Neural Decoding for Macaque's Finger Position: Convolutional Space Model

Haifeng Wu, Jingyi Feng, and Yu Zeng

***Abstract*— In this paper, we study how to use the number of spike signals in a macaque's motor cortex to estimate the position of its finger movement. First, we analyze the time correlation of a traditional state space model (SSM) and derive a convolutional space model (CSM) to decode the movement position of the macaque finger. Compared with the traditional model, the model can correlate the current moment state with the previous moment. In addition, we have improved the original SSM model using the CSM model. In its observation equation, the number of spike signals at one is expanded to the number of spike signals at multiple previous moments. In this way, the current time state in the improved model can be related to the number of spike signals at a plurality of previous times. In the experiment, a group of public data are used to validate the decoding performance of the model, and a least squares, a batch recursive least squares, and a gradient descent algorithms are used to train the model. The experimental results show that the decoding error of the CSM model and the improved SSM model is smaller than that of the traditional model, and thus, they have a better decoding accuracy. The results show that the CSM model has improved about 11.7% in x-axis decoding error performance than the traditional models.**

***Index Terms*— Neural decoding, TILM model, SSM model, CSM model.**

I. Introduction

NEURAL coding and decoding is an important field in neuroscience. The study of neural circuits and how to perceive external world and then generate behavior will reveal the working mechanism and rule of a brain, and can also make human body enhance the ability to sense and control the external world. At present, neural coding and decoding have been widely used in rehabilitation engineering [1]. In neural coding, for example, an artificial cochlear is implanted into the ear of a hearing-impaired person, and the sound signal is encoded as a digital signal that can stimulate the auditory nerve so that the patient has the ability to perceive sound [2]. In neural decoding, disabled people can use the spike signal from their cerebral motor cortex to directly control the movement of the device and realize brain control technology and intelligent life [3], [4], such as controlling the mouse, paraplegic patients' prosthesis, robot arm [5], [6] and so on.

Neural coding maps the external world to brain activity [7], and need to sort spike signals from brain regions via some classification methods. Then, the sorted spike signals will be related with bone or muscle action to the external world. Neural decoding is the inverse process of neural coding, and parses the human body's actions to the outside world from the brain activity. For example, the movement process of the body can be predicted or estimated through the sorted spike signals. The position estimation of a macaque's finger movements in this paper is a typical neural coding and decoding problem. The first step is neural coding, where the number of spike signals from a macaque's motor cortex will be related to the moving position of the macaque finger. Since the step has been discussed much in [8] and [9], this paper will no longer focus on it. The next step is neural decoding, where the moving finger's position is estimated through the established relationship between the number of spikes signals and the finger position. This paper will focus on how to better solve the problem of neural decoding.

The coding problem of the macaque finger movement is earlier described in [10], which find that there is a relationship between the direction and location of upper limb movement of a macaque and the spike signal in its motor cortex. The relationship is also confirmed in later literatures [11]. In traditional decoding methods, a time-independent linear method [12] is adopted earlier. The method uses a time-invariant linear model (TILM), where a movement state at one moment and a spike signal recorded at the moment are regarded as a time-invariant and proportional relationship. The advantage of the method is to implement and calculate easily. However, the accuracy of the estimation is not high since the state at each moment in a motion trajectory is considered as an independent process.

Now, a popular decoding method is to use a state space model (SSM) to solve the decoding problem and SSM has already been widely used in neuroscience [13].

Manuscript received June 10, 2018; revised October 18, 2018 and November 18, 2018; accepted January 9, 2019. Date of publication January 21, 2019; date of current version March 22, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61762093, in part by the 17th Batches of Young and Middle-Aged Leaders in Academic and Technical Reserved Talents Project of Yunnan Province under Grant 2014HB019, in part by the Key Applied and Basic Research Foundation of Yunnan Province under Grant 2018FA036, in part by the Program for Innovative Research Team (in Science and Technology), University of Yunnan Province, and in part by the Science Research Fund Graduate Program in Education Department of Yunnan Province under Grant 2018Y106. *(Corresponding author: Jingyi Feng.)*

The authors are with the Department of Electrical and Information Engineering, Yunnan Minzu University, Kunming 650500, China (e-mail: whf5469@gmail.com; fengjingyione@foxmail.com; yv.zeng@gmail.com).

Digital Object Identifier 10.1109/TNSRE.2019.2893406

Shanechi *et al.* [14] establish an SSM model and use an optimal feedback control to decode the state of a macaque movement. Aghagolzadeh*et al.* [15] use an SSM model to decode 3D stretching and grasping motions from a macaque's neuron activity in primary motor cortex. Feng *et al.* [16] use an unsupervised cubature Kalman filter (UCKF) to decode the position of a macaque's finger movements via an SSM model. Hotson *et al.* [17] use a recursive Bayesian estimate (RBE) to improve the decoding performance. Brockwell *et al.* [18] use successive state estimates to decode motor cortical signals. From the observation equation in the models for neural decoding, SSM is still a kind of TILM model. Compared with the independent linear method, however, SSM lets a movement state at one moment correlated with that at one previous moment, instead of regarding the finger movement state in the trajectory as a time-independent process. Thus, the estimation accuracy has been greatly enhanced.

After analyzing the temporal correlation of the TILM models, this paper will start from an SSM model, correlate a movement state at one moment with multiple states at several previous moments, and obtain another TILM model. The model expresses the moving position of a macaque's finger as a convolution of the spike signal vector and a group of constant coefficients, called a convolutional space model (CSM). The CSM is still based on the SSM model, except that the number of spike at a current state is extended to that at multiple previous moments. In order to solve the model, some popular algorithms such as a least squares (LS), a steepest gradient descent algorithm (GDA) and a Kalman filter are used to obtain the model parameters. At the same time, the influence of the temporal correlation on CSM model is also analyzed. In experiments, we use a group of public data recording real neural spike signals of a monkey's finger movement to verify the temporal correlation. The experimental results show that when the temporal correlation is considered, the decoding errors of the finger movement position in CSM are less than those in traditional models, such as the independent linear model and SSM model.

The rest of this paper is organized as follows. Section II contains the related work about an SSM model. Section III describes the decoding problem. We derive the CSM model and analyze the impact of temporal correlation on the model training in Section IV. Section V gives decoding algorithm steps for the CSM model and Section VI shows experimental results. In Section VII, We discuss some problems for the CSM model. Finally, conclusions are drawn in Section VIII.

II. Related Work: SSM Model

SSM is currently a very popular model in the neural decoding for macaque finger movement, and its state equation and observation equation is expressed as [7] and [17]–[19].

$$y_k = h(y_{k-1}) + \omega_k \tag{1-a}$$
$$\mathbf{s}_k = \mathbf{f}(y_k) + \mathbf{v}_k \tag{1-b}$$

where y_k is the position information of finger movement at time k, $k = 0, 1, \ldots K-1$, K is the total number of sampling points, $\mathbf{s}_k$ is a $N_e \times 1$ column vector which denotes the number of neuron spikes collected from N_e electrodes at time k, $h(\cdot)$ represents the function of a state equation, $\mathbf{f}(\cdot)$ represents the function of an observation equation, ω_k is a Gaussian white noise with a zero mean and variance σ^2 [16], $\mathbf{v}_k = [v_{0k}, v_{1k}, \ldots, v_{N_e-1,k}]^{\mathrm{T}}$ is a Gaussian white noise vector with a zero mean and a variance matrix $\mathbf{R}_v$, which is a diagonal matrix with a diagonal vector $[\sigma_0^2, \sigma_1^2, \ldots, \sigma_{N_e-1}^2]$ [16]. Usually, successive state estimation method could solve the state-space equation (1), such as Kalman filter (KF) [19], particle filter [18] and RBE [17]. Compared with the linear method [7] that treats $\{y_k, k = 0, 1, \ldots K-1\}$ as an independent process, SSM model would produce a more stable and smoother decoding curve for finger movement position.

