2021 43rd Annual International Conference of the
IEEE Engineering in Medicine & Biology Society (EMBC)
Oct 31 - Nov 4, 2021. Virtual Conference

Intention Estimation Based Adaptive Unscented Kalman Filter for Online Neural Decoding

Han Wei Ng, Brian Premchand, Kyaw Kyar Toe, Camilo Libedinsky, Rosa Q. So.

***Abstract*— The commonly used fixed discrete Kalman filters (DKF) in neural decoders do not generalize well to the actual relationship between neuronal firing rates and movement intention. This is due to the underlying assumption that the neural activity is linearly related to the output state. They also face the issues of requiring large amount of training datasets to achieve a robust model and a degradation of decoding performance over time. In this paper, an adaptive adjustment is made to the conventional unscented Kalman filter (UKF) via intention estimation. This is done by incorporating a history of newly collected state parameters to develop a new set of model parameters. At each time point, a comparative weighted sum of old and new model parameters using matrix squared sums is used to update the neural decoding model parameters. The effectiveness of the resulting adaptive unscented Kalman filter (AUKF) is compared against the discrete Kalman filter and unscented Kalman filter-based algorithms. The results show that the proposed new algorithm provides higher decoding accuracy and stability while requiring less training data.**

I. Introduction

In brain-machine interfaces (BMI), the conventionally used linear algorithms for neural decoding such as a linear discrete Kalman filters (DKF) utilizes methods that assumes a linear relationship between the neural activity and desired output [1]. Non-linear algorithms are generally considered to be more superior to linear decoders due to their ability to represent the true relationship between neural activity and desired output more accurately [2]. In particular, the unscented Kalman filter (UKF) is one of the most favored choice for non-linear neural decoding algorithms as the UKF is able to perform non-linear state estimations [3] while remaining computationally inexpensive compared to other non-linear models for online closed-loop neural decoding control [4].

However, the use of UKF still does not address the issues of requiring a large amount of training dataset in order to produce a robust model for accurate decoding [5] and the fixed nature of the model would lead to degradation of the model's accuracy over time [6]. These downsides associated with the implementation of the UKF can result in long training time for the user and decrease in the efficacy of the BMI over time.

In this paper, an adaptive unscented Kalman filter (AUKF) will be developed through the implementation of an adaptive adjustment to the UKF based on intention estimation. The adaptive modification is made through a combination of two methods. Firstly, the use of intention estimation at each time point using the user's current actuator position and the ideal target position to approximate the ideal movement vector of the actuator. Secondly, a matrix variable is added to contain the history of neural firing rates in a set of consecutive time points. The two methods combined allows for an adaptive tuning of the model that is easy to implement and computationally cheap for real-time online neural decoding.

Similar intention estimation methods have been used earlier in producing an adaptive DKF such as the recalibrated feedback intention-trained Kalman Filter (reFIT-KF) [7]. However, key differences in the proposed method include the use of a non-linear filter to perform state estimation as well as the ability for the learning rate to tune itself according to the performance of the user. The proposed algorithm also displays low run-time while utilising adaptation on a high nth- order UKF. Although other forms of adaptive filters exist, many of them have fixed learning rate algorithms or utilize linear regression methods [8].

II. Experimental Setup

In this study, a macaque (*Macaca fascicularis*) is trained to perform a cursor movement task. The macaque operates a joystick control capable of 360-degree motion to move a virtual cursor on a 2-dimensional screen. The goal is positioned some distance away from the cursor start position. The controlled virtual cursor starts at the centre of the screen and the target locations can be any one of the 8 pre-determined goals located radially around the centre of the screen.

All animal procedures were approved by and conducted in compliance with the standards of the Agri-Food and Veterinary Authority of Singapore and the National University of Singapore Institutional Animal Care and Use Committee (NUS IACUC R18-0295). The procedures also conformed to the recommendations described in Guidelines for the Care and Use of Mammals in Neuroscience and Behavioral Research [9].

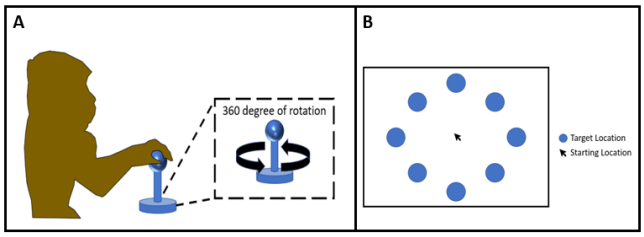

Figure 1. (A) Joystick control of virtual cursor for cursor movement. (B) The cursor movement task involves moving the cursor from the centre to any one of eight target locations.

