PAPER

Computationally inexpensive enhanced growing neural gas algorithm for real-time adaptive neural spike clustering

To cite this article: Zeinab Mohammadi *et al* 2019 *J. Neural Eng.* **16** 056007

View the article online for updates and enhancements.

Recent citations

- Low-latency single channel real-time neural spike sorting system based on template matching
  Pan Ke Wang *et al*

J. Neural Eng. **16** (2019) 056007 (18pp) https://doi.org/10.1088/1741-2552/ab208c

Computationally inexpensive enhanced growing neural gas algorithm for real-time adaptive neural spike clustering

Zeinab Mohammadi[1], John M Kincaid[1], Sio Hang Pun[2], Achim Klug[3], Chao Liu[1] and Tim C Lei[1,4]

[1] Department of Electrical Engineering, University of Colorado, Denver, CO 80204, United States of America
[2] State Key Laboratory of Analog and Mixed-Signal VLSI, University of Macau, People's Republic of China
[3] Department of Physiology and Biophysics, University of Colorado Anschutz Medical Campus, Aurora, CO 80045, United States of America

E-mail: tim.lei@ucdenver.edu

Received 26 October 2018, revised 17 April 2019
Accepted for publication 9 May 2019
Published 30 July 2019

Abstract

Objective. Real-time closed-loop neural feedback control requires the analysis of action potential traces within several milliseconds after they have been recorded from the brain. The current generation of spike clustering algorithms were mostly designed for off-line use and also require a significant amount of computational resources. A new spike clustering algorithm, termed 'enhanced growing neural gas (EGNG)', was therefore developed that is computationally lightweight and memory conserving. The EGNG algorithm can adapt to changes of the electrophysiological recording environment and can classify both pre-recorded and streaming action potentials. *Approach.* The algorithm only uses a small number of EGNG nodes and edges to learn the neural spike distributions which eliminates the need of retaining the neural data in the system memory to conserve computational resources. Most of the computations revolve around calculating Euclidian distances, which is computationally inexpensive and can be implemented in parallel using digital circuit technology. *Main results.* EGNG was evaluated off-line using both synthetic and pre-recorded neural spikes. Streaming synthetic neural spikes were also used to evaluate the ability of EGNG to classify action potentials in real-time. The algorithm was also implemented in hardware with a Field Programming Gate Array (FPGA) chip, and the worst-case clustering latency was 3.10 μs, allowing a minimum of 322 580 neural spikes to be clustered per second. *Significance.* The EGNG algorithm provides a viable solution to classification of neural spikes in real-time and can be implemented with limited computational resources as a front-end spike clustering unit for future tethered-free and miniaturized closed-loop neural feedback systems.

Keywords: spike sorting, electrophysiology, neural feedback control, *in vivo*, mutli-unit recording, single unit recording

S Supplementary material for this article is available online

(Some figures may appear in colour only in the online journal)

[4] Author to whom any correspondence should be addressed.

1. Introduction

In the pursuit of realizing miniaturized and portable systems for closed-loop neural control, one of the major challenges is to develop robust data analysis algorithms for analyzing neural signals, which are computationally inexpensive, use less memory, and are energy efficient, [1–5]. During *in vivo* recordings, which are often combined with other techniques (such as behavioral testing), action potentials are commonly recorded extracellularly from a biological subject with one or more electrodes [6–10]. Under this extracellular configuration, the electrode measures a time series of voltage signals containing extracellular action potentials originating from neighboring neurons. Ionic currents from a neuron that reach the recording electrode are highly dependent on the relative position between the electrode and the neuron, as well as the impedance of the extracellular fluid. Therefore, whenever several nearby neurons contribute to this voltage recorded by the electrode, these contributions differ in their temporal shapes [7]. These temporal shape differences are then used by spike sorting algorithms to classify neural spikes to the respective neurons.

For the past decade, several robust spike clustering algorithms were developed to classify pre-recorded neural activity in an off-line manner [11–17]. However, there is a growing need in neuroscience to perform spike clustering in real time in order to facilitate more sophisticated experimental testing but more importantly, to intervene and control neural activity in a closed loop manner based on brain state [4, 18–20]. Particularly with the advance of optogenetics, neurons can now be precisely stimulated or inhibited with proper combinations of optogenetic proteins, optical illumination wavelengths, and specific promotors, opening up new opportunities to conduct neuroscience experiments in a closed-loop manner [21–23]. However, in order to successfully modulate a neural circuit, the analysis and processing time can be no longer than several milliseconds, creating a significant technical challenge [4, 20]. In addition, most spike clustering algorithms were not designed to classify streaming neural spikes in real-time; thus developing new spike clustering algorithms that can classify streaming neural data in real-time accurately, rapidly, and adaptively is an important step for closed-loop neural controls.