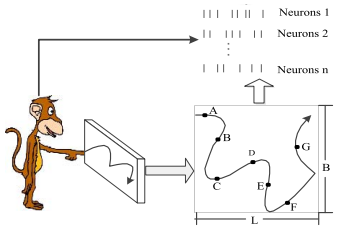

Fig. 1. Macaque finger movement track coding.

In addition, SSM model needs to know the functions $h(\cdot)$ and $\mathbf{f}(\cdot)$. A common method is to regard the observation function as linear ones and use training data to obtain them. In this case, (1) becomes

$$y_k = y_{k-1} + \omega_k \tag{2-a}$$
$$\mathbf{s}_k = \mathbf{a}_k y_k + \mathbf{b}_k + \mathbf{v}_k \tag{2-b}$$

where $\mathbf{a}_k$ and $\mathbf{b}_k$ are both $N_e \times 1$ column vectors. Since the coefficient vectors are constant, (2-b) is still a TILM model. In addition, an unsupervised UCKD method [16] rewrites (1) into two groups of state space equations, one for solving the finger movement position and the other for solving the $\mathbf{f}(\cdot)$ function, so that no training data are needed. From the SSM model on (1)-(2), the position state y_k at the current moment is only correlated to the previous moment state y_{k-1}. In the next section, we will analyze SSM model from the temporal correlations.

III. Problem for Decoding Movement Position and Temporal Correlation of Model

The estimation of a macaque's finger movement position is a typical neural coding and decoding process. First, we need to establish a one-to-one relationship between the coordinates of the finger movement at one moment and the number of spike signals from the macaque's brain at the moment. Second, via the established relationship, we will estimate the coordinates of the movement from the number of spikes. The illustration of the problem is shown in Fig.1.

Select a macaque and let its finger move in a screen of length L and width B. When a target appears on the screen, the macaque's finger will move to the target. Then, the target disappears and the macaque's finger moves to the next one. The training will repeat until the macaque is able to complete the task. Besides, we need to implant arrays of electrodes in

the brain regions associated with the finger movements. If the sampling period of the data is Δt, the electrodes will collect signals $\mathbf{s}_k$ with N_e neuron signals corresponding to the finger movement at time $k\Delta t$, $k = 0, 1, \ldots K-1$.

In addition, we also need to record the macaque's finger movement position at each moment. If the finger moves to point A at time $k\Delta t$ and the corresponding number of spike signals is $\mathbf{s}_k$, we will record an X or Y coordinate corresponding to the finger movement at the time, y_k. If the macaque finger movement spend a total of time $(K-1)\Delta t$, we will obtain the spike signals $\mathbf{s}_k$ and position coordinates y_k, $k = 0, 1, \ldots K-1$. In fact, the decoding is to obtain $\hat{y}_k$ from $\mathbf{s}_k$ and make $\hat{y}_k$ and y_k as similar as possible.

From the finger movement process of the macaque, the trajectory $\{y_k, k = 0, 1, \ldots, K-1\}$ of the moving position should be a continuous process with a speed and direction. It may be not optimal that traditional decoding methods based on the SSM model in (2) correlates the position state at one current moment only with that at one previous moment. The movement state at one current moment should be correlated with multiple states at several previous moments. In order to find a model with better temporal correlation, this paper attempts to establish a correlation between the current position state and the several previous states.

IV. Temporal Correlation of Time Invariant Linear Model

A. CSM Model

This section will focus on the temporal correlation of a decoding model, where we let the current moment state correlated with several previous moments. In the derivation of the model, we will start from the SSM model, and obtain a convolutional space model. If the position state y_k at time k in our model is correlated with the previous P moments k, $k-1\ldots$, k-$P+1$, then from (2-b), the position coordinates at time k-p will be

$$y_{k-p} = \mathbf{w}_p^{\mathrm{T}}\mathbf{s}_{k-p} + b'_p + \omega'_p, \quad p = 0, 1 \ldots P-1 \tag{3}$$

where $\mathbf{w}_p^{\mathrm{T}} = \mathbf{a}_{k-p}^{\dagger}$ is a row coefficient vector, $(\bullet)^{\mathrm{T}}$ denotes conjugate transpose, $(\bullet)^{\dagger}$ denotes a pseudo inverse, $b'_p = -\mathbf{w}_p^{\mathrm{T}}\mathbf{b}_{k-p}$, $\omega'_p = -\mathbf{w}_p^{\mathrm{T}}\mathbf{v}_{k-p}$ is still zero-mean Gaussian white noise [16]. Substituting (2-a) into (3) get

$$\begin{cases} y_k = \mathbf{w}_0^{\mathrm{T}}\mathbf{s}_k + b'_0 + \omega'_0 \\ y_k = \mathbf{w}_1^{\mathrm{T}}\mathbf{s}_{k-1} + b'_1 + \omega''_1 \\ \vdots \\ y_k = \mathbf{w}_{P-1}^{\mathrm{T}}\mathbf{s}_{k-P+1} + b'_{P-1} + \omega''_{P-1} \end{cases} \tag{4}$$

where $\omega''_p = \omega'_p + \omega_{k-1} + \omega_{k-2} + \ldots \omega_{k-p}$, $p = 0, 1 \ldots P-1$ is still Gaussian white noise [16]. Adding all of the formulas in (4) will have

$$y_k = \sum_{p=0}^{P-1} \mathbf{w'}_p^{\mathrm{T}}\mathbf{s}_{k-p} + B + \Omega_k \tag{5}$$

where $\mathbf{w'}_p^{\mathrm{T}} = \mathbf{w}_p^{\mathrm{T}}/P$ denotes a weight vector, $B = (b'_0 + b'_1 + \ldots b'_{P-1})/P$ denotes a bias, and $\Omega_k = (\omega'_0 + \omega''_1 + \omega''_2 + \ldots \omega''_{P-1})/P$ is still a zero-mean Gaussian white noise [16]. Note that $\mathbf{w'}_k$ and $\mathbf{s}_k$ in $\Sigma_{p=0}^{P-1}\mathbf{w'}_p^{\mathrm{T}}\mathbf{s}_{k-p}$ are both vectors and then let

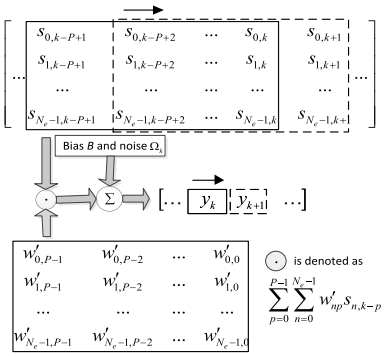

Fig. 2. Illustration of two-dimensional convolution space model.

$$\mathbf{w'}_p = [w'_{0p}, w'_{1p}, \ldots, w'_{N_e-1,p}]^{\mathrm{T}}$$
$$\mathbf{s}_{k-p} = [s_{0,k-p}, s_{1,k-p}, \ldots s_{N_e-1,k-p}]^{\mathrm{T}}$$

