *and* variable-arrival-time reaches? Our approach extends RSE with GPFD, a unifying design framework for neural prosthetic algorithms [2]. The new approach, GPFD-RSE, supports reaching movements to arbitrary targets while approximating an intended speed, even

Manuscript received May 3, 2010. This work was supported in part by the NIGMS NRSA T32 GM07753-28, UCLA Radiology, Singapore-MIT Gambit Game Lab, the NSF (2007043041, CCF-0810888), Adobe Systems, and Pixar Animation Studios, with software donations from Autodesk and Adobe.

Lakshminarayan Srinivasan is with the Neural Signal Processing Laboratory, Department of Radiology, University of California Los Angeles 90095 (email: ls2@nsplab.org).

Marco da Silva is with Boston Dynamics, Waltham, MA 02451 (email: dasilva@mit.edu).

when the intended arrival time is unknown to the prosthesis. We develop, benchmark, and analyze the technique *in silico*.

II. METHODS

This section describes how GPDF is used to expand the capability of RSE beyond the fixed-arrival-time assumption. The overall approach can be summarized in terms of the random variables displayed in Figure 1. We will define discrete-valued intentions (s_k) that represent both targets and arrival times. Each of these discrete-valued intentions will specify the dynamics of the arm state (x_k), a continuous-valued intention. GPFD then specifies recursive Bayesian estimation equations that can then be applied to automatically estimate the candidate arrival times, target locations, and arm trajectory.

A. Basic Reach State Equation

Our present work simulates the operation of a 3D prosthetic hand that is positioned onto a 2D workspace. For this application, x_k is defined as a 4x1 vector random variable that represents the intended position and velocity at time step k with time step width δ seconds. Recall that the basic RSE defines a minimal statistical constraint $p(x_0,\ldots,x_T)$ between target x_T and path $(x_0,\ldots,x_T)$ [4].

$$x_k = B_k x_{k-1} + f_k + \varepsilon_k \qquad (1)$$

Here, B_k is a 4x4 matrix that transfers velocity into position, f_k is a 4x1 forcing term that shapes the trajectory towards the target, and ε_k is 4x1 random variable that represents the prosthetic device's uncertainty about the user's intentions. Formulas for B_k, f_k, and ε_k are derived in [2] and reproduced in the Appendix. The 4x1 target state y_T enters into f_k and 4x4 uncertainty in the target state (Gaussian covariance $\prod_T$) enters into the covariance of ε_k given in equation (9). Accordingly, the RSE requires a known arrival time step T, but can be adjusted for arbitrary targets y_T using equation (1) without training data.

B. General Purpose Filter Design

Recall that General Purpose Filter Design (GPFD) is a versatile framework for conceptualizing the structure of neural signal filters across a variety of recording modalities and prosthetic devices [2]. At each time step k, a patient's intentions for a prosthetic device includes both discrete (s_k) and continuous-valued (x_k) components. The patient's neural activity (n_k) is determined by these intentions and prior neural activity $H_k = (n_1, n_2, ..., n_{k-1})$. GPFD uses a simple but rigorous probabilistic relationship between n_k, s_k, and x_k that is embodied in the graphical model (Fig. 1). If we specify n_k, s_k, and x_k and the conditional probability densities $p(x_{k+1}|x_k,s_{k+1})$, $p(s_{k+1}|s_k)$ and $p(n_{k+1}|H_{k+1}, x_{k+1}, s_{k+1})$, the GPFD framework automatically provides a neural signal processing algorithm for the desired prosthetic application. This versatility makes GPFD a natural choice for extending the RSE. The specifics of this procedure were previously derived, illustrated, and explained in detail [2]. Refer to the Appendix for an abbreviated presentation of the GPFD filter equations. We will now exploit GPFD to extend RSE beyond fixed arrival times.

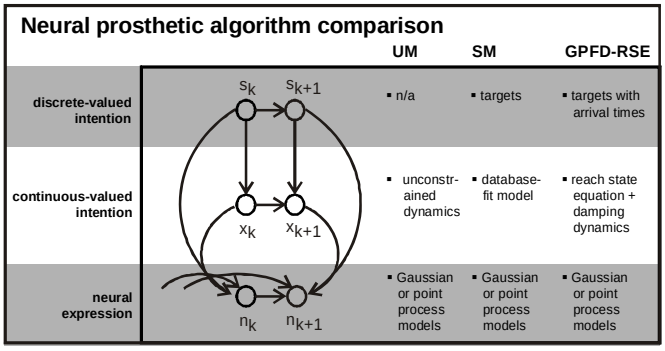

Fig. 1. Key characteristics of the Unconstrained Model (UM), Standard Model (SM), and the new approach (GPFD-RSE). All three methods are depicted with the GPFD graphical model. Nodes represent random variables, and arrows represent Markov statistical dependence. Any of these methods can decode reaching movements from neural activity, but only GPFD-RSE explicitly specifies variable arrival times in its description of the user's intentions. While the SM requires database training, GPFD-RSE exploits the RSE's capability to accommodate arbitrary, untrained target locations.

C. Extending the RSE to variable arrival times with GPFD

We will now use GPFD to extend the RSE to variable arrival times while maintaining its various desirable performance advances. We refer to this new variable-arrival-time RSE and its associated neural signal filter as GPFD-RSE.

Define $\Omega = \{\omega_1, \omega_2, \ldots, \omega_P\}$ as a finite, discrete set of desired arrival time steps. For each potential arrival time step $\omega \in \Omega$, the arm's continuous endpoint state x_k follows a modified RSE:

$$x_k = \begin{cases} B_k x_{k-1} + f_k + \varepsilon_k & , 0 < k < \omega \\ A_{damping}\, x_{k-1} & , \omega \le k \end{cases} \qquad (2)$$

where $A_{damping} = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & \kappa & 0 \\ 0 & 0 & 0 & \kappa \end{bmatrix}$, and $0 < \kappa < 1$. For the $0<k<\omega$ case, the parameters of the RSE are determined by the target position y_T and stopping time step ω. The rate at which the arm slows down is controlled by κ. Smaller κ results in faster damping. For the $\omega \le k$ condition, we assign $\kappa = 0.1$ (unitless). While the specific value of κ is somewhat arbitrary, the chosen order of magnitude creates the qualitative effect of crisp stopping movements. More concretely, with $\kappa = 0.1$, an arm velocity of (5 cm/sec, 5 cm/sec) at time step ω will be slowed to a velocity of (0.5 cm/sec, 0.5 cm/sec) at time step $\omega+1$.