There has been a sustained effort to develop better neural spike clustering algorithms. Reviews of various off-line spike clustering techniques can be found in [12, 24]. Generally speaking, spike clustering refers to associating each cluster with their respective firing neurons. The so-called 'hard clustering' methods assume the clusters are Gaussian distributed and the overall neural distribution is a sum of the Gaussian clusters in which k-means and its variations fall into this category. Other 'soft-clustering' techniques, including expectation maximization (EM), allow overlapping between these clusters to approximate the cluster distribution, within a given number of clusters [25]. Super-paramagnetic cluster (SPC) is another approach which borrows the physical concept of magnetic thermal interaction and models neural spikes as magnetic spin elements. At the transition temperature between paramagnetic and ferromagnetic states, the neural distribution fractures into different clusters and this concept is used for spike classification [26]. In addition, recent spike clustering developments have been focused on classifying massive amounts of neural data recorded from multielectrode arrays (MEAs). These algorithms require powerful computers with multiple central and graphical processors to classify the large amount of data collected off-line. For example, Masked-EM was recently applied to handle large amounts of neural spikes recorded from a MEA by introducing a geometric mask to reduce the data dimensionality [13]. ISO-SPLIT is another automatic spike clustering algorithm that projects the high-dimensional data onto the principal axes and finds the cluster boundaries by fitting to these projections [15]. ISO-SPLIT also eliminates the need for human intervention to allow a higher degree of automation. While these techniques are powerful, they are not designed to classify streaming neural spikes in real-time. In addition, these techniques require the retention of all the data in the system memory to compute the data recursively, making these techniques unsuitable for spike clustering in real-time with limited computing resources. On the hardware side, there were some recent exciting advancements. Q-sort was the earliest spike clustering algorithm designed for FPGA hardware but the algorithm is a simple classifier that does not take advantage of computational parallelism, nor does it have the capability to adapt to changes of the electrophysiological environment [17, 27]. Template matching is another recent approach in which spike cluster templates were learned during a training period and subsequent spikes are matched to these cluster templates for rapid sorting. Luan *et al* developed a 32-channel compact FPGA system utilizing template matching for real-time rapid spike sorting and tested the system to record the M1 region from a non-human primate; however, the system required a computer tethered to the FPGA to calculate the cluster templates during the training period [28]. Park *et al* developed a 128 channel FPGA system which can perform the template learning within the FPGA but their template generation algorithm lacks the ability to adapt to changes in spike shape during long measurement periods [29].

In this paper, our goal was to develop a neural spike clustering algorithm—enhanced growing neural gas (EGNG)—that can classify streaming neural spikes in real-time and provide immediate classified information for downstream neural decoding. The algorithm is also compatible with the design methodology and the parallel computation capability of modern digital integrated circuit technologies to allow hardware implementation in a field programmable gate array (FPGA) or an application-specific integrated circuit (ASIC) [21, 30]. The advantage of using this digital logic approach over general purpose processors (CPU) is that dedicated hardware logic can be built to perform parallel computations. In addition, EGNG can self-adapt to neurophysiological changes in real-time, such as new neural spike shapes appearing, to allow the clustering to be continually corrected.

During neural recordings, the electrode may move slightly within the brain due to physical movements or inflammatory responses, especially during long-term recordings. Thus, the temporal shapes of the neural spikes can change over time. In addition, neural spikes with new temporal shapes may be picked up by the electrode as it moves closer to some neurons that had no contribution to the recording waveform previously. Alternatively, a cluster may become obsolete when the electrode drifts too far away from a neuron. Therefore, clustering algorithms have to be able to adapt in order to handle these changes properly and EGNG was especially designed to have the capability to adapt to these changes. In addition, it has been suggested that the assumption that neural clusters have to be Gaussian distributed should be lifted in order to allow for handling of non-conventional noises [31, 32], and EGNG does not require the neural clusters to be Gaussian distributed.

In this paper, we present the EGNG algorithm which is specifically designed to perform neural spike clustering for both off-line and real-time situations, as well as the implementation of the EGNG algorithm in hardware using an FPGA to evaluate its clustering capability for real-time action potentials. The EGNG algorithm is computationally inexpensive with minimal requirements for processing power and system memory. The EGNG algorithm is designed to fit the needs for both off-line and real-time neural spike clustering, and can be used as a front-end spike clustering algorithm for downstream data analysis in closed-loop neural controls.

2. The EGNG algorithm and evaluation data

The EGNG algorithm has improved upon the original growing neural gas (GNG) algorithm developed by Bernd Fritzke [31, 33]. The original GNG algorithm is an unsupervised clustering algorithm which can be used to differentiate unconnected cluster groups in the vector space and therefore, the original GNG algorithm should in principle be able to classify spike clusters. However, the original GNG algorithm is an off-line algorithm which was not designed to handle streaming neural data in real-time. In addition, the original GNG algorithm does not handle terminal criteria well enough to properly separate closely spaced cluster groups for off-line pre-recorded neural data. Therefore, we modified and enhanced the original algorithm in several areas to allow the algorithm to classify both pre-recorded and streaming neural spikes. In addition, with these enhancements, the algorithm can automatically adapt to changes of the electrophysiology environment to correctly classify neural spikes. Compared to other simpler spike sorting algorithms, such as *k*-means [34], EGNG does not require neural clusters to be Gaussian-distributed, and can automatically determine the total number of clusters without user input. The algorithm can also distinguish neural spikes and reject noise outliers. In this section, the working principle of the EGNG algorithm and how the algorithm can be used to classify neural spikes for both off-line pre-recorded and real-time streaming neural spikes are discussed.

2.1. The EGNG algorithm

Similar to other spike sorting algorithms, neural spikes measured in the time domain are first transformed onto a multi-dimensional vector space using techniques such as principal component analysis (PCA) or wavelet transformation [12]. In this manner, neural spikes are transformed into neural points in a multi-dimensional vector space constructed by a chosen set of basis functions. EGNG does not handle the pre-processing steps in spike sorting, including spike detection, isolation, and transformation, but focuses on classifying the neural clusters once they have been properly processed and transformed using conventional methods. Here the term 'neural point' is defined as the feature vector in the vector space transformed from a neural spike in the time domain. Neural points with similar temporal spike profiles are consequentially clustered together into a group in the vector space. Therefore, the principle of the EGNG algorithm is to cover these neighboring neural points with EGNG nodes and interconnect them with EGNG edges to form EGNG clusters. In this way, EGNG can be considered as a process of forming clusters using EGNG nodes and edges on top of the distribution of neural data, and to constantly adapt and follow changes of the data distribution in the vector space.