Thus,

$$y_k = \sum_{p=0}^{P-1}\sum_{n=0}^{N_e-1} w'_{np}s_{n,k-p} + B + \Omega_k \tag{6}$$

Eq. (6) shows that the position state y_k is a two-dimensional (2D) convolution of the spike vector $\mathbf{s}_k$ with the weight vector $\mathbf{w'}_k$, as shown in Fig.2. For simplicity, let

$$\mathbf{W} = [\mathbf{w'}_0^{\mathrm{T}}, \mathbf{w'}_1^{\mathrm{T}}, \ldots \mathbf{w'}_{P-1}^{\mathrm{T}}, B]^{\mathrm{T}}$$
$$\mathbf{S}_k = [\mathbf{s}_k^{\mathrm{T}}, \mathbf{s}_{k-1}^{\mathrm{T}}, \ldots \mathbf{s}_{k-P+1}^{\mathrm{T}}, 1]^{\mathrm{T}}$$

And then we have

$$y_k = \mathbf{W}^{\mathrm{T}}\mathbf{S}_k + \Omega_k \tag{7}$$

Eq. (7) further shows that the position state y_k is the inner product of the spike matrix $\mathbf{S}_k$ and the weight matrix $\mathbf{W}$ plus the Gaussian white noise Ω_k. As can be seen from the 2D convolution model of (6-7), y_k is correlated not only with the spike vector $\mathbf{s}_k$ at time k, but also with $\mathbf{s}_{k-P+1}$, $\mathbf{s}_{k-P+2}$, $\ldots \mathbf{s}_{k-1}$. It should be noted that the two-dimensional convolution space model is more suitable for multi-dimensional spike signal processing. If the model in this paper is changed to one-dimensional convolution, only one-dimension spike signals can be processed.

From CSM model in (7), further, we will revise the original SSM in (2) into

$$y_k = y_{k-1} + \omega_k \tag{8}$$
$$\mathbf{S}_k = \mathbf{A}y_k + \mathbf{V}_k \tag{9}$$

where $\mathbf{A}$ is an coefficient matrix, $\mathbf{V}_k = [v_{0k}, v_{1k}, \ldots, v_{PN_e,k}]^{\mathrm{T}}$ is a Gaussian white noise vector with a zero mean and a variance matrix $\mathbf{R}_V$ which is a diagonal matrix with a vector $[\sigma_0^2, \sigma_1^2, \ldots, \sigma_{PN_e}^2]$. If $\mathbf{A}^{\dagger} = \mathbf{W}^{\mathrm{T}}$, multiplying both sides of (9) by $\mathbf{A}^{\dagger}$ will obtain (7). Thus, we will adopt a Kalman filter to

solve the SSM model, called CSM-KF, which is shown as [20]

$$\mathbf{G}_k = P_{k-1}A^{\mathbf{T}}(AP_{k-1}A^{\mathbf{T}}+\mathbf{R}_V)^{-1} \tag{10}$$
$$\alpha_k = \mathbf{S}_k - \mathbf{A}\hat{y}_{k-1} \tag{11}$$
$$\hat{y}_k = \hat{y}_{k1} + \mathbf{G}_k\alpha_k \tag{12}$$
$$P_k = P_{k-1} - \mathbf{G}_k\mathbf{A}P_{k-1} + \sigma^2 \tag{13}$$

where $\hat{y}_k$ is the estimated value of y_k at time k.

B. Influence of Temporal Correlation on Model Training

After the model is established, the model needs to be trained to obtain the model parameter. From Section IV-*A*, CSM model lets the current moment state correlated with P previous moments. Next, we will focus on analyzing the influence of the parameters P on the model training.

1) Influence of Time Correlation on Training Method: Here, we will adopt some traditional training methods, such as LS and GDA. It might get a reduction for decoding errors to try some newer algorithms. However, this will make some confusion that whether the reduction comes from the CSM model itself or the newer training algorithms. For a TILM model, if a noise is a zero mean Gaussian white noise, the model can be trained via methods such as LS and GDA. Since the noise in CSM model satisfies the condition of Gaussian white noise, LS can be used and shown as [21]

$$\hat{\mathbf{W}} = \bar{\mathbf{S}}^{\dagger}\mathbf{Y} \tag{14}$$

where $\bar{\mathbf{S}} = [\mathbf{S}_P, \mathbf{S}_{P+1}, \ldots, \mathbf{S}_K]^{\mathrm{T}}$, $\mathbf{Y} = [y_P, y_{P+1}, \ldots, y_K]^{\mathrm{T}}$ and $\hat{\mathbf{W}}$ is the estimated matrix of $\mathbf{W}$. Compared with the time-independent model of $P = 1$, the parameters of the CSM model are $P > 1$. For LS, therefore, the number of columns of the observation matrix $\bar{\mathbf{S}}$ in (14) will be changed from the original N_e+1 into PN_e+1. The change of the dimensions of the observation matrix directly affects the training complexity of the model. This will be analyzed in section IV-*B*-2.

In addition, we can use a batch recursive least squares (RLS) to train CSM model. Thus, we have [21]

$$e_k = y_k - \mathbf{S}_k^{\mathrm{T}}\tilde{\mathbf{W}}_{k-1} \tag{15}$$
$$\mathbf{K}_k = \mathbf{P}_{k-1}\mathbf{S}_k(\lambda + \mathbf{S}_k^{\mathrm{T}}\mathbf{P}_{k-1}\mathbf{S}_k)^{-1} \tag{16}$$
$$\tilde{\mathbf{W}}_k = \tilde{\mathbf{W}}_{k-1} + \mathbf{K}_k e_k \tag{17}$$
$$\mathbf{P}_k = [\mathbf{P}_{k-1} - K_k S_k^{\mathrm{T}}\mathbf{P}_{k-1}]/\lambda \tag{18}$$

where $\tilde{\mathbf{W}}_k$ is an iteration weight at the moment k, $\mathbf{K}_k$ is a gain vector at time k, $\mathbf{P}_k$ is an inverse correlation matrix at time k, $0 < \lambda \leq 1$ is a forgotten factor. Similarly, the number of columns of observation matrix $\mathbf{S}_k$ in (15-18) also changes from original $N_e + 1$ to $PN_e + 1$.

Besides, GDA algorithm is also a popular batch method training model, shown as [21]

$$\tilde{\mathbf{W}}_k = \tilde{\mathbf{W}}_{k-1} - 2\mu e_k \mathbf{S}_k \tag{19}$$

where μ denotes an iteration step. Similar to RLS, the observation matrix in GDA is also the matrix $\mathbf{S}_k$ with $PN_e + 1$ columns.