Multiple electrode arrays are implanted in the macaque in the hand/arm area of the primary motor cortex. Following which, the macaque is trained to use a joystick to control the virtual cursor. During the training, the implanted electrodes recorded the multi-unit spiking activity of the neurons. The

number of detected spikes is recorded in 100 ms steps in a 500 ms time window. The total number of these spikes is then divided by the length of the window to compute the firing rates for the time window.

The corresponding position of the cursor and joystick values are recorded as well. Within a single training session, a trial consists of the macaque moving the cursor from the start position to the target location. The recorded cursor positions can subsequently be plotted out to obtain the trajectory of the cursor throughout the training session.

The recorded neural firing rates during joystick control and their corresponding joystick values are subsequently used to train the neural decoding models. The neural decoders are implemented using MATLAB (Mathworks Inc, Massachusetts, USA). In total, four sessions of joystick control data are used for the training and testing of the neural decoders in this study. Of the four sessions, three sessions contained 16 successful trials, and one session contained 40 successful trials.

III. Discrete Kalman and Unscented Kalman Filters

A. Discrete Kalman Filter

A typical linear DKF can be broken down into two main steps, the prediction step, and the updating step [10]. In the prediction step, the *a priori* estimate of the predicted state, $x_{\bar{t}}$, is obtained through applying a linear state model on the latest known state estimate x_{t-1}. The predicted error covariance of the matrix, $P_{\bar{t}}$ is then computed.

In the updating step, the algorithm will correct the predicted state using Bayesian methods. The correction is done by using the observed measurements at the current time point and update the estimated state through computation of the posterior state estimate given the observed measurements. Full details of the DKF equations can be found in [10].

B. Unscented Kalman Filter

The UKF follows a similar mathematical process as the DKF. However, where the UKF differs from the DKF is that rather than assuming an underlying linear relationship, the UKF uses a non-linear function on Gaussian distributed random variables to calculate approximate solutions to the estimated state and error covariance [11].

Although the prediction step in the UKF normally utilizes a non-linear state model to predict the new state estimate and error covariance, for the purposes of this study the linear state model performs better and offers greater stability in translated movement. Thus, the prediction step of the UKF in this study is identical to that of the DKF.

For the updating step, the UKF generates a series of sigma points from the predicted state estimate and the state error covariance. Non-linear estimation is achieved by passing the original sigma points through a fixed non-linear function. The neural tuning model is then used to calculate a new best approximate gaussian from the transformed sigma points. The non-linear function used in this study relates the axis velocities of the cursor to the magnitude and is given by:

$$\sigma = (V_x, V_y, \sqrt{{V_x}^2 + {V_y}^2}) \quad (1)$$

The mean, covariance of the predicted neuronal firing rates and the state observation cross-covariance are then computed using the sigma points and the noise covariance of the observation model via a weighted sum.

Similar to the DKF, the Kalman gain is computed through the use of the predicted state error covariance and state observation cross-covariance. Full details of the original UKF equations can be found in [11].

IV. Intention Estimation Based Machine Learning

The adaptive modifications consist of two main parts. In the first part, the algorithm will approximate the movement intention of the subject based on the user's current position and the target position. In the second part, the algorithm will update the parameters using the newly measured neural firing rates and the newly calculated estimated intent of movement. This will be done through an update function which will compute a new set of parameters for the algorithm to use in subsequent steps of the process.

A. Intention Estimation

In the context of a cursor movement task via controlling a cursor on a 2D screen to reach a target, the ideal movement to reach the target location will be a straight line from the cursor's present location to the target location.

For the purposes of intention estimation, the intended movement of the cursor by the subject at each time point could be ideally approximated to be equal to that of the straight line towards the target location. This is a fair approximation since it can be assumed that for majority of the cursor movement task, the subject is trying to reach the target as fast as possible to maximise its rewards obtained.

Thus, since for a majority of the time the actual intended movement of the cursor can be closely approximated by the ideal straight-line movement towards the target location, the brain signals recorded at each time point can be said to closely represent the neural firing rates for the ideal straight-line movement.

