This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBME.2023.3347137,

Automatical Spike Sorting with Low-Rank and Sparse Representation

Libo Huang, *Student Member, IEEE*, Lu Gan, *Senior Member, IEEE*, Yan Zeng, *Student Member, IEEE*, and Bingo Wing-Kuen Ling, *Senior Member, IEEE*

***Abstract*— Spike sorting is crucial in studying neural individually and synergistically encoding and decoding behaviors. However, existent spike sorting algorithms perform unsatisfactorily in real scenarios where heavy noises and overlapping samples are commonly in the spikes, and the spikes from different neurons are similar. To address such challenging scenarios, we propose an automatic spike sporting method in this paper, which integrally combines low-rank and sparse representation (LRSR) into a unified model. In particular, LRSR models spikes through low-rank optimization, uncovering global data structure for handling similar and overlapped samples. To eliminate the influence of the embedded noises, LRSR uses a sparse constraint, effectively separating spikes from noise. The optimization is solved using alternate augmented Lagrange multipliers methods. Moreover, we conclude with an automatic spike-sorting framework that employs the spectral clustering theorem to estimate the number of neurons. Extensive experiments over various simulated and real-world datasets demonstrate that our proposed method, LRSR, can handle spike sorting effectively and efficiently.**

***Index Terms*— spike sorting, optimization, similar waveform, overlapped spike**

I. Introduction

EXTRACELLULAR recording, which traces the local electrical potentials around the implanted probe tip, is one of the most reliable methods for monitoring neural activities and unveiling the behavior of neural systems [1], [2]. By utilizing the extracellular recording technique, one can obtain the action potentials outside the neurons (i.e., spikes) [3]. Since the recorded spikes may come from multiple neurons, they need to be identified and assigned to the tentative neurons for further downstream studies, including brain-machine interfaces [4], treating paralysis, epilepsy, and memory loss, etc [5], [6]. Spike sorting aims to fill the identification and assignment gaps from the raw spikes, and has gained much attention recently [7]. As shown in Fig.1, the conventional pipeline of spike sorting includes filtering, spike detection, feature extraction, and spike clustering [8], [9]. Among them, the filtering and detection processes are straightforward, and performing simple filters and thresholding could yield acceptable results [10]. However, the last two steps, i.e., the spikes' feature extraction and clustering procedures, are far from settled [7]. This is mainly caused by unique characteristics that have not been thoroughly investigated yet, such as overlapping spikes and noise disturbance [11]. Hence, this paper focuses on such characteristics for more efficient and effective feature extraction and clustering.

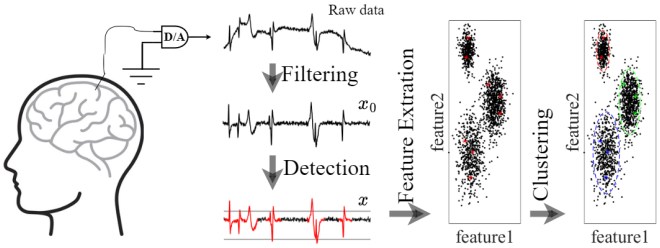


Fig. 1: Flowchart of the spike sorting framework.

Early endeavors of spike sorting originate from Grey, *et al.* [12], who introduced the cluster-cutting method by manually defining the boundaries of different classes. Before clustering, they directly extracted features from the spikes' waveform (e.g., peak-to-peak amplitudes [13] and waveform derivatives [14], etc.) or down-sampled from the aligned spikes [15], [5]. Though these features are intuitive, they are limited to being represented in a 2- or 3-dimensional space for carrying out the cluster cutting [16]. Moreover, the final clustering results would vary significantly with different extracted features and subjective human interventions [17].

Some recent efficient spike-sorting methods have emerged to mitigate the limitations of early endeavors. They could fall into two categories, i.e., the two-stage methods (performing the feature extraction and the clustering in turn) and the one-stage methods (performing both the feature extraction and the clustering simultaneously in a unified framework) [18], [19]. On

This work was supported partly by the National Nature Science Foundation of China (no. U1701266, no. 61372173, and no. 61671163), the Team Project of the Education Ministry of the Guangdong Province (no. 2017KCXTD011), the Guangdong Higher Education Engineering Technology Research Center for Big Data on Manufacturing Knowledge Patent (no. 501130144), the Guangdong Province Intellectual Property Key Laboratory Project (no. 2018B030322016) and Hong Kong Innovation and Technology Commission, Enterprise Support Scheme (no. S/E/070/17), and China Postdoctoral Science Foundation (2022M711812). (Corresponding author: Bingo Wing-Kuen Ling)

Libo Huang is with Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China (e-mail: www.huanglibo@gmail.com).

Lu Gan is with Department of Electrical and Computer Engineering, Brunel University London, UB8 3PH, U.K.(e-mail: lu.gan@brunel.ac.uk).

Yan Zeng is with Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China (e-mail: yanazeng013@gmail.com)

Bingo Wing-Kuen Ling is with School of Information Engineering, Guangdong University of Technology, Guangzhou, Guangdong, 510006, China (e-mail: yongquanling@gdut.edu.cn).

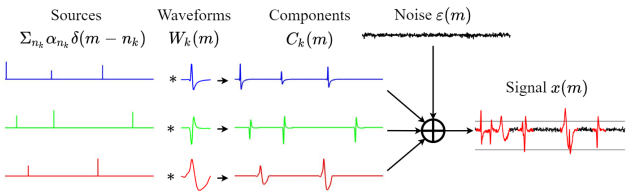


Fig. 2: Modeled spikes under the framework of one-stage methods. It views the signal x as the sum of components $C_k, k = \{1, 2, 3\}$ and noise ε. Here, the m-th sampling point of the k-th component $C_k(m)$ is formed by taking the convolution of the template waveform W_k with the time-shifted impulses $\delta(\cdot)$ amplified with α_{n_k}, where n_k is the number of impulse times of the corresponding k-th neuron. That is, $x(m) = \sum_k \sum_{n_k} \alpha_{n_k} W_k(m) * \delta(m - n_k) + \varepsilon(m)$, where $*$ represents the convolution operation.

the one hand, the two-stage methods include Wave_clus [20], [21] and High-Accuracy Spike Sorting (HASS) [22], [9], which extract features in a time-frequency domain. There also exist methods that extract features only in the time domain, including the features from principal component analysis [23], linear discriminant analysis [24], locality preserving projection [25], etc. Since these methods did not integrally consider the underlying assumptions behind the distinct extraction and clustering procedures, they may readily induce errors and render failures to the performance [7].

On the other hand, the one-stage methods usually perform better since they take the underlying assumptions of different data procedures into consideration and integrate feature extraction and clustering into a unified framework. These methods mainly include [6], [26], [27], [28], [29], etc. In particular, [26] specified the issued time of neurons as a prior Bernoulli distribution and modeled them as a generative process; [27] assumed the recorded voltage trace as a noisy linear superposition of different template waveforms and took a continuous basis pursuit method [30] for the solution. As demonstrated in Fig.2, both methods [26], [27] fix the template waveforms W_k to follow some distributions. Unlike fixing the template waveforms, other methods focus on dynamically estimating the waveforms. For instance, [6], [28] proposed a Focused Mixture Model (FMM) and used a Bayesian dictionary learning strategy to construct suitable suppositional waveforms automatically; [29] developed a Sparse Coding Spike Sorting algorithm (SCSS) that merges the neural impulse time δ and the amplitudes α (in Fig. 2) as one decipher vector s. In this way, SCSS could transform the convolution operation $*$ to the multiplication one, recasting the model in a matrix form. Please refer to the foundation in Section II for details [29].

Though these one-stage model-based methods have evaded the problem of inconsistent assumptions incurred in the two-stage methods, they cannot perform well in cases where the spikes are similar and overlapping or where there are heavy noises in the spikes. Such characteristics are prevalent in the extracellular recording [1], [2] and must be carefully considered to avoid high failure rates in the spike sorting. Further, the number of neurons needs to be known or assumed in advance, which hinders the achievement and application of the automatic sorting process.