2.1.1. Off-line EGNG to classify pre-recorded neural spikes. Figure 1 illustrates the EGNG algorithm configured to classify neural spikes off-line. In this off-line clustering configuration, neural spikes are pre-recorded and classified after the experiment has concluded. Consequentially, the EGNG algorithm does not need to be limited in terms of computing resources. Both computing power and memory are considered abundant such that all the neural data points can be stored in the system memory for recursive processing, in contrast to the case of classifying streaming neural data in real-time (see below). The parameters for the EGNG algorithm and the values used in this paper for classification are listed in table 1. Generally, only the maximum node count (N_{EGNG}) and the edge pruning threshold (a_{th}) are sensitive to the classification results. More discussion on the value selections of the parameters can be found in the supplementary information (stacks.iop.org/JNE/16/056007/mmedia).

The steps for off-line EGNG to classify pre-recorded neural spikes are illustrated in figure 1. Note that steps (1) to (5) were developed in the original GNG algorithm and step (6) is the new enhancement for better termination criteria to separate closely-packed neural clusters.

(1) Initialization: Several EGNG nodes connected by EGNG edges are randomly generated and placed among the neural data in the vector space. The locations of the EGNG nodes are denoted as w_j and the number of EGNG nodes created is denoted as N.

(2) Finding the closest EGNG node: A neural point x_i is randomly selected among the entire neural data distribution. The Euclidian distances ($d_j = \|x_i - w_j\|$) between the selected neural data point (x_i) and all the EGNG nodes (w_j) are then calculated. The EGNG nodes with the shortest and second shortest Euclidean distances are denoted as S_1 and S_2.

$$S_1 = argmin_{w_j \in N}(d_j)$$

$$S_2 = argmin_{w_j \in N \setminus \{S_1\}}(d_j)$$

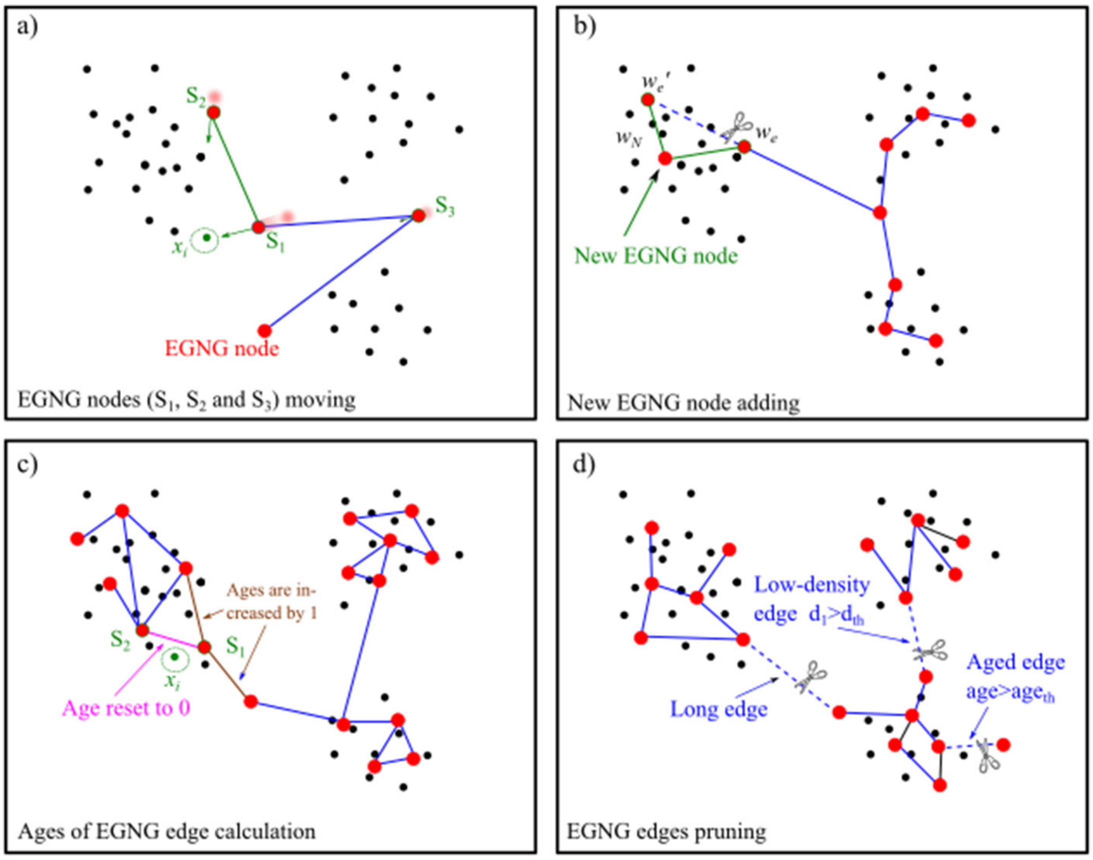


Figure 1. Illustration of the EGNG algorithm for sorting pre-recorded neural spikes off-line. (a) (Steps 1–3) At the beginning of the algorithm, a few EGNG nodes were generated and connected by EGNG edges. The closest EGNG node (S_1) and its neighboring nodes moved closer to a neural point (x_i) with different movement rates. (b) (Step 4) If the total number of EGNG nodes is less than the maximum node count (N_{EGNG}), a new EGNG node (S_n) is inserted between the node with the largest insertion value ($insert_{s_e}$) S_e and its closest neighboring EGNG node ($S_{e'}$). (c) (Step 5) As the EGNG tree grows larger covering the entire neural point distribution, EGNG edges connecting different neural clusters need to be pruned. For a neural point (x_i), the age of the edge between the closest (s_1) and the second closest (s_2) EGNG nodes ($a_{s_1,\ s_2}$) to is reset to zero, but the age of other neighboring edges to s_1 ($a_{s_1,\ s_j}$) are increased by 1. (d) (Step 5) When the age of an edge reaches the age threshold ($a > a_{th}$), the aged edge is pruned. Edges which are longer than the average edge length are considered long edges and will be pruned to help separate clusters. In addition, an edge is considered as a low-density edge if the average distance of the center of the edge to the five closest EGNG nodes is longer than the average.

Table 1. Parameters of the EGNG algorithm. The values used in the manuscript and the typical ranges of the parameters are listed.