This formulation can be extended further to include arbitrary discrete target sets $Z = \{z_1, z_2, \ldots, z_R\}$. This target set can be adapted from movement to movement based on the user's reachable environment and computer vision [30] or (with additional research) neural representations of visually identified objects [31]. To simultaneously incorporate both target and arrival time sets, we redefine the complete set of

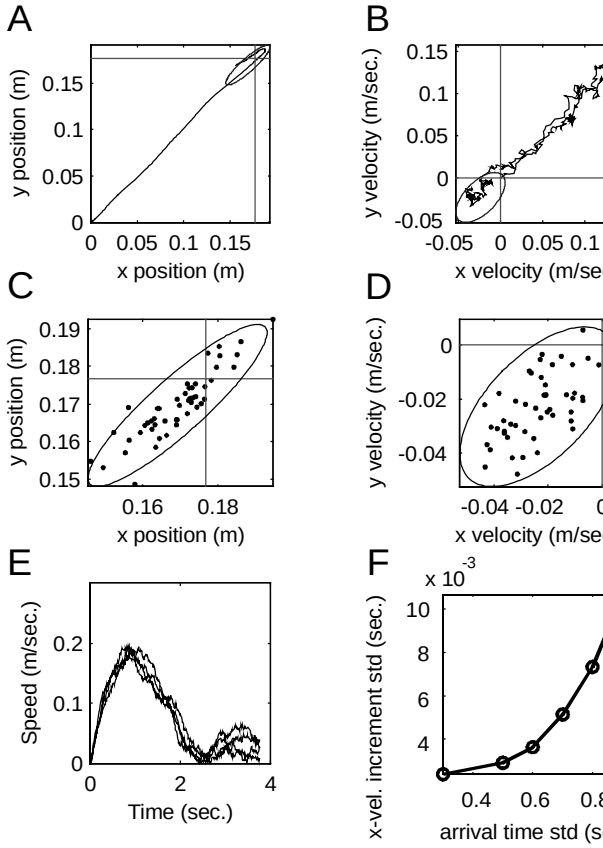

Fig. 2. Sample trajectories of the Standard Model (SM). The intended target is 0.25 meters from the origin at (0.1767, 0.1767) with arrival time ~N(2,0.4). (A) Single position trajectory in a 2D workspace. (B) Same as (A) with velocity. (C) Sample position endpoints with 95% ellipsoid. (D) Sample velocity endpoints. (E) Speed trajectories. Ellipsoids represent 95% confidence intervals (CI).

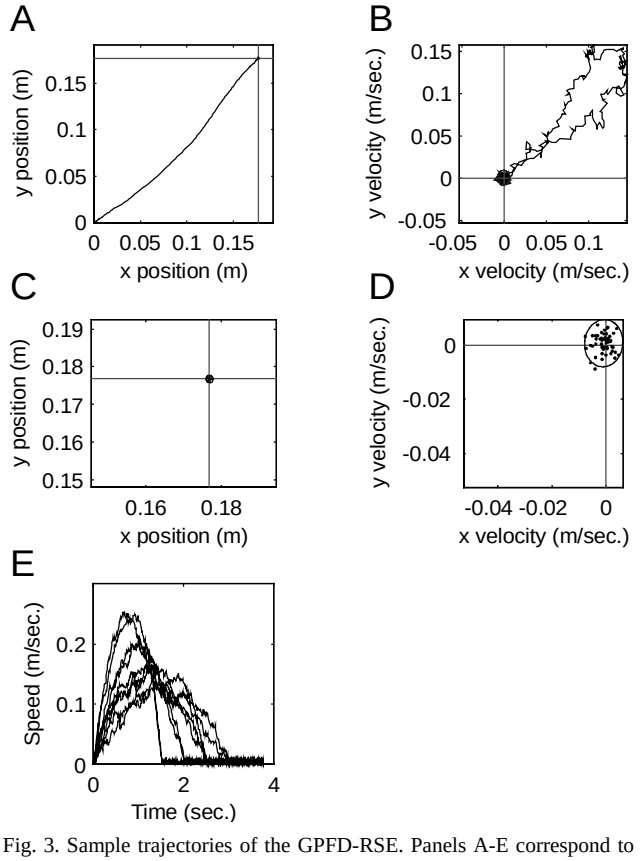

Fig. 3. Sample trajectories of the GPFD-RSE. Panels A-E correspond to Fig. 2(A-E), with identical scaling, intended target, and arrival time distributions. (A) Single position trajectory in a 2D workspace. (B) Same as (A) with velocity. (C) Sample position endpoints with 95% ellipsoid. (D) Sample velocity endpoints. (E) Speed trajectories. Ellipsoids represent 95% CI.

discrete states s_k from the GPFD framework to be $S = \Omega \times Z$. This represents all $P \times R$ pairs of possible arrival times and target locations.

To complete the GPFD-RSE specification, we must explicitly define $p(x_{k+1}|x_k,s_{k+1})$, $p(s_{k+1}|s_k)$ and $p(n_{k+1}|\ H_{k+1},\ x_{k+1},\ s_{k+1})$, as required by the graphical model (Fig. 1). For each discrete state $s_k \in S$, representing arrival time step ω and target location y_T, define the $p(x_{k+1}|x_k,s_k)$ by the RSE in (2) with $\prod_T=0$ (see Appendix and [4] for RSE formulas). Define $p(s_{k+1}|s_k)$ to be 1 if $s_{k+1} = s_k$, and 0 otherwise. This choice embodies the notion that the user will not switch their intended arrival time during the movement.

Finally, the observation model $p(n_{k+1}|H_{k+1},\ x_{k+1},\ s_{k+1})$ is determined by the specific recording modality (EEG, spikes, etc.). Previously, we demonstrated observation models for both spike and EEG-type signals [2]. In this paper's example, we decode simulated neural activity in primary motor cortex (MI) based on a velocity-dependent cosine tuning point process model [12, 32, 33]. The conditional intensity $\lambda(k|v_x,v_y)$ of this point process model defines the probability with which a neuron generates a spike for the intended arm movement at time step k:

$$\lambda(k \mid v_x, v_y) = \exp(\beta_0 + \beta_1 (v_x^2 + v_y^2)^{1/2} \cos(\theta - \theta_p)) \qquad (3)$$

where v_x and v_y are velocities in orthogonal directions. History dependence in the spiking patterns [32] can be readily accommodated as illustrated previously [2]. The model parameters are drawn to approximate empirically observed firing rates, direction tuning widths, and speed modulation in primate recordings of MI [32]: $\beta_0 = 2.28$ (unitless), $\beta_0 = 4.67$ s/m, and $\theta_p \sim$ uniform$(-\pi,\pi)$ (angle in radians). As we have shown previously, equation (3) determines $p(n_{k+1}|H_{k+1},\ x_{k+1},\ s_{k+1})$ [2].

We have now defined the components of the GPFD (n_k, s_k, x_k, $p(x_{k+1}|x_k,s_{k+1})$, $p(s_{k+1}|s_k)$ and $p(n_{k+1}|\ H_{k+1},\ x_{k+1},\ s_{k+1})$) in a manner that embodies the variable-arrival-time extension of the reach state equation. The GPFD machinery specifies a corresponding neural signal processing algorithm based on recursive Bayesian estimation as previously derived [2]. Although [2] provides a more expansive presentation of GPFD, the critical equations required to reproduce GPFD-RSE are provided in the Appendix of this paper.