In this paper, we propose a novel joint low-rank and sparse representation (LRSR) model for spike sorting in the matrix form. Specifically, we first utilize the low-rank model to handle the samples with similar and overlapped spikes. As the low-rank model defines the intrinsic relationships between different clusters, it could capture the spikes of each neuron in fine detail globally, separating the similar and overlapped spikes. We further engage a sparsity constraint to capture the local noises, with which the model is robust to the heavy noises in the spikes. By adopting the spectral clustering theory to estimate the number of clusters, we finally devise our one-stage automatic spike sorting method in LRSR. To summarize, our contributions are mainly three-fold [1]

- Based on the matrix forms of spikes, we model the feature extraction and clustering of spike sorting as a one-stage integrated problem with low-rank and sparsity optimizations. We theoretically analyze why our proposed model could handle cases with similar and overlapped spikes, as well as the embedded noise.
- We develop an automatic spike sorting framework, which includes the estimation of the number of neurons, and the optimization procedure of the proposed LRSR model.
- We conduct extensive experiments both on synthetic and real-world datasets to verify the effectiveness of our proposed method in sorting the similar and heavily noisy corrupted spikes, which also interestingly validate our superiority against the overlapped spikes.

The rest of the paper is organized as follows. We introduce the foundation model, SCSS, in Section II and propose our automatic spike sorting framework, LRSR, in Section III. Section IV presents the solution to the LRSR optimization problem as the core part. In Section V, we conduct extensive experiments compared with several state-of-the-art methods, verifying the LRSR's effectiveness. Finally, we conclude our paper in Section VI.

II. Foundation: SCSS

As shown in Fig.2, after filtering and spike detection, SCSS models the m_{th} sampling point of the recorded signal x as [27],

$$x(m) = \sum_k \sum_{n_k} \alpha_{n_k} W_k(m) * \delta(m - n_k) + \varepsilon(m), \quad (1)$$

where $\delta(\cdot)$ is the unit impulse function, α_{n_k} denotes the amplitude of the neural spike template W_k at the n_k-th time shift, where k ranges from 1 to K. K stands for the total number of neurons. Integrating the neural impulse time and the amplitudes into a decipher vector, and explicitly constructing W in the Toeplitz structure $W' \in \mathbb{R}^{d \times n_w}$, SCSS recasts the above model in a matrix form as [29],

$$X = W'S + E. \quad (2)$$

Here, $X \in \mathbb{R}^{d \times n}$ is the aligned spikes, $S \in \mathbb{R}^{n_w \times n}$ is the sparse coding matrix, and $E \in \mathbb{R}^{d \times n}$ is the noise matrix. d,

[1]This paper is an extension of our conference version [31].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBME.2023.3347137,

n, and n_w are the dimensions of spikes, the number of spikes, and the number of shifted waveform templates, respectively. The following non-negative least square sparse coding strategy is employed to solve the SCSS model [29],

$$\hat{\triangle} = \arg\min_{S} \kappa \left\| X - W'S \right\|_2^2 + \tau \|S\|_1, \tag{3}$$

where $\kappa, \tau \geq 0$ are two trade off parameters [29], [32]. After obtaining a proper solution S, one can determine the clusters that each spike belongs to with a posteriori map estimator,

$$\tilde{\triangle} = \arg\min_{S \in \Omega^N} \left\| X - W'S \right\|_2^2, \tag{4}$$

where Ω^N is the searching region, i.e., each individual template-constructed class.

III. Proposed Framework

In this section, we outline the proposed automatic spike sorting framework, which consists of the LRSR model for the detected spikes (Section III-A) and the automatic strategies (Section III-B), including the accelerated process and the estimation of the number of neurons. Key mathematical notations for LRSR are provided in Table I.

TABLE I: Key mathematical notations for LRSR

Symbols	Definitions
d	Dimension # of each detected spike
n	# of all detected spikes
λ	Trade-off parameter
r_d	Row rank # of the matrix X
$\hat{K}$	Estimated neurons #
J	Auxiliary matrix
$Y_i, i = 1, 2$	Lagrange multipliers
$X \in \mathbb{R}^{d \times n}$	Detected spikes matrix
$E \in \mathbb{R}^{d \times n}$	Noise matrix
$A \in \mathbb{R}^{d \times r_d}$	Dictionary
$Q \in \mathbb{R}^{n \times r_d}$	Linear set of orthonormal basis in X^T
$Z^* \in \mathbb{R}^{r_d \times n}$	Low-rank optimal solution matrix
$\hat{Z} \in \mathbb{R}^{n \times n}$	Block diagonal matrix which preserves the global structure of X
$F \in \mathbb{R}^{n \times n}$	Affinity matrix of spectral clustering
$(X)_+$	$(X)_+ = \max\{X, 0\}$
$\|X\|_p, p = 1, 2$	$\|X\|_p = \left(\sum_{i=1}^{d} \sum_{j=1}^{n} \|X_{ij}\|^p\right)^{1/p}$
$\|X\|_F$	$\|X\|_F = \|X\|_2$
$\|X\|_\infty$	$\|X\|_\infty = \max_{i=1,\ldots,d; j=1,\ldots,n} \{X_{ij}\}$
$\|X\|_*$	$\|X\|_* = \sum_{i=1}^{r_d} \sigma_i$, where σ_i is the singular value of X

A. LRSR Model

Similar to [21], [20], we filter the raw signal by using a fourth-order Butterworth bandpass filter in the range of 300-3,600 Hz and obtain the spike matrix $X \in \mathbb{R}^{d \times n}$ by using automatic threshold detection. In real-world spike sorting scenarios, the raw signals tend to have overlapping and similar spikes, which may be caused by the signal acquisition devices [8] and the characteristics of neuron distribution [9]. Additionally, various noise sources, including those from local field potentials and probe devices [9], are inevitably mixed in the acquired signals. Considering these factors, we propose a low-rank and sparse representation (LRSR) model for the detected spike X as follows,

$$X = AZ + E, \tag{5}$$

where A is an appropriate dictionary, Z and E are the low-rank matrix and the sparse noise matrix, respectively. The main differences between LRSR and the existing model-based spike sorting frameworks include: (I) The dictionary A in LRSR is automatically derived from the spike matrix X, rather than being manually defined based on the "ideal" and assumed-given templates, as demonstrated in prior works [27], [29]. (II) Unlike the sparsity constraint in the SCSS model or the reconstruction constraint of the template [33], [29] capture the local structure of the dataset, the matrix Z in LRSR contains the global structure underlying dataset [34]. (III) Unlike the existing model-based works [33], [27], [29] assume the noises are Gaussian distribution, LRSR does not make prior assumptions. Instead, LRSR claims that only a few spikes are corrupted by noises after filtering and detection and models the noise E as a sparse matrix.

One possible dictionary in LRSR is to set $A = X$ directly, where each entry Z_{ij} can be interpreted as the correlation degree between the i-th and j-th columns in X. In other words, Z_{ij} represents the correlation between i-th and j-th spikes. We can solve for Z and E through convex optimization by formulating the problem as follows [35]:

$$\begin{aligned} \min_{(Z,E)} \quad & \|Z\|_* + \lambda \|E\|_1, \\ s.t. \quad & X = AZ + E, \end{aligned} \tag{6}$$

where $\|Z\|_*$ represents the nuclear norm of Z, which is defined as the sum of the singular values of Z, and $\|E\|_1$ denotes the l_1-norm of E, which is defined as the sum of the absolute value of the elements in E. $\lambda \geq 0$ is a tradeoff parameter that allows us to balance the level of noise interference against the pure data. As λ increases, the model becomes less reliant on the structure present in the pure data. It is worth noting that noise is not always present in every spike, and in some cases, it may not heavily disrupt the pure spikes. This suggests that the sparse constraint on E is a valid assumption in our model.

As a quick demonstration, Fig.3 provides a graphical representation of the optimization results obtained from the model (6). In this example, the size of X is 64×3514. By solving the model (6) over X, we obtain the optimal values of Z^* and E^*. To provide a clear visual representation, only the first 200 entries of the detected spikes X are shown with their corresponding pure waveforms AZ^* and sparse noise E^*. Each column of X, AZ^*, and E^* represents one detected spike, its corresponding pure waveform, and the embedded noise, respectively. Compared with detected spikes X, the pure waveforms, AZ^*, display a more regular profile with less disturbance and exhibit near global structures, where the rank of Z^* is approximately 15. The noise is quite sparse, where only 5% of E^* columns present noise disruption.