The current position of the cursor at any point in time can be denoted as a 2D vector coordinate pair, (X_t, Y_t). The vector coordinates of the target location can be similarly denoted as a fixed coordinate pair (x_c, y_c), where x_c and y_c are fixed constants respectively representing the location of the target along the X and Y axis of the screen. The target location coordinates remain fixed for a single cursor movement task and would change for every new cursor movement task.

The estimated intended movement vector can then be computed using a simple calculation given by:

$$(V_x, V_y) = (x_c, y_c) - (X_t, Y_t) \quad (2)$$

Where V_x and V_y are the estimated intended velocities of the cursor in the X and Y axis respectively. V_x is taken to be positive in the right direction and V_y in the upwards direction.

In the instance whereby the cursor has entered within a predefined distance from the centre of the target location as shown by the blue circle in Fig. 3, the neuronal firing signal y_t can be approximated to be a stopping intention and its associated velocity vectors V_x and V_y are both set to be 0.

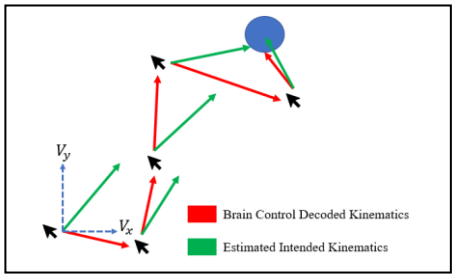

Figure 2. Example ideal movement vector (green) by the cursor on a 2-Dimensional screen at different timepoints in a single task.

The estimated intended velocity vector is then stored into a matrix, V_t , containing the vectors at time *t* through a series of concatenation. The intended velocity vector can then be paired with the measured neuronal signals at time *t* to form a single data pair linking the estimated intended velocity vector to the neuronal firing rates. To ensure robustness of the re-computed model parameters, a total of 100 data pairs is collected through concatenation of the data at each timepoint and stored in a matrix to be used for the adaptive learning function. At every iteration, the algorithm computes and collects a new set of neuronal firing rates and estimated intended velocity vector. Once 100 data pairs have been reached, the oldest data pair containing V_{t-99} and its associated neuronal firing rates y_{t-99} are removed and the newest pair is included. Thus, the updating algorithm achieves a history of 100 data pairs from the most recent timepoint *t* to *t-99*.

B. *Auto-tuning of Algorithm Parameters*

The AUKF performs the auto-tuning of the algorithm parameters at the end of a single iteration. The update function takes in the matrices containing the 100 data pairs of neuronal firing rates and estimated intended velocity as well as the static parameters for updating, namely the neural tuning model (*h*), the noise covariance of the observation model (*W*), the linear state model (*A*), and the associated noise covariance (*Q*). Since non-linear estimation is achieved by (1), the neural tuning model portion can be adapted through linear regression.

The update algorithm first computes a set of new parameters using solely the 100 data pairs given. First, the neural tuning model and its noise covariance are calculated by the following equations:

$$h_{\text{calc}} = y_{t..t-99} V_{t...t-99}{}^{T} (V_{t...t-99} V_{t...t-99}{}^{T})^{-1} \quad (3)$$

$$W_{calc} = \frac{(y_{t...t-99} - hV_{t...t-99})(y_{t...t-99} - hV_{t...t-99})^{T}}{N} \quad (4)$$

Where *V* represents the estimated intended velocity coordinate pair, *N* represents the history length and h_{calc} and W_{calc} are the newly calculated neuronal observation model and its associated noise covariance, respectively.

Next, the state model and its noise covariance are then calculated using the following equations:

$$A_{calc} = V_{t\text{-}1...t-99} V_{t...t-98}{}^{T} (V_{t...t-98} V_{t...t-98}{}^{T})^{-1} \quad (5)$$

$$Q_{calc} = \frac{(V_{t\text{-}1...t-99} - AV_{t...t-98})(V_{t\text{-}1...t-99} - AV_{t...t-98})^{T}}{N-1} \quad (6)$$

Where A_{calc} and Q_{calc} are the newly calculated state model and its noise covariance.

The set of calculated model parameters can then be used to update the original set of parameters through a weighted sum:

$$S_{i+1} = (1 - \alpha) S_i + \alpha S_{calc} \quad (7)$$

Where S_{i+1} represents the newly computed set of model parameters using the old set of parameters, S_i, and the latest calculated set of parameters, S_{calc}.