D. Unconstrained Model (UM) and Standard Model (SM)

Details of the UM [34] and SM [7, 8, 34] implementation have been discussed elsewhere. We previously demonstrated that the GPFD framework could be used to describe these algorithms as well [2]. Briefly, recall that the UM corresponds to defining $p(x_{k+1}|x_k)$ by a random walk of the form $x_k = x_{k-1} + w$ where w is a 4x1 zero-mean Gaussian random

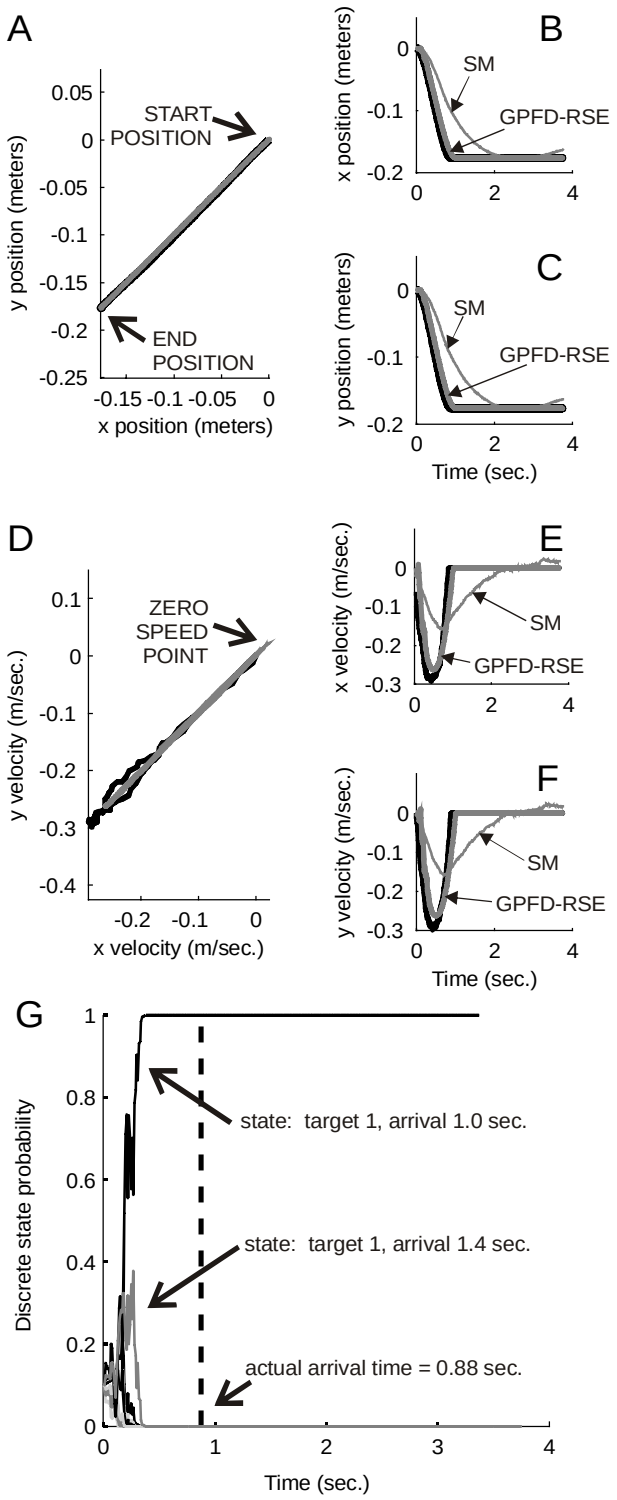

Fig. 4. Example decode, comparing SM and GPFD-RSE. The intended trajectory (thick black) begins at (0,0) and ends 0.25m away at (-0.1767, -0.1767) with an arrival time of 0.88 seconds. The decoded trajectories are superimposed, including SM (thin gray) and GPFD-RSE (thick gray). (A,B,C) Position, and (D,E,F) velocity trajectories. (G) Posterior probability on the discrete states of the GPFD-RSE.

vector. There are no discrete state variables s_k in this method. The Standard Model (SM) corresponds to defining $p(x_{k+1}|x_k,s_k)$ by a generic linear-Gaussian state equation $x_k = A_k x_{k-1} + w$ that is fit to a database . This database is comprised of human or primate-generated reaching movements to the targets of interest. The matrix A_k and covariance of w, are determined by a least squares fitting procedure.

Two versions of the SM have been described previously [7, 8]. In the first SM [7], each target location requires a separate database of reaches to run model fitting. In the second SM [8], one model is learned for arbitrary target locations. We focus on the second SM for its added feasibility. The state variable in this SM includes an 8x1 collection of two-dimensional arm position, velocity, acceleration, and target x- and y-coordinates. In this SM, the A_k is 8x8 and the w is an 8x1 zero mean Gaussian vector random variable.

E. Comparing algorithmic structure in the GPFD-RSE with the Standard Model (SM)

How do GPFD-RSE and SM differ in their algorithmic structure? In short, both methods involve interacting discrete and continuous intentions that influence the observed neural activity. For this reason, the graphical model that describes both approaches in an abstract sense is identical (Fig. 1). The critical difference is in the definition of s_k, x_k, and $p(x_{k+1}|x_k,s_k)$. These differences are summarized in the entries of Fig. 1 under the columns labeled "SM" and "GPFD-RSE". In the SM, the s_k represent individual target locations, whereas in the GPFD-RSE, the s_k represent pairs of target location and arrival time. In the SM, x_k is an 8x1 vector of various kinematic features, whereas in the GPFD-RSE, x_k is a 4x1 vector of positions and velocities (but can represent arbitrary arm states in general). In the SM, the $p(x_{k+1}|x_k,s_k)$ are governed by a database-driven model, whereas in the GPFD-RSE, the modified RSE in equation (2) describes this density for arbitrary target locations and arrival times. The validation process illustrates the impact of these various algorithmic differences on device performance.

F. Validating Neural Signal Filters

There are several types of validation studies for neural prosthetic device algorithms (Table I). This paper uses *in silico* validation. See the Appendix for details of our implementation. This type of validation uncovers essential algorithmic limitations of a technique before incurring the various expenses and risks of animal and human study. Our *in silico* validation procedure is based on point process models that relate intended arm movements to primary motor cortical activity. These models have been rigorously cross-validated against data from primates, both by their ability to predict velocity-dependent spike timing [32], and their ability to decode primary motor cortical activity [33, 35]. Although relatively new, this validation method has been critical in the development of the RSE [1, 13] and GPFD [2]. The clear benefit of *in silico* validation is the capability to rapidly prototype and test neuroprosthetic algorithms under thousands of simulated conditions. However, we expect that our performance estimates will differ from performance in the ultimate clinical application due to effects that were not modeled in simulation. Feedback and learning are generally expected to improve clinical performance over simulated predictions. Model mismatch is likely to degrade performance. In prior work, we used *in silico* methods to illustrate the effect

of model mismatch on RSE [4] and GPFD [2]. More recently, the use of simulated motor cortical data together with joystick entry has been proposed as a way of incorporating feedback effects in simulated validation, with cross-validation to primate data [36]. In general, concordance between various validation techniques is less well characterized. However, such studies will be important to expediting progress in algorithm design.