B. Automatic Spike Sorting Framework with LRSR

1) Accelerated Process of LRSR: Optimizing the $n \times n$ matrix Z from model (6) can be challenging because there

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBME.2023.3347137,

(a) Spikes X (b) Waveforms AZ^* (c) Noise E^*

Fig. 3: A quick demonstration of 200 detected spikes X represented by pure waveforms AZ^* and noise E^*. Here, each column in (a), (b), and (c) represents one spike, one pure waveform, and one noise vector embedded in the corresponding detected spike, respectively.

are usually many spikes, which leads to high time complexity. To achieve faster implementation, we can represent A as,

$$A = XQ \in \mathbb{R}^{d\times r_d}, \tag{7}$$

where $Q \in \mathbb{R}^{n\times r_d}$ serves as a set of orthonormal basis vectors for the transposed matrix X^T, and r_d is the rank of X and satisfies $r_d \leq d \ll n$. Note that the basis vectors could be calculated in advance by orthogonalizing the rows of X [36], i.e., $orth(X^T)$. This process effectively reduces the dimension of matrix Z from $(n \times n)$ to $(r_d \times n)$.

With Eq.(7) and assuming the optimal solution of model (6) is (Z^*, E^*), which is detailed in Section IV, we could recover the square matrix $\hat{Z} \in \mathbb{R}^{n\times n}$ by,

$$\hat{Z} = QZ^*. \tag{8}$$

Mathematically, under mild assumptions [36], [35], such as the independence or disjointness of data subspaces within the matrix X, the matrix $\hat{Z}$ exhibits a blocked diagonal structure. To enhance clustering performance, we choose to apply the skinny singular value decomposition (SVD) [37] to $\hat{Z}$. This enables us to robustly perform clustering analysis using various spectral clustering algorithms [38].

2) Estimating the Number of Neurons : Separating the estimation of the number of neurons from the spike sorting algorithm always results in much un-associated and superfluous work and even leads to worse results [10], [40]. To avoid these problems, we incorporate such estimation procedure into LRSR using spectral theorem [36], [38].

Specifically, we construct an affinity matrix F by performing skinny SVD on $\hat{Z}$ at first. Then, we estimate the number of neurons, K, by counting the number of ones in the singular values set $\{\sigma_i\}_{i=1}^n$ of the auxiliary matrix $L = D^{-\frac{1}{2}}FD^{-\frac{1}{2}}$, where D is a diagonal matrix whose diagonal element is the sum of the corresponding F's row [38]. In our method, we utilize soft thresholding to account for the unavoidable noise in the real scenario [36]. The estimated number of neurons, denoted as $\hat{K}$, is determined as follows,

$$\hat{K} = \text{round}\left(\sum_{i=1}^{n} f_\tau(\sigma_i)\right), \tag{9}$$

where $round(\cdot)$ is the operation of round towards the nearest integer, and $f_\tau(\cdot)$ is a soft thresholding operator defined as,

$$f_\tau(\sigma) = \begin{cases} 1, & if\ \sigma \geq \tau \\ 2^{\frac{\sigma^2}{\tau^2}} - 1, & otherwise \end{cases}, \tag{10}$$

Algorithm 1 Automatic spike sorting framework with LRSR

Input: Extracellularly recorded raw data.
Output: Sorted spikes.

1: Apply a Butterworth bandpass filter to the raw data with a passband from 300 to 6,000 Hz, resulting in filtered signal x_0.
2: Apply automatic threshold detection [20] on x_0 to obtain the spikes matrix X.
3: Normalize each row in X so that its elements fall within the range of 0.2 to 0.8 [39].
4: Execute LRSR method described in Algorithm 2 on matrix X to obtain matrix $\hat{Z}$ in a block diagonal form.
5: Perform skinny SVD on $\hat{Z}$, i.e. $\hat{Z} = U\Sigma V^T$, to get U and Σ.
6: Construct the intermediate matrix P with normalized rows by $U\Sigma^{1/2}$.
7: Calculate the affinity matrix F with its (i,j)-element as $F_{i,j} = \left(\left[PP^T\right]_{i,j}\right)^2$.
8: Calculate the diagonal matrix D whose diagonal element is the sum of corresponding F's row.
9: Construct the auxiliary matrix $L = D^{-\frac{1}{2}}FD^{-\frac{1}{2}}$, and estimate the number of neurons $\hat{K}$ by (9).
10: Perform Ng-Jordan-Weiss algorithm on matrix F with the number of neurons $\hat{K}$ to get the sorted spikes.

where $0 < \tau < 1$ is a parameter used to weigh the strength of the soft thresholding. As τ approaches 1, the impact of soft thresholding decreases. We perform feature extraction and clustering with the Ng Jordan Weiss [38] spectral clustering algorithm for four-step spike sorting. The entire spike sorting framework based on LRSR is outlined in Algorithm 1.

IV. LRSR Optimization Method

In this section, we present an efficient solution for the LRSR model by engaging the alternate augmented Lagrange multipliers strategy [41].

We start by introducing an auxiliary variable, J, to reformulate model (6) as follows,

$$\begin{aligned} \min_{(J,Z,E)} \ & \|J\|_* + \lambda\|E\|_1, \\ s.t.\ & X = AZ + E, \\ & Z = J. \end{aligned} \tag{11}$$

By introducing the Lagrange multiplier, we can get the corresponding unconstrained optimization model as,

$$\begin{aligned} \min_{(J,Z,E)} \|J\|_* + \lambda\|E\|_1 + \langle Y_1, X - AZ - E\rangle + \langle Y_2, Z - J\rangle \\ + \frac{\mu}{2}\left(\|X - AZ - E\|_F^2 + \|Z - J\|_F^2\right), \end{aligned} \tag{12}$$

where Y_1 and Y_2 are Lagrange multipliers, $\langle\cdot,\cdot\rangle$ and $\|\cdot\|_F^2$ denote the matrices inner product operator and matrices Frobenius norm, respectively, $\mu > 0$ is a penalty parameter. Below, we show how to update each variable alternatively.

A. Update J with Singular Value Thresholding

With the given matrices Z and E, J is updated by,

$$J^{t+1} = \arg\min_{J} \|J\|_* + \langle Y_2, Z^t - J\rangle + \frac{\mu}{2}\left\|Z^t - J\right\|_F^2, \quad (13)$$

where t is the index of iteration. Based on the equivalence operators on matrices A and B, $\langle A, A\rangle = \|A\|_F^2 = \mathrm{tr}(A^T A)$ and $\langle A, B\rangle = \mathrm{tr}(A^T B)$, we can recast (13) as,

$$\begin{aligned} J^{t+1} &= \arg\min_{J} \|J\|_* + \mathrm{tr}\left(Y_2^T Z^t - Y_2^T J\right) \\ &\quad + \frac{\mu}{2}\mathrm{tr}\left(Z^{tT} Z^t - Z^{tT} J - J^T Z^t + J^T J\right), \\ &= \arg\min_{J} \|J\|_* + \mathrm{tr}\Big(-Y_2^T J - \mu J^T Z^t + \frac{\mu}{2} J^T J\Big), \\ &= \arg\min_{J} \frac{1}{\mu}\|J\|_* + \frac{1}{2}\left\|J - \left(Z^t + \frac{Y_2}{\mu}\right)\right\|_F^2. \end{aligned} \quad (14)$$

where $tr(\cdot)$ is the matrix trace operator. The last two equations are derived from the fact that terms independent of J can be treated as constants. Adding or subtracting such terms does not impact the optimal solution of Problem (13). Finally, we can determine J^{t+1} by applying the singular value thresholding [42] on $(Z^t + \frac{Y_2}{\mu})$ with,

$$J^{t+1} = U\mathcal{D}_{1/\mu}(\Sigma)V^T, \quad (15)$$

where the matrices, U, Σ, and V, satisfy the SVD on $\left(Z^t + \frac{Y_2}{\mu}\right)$, i.e., $Z^t + \frac{Y_2}{\mu} = U\Sigma V^T$, and $\mathcal{D}_{1/\mu}(\Sigma)$ is the singular value shrinkage operator defined as,

$$\mathcal{D}_{1/\mu}(\Sigma) = \mathrm{diag}\left(\left(\sigma - \frac{1}{\mu}\right)_+\right), \quad (16)$$

where σ is a vector containing the diagonal elements of Σ, $z_+ = \max(z, 0)$, and $\mathrm{diag}(z)$ is the diagonal matrix by specifying the diagonal elements with vector z while other elements with zeros.