Parameters	Values used in this manuscript	Typical ranges
Maximum node count (N_{EGNG})	10–15 for most dataset and 100 for ring dataset	3 to max (depends on the data)
Edge pruning threshold (a_{th})	4–8 for most dataset and 30 for ring dataset	A fraction of total node count (e.g. half or one-third).
Number of initial nodes (N)	2	2
Moving rate of S_1 (e_{s_1})	0.1	(0, 1)
Moving rates of S_j (e_{nbr})	0.006	(0, 1), $e_{s_1} > e_{nbr}$
Insert parameter reduce rate of w_e and w'_e nodes (α)	0.5	(0, 1)
Insert parameter reduce rate of other nodes (β)	0.01	(0, 1)
Number of iterations before inserting a new node (λ)	10	5–50

If the two closest EGNG nodes (S_1 and S_2) have not been connected by an EGNG edge, a new EGNG edge is constructed to connect them.

(3) Moving the closest EGNG node and its neighboring nodes closer to the neural point $\boldsymbol{x_i}$: A neighboring node is another EGNG node that connects to an EGNG node directly by an EGNG edge, without another EGNG node crossed in between. In this step, S_j are denoted as all the neighboring nodes to the closest EGNG node (S_1). S_1 and all S_j are moved closer towards the neural data point (x_i), but with different moving rates.

$$S_1 \leftarrow S_1 + e_{s_1}(x_i - S_1)$$

$$S_j \leftarrow S_j + e_{nbr}(x_i - S_j)$$

where $e_{s_1}, e_{nbr} \in (0, 1)$ are the moving rates of S_1 and S_j, where $e_{s_1} > e_{nbr}$.

An 'Insertion' parameter ($insert_{w_i}$) is used to determine where a new EGNG node should be inserted. After S_1 has moved toward x_i, $insert_{s_1}$ is increased by

$$insert_{s_1} \leftarrow insert_{s_1} + \|x_i - s_1\|^2.$$

(4) Insertion of a new EGNG node using the insertion parameter: If the number of total EGNG nodes is less than the maximum node count (N_{EGNG}), a new EGNG node is added after λ of iteration. The EGNG node with the highest insertion and one of its neighbors with the highest insertion among all the neighboring nodes will be identified, and the positions of the two EGNG nodes are denoted as w_e and w_e'. The position of the new EGNG node (w_N) is equal to

$$w_N = \frac{1}{2}(w_e + w_{e'}).$$

EGNG will prune the EGNG edge connecting w_e and w_e' and will create two new EGNG edges connecting between w_e and w_N and between $w_{e'}$ and w_N. After the insertion, the insertion parameters of the two nodes are reduced to avoid repetitive EGNG node creation,

$$insert_{w_e} \leftarrow \alpha \cdot insert_{w_e}$$

$$insert_{w_{e'}} \leftarrow \alpha \cdot insert_{w_e'}$$

$$insert_{w_N} \leftarrow insert_{w_e}.$$

In addition, the insertion parameters of all the other EGNG nodes are reduced by β, where $\alpha, \beta \in (0, 1)$ are the insertion reduction rates.

(5) Edge pruning with the age parameter: If the EGNG edge is spanning over an area with few neural points, the EGNG edge will be removed to allow cluster separation. In order to allow pruning, an 'age' parameter (a_k) is assigned to all EGNG edges to determine if an EGNG edge should be pruned. For the EGNG edge connecting (S_1 and S_2), the age (a_{S_1,S_2}) of this edge is reset to zero, indicating this edge is close to the neural point (x_i) and the EGNG edge should be retained.

$$a_{S_1,S_2} \leftarrow 0.$$

However, for other neighboring EGNG edges connecting S_j to S_1 , the ages (a_{s_1,S_j}) of these edges are increased by 1, indicating that these edges are farther away from the neural point (x_i), leading to a higher chance of being eventually pruned.

$$a_{s_1,S_j} \leftarrow a_{s_1,S_j} + 1.$$

With this strategy, the ages of remote EGNG edges will be steadily increased. When the age of an edge is higher than the edge pruning threshold (a_{th} ($a_{S_i,S_j} > a_{th}$)), the EGNG edge is pruned to allow cluster separation.

(6) Termination criteria: After all the neural points have been processed and if the number of EGNG nodes (N) created is less than the maximum node count (N_{EGNG}), the neural points will be processed one more time to allow all the EGNG nodes to be generated for better clustering. After all the EGNG nodes are generated, the EGNG clustering will be terminated. This termination criteria generally results in well-separated EGNG clusters if the neural clusters are far from one another and the neural points are densely clustered in the cluster centers. However, if the neural clusters are close to one another and the cluster centers are not densely packed, this termination criteria is not strong enough to ensure sufficient separation between neural clusters. For this reason, two strategies—removing long edges and removing low-density edges—are used to allow the EGNG algorithm to better separate the neural clusters even if they are loosely packed and close to one another. Closely-packed and loose neural data distributions are common for actual electrophysiology recordings in animals.

(a) Removing long edges: The length of an EGNG edge is $l_{w_i,w_j} = \|w_j - w_i\|$. The EGNG edge is removed if its length (l_{w_i,w_j}) is larger than the average length of all the EGNG edges (l_{ave}), that is $l_{w_i,w_j} > l_{ave}$.

(b) Removing low-density edges: The average Euclidian distance of five neural points closest to the middle point of an EGNG edge is calculated as l_{mid}, and $\overline{l_{mid}}$ is the average l_{mid} of all the EGNG edges. An EGNG edge is considered as a low-density edge and needed to be removed if $l_{mid} > l_{mid}$.