G. Controlling a virtual over-actuated prosthetic arm

Prosthetic and human arms are vastly overactuated, with as many as 25 DOF [37]. As we illustrate here, hierarchical autonomous control of the remaining DOF can allow the user to achieve global objectives for the device without having to specify every DOF. Working in a three-dimensional virtual environment, we use the output of each neural signal filter (GPFD-RSE, SM, or UM) to control an overactuated robotic arm with 9 DOF that must configure itself in 3D to achieve the estimated hand position within a 2D workspace. The resulting simulation is rendered in a movie (online, [38]). Every time step is displayed without any post-analysis smoothing effects. The arm includes three ball-in-socket joints, each with 3 DOF, controlled through a damped inverse kinematics algorithm [39, 40] that operates in real-time. This method is capable of achieving dynamic secondary constraints that might be required when reaching through a cluttered environment like a refrigerator or dining table. Our implementation uses C++, OpenGL, Lapack, and a standard desktop computer running Macintosh OS or Linux.

III. Results

A. Properties of the Standard Model versus GPFD-RSE

We first generated sample trajectories of the SM and GPFD-RSE models to visualize whether the models captured essential qualities of variable arrival time reaches.

Sample paths of the SM (Fig. 2) illustrate that the model deviates from defining features of a reaching movement. A single SM trajectory does not end definitively at the target, indicated by the x- and y- crosshairs (Fig. 2A). There is no definitive stopping point to the SM-simulated reach velocity (Fig. 2B). Endpoint position (Fig. 2C) and velocity (Fig. 2D) across multiple sample trajectories show large spread. Ideally, endpoint position should be focused definitively on the target position in crosshairs, and endpoint velocity should be zero. The speed profile across several SM trajectories further emphasize that individual trajectories fail to achieve definitive stopping times (Fig. 2E). As variance in the database of arrival times grows, the best-fit SM model has increased increment covariance. This means that the SM model finds it difficult to simultaneously achieve both smooth sample velocity trajectories and highly variable arrival times.

Trajectories of the GPFD-RSE model (Fig. 3), capture defining features of a reaching movement that the SM was unable to reproduce. A sample trajectory arrives at the target (Fig. 3A), and stops definitively (Fig. 3B). Across several sample GPFD-RSE trajectories, the endpoint achieves target positions (Fig. 3C) and stopping velocities (Fig. 3D) with greater precision and accuracy than the SM trajectories. Stopping velocities are not exactly zero because GPFD-RSE uses damping dynamics rather than suddenly bringing the arm to zero velocity. GPFD-RSE generates a wide range of definitive stopping times (Fig. 3E) that are independent of increment covariance unlike SM (Fig. 2 E,F).

TABLE I
VARIOUS APPROACHES TO VALIDATING NEURAL PROSTHETIC ALGORITHMS

Type of neural prosthetic device validation study	Examples
In silico analysis using statistically validated models of neural signals	[1-5]
Animal subject, healthy: open-loop neural signal filtering	[6-8]
Animal subject, healthy: closed-loop neural signal filtering	[8, 17-24]
Animal subject, disease model	[25]
Human subject, healthy	[26, 27]
Human subject, different disease e.g. subdural grid validation for cursor control during seizure localization for pre-surgical planning	[21]
Human subject, target disease: case studies	[23, 28, 29]
Human subject, target disease: randomized controlled trials	None

The common objective of these approaches is to produce realistic predictions about device performance in the target patient population.

B. Decoding Results

We apply the basic *in silico* validation process described in Section II. to produce decoding results from the UM, SM, and GPFD-RSE models. A single decoded trial (Fig. 4) illustrates the filters' typical behaviors. The SM decoding result (thin gray line) achieves the same position path in space (Fig. 4A), but lags the desired movement through time (Fig. 4B,C). The GPFD-RSE result (thick gray line) is superimposed on the desired trajectory in space *and* time. These discrepancies are reiterated in velocity trajectories (Fig. 4D-F). Moreover, just as the sample SM trajectories did not show definitive stopping (Fig. 2D), the SM-decoded arm continues to drift long after the stopping time desired by the user (Fig. 2E,F). The GPFD-RSE model effectively stores multiple hypotheses about the intended arrival time and adjusts its belief about the true arrival time in real-time based on incoming neural signals. The discrete state probability (Fig. 4G) represents this adjusting belief, where the discrete state corresponding to the (-0.1767, -0.1767) target with arrival at 1 second quickly dominates alternative discrete hypotheses distributed over 1 to 3 seconds based on accumulated neural observations reflecting the user's intended arrival time of 0.88 seconds. This example also demonstrates that GPFD-RSE can achieve intermediate arrival times that are not explicitly included in the discrete set *S*. This is because trajectory estimates are calculated by the mean of the posterior density. The UM decoded trajectory is omitted from Fig. 4 for clarity.

How well do the UM, SM, and GPFD-RSE algorithms perform on average? We first examined performance errors post-movement-onset and post-stop-time, averaged over all reach durations (Fig. 5). Error was reported as root mean squared error (RMSE) in position and velocity. Position RMSE errors accumulate early in the movement, and decrease later in time post-movement onset (Fig. 5A). This effect is most pronounced with GPFD-RSE (black line), because it more definitively represents the target position than SM (thin

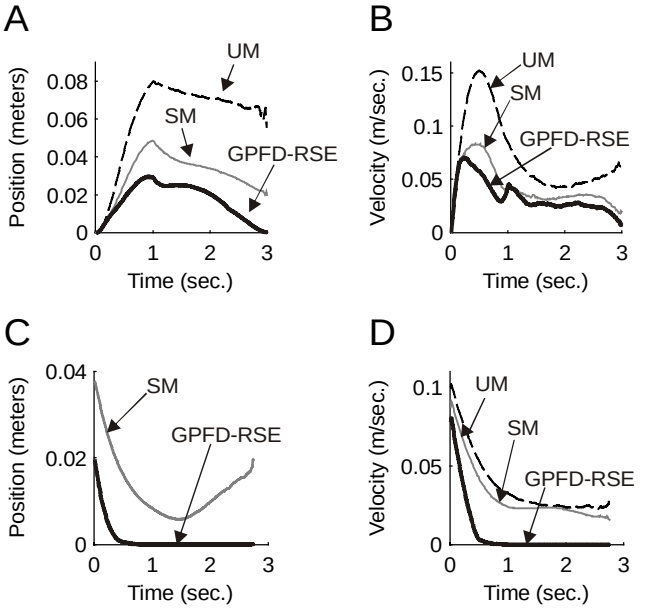

Fig. 5. Performance errors as a function of time-into-movement. GPFD-RSE (solid black), SM (solid gray), and UM (dotted gray) errors are averaged across reaches of various durations. (A, B) RMSE in (A) position and (B) velocity as a function of time post movement onset. (C, D) RMSE in (C)