B. Update Z with Matrix Derivation Rule

With the given matrices J and E, Z is updated by,

$$\begin{aligned} Z^{t+1} &= \arg\min_{Z} \langle Y_1, X - AZ - E^t\rangle + \langle Y_2, Z - J^{t+1}\rangle \\ &\quad + \frac{\mu}{2}\Big(\|X - AZ - E^t\|_F^2 + \|Z - J^{t+1}\|_F^2\Big), \\ &= \arg\min_{Z} \mathrm{tr}\left[\frac{\mu}{2} Z^T\left(A^T A + I\right)Z\right. \\ &\quad \left. - \left(-Y_2^T + \mu X^T A - \mu {E^t}^T A + \mu {J^{t+1}}^T\right)Z\right], \end{aligned} \quad (17)$$

where I is the identity matrix with the same dimensions as $A^T A$. By using the first-order condition and the following properties of the matrix trace operator,

$$\frac{\partial \mathrm{tr}\left(A^T B\right)}{\partial A} = B, \quad \frac{\partial \mathrm{tr}\left(A^T B A\right)}{\partial A} = \left(B^T + B\right)A, \quad (18)$$

we can recast (17) as,

$$Z^{t+1} = \left(A^T A + I\right)^{-1} \cdot \left(\frac{A^T Y_1 - Y_2}{\mu} + A^T X - A^T E^t + J^{t+1}\right). \quad (19)$$

where A^{-1} is the inverse of matrix A.

Algorithm 2 Solution for LRSR model

Input: Spikes matrix, $X \in \mathbb{R}^{d\times n}$, and parameter λ.
Output: Block diagonal matrix, $\hat{Z} \in \mathbb{R}^{n\times n}$, and noise matrix, $E^* \in \mathbb{R}^{d\times n}$.

1: Initialization: $Q \leftarrow \mathrm{orth}(X^T) \in \mathbb{R}^{n\times r_d}$, $A \leftarrow XQ \in \mathbb{R}^{d\times r_d}$, $Z^0 = J^0 = Y_2^0 \leftarrow 0_{r_d\times n}$, $E^0 = Y_1^0 \leftarrow 0_{d\times n}$, $\mu^0 = 10^{-6}$ and $t \leftarrow 0$.
2: Update J, Z, E and other parameters sequentially following Eqs.(15), (19), (23) and (24).
3: Quit the loop if the convergence criterion (26) holds, otherwise, set $t \leftarrow t+1$ and go back to step 2.
4: $Z^* \leftarrow Z^{t+1}$, $E^* \leftarrow X - AZ^*$.
5: Return E^* and $\hat{Z} \leftarrow QZ^*$.

C. Update E with Soft Thresholding

With the given matrices J and Z, E is updated by,

$$\begin{aligned} E^{t+1} &= \arg\min_{E} \lambda\|E\|_1 + \left\langle Y_1, X - AZ^{t+1} - E\right\rangle \\ &\quad + \frac{\mu}{2}\left\|X - AZ^{t+1} - E\right\|_F^2. \end{aligned} \quad (20)$$

Similar to the derivations of (14) and (19), we can recast (20) as,

$$E^{t+1} = \arg\min_E \tfrac{\lambda}{\mu}\|E\|_1 + \tfrac{1}{2}\left\|E - \left(\tfrac{Y_1}{\mu} + X - AZ^{t+1}\right)\right\|_F^2. \quad (21)$$

(21) is a standard sparse optimization problem. By conducting the first-order condition on (21) in the form of,

$$\frac{\partial\left(\frac{\lambda}{\mu}\|E\|_1 + \frac{1}{2}\|E - G\|_F^2\right)}{\partial E} = 0, \quad (22)$$

and all boundary points condition elementwisely, where $G = \frac{Y_1}{\mu} + X - AZ^{t+1}$ and $\frac{\partial}{\partial E}$ is a gradient operator on E, we can update E^{t+1} by,

$$E^{t+1} = \left(\mathrm{abs}\left(G\right) - \frac{\lambda}{\mu}\right)_+ \cdot \mathrm{sgn}(G), \quad (23)$$

where (G) and $sgn(G)$ are the absolute value and sign operators on matrix G, respectively.

D. Update Other Parameters Automatically

Finally, we update other parameters, including two Lagrange multipliers Y_1 and Y_2, and the penalty parameter μ, by,

$$\begin{aligned} Y_1^{t+1} &= Y_1^t + \mu^t \Delta Y_1^{t+1}, \\ Y_2^{t+1} &= Y_2^t + \mu^t \Delta Y_2^{t+1}, \\ \mu^{t+1} &= \min\left(\delta \times \mu^t, \mu_{max}\right), \end{aligned} \quad (24)$$

where $\delta > 1$ and μ_{max} stand for the step size and the maximal value regarding the parameter μ, and in our model, we fix them with 1.1 and 10^6, respectively. We define $\Delta Y_i^{t+1}, i = 1, 2$ as,

$$\begin{aligned} \Delta Y_1^{t+1} &= X - AZ^{t+1} - E^{t+1}, \\ \Delta Y_2^{t+1} &= Z^{t+1} - J^{t+1}, \end{aligned} \quad (25)$$

and use the following convergence criterion to determine the termination,

$$\max\left(\|\Delta Y_1^{t+1}\|_\infty, \|\Delta Y_2^{t+1}\|_\infty\right) < \epsilon, \quad (26)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBME.2023.3347137,

where ϵ is a small enough precision to control the model convergence, and $\|\cdot\|_\infty$ refers to the maximum entry of a matrix. We summarize the overall solution for the LRSR model in Algorithm 2.

V. Experiments

In this section, we validate the effectiveness of the proposed method, LRSR, on various datasets. All experiments are conducted in the same PC with Intel Core i7-8750H, 2.21 GHz CPU, and 32GB main memory.

A. Experimental Settings

1) Datasets: We utilize simulated and real-world benchmark datasets in our experiments.

The first simulated dataset is proposed by Quiroga Rodrigo Quian *et al.* [21], [20], [43], which consists of 20 sub-datasets originating from 594 neurons in the neocortex and basal ganglia. Each sub-dataset includes three distinct neurons with known labels, varying levels of noise (with a standard deviation ranging from 0.05 to 0.4), overlapping spikes, and similar waveforms. The noise in these sub-datasets mimics background activity generated by distant neurons. The degree of waveform similarity is denoted as "easy" or "difficult" on the sub-dataset name, where "easy" indicates low similarity and "difficult" suggests high waveform similarity. More details about this dataset, including their names (DS), noise levels (NL), the number of spikes/overlapped spikes (SN(O)), and the number of spikes/overlapped spikes per neuron (SN(O)/Neuron), can be found in Table II. Besides, we also utilize another simulated dataset proposed by Pedreira *et al.* [17], [21], which contains 95 sub-datasets. Each set contains background noise, multiunit activity, and 2 to 20 neurons (5 sub-datasets for each neuron count). This dataset is primarily used to evaluate the effectiveness of the estimation process of the automatic spike sorting methods.

We engage the real-world vivo dataset, HC1 [16], [27], in our experiments. HC1 is recorded from the hippocampus of an anesthetized rat. It contains unlabeled spikes of several unknown neurons from the implanted extracellular tetrode and labeled spikes of one known neuron from an implanted intracellular electrode. Based on these labeled spikes, we can sort and analyze all spikes to study different sorting methods [27], [7].

2) Evaluation Metrics: Since the proposed LRSR is fully unsupervised, we use the *Adjusted Rand Index (ARI)* [44], [10], [45] as our main performance metric. ARI is an improved metric to eliminate the bias chance from the Rand Index (RI) that is formulated as $(TP+TN)/(TP+FP+FN+TN)$, where TP, TN, FN, and FP represent true positives, true negatives, false positives, and false negatives, respectively. RI values range from 0 to 1, with 0 indicating no agreement between two clustering results and 1 indicating perfect agreement. Given a sample-set S with n items, a known ground truth partition of c clusters denoted as $V=\{V_1,V_2,\ldots,V_c\}$, and another validated partition from any clustering algorithm of r clusters denoted as $U=\{U_1,U_2,\ldots,U_r\}$ (r may not equal c), we can outline the overlapping elements between U and V in the contingency table (27),

$U \backslash V$	V_1	V_2	$\ldots$	V_c	Sums
U_1	n_{11}	n_{12}	$\ldots$	n_{1c}	$n_{1\cdot}$
U_2	n_{21}	n_{22}	$\ldots$	n_{2s}	$n_{2\cdot}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
U_r	n_{r1}	n_{r2}	$\ldots$	n_{rc}	$n_{r\cdot}$
Sums	$n_{\cdot 1}$	$n_{\cdot 2}$	$\ldots$	$n_{\cdot c}$	n

(27)

where n_{ij} is the number of the overlapped samples between sets U_i and V_j, i.e., $n_{ij}=|U_i\cap V_j|$, and ARI is defined as,

$$ARI=\frac{\sum_{ij}\binom{n_{ij}}{2}-\left[\sum_i\binom{n_{i\cdot}}{2}\cdot\sum_j\binom{n_{\cdot j}}{2}\right]/\binom{n}{2}}{\left[\sum_i\binom{n_{i\cdot}}{2}+\sum_j\binom{n_{\cdot j}}{2}\right]/2-\left[\sum_i\binom{n_{i\cdot}}{2}\cdot\sum_j\binom{n_{\cdot j}}{2}\right]/\binom{n}{2}},$$

where $\binom{n}{2}=n(n-1)/2$. ARI can be negative if predicted clusters are fewer than expected, while RI ranges from 0 to 1.