TABLE I
TRAINING COMPLEXITY ABOUT PARAMETER P

CSM algorithm	LS	RLS	GDA	CSM-KF
Complexity	$O(P^3)$	$O(TP^3)$	$O(TP^2)$	$O(P^3)$

In addition, CSM-KF requires training the parameter vector $\mathbf{A}$ in (9). Similar to LS in (14), $\mathbf{A}$ can also be obtained via an LS algorithm, shown as

$$\hat{\mathbf{A}}^{\mathrm{T}} = \dot{\mathbf{Y}}^{\dagger}\dot{\mathbf{S}} \tag{20}$$

where $\hat{\mathbf{A}}$ is the estimated matrix of $\mathbf{A}$, $\dot{\mathbf{S}} = [\dot{\mathbf{S}}_P, \dot{\mathbf{S}}_{P+1}, \ldots, \dot{\mathbf{S}}_K]^{\mathrm{T}}$, $\dot{\mathbf{S}}_k = [\mathbf{s}_k^{\mathrm{T}}, \mathbf{s}_{k-1}^{\mathrm{T}}, \ldots \mathbf{s}_{k-P+1}^{\mathrm{T}}]^{\mathrm{T}}$, $\dot{\mathbf{Y}} = [\dot{\mathbf{y}}_P, \dot{\mathbf{y}}_{P+1}, \ldots, \dot{\mathbf{y}}_K]^{\mathrm{T}}$, and $\dot{\mathbf{y}}_k = [y_k, 1]^{\mathrm{T}}$.

2) Influence of Time Correlation on Training Complexity: In CSM model, $\bar{\mathbf{S}}$ in LS is a $(K\text{-}P + 1) \times (PN_e + 1)$ matrix. Hence, the pseudo-inverse computation of the matrix $\bar{\mathbf{S}}$ require Ln^3 multiplications, where

$$L = K - P + 1 \tag{21}$$
$$n = PN_e + 1 \tag{22}$$

Ignoring low-order terms and plus or minus terms, the computational complexity of the LS w.r.t. P can be expressed as $O(P^3)$. In RLS, the solution of $\tilde{\mathbf{W}}_k$ in step k needs $n^3 + 2n^2 + n$ multiplications and divisions and needs L steps. If $\tilde{\mathbf{W}}_k$ needs a total of T iteration cycles to converge,

RLS will require a total of $TL(n^3+2n^2+n)$ multiplications and divisions. Also ignoring low-order terms and plus or minus terms, the computational complexity of RLS w.r.t. the parameter P can be expressed as $O(TP^3)$. For GDA, the solution of $\tilde{\mathbf{W}}_k$ in step k needs n^2+2n multiplications and divisions, and a total of L steps are required. If $\tilde{\mathbf{W}}_k$ needs a total of T cycles to converge, GDA will require a total of $TL(n^2+2n)$ multiplications and divisions. Therefore, the computational complexity of GDA w.r.t. the parameter P can be expressed as $O(TP^2)$. Finally, since CSM-KF adopts an LS training method, its computational complexity w.r.t. P can be expressed as $O(P^3)$.

Table I shows the computational complexity of CSM model training w.r.t. P. When the value of the parameter P is 1, CSM model is changed into a linear time-independent model. When $P > 1$, the complexity of LS, RLS, GDA and CSM-KF would be increased by P^3,P^3,P^2 and P^3 times, respectively.

3) Influence of Temporal Correlation on Model Weights: In CSM model, the number of parameters in the convolution kernel $\mathbf{W}$ is PN_e+1, which is approximately P times greater than the number of parameters of the time-independent model.

The training results of the convolution kernel $\mathbf{W}$ of the CSM model in four groups of experiments is shown in Fig. 3, where the weight value is the 2 norm of $\mathbf{w}'_p$, $p = 1, 2, \ldots, P$, $P = 10$. More detailed parameters in the four groups of experiments could be seen in section VI-*B*. From the figure, LS, RLS and GDA has nearly same pattern for the distribution of the kernel weights. As the value of P increases, the amplitude of the convolution kernel weight gradually decreases until at $P = 10$ where almost no decrement happen. This result shows that the value of P is not necessarily too large because

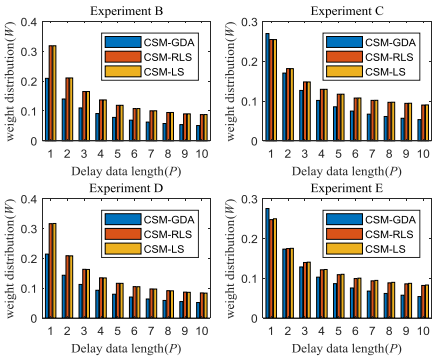

Fig. 3. Convolutional kernel weight distribution about temporal correlation.

TABLE II
LS ALGORITHM STEPS FOR NEURAL DECODING OF MACAQUE FINGER MOVEMENT POSITION

Input: Spike signal $\mathbf{s}_k$, X or Y coordinate value y_k in training data, and spike signal $\mathbf{s}'_k$ in test data
Output: Weight $\hat{\mathbf{W}}$, decoded X or Y coordinate $\hat{y}_k$ for test data
Algorithm 1 LS algorithm
Preprocessing:
1. change spike signal vector $\mathbf{s}_k$ into matrix $\overline{\mathbf{S}}$ in (14);
Training:
2. obtain weight $\hat{\mathbf{W}}$ in (14) via $\mathbf{s}_k$ and y_k;
Decoding:
3. substitute $\mathbf{s}'_k$ and $\hat{\mathbf{W}}$ into (5-7) and obtain $\hat{y}_k$

the contribution of its weight to the model time correlation becomes smaller, with the increase of P.

V. NEURAL DECODING

After the weight training in the model is completed, substituting the number of neuron spikes in test data into CSM model will obtain a decoding value, which is in fact the coordinate of the finger movement. Table II - IV show total algorithm steps for the neural decoding of the macaque finger movement position.

VI. EXPERIMENT RESULTS AND ANALYSIS

A. Experiment Data

This experiment data is provided by Hatsopoulos Laboratories [7] and the download address is http://booksite.elsevier.com/0780123838360. The details for the data acquisition can be seen in Section III problem description, and the other relevant parameters in the data are as follows.

1) Macaque's finger moving range $L = 25$cm, $B = 18$cm
2) The number of collecting electrodes in Macaque's brain $N_e = 42$
3) Sampling period $\Delta t = 70$ms
4) Data length $K = 3101$

The downloaded data has two groups and the details are as follows.

1) Data 1: a $K \times N_e$ data matrix records the number of neuron spikes, where the length is K and the number of electrodes is N_e. Another $K \times 2$ data matrix records X and Y coordinates, where the length is also K, the first column is for X axis and the second column is for Y axis.
2) Data 2: two data matrices records the number of spikes and X and Y coordinates, and has the same format as Data 1.

Data 1 and Data 2 are all acquired under the same conditions, except that the finger in data 1 does not move in horizontal or vertical direction and data 2 has horizontal or vertical movement. Also, a small number of position sample points exceed the finger's range of motion in the Y axis of the data set. Hence, the position sample points of the data set beyond the active area will be removed.