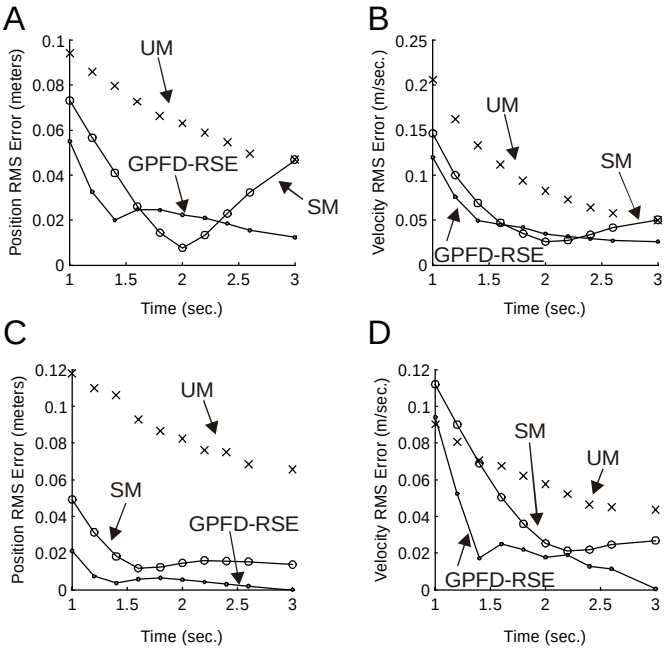

Fig. 6. Performance errors as a function of intended arrival time. GPFD-RSE (small circles), SM (large circles), and UM (cross marks) errors are averaged over the intended movement period (A,B) and intended post-arrival period (C,D). (A, B) RMSE in (A) position and (B) velocity over the entire movement trajectory as a function of intended arrival time. (C, D) RMSE in (C) position and (D) velocity as a function of intended arrival time.

gray line). UM accumulates significantly more error over all times post-movement-onset than SM or GPFD-RSE because it has no representation of potential targets to propagate into its trajectory estimates. Similar trends are observed in velocity RMSE post-movement onset (Fig. 5B). Position RMSE post-stop-time (Fig. 5 C, D) demonstrates that GPFD-RSE achieves near-zero error very shortly after the desired endpoint arrival time, whereas SM has residual and even increasing endpoint error with time (Fig. 5C). This reflects the definitive stopping behavior in GPFD-RSE (Fig 3E) that was absent in SM (Fig. 2E). Position RMSE for UM is omitted because it performs substantially worse than SM and GPFD-RSE in this measure; plotting UM in the same graph would obscure the difference in performance between SM and GPFD-RSE. In velocity RMSE post-stop-time (Fig. 5D), SM and UM perform approximately equally, maintaining residual error while the GPFD-RSE has locked quickly to the zero velocity state, within approximately 0.5 seconds of the desired stopping time. This is again likely related to the definitive stopping property of GPFD-RSE that is absent in SM.

We next examined errors averaged individually for each reach duration. Interestingly, SM (large circles) performs better for reach durations around the mid-point of the uniform(1 sec., 3 sec.) arrival time distribution than GPFD-RSE (small circles) or UM (cross marks) in terms of the pre-arrival-time trajectory (Fig. 6 A, B). This is likely because the SM model better represents this average case than the GPFD-RSE model. Nevertheless, SM performance during the pre-arrival-time period degrades for reaching movements that are faster or slower than the average arrival time, whereas GPFD-RSE maintains RMSE performance over a wider range of arrival times. All three filters produce degraded behavior with the fastest movements, which are more difficult to track in general because previous neural signals are less informative about current arm states in rapidly changing versus gradually changing movements. In the post-arrival-time period (Fig. 6 C, D), the GPFD-RSE method still outperforms SM and UM across all reach arrival times.

Finally, we demonstrated control of a virtual overactuated robotic arm by combining UM, SM, and GPFD-RSE neural filters with real-time inverse kinematics (see online movie, [38]). Results are displayed for a fast reaching movement (example 1) and a slow reaching movement (example 2). The starting hand position (green ball), target hand position (red ball), and intended trajectory (animated blue ball) represent the user's intentions. The position of the robotic end-actuator represents the decoded intention based on UM, SM, and GPFD-RSE neural filters.

For the fast reach movie segment, SM *lags* the desired hand location throughout the trajectory, and continues to move when the intended trajectory has stopped. For the slow reach, SM *leads* the desired hand location, arriving at the target before the intended time. The SM model behavior resembles the average trajectory case, which is inaccurate for faster (~ 1.4 sec.) and slower (~2.8 sec.) arrival times. In contrast, the GPFD-RSE model represents various arrival times as separate hypotheses, allowing it to track arrival times that differ more substantially in time and achieve crisp stopping behavior.

IV. Discussion

We rely on our ability to produce reaching movements to various targets *and* with different arrival times - gingerly bringing a hot cup of coffee to our lips, or briskly opening the door to the office. By combining plan and execution activity, the RSE represented the beginning of an evolving design

principle where the neural interface could achieve this type of behavior. However, the use of RSE in animal and human prototypes has been limited by its assumption that the arrival time of a desired reaching movement is known. In this paper, we developed and analyzed a new method (GPFD-RSE) that breaks the fixed arrival time assumption.

We used sample model trajectories to understand the intrinsic advantages of GPFD-RSE over SM. Our *in silico* decoding results predict that GPFD-RSE performs more consistently across a wider range of arrival times than SM. We then coupled the various neural signal filter outputs with a real-time inverse kinematics algorithm to control an overactuated robotic arm on a 2D workspace in a 3D virtual environment (online movie, [38]). This work demonstrate that GPFD-RSE provides accurate stop-times, definitive stop velocities, and better trajectory tracking than the SM or UM. We also showed (Fig. 4) that the GPFD-RSE is capable of achieving arrival times that are not explicitly included in the discrete set *S*. Intermediate arrival times are achieved because the decoded arm position equals the mean of the posterior density. Finally, we illustrated the practical application of hierarchical design for achieving neural control of overactuated arms like the DARPA HAND prototypes [37].

This paper used RMS error to characterize device performance. This criterion can be interpreted as a standard deviation. Error bars constructed on this standard deviation give us a sense of the typical discrepancy between intended and achieved states if we assume a Gaussian error distribution. When interpreting RMS position error, it is important to consider the size of the workspace. For example, a two centimeter increase in RMS endpoint position error over a 25 cm reach (Fig. 5C) broadens the typical (66% confidence interval) spread in endpoint error by over 10% of the reaching distance. For RMS endpoint velocity error (Fig. 5D), a residual movement of +/- 2.5 cm/sec. (66% confidence interval) in the SM model compared to crisp stopping in the GPFD-RSE model is more directly appreciated. The online movie [38] helps to visualize the extent of functional deficits that might be induced by these levels of error.

The GPFD approach and the SM approaches share fundamentally similar mathematical structure (Fig. 1). They are all recursive Bayesian techniques with discrete and continuous interacting states. This similarity was the focus of our previous manuscript that used graphical models to relate various different neural prosthetic algorithms under a common GPFD framework [2]. There are also key differences between the GPFD-RSE and SM. The GPFD-RSE does not use training reaches for each target location. As a result, the GPFD-RSE can support novel reach targets and arrival times. This stems from our use of the reach state equation [4]. In contrast, the parameters of the SM approaches [7, 8] are selected by fitting the model to example trajectories. Second, the original SM approaches use different methods for computing the posterior densities from point process data, including a Kalman filter where spike trains are first convolved with a Gaussian kernel [8], and an approximate point process filter that requires executing Newton's method at each step [7, 41]. GPFD-RSE uses a streamlined approximate point process filter detailed previously [2, 33] that eliminates the need for a Newton's method at each iteration.