Besides, we also engage the quantitative metrics of misclassification number, error rate, and mean processing time. The *misclassification number* is the count of missed target spikes, the *error rate* is the percentage of incorrectly classified spikes, and the *mean processing time* is the total processing time divided by the number of datasets. For HC1 evaluation, false negative rate (FNR), false positive rate (FPR), and scatter plots are used to demonstrate method effectiveness in spike sorting [24], [7].

3) Baseline Methods: We compare four state-of-the-art spike sorting methods, HASS [22], SCSS [29], FMM [6] and Wave_clus [21]. HASS and SCSS require the manually defined neuron number, while FMM and Wave_clus are fully automatic. Additionally, we also compare the baseline, LR, an ablation method that pays no attention to noise interference compared with LRSR. For fairness, The raw signal filtering and spike detection in all comparisons follow the methodology outlined in Algorithm 1. Unless specified otherwise, the parameters of baselines align with those in the original paper, maintaining consistency.

B. Evaluation on Simulated Dataset

1) Overall Experiments: We conduct extensive experiments to evaluate the performance of all baseline methods in spike-sorting on the simulated datasets.

As shown in Table II, the results include ARI, neuron number (NN), and average processing time, where ARI* represents ARI with the fixed ground-truth neuron number of 3. Overall, all methods perform well on low-noise, distinct waveform datasets labeled "Easy". LRSR and HASS outperform Wave_clus in ARI on most datasets, while SCSS and FMM struggle with high-noise datasets. LRSR's performance is comparable to HASS, especially in high-noise situations. It is worth noting that HASS requires the prior number of neurons, whereas LRSR is fully automatic. Besides, FMM yields a negative ARI on "Difficult1" when predicting fewer neurons than the expected 3, indicating ARI's sensitivity to the estimation results. SCSS excels in handling overlapping spikes but suffers on heavily noise-disrupted datasets, possibly due to the fixed weight of the l_0 norm term as indicated in Eq.(3). In construct, LRSR both automatically estimates the number of

TABLE II: Details of the simulated dataset and the results of ARI value, estimated number of neurons (NN) of various methods, where ARI* denotes the results with a manually fixed NN of 3.

DS	NL	SN(O)	SN(O)/Neuron	SCSS	HASS	LR	FMM		Wave_clus		LRSR	
				ARI*	ARI*	ARI*	ARI	NN	ARI	NN	ARI	NN
Easy1	005	3514(785)	1165(250) 1157(275) 1192(260)	0.9633	**0.9872**	0.8405	0.9005	8	0.8906	3	**0.9668**	3
	010	3522(769)	1151(248) 1134(264) 1237(257)	0.9563	**0.9854**	0.7823	0.9370	6	0.9561	3	**0.9719**	3
	015	3477(784)	1132(242) 1188(272) 1157(270)	0.8973	**0.9862**	0.3855	0.9379	5	**0.9685**	3	0.9682	3
	020	3474(796)	1198(279) 1128(248) 1148(269)	0.8223	0.9786	0.3703	0.9528	5	**0.9791**	3	0.9768	3
	025	3298(712)	1094(237) 1089(229) 1115(246)	0.7911	**0.9827**	0.1533	0.9627	4	0.8000	6	**0.9819**	3
	030	3475(846)	1162(294) 1164(285) 1149(267)	0.7403	0.9735	0.3233	0.9384	4	0.8248	3	**0.9785**	3
	035	3534(832)	1208(285) 1137(269) 1189(278)	0.6880	0.9705	0.3012	0.8922	3	0.7893	4	**0.9711**	3
	040	3386(741)	1079(238) 1158(261) 1149(242)	0.6358	**0.9806**	0.2377	0.8274	3	0.9597	3	**0.9609**	3
Easy2	005	3410(791)	1130(274) 1113(257) 1167(260)	0.9114	**0.9576**	0.8537	**0.8999**	6	0.6998	6	0.8475	4
	010	3520(826)	1160(269) 1146(280) 1214(277)	0.5666	**0.9855**	0.8342	0.8593	6	0.9604	3	**0.9789**	3
	015	3411(763)	1181(265) 1098(237) 1132(261)	0.5651	0.9505	0.5301	0.5390	3	0.9641	3	**0.9649**	3
	020	3526(811)	1186(262) 1188(278) 1152(271)	0.5523	0.9629	0.4022	0.5348	3	0.9553	3	**0.9713**	3
Difficult1	005	3383(767)	1115(244) 1113(256) 1155(267)	0.7956	0.8442	0.9069	0.7743	6	0.7336	4	**0.9664**	3
	010	3448(810)	1164(260) 1155(269) 1129(281)	0.4467	0.7903	0.8846	0.6893	4	0.9131	3	**0.9460**	3
	015	3472(812)	1159(275) 1172(260) 1141(277)	0.0073	0.6187	0.8113	0.2619	4	0.6121	3	**0.8499**	3
	020	3414(790)	1136(267) 1099(257) 1179(266)	0.0004	0.5919	0.6450	-0.0001	2	0.7297	4	**0.7474**	3
Difficult2	005	3364(829)	1120(271) 1109(274) 1135(284)	0.9507	**0.9646**	0.9078	**0.8715**	6	0.7471	4	0.8265	4
	010	3462(720)	1187(230) 1136(238) 1139(252)	0.2848	**0.9683**	0.2793	0.8651	5	0.9630	3	**0.9665**	3
	015	3440(809)	1142(284) 1113(262) 1185(263)	0.4173	0.9346	0.2216	0.5323	3	0.8010	4	**0.9622**	3
	020	3493(777)	1151(260) 1195(277) 1147(240)	0.2072	0.9289	0.1640	0.5405	3	0.9704	3	**0.9820**	3
Mean processing time (in seconds)				137.55	394.35	24.19	8.27		3.47		16.21	

neurons as shown in section III-B.2 and sets the parameters as analyzed in section IV-D, indicating the effectiveness of our proposed spike sorting framework.

As for the ablation model, LR excels on "Difficult" datasets with notable ARI values due to its focus on low-rank representation, effectively sorting similar spikes. However, its performance drops on high-noise datasets due to the lack of a noise cancellation strategy. In contrast, LRSR significantly improves by incorporating a noise term. Interestingly, the time complexity of LRSR is less than that of LR. There may be owing to two main reasons. Firstly, the rank of Z in the LR model is large as the presence of noise, which results in more iterations and SVD operations in the solution process. Secondly, compared with the LR model, the noise term added in the LRSR is element-wisely updated using a soft threshold algorithm, whose complexity is low and negligible. The processing time of LRSR is also acceptable for the online spike sorting applications [46], as evidenced by its average processing time of 16.21 seconds, less than that of the recording period, 60 seconds, of each dataset.

In summary, as one of the automatic methods, LRSR outperforms others in achieving the best ARI and estimated neurons, comparable to supervised spike sorting methods. By incorporating a noise cancellation strategy, LRSR outperforms LR in sorting performance and time complexity, which is also acceptable for online spike sorting applications.

2) Effects of Overlapping and Noise Interference: We only report the results of LRSR and HASS here, since the ARI values of other methods (SCSS, LR, FMM, and Wave_clus) were relatively much lower, as shown in Table II. We use 3 artificially assigned neurons to conduct the following experiments to ensure fair evaluations.