TABLE III
BATCH ALGORITHM STEPS FOR NEURAL DECODING OF MACAQUE FINGER MOVEMENT POSITION

Input: Spike signal $\mathbf{s}_k$, X or Y coordinate value y_k in training data, and spike signal $\mathbf{s}'_k$ in test data
Output: Weight $\hat{\mathbf{W}}$ and decoded X or Y coordinate $\hat{y}_k$ for test data
Initialized value: $k = 0$, $t = 0$, $\mu = 2\times10^{-6}$, $\lambda = 0.9999$, $\hat{\mathbf{W}}_0 = 0$, $\boldsymbol{P}_0 = \delta^{-1}\mathrm{I}$, $\delta = 1$.
Algorithm 1 RLS algorithm
Preprocessing:
1. change spike signal $\mathbf{s}_k$ into $\mathbf{S}_k$ in (7);
Training:
2. obtain $\tilde{\mathbf{W}}_k$ in (15-18) via $\mathbf{s}_k$ and y_k;
3. let $k = k+1$, repeat steps 2-3 until $k = K-1$;
4. let $\tilde{\mathbf{W}}_0 = \tilde{\mathbf{W}}_k$, $k = 0$, $t = t+1$, repeat steps 2-4 until $t = T-1$;
Decoding:
5. Substitute $\mathbf{s}'_k$ and $\hat{\mathbf{W}}_k$ into (5-7) and obtain $\hat{y}_k$
Algorithm 2 GDA algorithm
Preprocessing:
1. Change spike signal $\mathbf{s}_k$ into $\mathbf{S}_k$ in (7);
Training:
2. obtain $\tilde{\mathbf{W}}_k$ in (19) via $\mathbf{s}_k$ and y_k;
3. let $k = k+1$, and repeat steps 2-3 until $k = K-1$;
4. let $\tilde{\mathbf{W}}_0 = \tilde{\mathbf{W}}_k$, $k = 0$, $k = k+1$, and repeat steps 2-4 until $k = T-1$;
Decoding:
5. Substitute $\mathbf{s}'_k$ and $\hat{\mathbf{W}}_k$ into (5-7) and obtain $\hat{y}_k$

TABLE IV
CSM-KF ALGORITHM STEPS FOR NEURAL DECODING OF MACAQUE FINGER MOVEMENT POSITION

Input: Spike signal $\mathbf{s}_k$, X or Y coordinate value y_k in training data, and spike signal $\mathbf{s}'_k$ in test data
Output: Weight $\hat{\mathbf{W}}$, decoded X or Y coordinate $\hat{y}_k$ for test data
Initialized value: $\hat{y}_0 = 0$, $P_0 = 1$
Algorithm 1 CSM-KF algorithm
Preprocessing:
1. change spike signal vector $\mathbf{s}_k$ into matrix $\overline{\mathbf{S}}$ in (20);
Training:
2. obtain weight $\hat{\mathbf{A}}$ in (20);
Decoding:
3. substitute $\mathbf{s}'_k$ and $\hat{\mathbf{A}}$ into (10-13) and obtain $\hat{y}_k$

B. Experiment and Parameter Setting

Since data 1 and 2 have different characteristics, we consider processing data from multiple cross validations. In this section, we will use five groups of experiments to evaluate the performance of neural decoding methods. The data processing in the five groups of experiments are as follows.

1) *Experiment A:* Holdout verification [22] on data 1, 70% for training, and 30% for testing;
2) *Experiment B:* M-fold cross validation [22] on data 1, where $M = 10$;
3) *Experiment C:* M-fold cross validation [22] on data 2, where $M = 10$;
4) *Experiment D:* Data 1 for training, data 2 for testing;
5) *Experiment E:* Data 2 for training, data 1 for testing.

For M-fold cross-validation in *Experiment B* and *C*, a root mean square error (RMSE) e_c is used to evaluate the performance of decoding methods and defined as follows

$$e_c = \frac{1}{M}\sum_{m=1}^{M}\sqrt{\frac{1}{K'}\sum_{k=0}^{K'-1}(\hat{y}_{m,k} - y_{m,k})^2} \tag{23}$$

where $\hat{y}_{m,k}$ and $y_{m,k}$ are the decoded coordinate values and the real ones, respectively and K' is the cross-validated data length. *Experiments D* and *E* use another RMSE, e_r to evaluate the performance, and the definition of e_r is

$$e_r = \sqrt{\frac{1}{K}\sum_{k=0}^{K-1}(\hat{y}_k - y_k)^2} \tag{24}$$

where $\hat{y}_k$ and y_k are the decoded coordinates and the real ones in the test data, respectively and K is the length of the test data. Also, e_c and e_r could reflect the generalization ability of decoding methods in *Experiment B* to *E*. And, e_c is for *Experiment B* and *Experiment C*, where the M-fold cross-validation method is adopted. e_r is for *Experiment D* and *Experiment E*, where a training data set and a test data set are required.

The above five groups of experiments will calculate e_c and e_r for Linear [7], KF[19], RBE[17], UCKD[16] and CSM methods, respectively for comparison and some parameters of these methods are set as follows.

1) Linear: train a time-independent model through training data, i.e. the model in (3) where $P = 1$ and then decode movement position through testing data.
2) KF: train the model in (2) via LS, and then use Kalman filter to decode movement position. The parameters in Kalman filter are, an initial decoding value $\hat{y}_0 = 10$, covariance $P_{y,0|0} = 1$, a state noise variance $\sigma^2 = 0.8$, an observe noise variance $\mathbf{R}_v = (\bar{\mathbf{s}} - \hat{\mathbf{s}})^{\mathrm{T}}(\bar{\mathbf{s}} - \hat{\mathbf{s}})/(K-1)$, where $\bar{\mathbf{s}} = [\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{K-1}]^{\mathrm{T}}$ and $\hat{\mathbf{s}} = \ddot{\mathbf{Y}}\ddot{\mathbf{W}}^{\mathrm{T}}$, $\ddot{\mathbf{Y}} = [\ddot{\mathbf{y}}_0, \ddot{\mathbf{y}}_1, \ldots, \ddot{\mathbf{y}}_{K-1}]^{\mathrm{T}}$, $\ddot{\mathbf{y}}_k = [y_k, 1]^{\mathrm{T}}$, $\ddot{\mathbf{W}} = \ddot{\mathbf{Y}}^{\dagger}\bar{\mathbf{s}}$.
3) RBE: an X-axis estimated range is 0.5: 1: 24.5 (denoted as a vector starting from 0.5, ending at 24.5 and spacing on 1), and a Y-axis estimated range is 0.5: 1: 14.5.
4) UCKD: an initial decoding value $\hat{y}_0 = 0$, a weight $\hat{\boldsymbol{w}}_0$ is uniformly distributed random signal whose mean value is 0, a covariance $P_{y,0|0} = 1$, $\mathbf{P}_{\mathbf{w},0|0} = 0.05\ \mathbf{I}$ where $\mathbf{I}$ is an identity matrix, a state noise variance $\sigma^2 = 0.8$, a weight noise covariance $\mathbf{R}_{\mathbf{w}} = 0.01\ \mathbf{I}$, a forgetting factor $\lambda = 0.005$, and an observed noise variance $\mathbf{R}_v$ is the same as KF method.
5) CSM: Unless otherwise stated, a forgotten factor $\lambda = 0.9999$, an iteration step $\mu = 2 \times 10^{-6}$, a temporal-correlation parameter $P = 10$, an iteration cycle for RLS $T = 3$ and an iteration cycle for GDA $T = 100$.

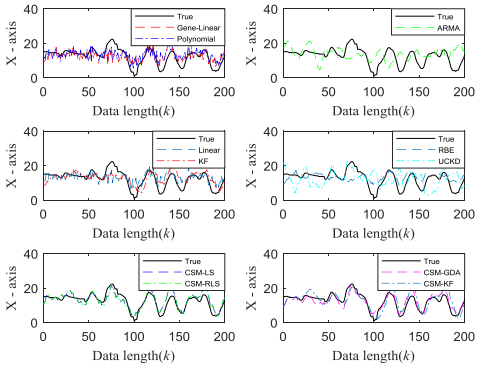

Fig. 4. Curves for decoding finger position in *Experiment A*.