Hardware that implements GPFD-RSE must be able to parallelize or multi-thread the execution of several point process or continuous-valued filters under operating constraints. GPFD-RSE requires one basic filter for each of $P \times R$ discrete states. Restricting $P \times R$ improves efficiency *and* accuracy. In practice, $P \times R$ can be kept small by automatically extracting potential targets (R) in the visual scene, and by supporting a restricted set of definitive stopping times (P). Alternatively, neural signals could be incorporated to constrain the desired arrival time before the movement is executed [42].

Ultimately, neural prosthetic devices will transition from prototype to production. Patients will demand reliable, uncompromising performance from these algorithms to restore dexterous reaching movements in their everyday lives. Support for variable arrival times with definitive stopping is one critical step towards this ambitious therapeutic goal.

APPENDIX

IMPLEMENTING THE SIMULATION

The entire simulation process involves four steps. First, we draw the arrival time uniformly from a continuous interval between 1 and 3 seconds. Next, we draw the target location at random (with equal probability) between two target locations at (0.1767,0.1767) and (-0.1767,-0.1767) meters from the hand's starting position at (0,0).

Second, we simulate a trajectory at 10 millisecond time steps using the RSE corresponding to the drawn arrival time and target location. For time steps greater than the arrival time, the hand's position is assigned to the target location with zero velocity. Each trial lasts a total of 3.75 seconds, including both the reaching and stopped periods.

Third, to simulate the user's spiking neural activity, we apply the empirically-derived observation model of primary motor cortex (MI) [43] represented in equation (3). Spikes are simulated using the time rescaling theorem as described previously [43]. This ensemble size (9 neurons) corresponds approximately to a recent animal validation of the Standard Model [8].

Fourth, we apply three competing filtering algorithms to reconstruct trajectories. These are the point process filters that correspond to the GPFD-RSE, SM, and UM. The filters use the same observation model (3), but differ in their state space model – the component that describes the expected class of intended movements. The GPFD-RSE and SM were provided exact coordinates of the two target locations. The GPFD-RSE represented the potential set of arrival times between 1 and 3 seconds with discrete state variables corresponding to arrivals at 1.0, 1.7, 2.3, and 3.0 seconds at each target location.

REACH STATE EQUATION FORMULAS

Formulas for the RSE are derived in [4] and reproduced below for the reader's convenience. Refer to the reach state equation defined in equation (1) for basic definitions of B_k, $f_{k,}$

and ε_k. The free parameters of the general RSE are a target spread covariance matrix ($\prod_T$), a trajectory smoothness covariance matrix (Q_k), and a basic inertia matrix (A_k) that translates velocity into time. $\prod_T$ represents expected spread in each target endpoint, which should be orders of magnitude smaller than the size of the workspace because we have a collection of discrete targets in this application. Accordingly, we chose $\prod_T = 10^{-10} m^2 I_{4x4}$, where I_{4x4} is a 4x4 identity matrix and m^2 denotes square meters. Consistent with prior work [2, 4, 5], A_k and Q_k were fixed in time, with $A_k = \begin{bmatrix} 0 & 0 & \delta & 0 \\ 0 & 0 & 0 & \delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and $Q_k = (10^{-5} m^2) \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, where δ = 0.01 sec. was the time step used here.

The remaining expressions in the RSE are derived from $\prod_T$, Q_k, and A_k. $\phi(k,s)$ is a state transition matrix (4x4 in this paper's example) given by:

$$\phi(k,s) = \begin{cases} \prod_{i=1+\min(k,s)}^{\max(k,s)} A_i^{sign(k-s)}, k \neq s \\ \mathrm{I}, k = s \end{cases} \quad (4)$$

$\prod(k,T)$ is a set of recursively computed matrices (4x4 in this paper's example) given by:

$$\Pi(T,T) = \Pi_T + Q_T \quad (5)$$

$$\Pi(k-1,T) = \phi(k-1,k)\Pi(k,T)\phi'(k-1,k) + \phi(t-1,k)Q_{k-1}\phi'(k-1,k) \quad (6)$$

The B_k, f_k, and ε_k terms that appear in the RSE of equation (1) are written using the above terms:

$$B_k = [I - Q_k \Pi^{-1}(k,T)]A_k \quad (7)$$

$$f_k = Q_k \Pi^{-1}(k,T)\phi(k,T)y_T \quad (8)$$

$$\varepsilon_k : N\left(0, Q_k - Q_k \Pi^{-1}(k,T)Q_k'\right) \quad (9)$$

Refer to the original paper [4] for a detailed presentation.

GPFD Filter Equations

Refer to the introductory description of GPFD in II.B. Formulas for GPFD filtering are derived in [2] and reproduced below for the reader's convenience. We begin with basic notation.

For the k^{th} discrete time step of length δ_k seconds, the conditional intensity of neuron c is represented as λ_k^c, in units of spikes per second. Spiking activity at the k^{th} time step is summarized by a vector $n_k^{1:C} = \left(n_k^1, n_k^2, ..., n_k^C\right)$ of binned spike counts. The c^{th} element of $n_k^{1:C}$ contains the total number of spikes generated by the c^{th} neuron in the respective δ_k-second interval. Spike history is represented by $H_k = \left(n_1^{1:C}, n_2^{1:C}, ..., n_{k-1}^{1:C}\right)$.

Each iteration of the point process hybrid filter involves nine basic steps. The quantities $p(s_k \mid H_{k+1})$ and $p(x_k \mid s_k, H_{k+1})$ come from the previous iteration, where $p(s_1)$ and $p(x_1)$ are used instead for the first iteration. A Gaussian approximation to a probability density on the continuous state x_t is specified by a mean and covariance matrix. A probability mass function on the discrete state s_k is specified by a list of probabilities for each possible value of s_k.

Step 1. Compute $p(s_{k+1} \mid H_{k+1}) = \sum_{s_k} p(s_{k+1} \mid s_k) p(s_k \mid H_{k+1})$

Step 2. Compute $p(s_k \mid s_{k+1}, H_{k+1}) = \frac{p(s_{k+1} \mid s_k) p(s_k \mid H_{k+1})}{\sum_{s_k} p(s_{k+1} \mid s_k) p(s_k \mid H_{k+1})}$

Step 3. $p(x_k \mid s_{k+1}, H_{k+1}) = \sum_{s_k} p(s_k \mid s_{k+1}, H_{k+1}) p(x_k \mid s_k, H_{k+1})$.

Approximate this mixture of Gaussians as a single Gaussian (see section below for approximation formula).

Step 4. Calculate the Gaussian approximation to $p(x_{k+1} \mid s_{k+1}, n_{k+1}^{1:C}, H_{k+1})$ as follows. For each value that s_{k+1} can take on, send $p(x_k \mid s_{k+1}, H_{k+1})$ through one full iteration of a point process filter (see next section below) with $p(n_k^{1:C} \mid x_k, s_k, H_k) \propto \prod_{c=1}^{C} \exp(n_k^c \log(\lambda_k^c \delta_k) - \lambda_k^c \delta_k)$ and state equation $p(x_{k+1} \mid x_k, s_{k+1})$. Retain these densities (one Gaussian for each possible value of s_{k+1}) for the next iteration.