2.1.2. On-line EGNG to classify streaming neural spikes. The EGNG algorithm for on-line sorting is designed to classify streaming neural spikes rapidly and accurately with limited processing power and memory. The limited use of computational power and memory allow for hardware miniaturization to fit all the necessary circuits into a small FPGA or ASIC to avoid cable tethering to a main computer. Another benefit

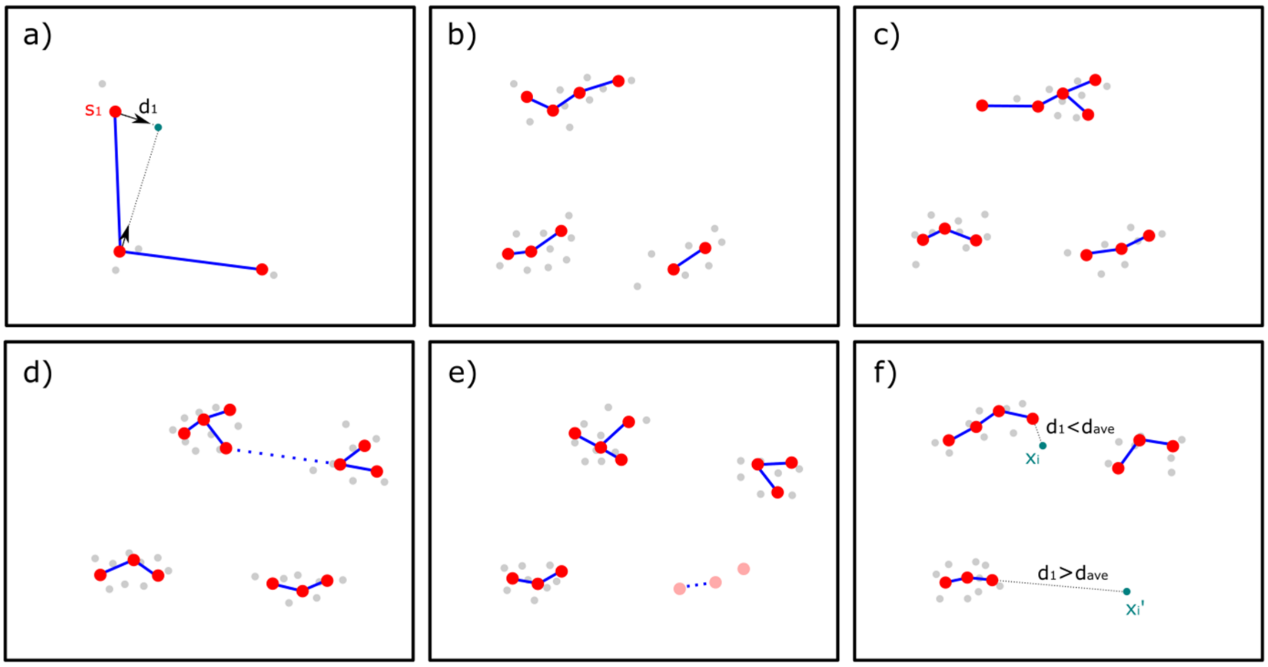


Figure 2. Illustration of the EGNG algorithm sorting streaming neural spikes in real-time. (a) At the beginning of the sorting, very few neural points are streamed in and only a few EGNG nodes are created. The closest EGNG node (s_1) and its neighing nodes are moved towards the incoming neural point (blue). (b) The methods to add an EGNG node and to prune an EGNG edge are the same as in off-line mode. A stable EGNG cluster distribution will emerge after enough neural points are processed, allowing correct classification for subsequent incoming neural spikes. (c) The EGNG cluster can move towards a new position if the temporal shape of the neural spikes changes for continuous correct classification. (d) A new EGNG cluster can be formed if a steady stream of neural points appears in a more distant location by the regular EGNG edge pruning. (e) An existing EGNG cluster can be eliminated if no neural points are coming in for the cluster group for an extended period of time. (f) A neural point (x_i) having a Euclidean distance to the shortest EGNG node (d_1) shorter than the average Euclidean distance of all EGNG points ($d_1 < d_{ave}$) is immediately classified as belonging to the EGNG group with the closest EGNG node. In contrast, a neural point (x_i') having a Euclidean distance to the closest EGNG node (d_1) longer than the average Euclidean distance of all EGNG points ($d_1 > d_{ave}$) is classified as noise.

of rapid spike sorting is that it may enable immediate neural decoding such that the decoded information can be used to drive closed-loop neural feedback control, such as optogenetic stimulation or inhibition. Unlike off-line EGNG where all the neural points are stored in the memory and processed recursively, real-time EGNG does not retain the neural points in the memory once they are processed. Instead, EGNG learns the cluster distribution from the neural points and only stores the learned distribution into the EGNG node positions and edge connections. This approach is similar to Bayesian inference in which the streaming of neural points serve as priors, and the EGNG clusters will approximate the neural data distribution as more and more spikes are streamed in and processed, resulting in more accurate classifications for subsequent neural spikes [35]. Figure 2 illustrates the process of on-line EGNG, as well as showing how the algorithm can adapt to changes of the neural data distribution dynamically.

(1) Update of the EGNG nodes based on the incoming neural spike: Unlike off-line clustering where all neural points are available, in real-time clustering, neural spikes are recorded and streamed into the algorithm individually and sequentially. Despite this acquisition difference, the basic EGNG algorithm of creating, deleting, and moving EGNG nodes and edges is the same in both modes. However, in real-time sorting, steps 1 to 6 of the off-line mode will be used continuously to build up the EGNG clusters, as shown in figures 2(a) and (b). The main difference is that instead of selecting a neural point x_i from the pre-recorded data pool, the neural point x_i is now the most recently streamed-in neural point.

(2) Immediate classification of the incoming neural spike: Immediate classification of the neural spikes is particularly important to allow downstream data analysis to estimate brain state and respond with proper interventions. Therefore, the classification of the incoming data point is performed immediately by determining which EGNG cluster is closest to the neural point. To realize this, EGNG calculates the Euclidean distances between the new neural point x_i and all the EGNG nodes w_j,

$$d_j = \|x_i - w_j\|, j = 1 \ldots N.$$

The EGNG node with the shortest distance to the new neural point x_i is identified and the classification is based on identifying the EGNG cluster containing this EGNG node.