Fig.4 shows the total numbers of the missed target spikes, including the non-overlapped spikes and the overlapped spikes obtained by both LRSR and HASS methods. It can be seen that 15 datasets are well classified without any non-overlapped missed target spike for LRSR, while only 10 for HASS. There are five datasets with non-overlapped missed target spikes for LRSR, "Easy1_noise035", "Easy1_noise040", "Difficult1_noise010", "Difficult1_noise015" and "Difficult1_noise020", and all these

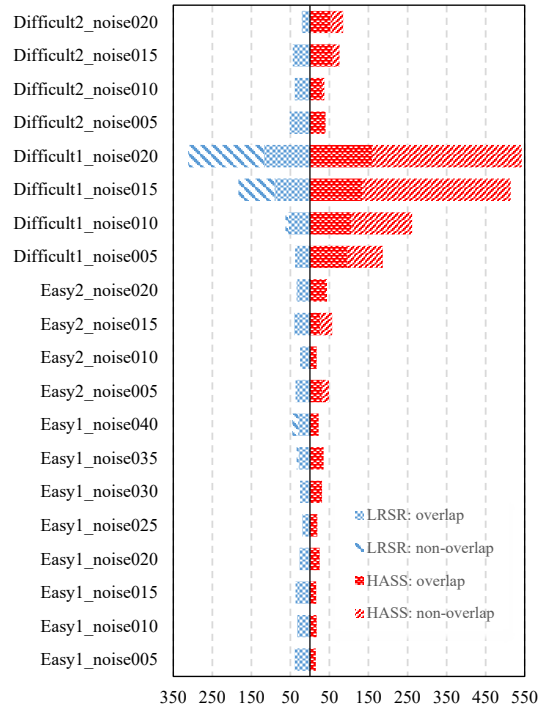


Fig. 4: The number of miss-classified spikes (including the non-overlapped and overlapped spikes) of LRSR and HASS.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBME.2023.3347137,

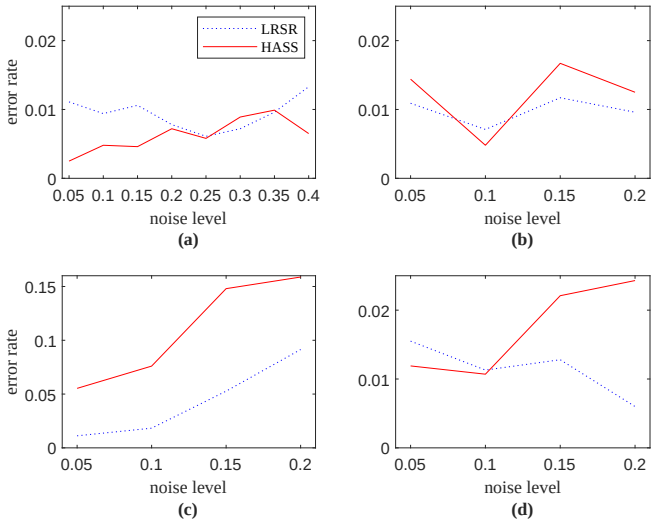


Fig. 5: Error rates of LRSR and HASS on datasets, (a) "Easy1", (b) "Easy2", (c) "Difficult1", and (d) "Difficult2", disrupted by different noise levels.

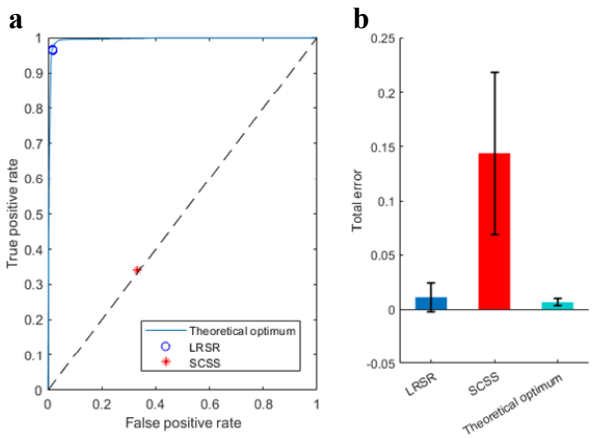


Fig. 6: (**a**) The influence of parameter λ in LRSR compared with the theoretical optimum performance on "Difficult1_noise020" dataset and (**b**) the mean deviation of total errors over all datasets.

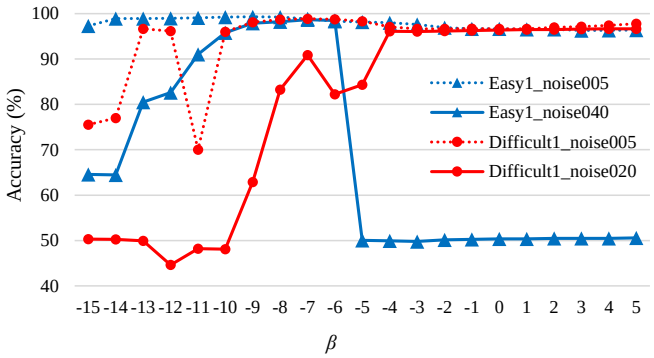


Fig. 7: Effects of the parameter $\lambda = 2^{\beta}$ on the accuracy of spike sorting.

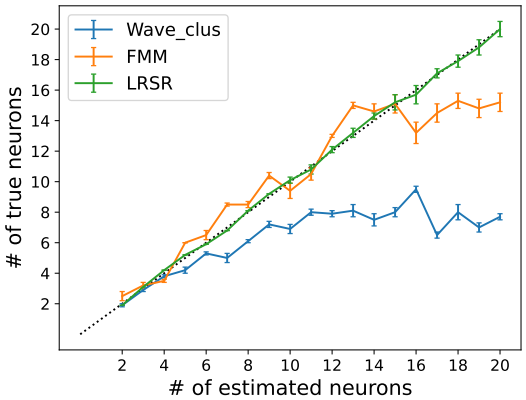


Fig. 8: Estimation results on the simulated dataset of automatic spike sorting methods.

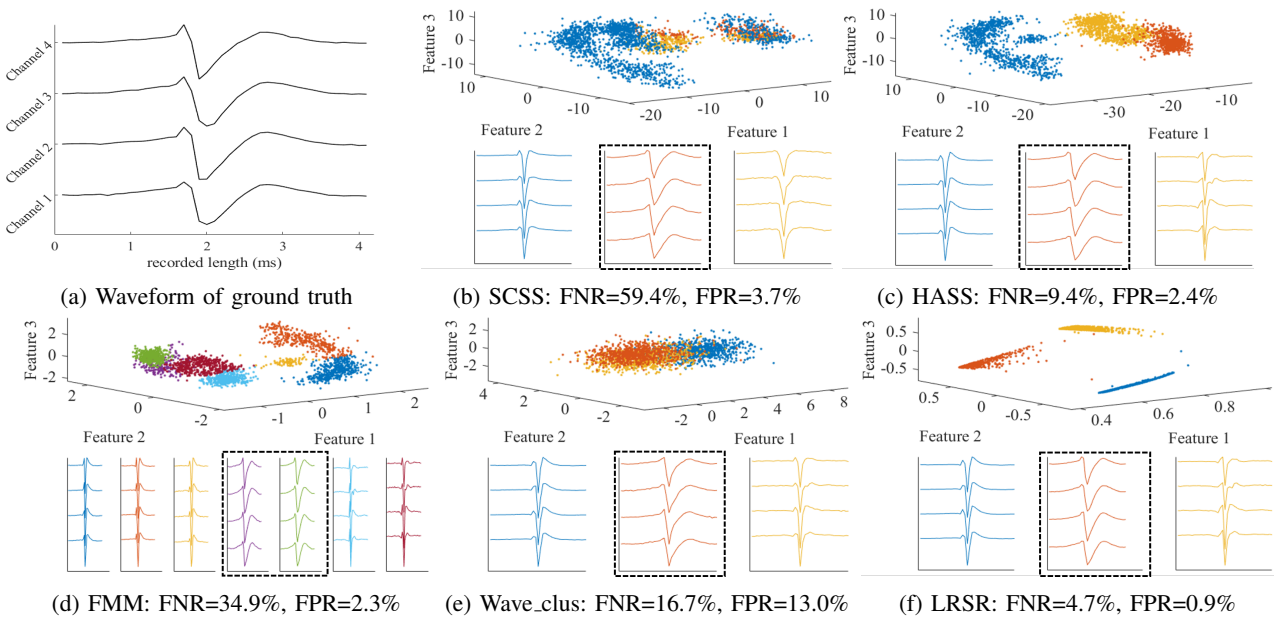
waveforms are corrupted by the high-level noises. HASS performs worse on them. Besides, the misclassification spikes of "Difficult1" are relatively higher than other datasets. It is mainly because the similarities of the waveforms in the "Difficult1" are shallow, and the high-level noises corrupt these waveforms. Overall, the total numbers of the missed target overlapped spikes of most datasets, except for "Difficult1_noise015" and "Difficult1_noise020", are under 50 for LRSR. The sorting results of overlapping spikes are much worse for HASS, especially on the "Difficult1" datasets, demonstrating the satisfactory performance of our proposed LRSR.