Step 5. Calculate

$$p(n_{k+1}^{1:C} \mid s_{k+1}, H_{k+1}) \approx \left. \frac{\left|W_{k+1|k+1,s_{k+1}}\right|^{1/2}}{\left|W_{k+1|k,s_{k+1}}\right|^{1/2}} \prod_{c=1}^{C} \exp(n_{k+1}^c \log(\lambda_{k+1}^c \delta_{k+1}) - \lambda_{k+1}^c \delta_{k+1}) \right|_{s_{k+1}=x_{k+1|k,s_{k+1}}}$$

Note $W_{k+1|k+1,s_{k+1}}$ and $W_{k+1|k,s_{k+1}}$ are posterior and prediction covariance terms from Step 4.

Step 6. Calculate

$$p(s_{k+1} \mid n_{k+1}^{1:C}, H_{k+1}) = \frac{p(n_{k+1}^{1:C} \mid s_{k+1}, H_{k+1}) p(s_{k+1} \mid H_{k+1})}{\sum_{s_{k+1}} p(n_{k+1}^{1:C} \mid s_{k+1}, H_{k+1}) p(s_{k+1} \mid H_{k+1})}.$$

Step 7.
Calculate

$$p(x_{k+1} \mid n_{k+1}^{1:C}, H_{k+1}) = \sum_{s_{k+1}} p(x_{k+1} \mid s_{k+1}, n_{k+1}^{1:C}, H_{k+1}) p(s_{k+1} \mid n_{k+1}^{1:C}, H_{k+1})$$

using steps 4 and 6.

Step 8. Place the actuator at a location defined by step 7 and your performance criteria. For example, use the mean of this distribution to minimize mean squared error.

Step 9. Return to step 1.

APPROXIMATE POINT PROCESS FILTER EQUATIONS

Step 4 of the specific GPFD filter equations used above requires implementation of an approximate point process filter that produces a Gaussian approximation to the posterior density. One example of such equations is provided in [2], derived in [33], and recapitulated below for the reader's convenience.

Consider a Gauss-Markov trajectory model $p(x_{k+1} \mid x_k)$: $N(F_k x_k + b_k, Q_k)$. The prediction density mean $x_{k+1|k}$ and covariance $W_{k+1|k}$ are:

$$x_{k+1|k} = F_k x_{k|k} + b_k \tag{10}$$

$$W_{k+1|k} = F_k W_{k|k} F_k^{'} + Q_k \tag{11}$$

The posterior density covariance $W_{k+1|k+1}$ and mean $x_{k+1|k+1}$ are:

$$(W_{k+1|k+1})^{-1} = (W_{k+1|k})^{-1} + \sum_{c=1}^{C}\left[\left(\frac{\partial \log \lambda_k^c}{\partial x_k}\right)^{'}\left[\lambda_k^c \delta_k\right]\left(\frac{\partial \log \lambda_k^c}{\partial x_k}\right) - \left(n_k^c - \lambda_k^c \delta_k\right)\frac{\partial^2 \log \lambda_k^c}{\partial x_k \partial x_k^{'}}\right]_{x_{k+1|k}} \tag{12}$$

$$x_{k+1|k+1} = x_{k+1|k} + W_{k+1|k+1}\sum_{c=1}^{C}\left[\left(\frac{\partial \log \lambda_k^c}{\partial x_k}\right)^{'}\left(n_k^c - \lambda_k^c \delta_k\right)\right]_{x_{k+1|k}} \tag{13}$$

GAUSSIAN APPROXIMATION TO MIXTURE OF GAUSSIANS

A mixture of Gaussians is a weighted average of R multidimensional Gaussians:

$$p(x) = \sum_{i=1}^{R} d_i N(x; \mu_i, W_i) \tag{14}$$

with weights d_i, and where $N(x; \mu_i, W_i)$ denotes the Gaussian probability density function with mean μ_i, and covariance W_i.

This distribution can be approximated by a single Gaussian by calculating the mean and covariance of $p(x)$ [44]:

$$p(x) \approx N(x; m, K) \tag{15}$$

where

$$m = \sum_{i=1}^{R} d_i \mu_i \tag{16}$$

$$K = \sum_{i=1}^{R} d_i \times [W_i + (\mu_i - m)(\mu_i - m)^{'}] \tag{17}$$

ACKNOWLEDGMENT

The authors thank Jernej Barbic (University of Southern California) for initial discussions on rendering and animation. LS thanks Rengaswamy, Uma, and Shyam Srinivasan for scientific discussions. LS wishes to acknowledge institutional support during this research period from Emad N. Eskandar (Massachusetts General Hospital), Alan Willsky (Massachusetts Institute of Technology), and Terrence J. Sejnowski (Salk Institute). Finally, LS acknowledges the UCLA Department of Radiology for institutional support, including Carole Barrinuevo, Chris L. Chang, Dieter Enzmann, Audrey Gallego, Robert D. Suh, and Fernando Vinuela.

REFERENCES

[1] L. Srinivasan, U. T. Eden, A. S. Willsky, and E. N. Brown, "Goal-directed state equation for tracking reaching movements using neural signals," *Proc 2nd Internat IEEE EMBS Conf on Neural Engineering*, pp. 352-355, 2005.

[2] L. Srinivasan, U. T. Eden, S. K. Mitter, and E. N. Brown, "General Purpose Filter Design for Neural Prosthetic Devices," *J Neurophysiol*, vol. 98, pp. 2456-2475, 2007.

[3] L. Srinivasan and E. N. Brown, "Dynamic-goal state equations for tracking reaching movements using neural signals," *1st IEEE RAS-EMBS Internat Conf on Biomed Robotics and Biomechatronics (BioRob '06)*, 2006.

[4] L. Srinivasan, U. T. Eden, A. S. Willsky, and E. N. Brown, "A State-Space Analysis for Reconstruction of Goal-Directed Movements using Neural Signals," *Neural Computation*, vol. 18 (10), 2006.

[5] L. Srinivasan and E. N. Brown, "A State Space Framework for Movement Control to Dynamic Goals Through Brain-Driven Interfaces," *IEEE Transactions on Biomedical Engineering*, vol. 54, pp. 526-535, 2007.

[6] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivasan, and M. A. Nicolelis, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature*, vol. 408, pp. 361-5, 2000.

[7] B. M. Yu, C. Kemere, G. Santhanam, A. Afshar, S. I. Ryu, T. H. Meng, M. Sahani, and K. V. Shenoy, "Mixture of Trajectory Models for Neural Decoding of Goal-Directed Movements," *J Neurophysiol*, 2007.

[8] G. H. Mulliken, S. Musallam, and R. A. Andersen, "Decoding trajectories from posterior parietal cortex ensembles," *J Neurosci*, vol. 28, pp. 12913-26, 2008.