(3) Movement of EGNG clusters: If the temporal shape of a neural cluster has changed over time due to probe movements or other electrophysiological reasons, the closest EGNG cluster will move towards the new position, as shown in figure 2(c). This movement is facilitated by the normal EGNG node movement towards the most recent neural points. This adaptability allows continued correct classification of the neural data, even if the temporal shape of the neural spikes has changed, which is particularly important for long-term recording.

(4) Deletion of obsolete EGNG clusters: Similarly, if neural spikes from a particular neuron are no longer recorded by the electrode, the corresponding EGNG cluster will be removed automatically. For the EGNG cluster no longer receives new incoming neural spikes, EGNG nodes and edges are deleted once the age parameters are matured. This removal process results in the complete deletion of the obsolete EGNG cluster, as shown in figure 2(e).

(5) Creation of new EGNG cluster and rejection of noisy signals: Neural points can appear far away from existing EGNG clusters for two reasons. The first is that these neural points belong to a new cluster which the electrode was not able to pick up previously. The second reason is that these neural points are actually noise signals which exceed the spike threshold and thereby enter the clustering algorithm inappropriately. In order to differentiate between these two conditions, the shortest Euclidian distance between the neural point and the EGNG nodes is calculated, and if this distance is larger than the outlier threshold d_{out}, this neural point is considered as an outlier. The EGNG algorithm will further calculate the running average position x_{ave} of these outliers with

$$x_{ave} \leftarrow \frac{x_i}{N_{out}} + \frac{N_{out} - 1}{N_{out}} x_{ave}$$

where N_{out} is the number of recent outliers for the average. For the subsequent streaming neural points, if the Euclidian distance between the neural point and the running average x_{ave} is larger than the noise rejection threshold d_n, the neural point is categorized as noise and discarded from the algorithm. However, if the Euclidian distance of the outlier is smaller than d_n, the outlier is considered as a neural point that belongs to a new neuronal recording, and the location of the neural point will be retained in the system. In addition, if the total count of these retained neural points has reached the new cluster creation threshold N_c, a new EGNG cluster will be created around these retained neural points. Once the new EGNG cluster has been created, these few retained neural points will be discarded to save system memory. Neural points belonging to this new cell subsequently streamed into the algorithm will be correctly categorized due to the creation of the new EGNG cluster.

2.2. Simulated and pre-recorded electrophysiology neural data for evaluating EGNG

Both simulated and pre-recorded electrophysiology data were used to evaluate the performance and accuracy of the EGNG algorithm for both the off-line and the real-time modes, as well as the FPGA hardware implementation of EGNG. The same data were also used to compare EGNG to four other existing off-line spike sorting algorithms for classification accuracies and processing times.

2.2.1. Simulated neural data. The off-line EGNG algorithm was first evaluated with a simulated data set containing five Gaussian clusters, each with 200, 150, 150, 100 and 100 neural points, respectively. Three of these Gaussian clusters are circularly distributed and the remaining two are elliptically distributed [36]. Another set of simulated data distributed as three concentric rings were used to test whether the EGNG algorithm can handle non-Gaussian data distributions. The three concentric rings have radii of 1.0, 2.0 and 3.0 containing 4800, 3200 and 1600 data points with a Gaussian standard derivation of 0.14 [16]. The noisy moon data were also used to compare the EGNG algorithm to four other clustering algorithms (see below) and the data set was generated using the make_moons function in the scikit-learn library with a total of 1500 sample data points [37]. In order to test the on-line clustering capability of the EGNG algorithm implemented in both software and hardware, another set of simulated data obtained from [26] was used as streaming neural data. The data set contains 1440 000 data points and only the first 200 000 data points were used for the elevation. The data set were also used to evaluate the EGNG hardware implementation with an FPGA chip and the hardware clustering result was compared to that generated by the software implementation.

2.2.2. Pre-recorded neural data. The algorithm was also tested with actual neural data. All experimental procedures complied with all applicable laws and National Institutes of Health guidelines and were approved by the University of Colorado Institutional Animal Care and Use Committee (IACUC). The details of the experimental setup and the recording procedure were discussed in our previous publication [21] and are not repeated here in detail. In brief, the extracellular voltage trace was recorded with a high-impedance Tungsten metal electrode (WEPT33.0B10, MicroProbes, Gaithersburg, MD, USA). The electrode was placed into the fiber tract of the fifth (trigeminal) nerve within the brain stem of an anesthetized Mongolian gerbil (Meriones unguiculatus), and the neural spikes recorded had an average signal-to-noise ratio (SNR) of 2.7. This relatively low SNR was used intentionally to test the clustering capabilities of the algorithm. Individual neural spikes were isolated from the recorded trace through setting a threshold two times above the noise average and subsequently converted to feature vectors using PCA before sending to the EGNG algorithm for clustering.