Fig.5 shows the error rates obtained by LRSR and HASS methods at different noise levels. Overall, with the increasing noise levels, the classification error rate of LRSR is more stable than HASS. This conclusion is particularly evident in the datasets "Easy2", "Difficult1", and "Difficult2". For the dataset "Easy1", although the classification error rate of LRSR is slightly worse than that of HASS, they always remain near the error rate of 1%. In the "Difficult1" and "Difficult2" datasets with the high spikes' similarity, LRSR achieves better results than HASS, which generally confirms the sound anti-noise performance of our proposed model.

3) Parameter Analysis: Algorithm 2 implies that the LRSR method has only one parameter, λ, to decide. In general, the choice of this parameter depends on the prior knowledge of the noise level of a signal, which many existing methods could estimate [47]. When the noise is slight, we should use a relatively large λ, while a minor λ is recommended with heavy noises.

We conducted experiments of parameter analysis on four datasets, including both "easy" and "difficult". In particular, we used two datasets with the lowest noise level, 0.05, named "Easy1_noise005" and "Difficult1_noise005", and the other two with the highest noise level, 0.4 for "easy" and 0.2 for "difficult", named "Easy1_noise040" and "Difficult1_noise020". We set $\lambda = 2^{\beta}$, where β was set from -15 to 5 with the increasing step 1 so that the conducted experiments are in a wide enough range of λ. The other steps are the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TBME.2023.3347137,



(a) Waveform of ground truth (b) SCSS: FNR=59.4%, FPR=3.7% (c) HASS: FNR=9.4%, FPR=2.4%

(d) FMM: FNR=34.9%, FPR=2.3% (e) Wave_clus: FNR=16.7%, FPR=13.0% (f) LRSR: FNR=4.7%, FPR=0.9%

Fig. 9: Results on the real-world dataset, including (a) the mean waveform of the ground truth, and the low-dimensional distributions of spikes and the corresponding mean waveforms of each tentative neuron got from the spike sorting methods, (b) SCSS, (c) HASS, (d) FMM, (e) Wave_clus, and (f) LRSR. We provide the FNR and FPR results of each method in the caption accordingly.

same as Algorithm 1, and we recorded the final results of the accuracy with the increasing parameter β.

As shown in Fig.7, by setting a relatively more minor β, the performance on such two high noise level disrupted data, "Easy1_noise040" and "Difficult1_noise020", are satisfactory, and vice versa for the slight noise corrupted dataset.

Also, to evaluate the influence of λ in LRSR compared with the theoretical optimum performance, we employed the best ellipsoid error rate (BEER) measure [16], [48] on the "Difficult1_noise020" dataset. We engaged one two-layer feed-forward neural network and a 5-fold cross-validation method in a supervised manner, and plotted the final theoretical upper bound region of convergence (ROC) curve [48] with cyan color in Fig.6.a. Based on the prior fact that the noises in the selected dataset are not too heavy, we assigned λ of LRSR in a relatively reasonable guessing range, from 1 to 50, for experiments. Fig.6.a demonstrates 50 blue dots, representing the mean FPR and TPR about the multi-classification performance of LRSR. For comparison, we also conducted the SCSS method by manually setting the number of neurons with 3, and plotted the result in Fig.6.a with one red star dot. The corresponding total error's mean and standard deviation over all twenty datasets for theoretical optimum (BEER measure), LRSR, and SCSS are given in Fig.6(b).

On the dataset "Difficult1_noise020", the performance of LRSR is close to the theoretical optimum obtained by BEER measure (Fig.6(a))), while the performance of SCSS is inferior, with error rates typically exceeding 20% (Fig.6(b)). Among all twenty datasets, we found no significant difference in the total error between the LRSR and theoretical optimum, but a distinct difference between the performances of LRSR and SCSS. Thereafter, we fixed the β with -7, i.e., $\lambda = 2^{-7}$, for simplifying the procedure in the whole experiments.

4) Estimating Number of Neurons: To evaluate the effectiveness of our proposed automatic spike sorting method, specifically in estimating the neuron numbers, we conduct experiments on the simulations with various neurons from 2 to 20 [17], [21], [45]. With each neuron count, there are 5 different sub-datasets. We conduct 20 experiments for each neuron count and report the final average estimated results and its standard deviation accordingly. As shown in Fig.8, Wave_clus exhibits a similar trend as their primary work [17], illustrating that Wave_clus could estimate up to 8 neurons. FMM improves this estimation to around 14, although its standard deviation is larger than Wave_clus. In contrast, LRSR produces a stable and accurate estimation from 2 to 20 neurons, verifying the effectiveness of our estimation strategy. This is mainly attributed to the well-constructed low-rank matrix and the robust thresholding estimator given in section III-B.2.

C. Evaluation on Real-World Benchmark

To evaluate the effectiveness of LRSR, we conducted experiments on the real-world dataset HC1. Refers to the existing works [24], [27], we filter the raw signal with a highpass Butterworth filter at 250 Hz with an order of 50 at first. Then, we detect all spikes by a thresholding method [27]. Each spike is 4×41, as indicated in Fig.9a. We concatenate the spike along the channel to a 164-dimension signal for automatic sorting [27], [7]. It is worth noting that we manually fixed the neuron number of SCSS and HASS with the empirical number 3 as they could not estimate the number automatically [27].

As shown in Figs.9b-f, we illustrate the low-dimensional spikes in their feature space to indicate the separations of each

method. Below the feature space illustration, we also provide the corresponding mean waveforms of each method, where the most similar one(s) to the ground truth is(are) emphasized with the dashed box. The quantitative results of FNR and FPR are given in the caption of each figure. Generatively, LRSR outperforms others in both the separations in the feature space and the quantitative results. From the perspective of the mean waveforms, LRSR (Fig.9f) and Wave_clus (Fig.9e) estimated 3 neurons in line with the previous works [27], [7]. FMM (Fig.9d)intends to estimate more neurons, causing the third and fourth mean waveforms to share a similar profile. In summary, LRSR exhibits excellent accuracy and stability on synthetic and real-world datasets.

VI. Conclusion and Future Works

In this paper, a joint low-rank and sparse representation (LRSR) model has been proposed for automatic spike sorting. LRSR efficiently recognizes noise corruption or similar spikes, and is robust to overlapped spikes. Extensive experimental results demonstrated the superiority of LRSR over the other four state-of-the-art methods. In the future, we aim to improve the computational speed of our approach further. Besides, our current work is limited to extracellular single-channel and vivo spike sorting processing. It is potential and desirable to extend LRSR with tensor optimization for signals recorded from the multi-channel, multi-electrode array, and nonlinear spatial arrays.

References

[1] Christophe Pouzat, Ofer Mazor, and Gilles Laurent. Using noise signature to optimize spike-sorting and to assess neuronal classification quality. *Journal of Neuroscience Methods*, 122(1):43–57, 2002.

[2] Robert Bestel, Andreas W Daus, and Christiane Thielemann. A novel automated spike sorting algorithm with adaptable feature extraction. *Journal of Neuroscience Methods*, 211(1):168–178, 2012.

[3] Marius Pachitariu, Nicholas Steinmetz, Shabnam Kadir, Matteo Carandini, and Harris Kenneth D. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *BioRxiv*, page 061481, 2016.

[4] Miguel AL Nicolelis and Mikhail A Lebedev. Principles of neural ensemble physiology underlying the operation of brain-machine interfaces. *Nature Reviews Neuroscience*, 10(7):530, 2009.

[5] Sarah Gibson, Jack W Judy, and Dejan Marković. Spike sorting: The first step in decoding the brain. *IEEE Signal Processing Magazine*, 29(1):124–143, 2011.