[9] M. Weinrich and S. P. Wise, "The premotor cortex of the monkey," *J Neurosci*, vol. 2, pp. 1329-45, 1982.

[10] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner, "Neuronal population coding of movement direction," *Science*, vol. 233, pp. 1416-9, 1986.

[11] A. P. Batista, C. A. Buneo, L. H. Snyder, and R. A. Andersen, "Reach plans in eye-centered coordinates," *Science*, vol. 285, pp. 257-60, 1999.

[12] D. W. Moran and A. B. Schwartz, "Motor cortical representation of speed and direction during reaching," *J Neurophysiol*, vol. 82, pp. 2676-92, 1999.

[13] C. Kemere and T. H. Meng, "Optimal estimation of feed-forward-controlled linear systems," *Proc IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '05)*, vol. 5, pp. 353-356, 2005.

[14] C. Kemere, M. Sahani, and T. H. Meng, "Robust neural decoding of reaching movements for prosthetic systems," *Proc 25th Annual Meeting IEEE EMBS*, vol. 3, pp. 2079 - 2082, 2003.

[15] C. Kemere, K. V. Shenoy, and T. H. Meng, "Model-based neural decoding of reaching movements: a maximum likelihood approach," *IEEE Trans Biomed Eng*, vol. 51, pp. 925-32, 2004.

[16] J. Kulkarni and L. Paninski, "State-Space Decoding of Goal-Directed Movements," *IEEE Signal Processing Magazine*, vol. 25, pp. 78-86, 2008.

[17] 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. 141-2, 2002.

[18] D. M. Taylor, S. I. Tillery, and A. B. Schwartz, "Direct cortical control of 3D neuroprosthetic devices," *Science*, vol. 296, pp. 1829-32, 2002.

[19] 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 Biol*, vol. 1, pp. E42, 2003.

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

[21] E. C. Leuthardt, G. Schalk, J. R. Wolpaw, J. G. Ojemann, and D. W. Moran, "A brain-computer interface using electrocorticographic signals in humans," *J Neural Eng*, vol. 1, pp. 63-71, 2004.

[22] S. Musallam, B. D. Corneil, B. Greger, H. Scherberger, and R. A. Andersen, "Cognitive control signals for neural prosthetics," *Science*, vol. 305, pp. 258-62, 2004.

[23] 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. 164-71, 2006.

[24] R. Wahnoun, J. He, and S. I. Helms Tillery, "Selection and parameterization of cortical neurons for neuroprosthetic control," *J Neural Eng*, vol. 3, pp. 162-71, 2006.

[25] E. A. Pohlmeyer, E. R. Oby, E. J. Perreault, S. A. Solla, K. L. Kilgore, R. F. Kirsch, and L. E. Miller, "Toward the restoration of hand use to a paralyzed monkey: brain-controlled functional electrical stimulation of forearm muscles," *PLoS One*, vol. 4, pp. e5924, 2009.

[26] T. M. Cowan and D. M. Taylor, "Predicting reach goal in a continuous workspace for command of a brain-controlled upper-limb neuroprosthesis," *Proc 2nd Internat IEEE EMBS Conf on Neural Engineering*, pp. 74, 2005.

[27] J. R. Wolpaw and D. J. McFarland, "Multichannel EEG-based brain-computer communication," *Electroencephalogr Clin Neurophysiol*, vol. 90, pp. 444-9, 1994.

[28] N. Birbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kubler, J. Perelmouter, E. Taub, and H. Flor, "A spelling device for the paralysed," *Nature*, vol. 398, pp. 297-8, 1999.

[29] J. R. Wolpaw and D. J. McFarland, "Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans," *Proc Natl Acad Sci U S A*, vol. 101, pp. 17849-54, 2004.

[30] P. Viola and M. J. Jones, "Robust Real-time Object Detection," *HP Labs Technical Reports*, vol. CRL-2001-1, http://www.hpl.hp.com/techreports/Compaq-DEC/CRL-2001-1.html, 2001.

[31] G. Kreiman, C. Koch, and I. Fried, "Imagery neurons in the human brain," *Nature*, vol. 408, pp. 357-61, 2000.

[32] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, "A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects," *J Neurophysiol*, vol. 93, pp. 1074-89, 2005.

[33] U. T. Eden, L. M. Frank, R. Barbieri, V. Solo, and E. N. Brown, "Dynamic analysis of neural encoding by point process adaptive filtering," *Neural Computation*, vol. 16, pp. 971-98, 2004.

[34] 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 Comput*, vol. 18, pp. 80-118, 2006.

[35] U. T. Eden, W. Truccolo, M. R. Fellows, J. P. Donoghue, and E. N. Brown, "Reconstruction of hand movement trajectories from a dynamic ensemble of spiking motor cortical neurons," *Proc 26th IEEE Engineering in Medicine and Biology Society Annual Conference (EMBC '04)*, vol. 2, pp. 4017- 4020, 2004.

[36] 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," *J Neurophysiol*, 2010.

[37] S. Adee, "The revolution will be prosthetized," *IEEE Spectrum*, vol. January 2009, 2009.

[38] L. Srinivasan and M. J. da Silva, "Online Movie: Variable arrival time reaching movements of an overactuated virtual 3D robotic arm operating in a 2D workspace driven by three prosthetic algorithms." http://hdl.handle.net/1721.1/60298, 2010.

[39] S. R. Buss, "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods," *University of California San Diego online instructional material*, 2004.

[40] S. R. Buss, "3-D computer graphics a mathematical introduction with OpenGL." Cambridge, Eng. ; New York: Cambridge University Press, 2003.

[41] E. N. Brown, L. M. Frank, D. Tang, M. C. Quirk, and M. A. Wilson, "A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells," *J Neurosci*, vol. 18, pp. 7411-25, 1998.

[42] M. M. Churchland, G. Santhanam, and K. V. Shenoy, "Preparatory activity in premotor and motor cortex reflects the speed of the upcoming reach," *J Neurophysiol*, vol. 96, pp. 3130-46, 2006.

[43] E. N. Brown, R. Barbieri, V. Ventura, R. E. Kass, and L. M. Frank, "The time-rescaling theorem and its application to neural spike train data analysis," *Neural Comput*, vol. 14, pp. 325-46, 2002.

[44] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation : theory, algorithms and software*. New York: Wiley, 2001.

Lakshminarayan Srinivasan (M'05) received the B.S. degree in electrical and computer engineering (with honors) from the California Institute of Technology, Pasadena, CA, the S.M. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, and the M.D. degree from Harvard Medical School. He was also a postdoctoral researcher at Massachusetts General Hospital, and subsequently a visiting scientist at the Salk Institute. He is currently a principal investigator in the Department of Radiology at the David Geffen School of Medicine, University of California Los Angeles, and a research affiliate of the MIT Laboratory for Information & Decision Systems.

Marco da Silva received the Sc.B. degree in mathematics and computer science (with honors) from Brown University, Providence, RI and the S.M. and Ph.D. degrees in computer science from the Massachusetts Institute of Technology, Cambridge, MA. He is currently a robotics engineer at Boston Dynamics, Waltham, MA.