2.2.3. Publically available neural data sets. Two publically available neural data sets were used to further evaluate the classification capability of the off-line EGNG algorithm. The first data set (hc-1, animal: 166, file: d16613.001.dat) was recorded from the CA1 hippocampal region and was developed as a benchmark for spike detection and sorting [38, 39]. The neural spikes were isolated by a threshold of two times above the noise level, and were converted to neural points in the vector space using PCA, followed by EGNG classification. The second data set (NCHybrid136) of [40] were constructed from two neural datasets on a 32-channel probe in the rat cortex. Since the second data set contains 32 channels, channel 3 was randomly chosen for the analysis. The

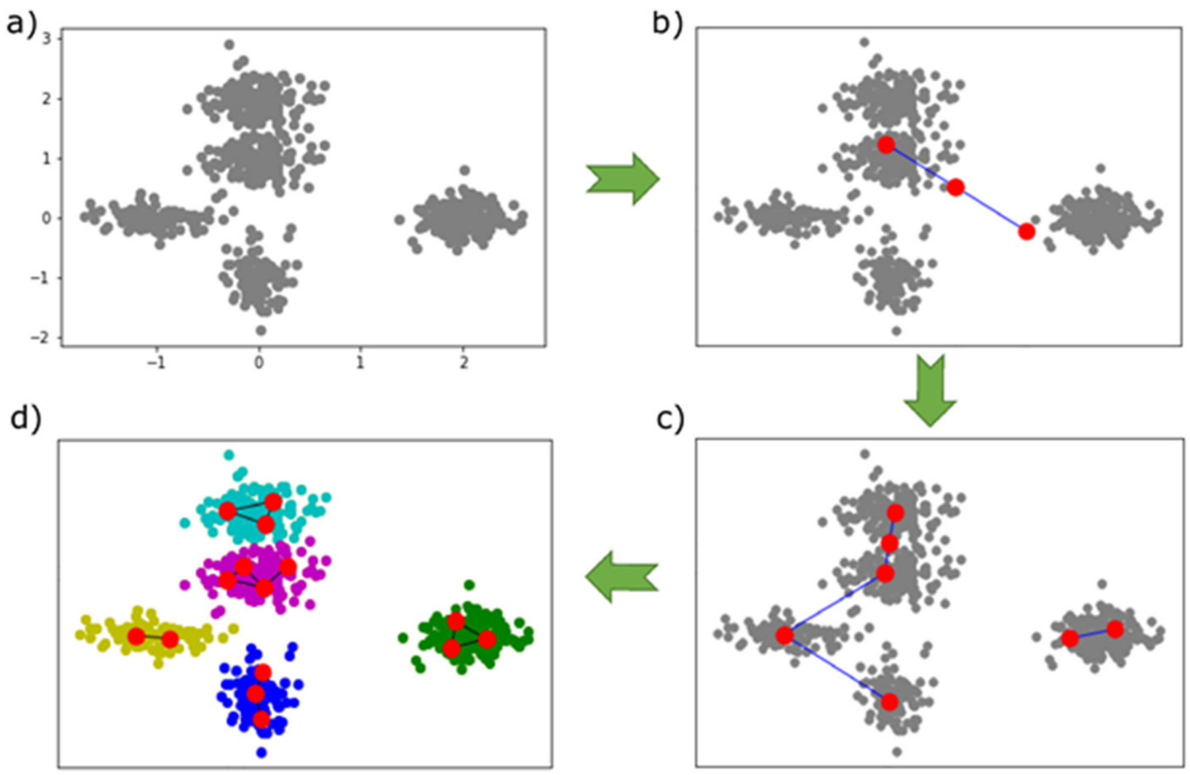


Figure 3. EGNG algorithm to sort neural spikes in off-line mode. (a) Artificially generated neural points (grey dots) simulating neural spikes generated from five different nearby neurons. (b) Three EGNG nodes (red) were interconnected by two EGNG edges (blue line) at the beginning of the algorithm. (c) More EGNG nodes were generated to distribute across the entire distribution of neural data. Edge pruning has also occurred to separate an EGNG cluster away from the main EGNG tree. (d) At the end of the algorithm, five EGNG clusters were formed and the neural points were classified by identifying the nearest EGNG node (five colors representing five separated cluster groups).

recording was initially processed by a 5th-order Butterworth high-pass filter with a cut-off frequency of 200 Hz to remove the low-frequency local field potential component, followed by a threshold of two times above the noise average to isolate the neural spikes. A total of 1902 neural spikes were isolated from the first 500 000 data points of the recording and was converted to the vector space into neural points using PCA for EGNG classification.

2.3. Comparing EGNG to other off-line clustering algorithms

The EGNG algorithm in off-line mode was compared to four other off-line clustering techniques—Density-Based Spatial clustering of Applications with Noise (DBSCAN), *K*-means, EM, and SPC—to evaluate the performance of these algorithms compared to EGNG. DBSCAN considers data points with close Euclidian distances as neighbors and assigns these neighbors as a group if the local density is higher than a threshold. Otherwise, the neighbors are considered as outliers [41]. *K*-mean assumes neural clusters are Gaussian distributed with a known total number of clusters, and moves the centers of these Gaussian clusters towards the centers of the neural points in order to best cover them [34]. EM is one of the so-called 'soft clustering methods' which applies two steps—the expectation (E) step and the maximization (M) step—iteratively to maximize the posteriori (MAP) or maximum likelihood (ML) estimates for the Gaussian clusters to best cover the neural distribution [42]. Custom python code was written for DBSCAN, *K*-means and EM using the built-in functions of the scikit-learn library [37] and the software package Wave_Clus was used for SPC classification [26, 43]. The aforementioned simulated data sets were fed into the algorithms for result comparisons with off-line EGNG.

3. Results

The spike sorting capabilities of the EGNG algorithm for both off-line and on-line modes are demonstrated and evaluated in this section. For the off-line mode, the evaluations were focused on demonstrating the sorting accuracy and the ability to handle non-Gaussian distributed clusters. Simulated and actual pre-recorded neural data were used for the evaluations and the EGNG algorithm was also compared to four other off-line sorting algorithms. For the on-line mode, the ability of the EGNG algorithm to rapidly classify streaming neural data was demonstrated in both software and hardware implementations. In addition, the flexibility of the EGNG algorithm to correctly and continually classify neural spikes, even in cases where the distributions are changing, was demonstrated in software.