The weight, α, is computed anew at each iteration using the sum of squared errors between the old and calculated neuronal observation models. The weight, α, is thus given by:

$$\alpha = [a(1 - \frac{SSE}{b})]^{-1} \quad (8)$$

Where *a* and *b* are constants and have a numerical value greater than 1. The term *SSE* represents the sum of squared errors between the old and calculated neuronal observation models within a single iteration of the UKF algorithm.

In this manner, the learning rate of the algorithm changes accordingly to how different the old and calculated neural observation models are at each iteration of the UKF algorithm. A bigger difference between the observation models would result in a larger sum of squared errors and therefore α will increase but its maximum value will remain below 1. As a result, the weight given to the newly calculated parameters increases whenever the algorithm detects a bigger difference in the old and calculated observation models.

Once the weight has been calculated, the original parameters are then updated accordingly using the weighted sum method in (7). The updated parameters are subsequently returned to the algorithm to be used in the following iteration, thus the UKF algorithm uses a re-tuned set of parameters for online neural decoding which would have been otherwise static in the standard UKF algorithm.

V. Testing Results and Discussion

The AUKF developed is tested alongside the DKF and the UKF algorithms in an offline simulation on MATLAB using pre-recorded brain signals of a monkey performing a simple linear cursor movement task using joystick control to move a virtual cursor towards a target location. A total of four sessions are used for the testing with three sessions containing 16 successful trials, and one session containing 40 successful trials.

For each of the filters, a four-fold cross validation is done whereby the starting algorithm model parameters are trained using only a single session's worth of data, leaving three other sessions available for testing. This is to emulate the usage of the BMI with low amounts of training data for the purpose of reduced training timing. The trained filter is then tested on the other three trials that are not used in the training phase. The accuracy is determined by the correlation coefficient in the X and Y axis between the recorded joystick values and the decoded joystick values.

Results showed that while the UKF is only slightly advantageous over the DKF in terms of neural decoding accuracy, the AUKF shows significant decoding performance improvements compared to the other two filters. The UKF has long been established to be superior to the DKF due to its

ability to perform non-linear state estimation [3]. However, due to the low amounts of initial training data used for the calibration of the models, this leads to poorly trained models which would result in worse performing algorithm models in both filters. Any benefit that the UKF has over the DKF becomes obscured which is reflected in the small difference between the poorly trained DKF and UKF.

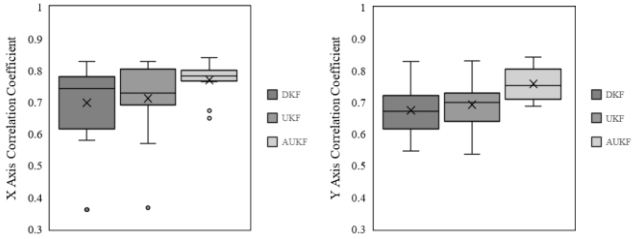

Figure 3. Comparison of correlation coefficient values of test sessions between the Discrete Kalman Filter, Unscented Kalman Filter and Adaptive Unscented Kalman Filter in the X and Y axis.

The advantage for the AUKF under low training scenarios is clear where it is able to outperform both the DKF and UKF in almost every tested session. This is to be expected since the AUKF is able to re-train and update its model towards newly received data and therefore an initially poorly trained model can be tuned towards new changes.

The different algorithms are also tested for the average run-time per time point in each trial across the sessions (Intel Core i7-7700HQ 2.8GHz, 2 x 8GB DDR4 RAM). Since the number of spikes are recorded using 100 ms steps, a low run-time of below 100 ms is necessary to perform online neural decoding without experiencing any lag time.

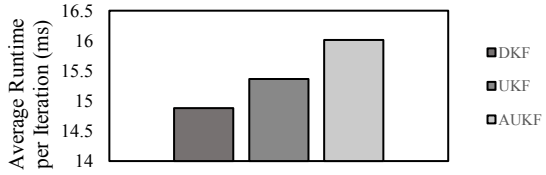

Figure 4. Comparison in average execution time per iteration.