[6] David E Carlson, Joshua T Vogelstein, Qisong Wu, Wenzhao Lian, Mingyuan Zhou, Colin R Stoetzner, Daryl Kipke, Douglas Weber, David B Dunson, and Lawrence Carin. Multichannel electrophysiological spike sorting via joint dictionary learning and mixture modeling. *IEEE Transactions on Biomedical Engineering*, 61(1):41–54, 2013.

[7] Libo Huang, Lu Gan, and Bingo Wing-Kuen Ling. A unified optimization model of feature extraction and clustering for spike sorting. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:750–759, 2021.

[8] Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. Past, present and future of spike sorting techniques. *Brain Research Bulletin*, 119:106–117, 2015.

[9] Libo Huang, Bingo Wing-Kuen Ling, Ruichu Cai, Yan Zeng, Jiong He, and Yao Chen. Wmsorting: Wavelet packets decomposition and mutual information based spike sorting method. *IEEE Transactions on Nanobioscience*, 2019.

[10] Sara Mahallati, James C Bezdek, Milos R Popovic, and Taufik A Valiante. Cluster tendency assessment in neuronal spike data. *PloS one*, 14(11), 2019.

[11] Mehdi Shirzadi, Hamid R Marateb, Kevin C McGill, Silvia Muceli, Miguel A Mañanas, and Dario Farina. An accurate and real-time method for resolving superimposed action potentials in multiunit recordings. *IEEE Transactions on Biomedical Engineering*, 70(1):378–389, 2022.

[12] Charles M Gray, Pedro E Maldonado, Mathew Wilson, and Bruce McNaughton. Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. *Journal of Neuroscience Methods*, 63(1-2):43–54, 1995.

[13] Michael S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78, 1998.

[14] Zhi Yang, Qi Zhao, and Wentai Liu. Improving spike separation using waveform derivatives. *Journal of neural engineering*, 6(4):046006, 2009.

[15] Sarah Gibson, Jack W Judy, and Dejan Markovic. Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(5):469–478, 2010.

[16] Kenneth D Harris, Darrell A Henze, Jozsef Csicsvari, Hajime Hirase, and Gyorgy Buzsaki. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *Journal of Neurophysiology*, 84(1):401–414, 2000.

[17] Carlos Pedreira, Juan Martinez, Matias J Ison, and Rodrigo Quian Quiroga. How many neurons can we see with current spike sorting algorithms? *Journal of Neuroscience Methods*, 211(1):58–65, 2012.

[18] Maneesh Sahani. Latent variable models for neural data analysis. *California Institute of Technology*, 1999.

[19] Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995, 2008.

[20] R Quian Quiroga, Zoltan Nadasdy, and Yoram Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Computation*, 16(8):1661–1687, 2004.

[21] Fernando J Chaure, Hernan G Rey, and Rodrigo Quian Quiroga. A novel and fully automatic spike-sorting implementation with variable number of features. *Journal of Neurophysiology*, 120(4):1859–1871, 2018.

[22] Yao Chen, Libo Huang, Jiong He, Kunyao Zhao, Ruichu Cai, and Zhifeng Hao. Hass: High accuracy spike sorting with wavelet package decomposition and mutual information. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 831–838. IEEE, 2018.

[23] Dimitrios A Adamos, Efstratios K Kosmidis, and George Theophilidis. Performance evaluation of pca-based spike sorting algorithms. *Computer Methods and Programs in Biomedicine*, 91(3):232–244, 2008.

[24] Mohammad Reza Keshtkaran and Zhi Yang. Noise-robust unsupervised spike sorting based on discriminative subspace learning with outlier handling. *Journal of Neural Engineering*, 14(3):036003, 2017.

[25] Thanh Nguyen, Abbas Khosravi, Douglas Creighton, and Saeid Nahavandi. Spike sorting using locality preserving projection with gap statistics and landmark-based spectral clustering. *Journal of Neuroscience Methods*, 238:43–53, 2014.

[26] Jonathan W Pillow, Jonathon Shlens, EJ Chichilnisky, and Eero P Simoncelli. A model-based spike sorting algorithm for removing correlation artifacts in multi-neuron recordings. *PloS one*, 8(5):e62123, 2013.

[27] Chaitanya Ekanadham, Daniel Tranchina, and Eero P Simoncelli. A unified framework and method for automatic neural spike identification. *Journal of Neuroscience Methods*, 222:47–55, 2014.

[28] Carlson. Fmmspikesorter. `https://github.com/decarlson/FMMSpikeSorter`, April 2014. Accessed: 2019-07-29.

[29] Haifeng Wu, Kai Yang, and Yu Zeng. Sparse coding and compressive sensing for overlapping neural spike sorting. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(8):1516–1525, 2018.

[30] Chaitanya Ekanadham, Daniel Tranchina, and Eero P Simoncelli. Recovery of sparse translation-invariant signals with continuous basis pursuit. *IEEE Transactions on Signal Processing*, 59(10):4735–4744, 2011.

[31] Libo Huang, Bingo Wing-Kuen Ling, Yan Zeng, and Lu Gan. Spike sorting based on low-rank and sparse representation. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.

[32] Kai Yang. Sparse coding and compressive sensing for overlapping neural spike sorting. `https://github.com/yangkai12/SpikeSorting`, April 2018. Accessed: 2019-09-30.

[33] Pu-Ming Zhang, Jin-Yong Wu, Yi Zhou, Pei-Ji Liang, and Jing-Qi Yuan. Spike sorting based on automatic template reconstruction with a partial solution to the overlapping problem. *Journal of Neuroscience Methods*, 135(1-2):55–65, 2004.

[34] Maryam Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford University, 2002.
[35] Kewei Tang, Risheng Liu, Zhixun Su, and Jie Zhang. Structure-constrained low-rank representation. *IEEE Transactions on Neural Networks and Learning Systems*, 25(12):2167–2179, 2014.
[36] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2012.
[37] Guangming Shi, Zuozhi Liu, Xiaotian Wang, Chengyu T Li, and Xiaowei Gu. Object-dependent sparse representation for extracellular spike detection. *Neurocomputing*, 266:674–686, 2017.
[38] Desmond J Higham, Gabriela Kalna, and Milla Kibble. Spectral clustering and its use in bioinformatics. *Journal of Computational and Applied Mathematics*, 204(1):25–37, 2007.
[39] Adarsh Singh, Rabindra K Panda, and Niranjan Pramanik. Appropriate data normalization range for daily river flow forecasting using an artificial neural network. *IAHS Publication*, 331:51, 2009.
[40] Rubén Armañanzas and Giorgio A Ascoli. Towards the automatic classification of neurons. *Trends in neurosciences*, 38(5):307–318, 2015.
[41] Zhouchen Lin, Minming Chen, Leqin Wu, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *Coordinated Science Laboratory Report No. UILU-ENG-09-2215, DC-247*, 2010.
[42] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.
[43] Junsik Eom, In Yong Park, Sewon Kim, Hanbyol Jang, Sanggeon Park, Yeowool Huh, and Dosik Hwang. Deep-learned spike representations and sorting via an ensemble of auto-encoders. *Neural Networks*, 134:131–142, 2021.
[44] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.
[45] Eugen-Richard Ardelean, Andreea Coporîie, Ana-Maria Ichim, Mihaela Dînșoreanu, and Raul Cristian Mureșan. A study of autoencoders as a feature extraction technique for spike sorting. *Plos One*, 18(3):e0282810, 2023.
[46] Laszlo Schäffer, Zoltan Nagy, Zoltan Kincses, Richard Fiáth, and Istvan Ulbert. Spatial information based osort for real-time spike sorting using fpga. *IEEE Transactions on Biomedical Engineering*, 68(1):99–108, 2020.
[47] Jiri Wild, Zoltan Prekopcsak, Tomas Sieger, Daniel Novak, and Robert Jech. Performance comparison of extracellular spike sorting algorithms for single-channel recordings. *Journal of Neuroscience Methods*, 203(2):369–376, 2012.
[48] Cyrille Rossant, Shabnam N Kadir, Dan FM Goodman, John Schulman, Maximilian LD Hunter, Aman B Saleem, Andres Grosmark, Mariano Belluscio, George H Denfield, Alexander S Ecker, et al. Spike sorting for large, dense electrode arrays. *Nature Neuroscience*, 19(4):634, 2016.