In addition, we have considered three traditional models and an SSM model from CSM. Their parameters are as follows.

1) Polynomial: the model is expressed as $y_k = \ddot{\mathbf{W}}^{\mathrm{T}}\mathbf{s}_k^2 + \dot{\mathbf{W}}^{\mathrm{T}}\mathbf{s}_k + B$, where $\ddot{\mathbf{W}}$ and $\dot{\mathbf{W}}$ is are both coefficient vector, and $\mathbf{s}_k^2$ is also a vector whose elements are the squares of the elements of $\mathbf{s}_k$.
2) Gen-Linear: the model is expressed as $y_k = \exp(\dot{\mathbf{W}}^{\mathrm{T}}\mathbf{s}_k + B)$.
3) ARMA: the model is expressed as $\Sigma_{q=0}^{Q-1} w_q'' y_{k-q} = \Sigma_{p=0}^{P-1}\mathbf{w}'^{\mathrm{T}}_p\mathbf{s}_{k-p} + B + \Omega_k$, where w_q'' is dented as AR coefficients, $Q = 10$ and $P = 10$.
4) CSM-KF: a state noise variance $\sigma^2 = 0.8$, and an observe noise variance $\mathbf{R}_V = (\dot{\mathbf{S}} - \hat{\dot{\mathbf{S}}})^{\mathrm{T}}(\dot{\mathbf{S}} - \hat{\dot{\mathbf{S}}})/(K-1)$ where $\hat{\dot{\mathbf{S}}} = \dot{\mathbf{Y}}\hat{\mathbf{A}}^{\mathrm{T}}$.

C. Decoding Result Analysis

Fig.4 shows the X-axis decoding curves of 200 sampling test data points for each algorithm in *Experiment A*. From the figure, all of methods can approximately track the real finger moving curve, but for each algorithm, their performance is not exactly the same. The decoding curves for the three traditional regression methods, polynomial fitting, generalized linear function and ARMA model have some glitches or jitters, which will result in poor overall estimation performance. The reason is that these methods do not consider the temporal correlation in regression models. Also, the decoding curve of Linear shows more jitter. For example, the curve fluctuates frequently around points 80 to 100. The reason is also that linear algorithm considers each moving coordinate state as an independent process and discards the correlation of adjacent data points. Compared with other decoding algorithms, UCKD have a larger deviation, and especially has a reverse direction to the actual decoding curve in points 120-130 and 150-160 because the algorithm uses an unsupervised method. When decoding the X-axis, UCKD sometimes does not recognize the direction of the finger movement, but the algorithm has good recognition of the Y-axis from Table VI. The curves of KF and RBE algorithm both based on SSM model is smoother and less jerky than Linear algorithm because SSM model correlates a state at one current time with one previous moment. However, it should be noted that the curves do not

TABLE V
X-AXIS AND Y-AXIS (IN PARENTHESES) ESTIMATION ERROR, UNIT: cm

Alg X(Y)	Exp B	Exp C	Exp D	Exp E	average
Gene-Linear	4.036(2.386)	4.119(3.143)	4.101(2.952)	4.292(2.881)	4.137(2.841)
Polynomial	3.790(1.968)	3.825(2.436)	4.651(3.012)	3.887(2.138)	4.038(2.389)
ARMA	3.025(1.661)	3.743(2.257)	5.242(2.614)	3.150(1.888)	3.790(2.105)

TABLE VI
X-AXIS AND Y-AXIS (IN PARENTHESES) ESTIMATION ERROR, UNIT: cm

Alg X(Y)	Exp B	Exp C	Exp D	Exp E	average
Linear	3.813(2.003)	3.898(2.460)	4.084(2.934)	3.954(2.132)	3.937(2.382)
KF	3.060(1.498)	3.877(1.956)	4.569(2.887)	3.602(1.649)	3.777(1.998)
RBE	3.637(1.727)	4.660(2.227)	4.179(2.330)	4.345(1.923)	4.205(2.052)
UCKD	6.856(2.934)	6.575(5.141)	6.679(2.471)	7.551(4.612)	6.915(3.790)
CSM-LS	2.959(1.411)	3.385(2.096)	3.887(2.841)	3.103(1.688)	3.334(2.009)
CSM-RLS	2.960(1.411)	3.382(2.096)	3.908(2.811)	3.099(1.689)	3.337(2.002)
CSM-GDA	2.896(1.500)	3.222(2.033)	4.560(2.305)	3.239(1.666)	3.479(1.876)
CSM-KF	3.004(1.494)	3.607(2.126)	4.320(2.995)	3.273(1.796)	3.551(2.103)

fully track the actual curve in some places where the trend reverses. For example, near points 40-60, the decoding curves do not fully reflect the trend of the actual curve. On the other hand, LS, RLS and GDA algorithms based on CSM model can track the actual curves even at trend reversals, such as points 20-40, 40-60 and 80-100. The results show that the correlation established between the current state and the states at several previous moments achieves good tracking performance.

The above results give a qualitative analysis. Next, we will give a quantitative analysis through the evaluating metric of RMSE. Table VI shows X-axis decoding results for each algorithm in *Experiments B* to E, where *Experiment B* and *C* show the metric of e_c, and *Experiment D* and *E* show the metric of e_r. The average values of the four groups of experimental results ranked from the highest to the lowest are UCKD, RBE, Linear, KF, CSM-KF, CSM-GDA, CSM-RLS and CSM-LS, respectively. Especially, LS's minimum RMSE is about 11.7% smaller than KF's minimum RMSE. In addition, for each group of experiments' RMSE, the four decoding algorithms based on CSM model are less than or close to the traditional algorithm, except that GDA's RMSE is 4.56cm and CSM-KF's RMSE is 4.32cm in *Experiment D*, higher than traditional Linear and RBE, which shows that the GDA and CSM-KF have a weak generalization ability to the data in *Experiment D*.

In addition, Table VI also gives Y-axis decoding results for each algorithm. The average RMSE values of the four experimental results are UCKD, Linear, CSM-KF, RBE, CSM-LS, CSM-RLS, KF, and CSM-GDA from the highest to the lowest. For the results of the four groups of experiments, the errors of the decoding algorithms based on CSM are less than or close to traditional algorithms. However, GDA with the least errors is only 6.1% smaller than KF with the least errors in the traditional algorithms. The performance improvement is not as good as the results for the X axis. The result also shows that the CSM model for the X axis has better results than that for the Y axis. That is, CSM's generalization for the Y-axis is stronger than that for the X-axis. One of the reasons is that the Y-axis test data has more similarity to

TABLE VII
THE ESTIMATION ERROR OF THE TWO-DIMENSIONAL PLANE, UNIT: cm

Alg XY	Exp B	Exp C	Exp D	Exp E	average
Linear	4.307	4.609	5.029	4.492	4.609
KF	3.407	4.343	5.405	3.962	4.279
RBE	4.026	5.165	4.785	4.752	4.682
UCKD	7.457	8.346	7.121	8.848	7.943
CSM-LS	3.278	3.981	4.815	3.532	3.902
CSM-RLS	3.279	3.979	4.814	3.529	3.900
CSM-GDA	3.261	3.810	5.110	3.642	3.956
CSM-KF	3.355	4.187	5.257	3.733	4.133

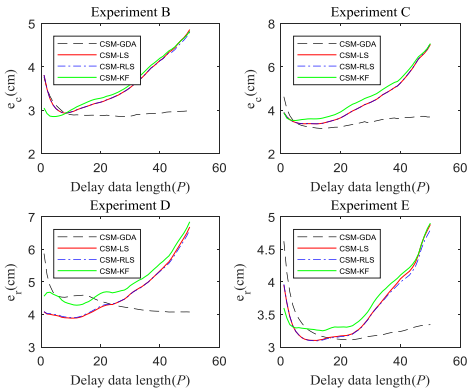

Fig. 5. Estimation error on X-axis v.s. temporal correlation parameter P in *Experiment B* to *E*.

the training data set. Finally, Table VII shows the estimation error results of each algorithm for 2D planes. The decoding results are similar to Table VI, and the algorithms with the least errors are still the algorithms based on CSM model.