From the comparisons, the DKF has the fastest runtime followed by the UKF and finally the AUKF. This is likely due to the higher time complexity of the AUKF algorithm because of the adaptive modifications made. Considering the inexpensive nature of the modifications, the time increment is minimal, and the average runtime remains well below 100 ms. Therefore, the AUKF is capable of performing online neural decoding with no noticeable lag time. The small increment in runtime may be well worth for the ability to continuously adapt the model parameters through the AUKF algorithm.

VI. Conclusion

In this study, we developed an intention estimation based adaptive training function that shows how a combination of intention estimation and a concatenated history of neural data can be used to perform a weighted adjustment of the model parameters at every iteration. Through the dynamic adaption of the parameters, the algorithm can capture and incorporate fresh data and adapt towards changes in the environment in an unsupervised manner. This better reflects real-world scenarios whereby the usage of BMIs is often under the context of constantly changing dynamic environments. An adaptive model of the UKF will therefore display superior decoding abilities and stability compared to the static model counterparts and brings the added benefit of requiring less initial training. The offline comparison of the algorithms has thus demonstrated that there is promise in working towards decoding models capable of adaption and the method proposed in this paper is just one of many possibilities.

Future possible improvements include potentially combining intention estimation alongside retrospective estimation. The idea of retrospective estimation is like that of intention estimation, only that the velocity vectors are found after the controlled actuator has reached the target which is useful in scenarios where targets are not known *a priori*. Since both methods are suitable for cursor movement-based tasks, a combination of both methods might lead to improved performance. In the event where there are multiple possible targets, a target determining algorithm can be implemented using the distance from targets to initial brain control decoded kinematics as the deciding condition. In addition, online closed-loop testing of the AUKF will have to be done to reaffirm its effectiveness in actual BMI use.

References

[1] W. Wu, Y. Gao, E. Bienenstock, J. Donoghue and M. Black, "Bayesian Population Decoding of Motor Cortical Activity Using a Kalman Filter", *Neural Computation*, vol. 18, no. 1, pp. 80-118, 2006. Available: 10.1162/089976606774841585.

[2] Y. Gao, M. Black, E. Bienenstock, W. Wu and J. Donoghue, "A quantitative comparison of linear and non-linear models of motor cortical activity for the encoding and decoding of arm motions", First International IEEE EMBS Conference on Neural Engineering, 2003. Conference Proceedings.

[3] Z. Li, J. O'Doherty, T. Hanson, M. Lebedev, C. Henriquez and M. Nicolelis, "Unscented Kalman Filter for Brain-Machine Interfaces", *PLoS ONE*, vol. 4, no. 7, p. e6243, 2009.

[4] T. Luu, Y. He, S. Brown, S. Nakagome and J. Contreras-Vidal, "Gait adaptation to visual kinematic perturbations using a real-time closed-loop brain–computer interface to a virtual reality avatar", *Journal of Neural Engineering*, vol. 13, no. 3, p. 036006, 2016.

[5] C. Huang, Z. Wang, Z. Zhao, L. Wang, C. Lai and D. Wang, "Robustness Evaluation of Extended and Unscented Kalman Filter for Battery State of Charge Estimation", *IEEE Access*, vol. 6, pp. 27617-27628, 2018.

[6] D. Brandman, M. Burkhart, J. Kelemen, B. Franco, M. Harrison and L. Hochberg, "Robust Closed-Loop Control of a Cursor in a Person with Tetraplegia using Gaussian Process Regression", *Neural Computation*, vol. 30, no. 11, pp. 2986-3008, 2018.

[7] V. Gilja et al., "A high-performance neural prosthesis enabled by control algorithm design", Nature Neuroscience, vol. 15, no. 12, pp. 1752-1757, 2012.

[8] M. Shanechi, "Brain–Machine Interface Control Algorithms", IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 25, no. 10, pp. 1725-1734, 2017.

[9] Van Sluyters, R. C., & Obernier, J. A, "Guidelines for the care and use of mammals in neuroscience and behavioral research.", *Contemporary topics in laboratory animal science*, *43*(2), 48-+.

[10] S. Enshaeifar, L. Spyrou, S. Sanei and C. Took, "A regularised EEG informed Kalman filtering algorithm", *Biomedical Signal Processing and Control*, vol. 25, pp. 196-200, 2016.

[11] R. Kandepu, B. Foss and L. Imsland, "Applying the unscented Kalman filter for nonlinear state estimation", Journal of Process Control, vol. 18, no. 7-8, pp. 753-768, 2008.