Fig.5 shows the impact of P on the decoding performance of CSM model, and gives the RMSE curves of LS, RLS, GDA and CSM-KF in *Experiment B* to E. As can be seen from the figure, the curves of LS and RLS are almost coincident, and after about $P = 20$, the curves begin to rise. In *Experiment B*, C and E, the curves of LS, RLS and GDA based on CSM and CSM-KF all drop to about 3cm before $P = 10$. In *Experiment D*, the curve trend of GDA is, dropping, rising, dropping and then being stable, while the curve trend of LS, RLS and CSM-KF is rising, being stable and then rising. It should be noted that the errors for the four algorithm are all be smaller at about $P = 10$. P is in fact a parameter about temporal correlation in the process. Excessive small value of P will cause under-fitting. Of course, excessive large value of P will cause over-fitting. Both of under- and over-fitting will make the decoding errors increase. Therefore, it can be analyzed from the figure that $P = 10$ can better guarantee that the three algorithms based on CSM have smaller errors for the finger position decoding.

Fig.6 shows the impact of iteration cycles T on the decoding performance of CSM and gives the RMSE v.s. T curves of RLS and GDA in Experiment B to E, respectively. As can be seen from the figure, the curve of GDA drops rapidly when T varies from 1 to 10, drops slowly when T varies from 10 to 40, and tends to be a straight line after about $T = 40$. On the other hand, the curve of RLS is always in a steady state whatever T is. From the above analysis, the value of T

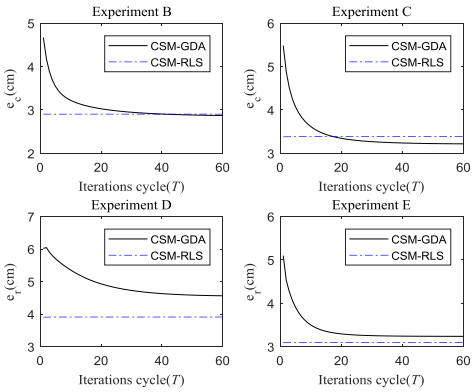

Fig. 6. Estimation error on X-axis v.s. the number of iterations cycle T in *Experiment B* to *E*.

TABLE VIII
THE TRAINING TIME OF CSM MODEL ALGORITHM, UNIT: SECONDS

Algorithm	Train time	Algorithm	Train time	Algorithm	Train time
Linear	0.009	UCKD	296.955	CSM-GDA	2.058
KF	0.005	CSM-LS	0.125	CSM-KF	0.011
RBE	1.351	CSM-RLS	26.802		

for GDA can be set as low as 40, because the errors do not decline much even if T is greater than 40. For RLS, only one iteration cycle can guarantee a smaller decoding error because its error curve is a straight line.

To evaluate the computational complexity of the four algorithms in the CSM model, finally, Table VIII shows the training time of conventional methods and CSM model in *Experiment D*, where $T = 30$ in UCKD, $T = 3$ in RLS and $T = 60$ in GDA. The PC computer operation system running the decoding algorithms in the table is Windows 7 Ultimate 64-bit SP1, the processor is Intel Corei5-6400@2.70GHz quad-core and the processing software is MatLabR2017b. The training method based on the CSM model, CSM-LS and CSM-KF's training time is more than Linear and KF, although all of them use Least Square. And, CSM-RLS and CSM-GDA's training time is more than RBE, although all of them use iteration methods. The reason is that the number of training weights in the CSM model increases when the parameter P is introduced. Besides, the reason for UCKD's more training time is that it adopts two SSM model. One is for weight training, and the other is for position estimation.

VII. DISCUSS

In this paper, we study the temporal correlation of the decoding model for macaque's finger position, use CSM model to decode the position, and train CSM model through three algorithms. Compared with the traditional decoding models, CSM model can reduce decoding error. However, there are some problems which need further discussion about CSM model and the training algorithms.

First, after analyzing the temporal correlation, the introduction of the parameter P increases the number of parameters that need to be trained. The number of weights $\mathbf{a}$ and bases $\mathbf{b}$ trained in the traditional SSM model (2) is $2N_e$, but this number will become $PN_e + 1$ in CSM model. Due to the increase of training parameters, the complexity of training algorithm will inevitably increase. However, since the training is completed only in a training step and only several corresponding multiplications are added in a decoding step, the increase in complexity has little effect on the decoding. Second, the value of temporal-correlation parameter P can have a significant impact on the decoding performance. Too large value of P will result in over-fitting and too small value will result in under-fitting. To determine an appropriate P, a data set can be divided into three parts, a training set, a test set, and a validation set. Given a group of P values, the training weights are obtained in the training set, and then the decoding errors are obtained in the test set. The P value corresponding to the smallest error will be the optimal value. The optimal P value and weights substituting into the validation set will produce the final decoding error, which can measure the decoding performance of the model.

In addition, it can be seen from the experimental data that although the performance of CSM model is mainly improved on the X-axis instead of the Y-axis, the decoding errors of the 2-D plane are greatly reduced.

For batch training methods, besides, RLS and GDA require some iteration cycles. From the experimental results, the performance of RLS with one cycle is close to that with several cycles. Thus, the number of cycle can be set as 1. For GDA, the number of cycles needs more than 40 to be convergent, which in turn will increase computational complexity. However, GDA's error decreases rapidly in the first 10 cycles, and is still decreasing but tends to be flat in the subsequent cycles. In order to reduce the cycle time, thus, it is not necessary to set too many cycles to be convergent, and $T = 10$ is a feasible option. Moreover, the computational complexity for GDA can be denoted as $O(TP^2)$. Compared with the other two algorithms' $O(P^3)$ and $O(TP^2)$, hence, the increase of T sometimes does not necessarily produce too much complexity.

It should be noted that the training CSM method in this paper is still a supervised training method, because the decoding cannot be done without training data. In fact, human learning is a kind of unsupervised learning. If there is an unsupervised learning method which will be able to decode finger movements without training set, the method would be a more practical learning method. Although UCKD is an unsupervised method, unfortunately, it does not have better performance for the X-axis decoding. In addition, semi-supervised or weak-supervised decoding methods can also be considered even if training data's label information is incomplete. For example, there are only some approximate information of the finger's movement position rather than the exact information in training data. This may be one next study direction.

Finally, CSM model itself is a convolutional model and the recent popular deep convolutional neural network (CNN) seems to be used for the model. Here, we give some preliminary results. Fig.7 shows the X-axis decoding curves of a deep convolutional neural network algorithm in *Experiment A*. The code of convolutional neural networks is from https://github.com/rasmusbergpalm/DeepLearnToolbox. From the figure, the decoding curve can nearly track the real curve, but it has the following problem. The first one is, the convolutional

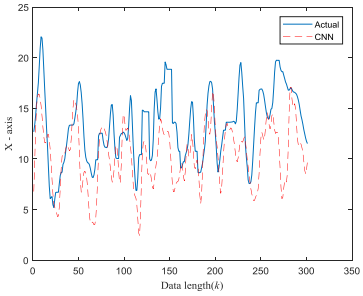

Fig. 7. Estimation error curves of finger movement on X-axis via a convolutional neural network in *Experiment A*.

network used in the experiment randomly generates initial parameters. Thus, its decoding performance becomes random. How to determine a good initial value is an important problem. Besides, the decoding curve by the convolution neural network is sometimes in an opposite direction to the real curve. From the figure, most of points from 1 to 200 can keep up with the real movement position points, but some inversions happen in points from 200 to 350. In addition, the convolutional neural networks are currently more used for classification. When it is used for regression of this position decoding, our method is to divide continuous values into several intervals. Each of intervals corresponds a class. However, the number of intervals will have an impact on decoding accuracy and complexity.

VIII. Conclusion

It is a typical neural decoding problem to estimate a macaque's finger movement position through spike signals in the macaque's motor cortex neurons. This paper studies the temporal correlation of the decoding model and uses a linear time-invariant convolutional model, called CSM model to decode the finger position. Then, LS, RLS, GDA and KF algorithms is used to solve the model. After analyzing the temporal correlation, P value in CSM model will affect the performance of neural decoding. When $P = 1$, the model is in fact a linear time-independent model, and when $P > 1$, it is the convolution space model, where the finger movement position can be expressed as the 2D convolution of a cluster vector of spike signals and a group of constant coefficients.

In the experiments, we use the public data to give five groups of experimental results, where Holdout and Cross-validation method are adopted to evaluate the decoding performance of CSM and traditional methods. From the results, CSM model has improved about 11.7% in X-axis decoding error performance than the traditional models.

In addition, the experimental results also confirm that the correlation between the state at one current moment and that at ten previous moments for the public data used in this paper, has better decoding error performance. Finally, RLS and GDA based on CSM model adopt the batch method. From the results, RLS's convergence needs only one iteration cycle, but GDA algorithm has more iteration cycles.

From the decoding error in Table VI and Table VII, GDA algorithm in the CSM model has fewer decoding errors than the other three algorithms. And, from the training time in Table VIII, GDA's time is less than RLS and is not much more than LS and CSM-KF. Therefore, GDA should be a better method because it has better performance on decoding errors and training time.

References

[1] T. Kapelner, F. Negro, O. C. Aszmann, and D. Farina, "Decoding motor unit activity from forearm muscles: Perspectives for myoelectric control," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 26, no. 1, pp. 244–251, Jan. 2018.

[2] S. Tabibi, A. Kegel, W. K. Lai, and N. Dillier, "Investigating the use of a Gammatone filterbank for a cochlear implant coding strategy," *J. Neurosci. Methods*, vol. 277, pp. 63–74, Feb. 2017.

[3] F.-V. Jacobo, L. Y. Chu, K. Kahori, and W. Yu, "3D continuos hand motion reconstruction from dual EEG and EMG recordings," in *Proc. Int. Conf. Intell. Inform. Biomed. Sci. (ICIIBMS)*, Okinawa, Japan, Nov. 2015, pp. 101–108.

[4] M. Hamedi, S. H. Salleh, and A. M. Noor, "Electroencephalographic motor imagery brain connectivity analysis for BCI: A review," *Neural Comput.*, vol. 28, no. 6, pp. 999–1041, Jun. 2016.

[5] K.-T. Kim, H.-I. Suk, and S.-W. Lee, "Commanding a brain-controlled wheelchair using steady-state somatosensory evoked potentials," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 26, no. 3, pp. 654–665, Mar. 2018.

[6] X. Gao, D. Xu, M. Cheng, and S. Gao, "A BCI-based environmental controller for the motion-disabled," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 11, no. 2, pp. 137–140, Feb. 2003.

[7] P. Wallisch, M. E. Lusignan, M. D. Benayoun, T. L. Baker, A. S. Dickey, and N. G. Hatsopoulos, *MATLAB For Neuroscientists*, 2nd ed. London, U.K.: Academic, 2014.

[8] E. Borra, M. Gerbella, S. Rozzi, and G. Luppino, "The macaque lateral grasping network: A neural substrate for generating purposeful hand actions," *Neurosci. Biobehav. Rev.*, vol. 75, pp. 65–90, Apr. 2017.

[9] J. A. Michaels and H. Scherberger, "Population coding of grasp and laterality-related information in the macaque fronto-parietal network," *Sci. Rep.*, vol. 8, Jan. 2018, Art. no. 1710. doi: 10.1038/s41598-018-20051-7.

[10] A. P. Georgopoulos, J. T. Lurito, M. Petrides, A. B. Schwartz, and J. T. Massey, "Mental rotation of the neuronal population vector," *Science*, vol. 243, pp. 234–236, Jan. 1989.

[11] S. Schaffelhofer, A. Agudelo-Toro, and H. Scherberger, "Decoding a wide range of hand configurations from macaque motor, premotor, and parietal cortices," *J. Neurosci.*, vol. 35, pp. 1068–1081, Jan. 2015.

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

[13] Z. Chen, *Advanced State Space Methods for Neural and Clinical Data*. Oxford, U.K.: Cambridge Univ. Press, 2015.

[14] M. M. Shanechi, G. W. Wornell, Z. M. Williams, and E. N. Brown, "Feedback-controlled parallel point process filter for estimation of goal-directed movements from neural signals," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 21, no. 1, pp. 129–140, Jan. 2013.

[15] M. Aghagolzadeh and W. Truccolo, "Latent state-space models for neural decoding," in *Proc. IEEE 36th Annu. Int. Conf. Eng. Med. Biol. Soc.*, Chicago, IL, USA, Aug. 2014, pp. 3033–3036.

[16] J. Y. Feng, H. F. Wu, and Y. Zeng, "Time correlation of time-invariant linear models in neural decoding for the macaque's moving finger," *Acta Autom. Sinica*, to be published.

[17] G. Hotson, R. J. Smith, A. G. Rouse, M. H. Schieber, N. V. Thakor, and B. A. Wester, "High precision neural decoding of complex movement trajectories using recursive Bayesian estimation with dynamic movement primitives," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 676–683, Jul. 2016.

[18] A. E. Brockwell, A. L. Rojas, and R. E. Kass, "Recursive Bayesian decoding of motor cortical signals by particle filtering," *J. Neurophysiol.*, vol. 91, pp. 1899–1907, Apr. 2004.

[19] W. Wu, A. Shaikhouni, J. P. Donoghue, and M. J. Black, "Closed-loop neural control of cursor motion using a Kalman filter," in *Proc. IEEE 26th Annu. Int. Conf. Eng. Med. Biol. Soc. (IEMBS)*, San Francisco, CA, USA, Sep. 2004, pp. 4126–4129.

[20] S. O. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2008.

[21] X. D. Zhang, *Modern Signal Processing*, 3rd ed. Beijing, China: Tsinghua Univ. Press, 2015.

[22] S. Raschka, *Python Machine Learning*, 2nd ed. Bermingham, U.K.: Packt, 2015.