3.1. EGNG algorithm to classify off-line simulated neural data

Figure 3 demonstrates the EGNG algorithm in off-line mode when classifying simulated neural data (grey dots) belonging

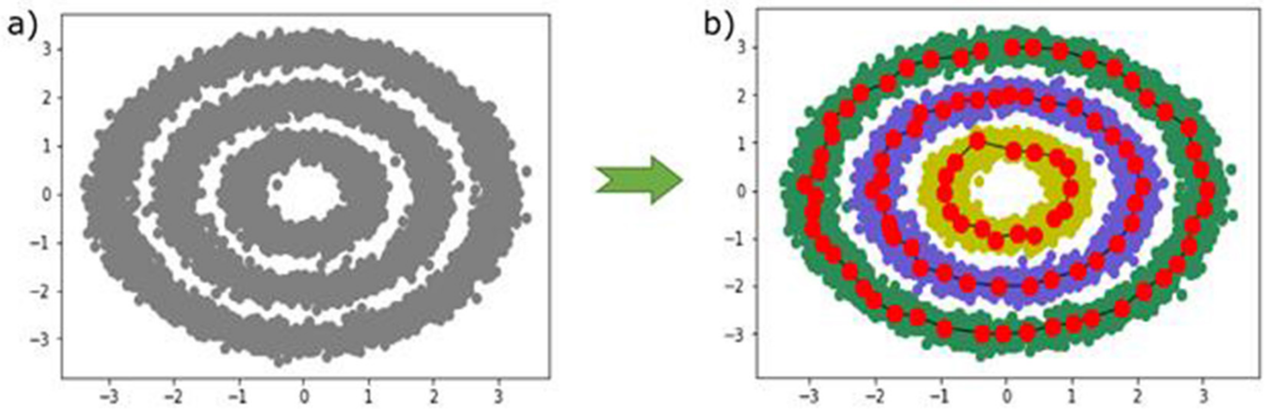


Figure 4. EGNG algorithm to sort non-Gaussian distributed neural points. (a) Three concentrically distributed neural groups were generated with radii of 3.0, 2.0, and 1.0 with a Gaussian noise standard deviation of 0.14. (b) The EGNG algorithm separated the concentric rings and correctly classified the neural spikes.

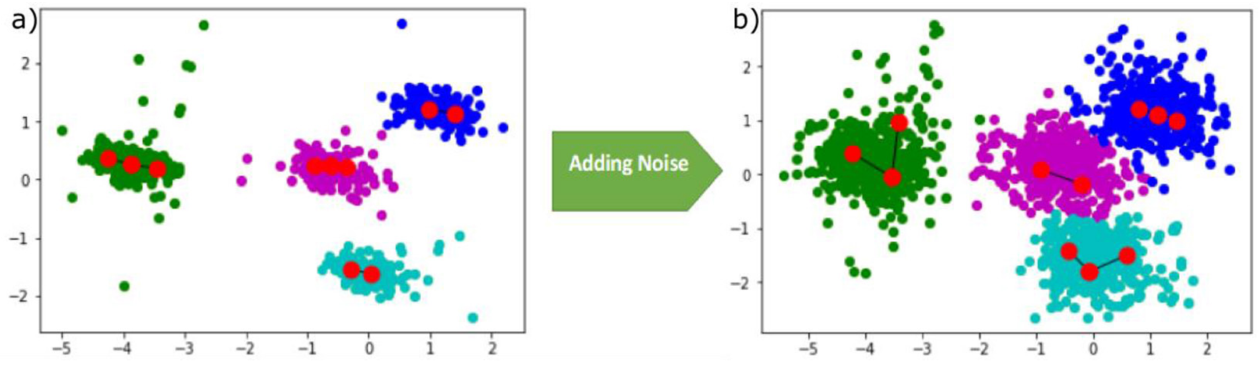


Figure 5. The EGNG algorithm can correctly classify closed-packed neural clusters. (a) Loosely spaced neural groups were successfully separately by the EGNG algorithm. (b) Additional normally distributed noise was added to the four neural groups to allow the groups to stay much closer together to a degree where the two top left groups were about to merge together. Despite the small separations, the EGNG algorithm can still successfully separate the four neural groups.

to five different neurons. Five non-overlapping neural clusters were distributed across the vector space as shown in figure 3(a). At the beginning of the classification, there were only three EGNG nodes (red dots) interconnected by two EGNG edges (blue lines) among the neural data as in figure 3(b). As the clustering proceeds, as shown in figure 3(c), more EGNG nodes and edges were generated to loosely cover the entire neural distribution with EGNG nodes still interconnecting with one another. By the end of the clustering, some EGNG edges were pruned and the EGNG nodes were separated into five different EGNG clusters. At the point, the Euclidian distances from the neural points to their closest EGNG nodes were determined. The neural points were classified based on the EGNG cluster containing the closest EGNG node to the neural point. For demonstration purposes, neural points belonging to the same cluster were labeled with the same color for easy identification in figure 3(d).

3.2. EGNG algorithm is robust for clustering non-Gaussian neural spikes

The EGNG algorithm makes no presumption about the nature of the data distribution and is robust in classifying neural clusters with non-Gaussian distributions. Figure 4(a) shows synthetic neural points distributed in three concentric rings. At the beginning, EGNG nodes were randomly generated and interconnected by EGNG edges covering the three rings as one large group. After some iterations, the EGNG edges between the rings were pruned and resulted in three concentric EGNG circles. Figure 4(b) shows the final classification results indicating that the neural points were successfully classified into three separate clusters and labeled with three different colors to indicate which EGNG cluster the points were classified.

3.3. EGNG algorithm can classify closely packed neural clusters

Figure 5(a) is the result of using the EGNG algorithm to classify simulated neural points distributed in four neural clusters. Despite the four neural clusters being close to one another, there were still low-density gaps between the clusters to allow them to be separated. In order to test the capability of the EGNG algorithm to classify closely spaced neural clusters, additional Gaussian noise with a variance of 0.16 was added to make the neural clusters overlap closer. With the help of edge pruning at low-density areas, the EGNG algorithm can separate the neural clusters successfully, as demonstrated in figure 5(b).

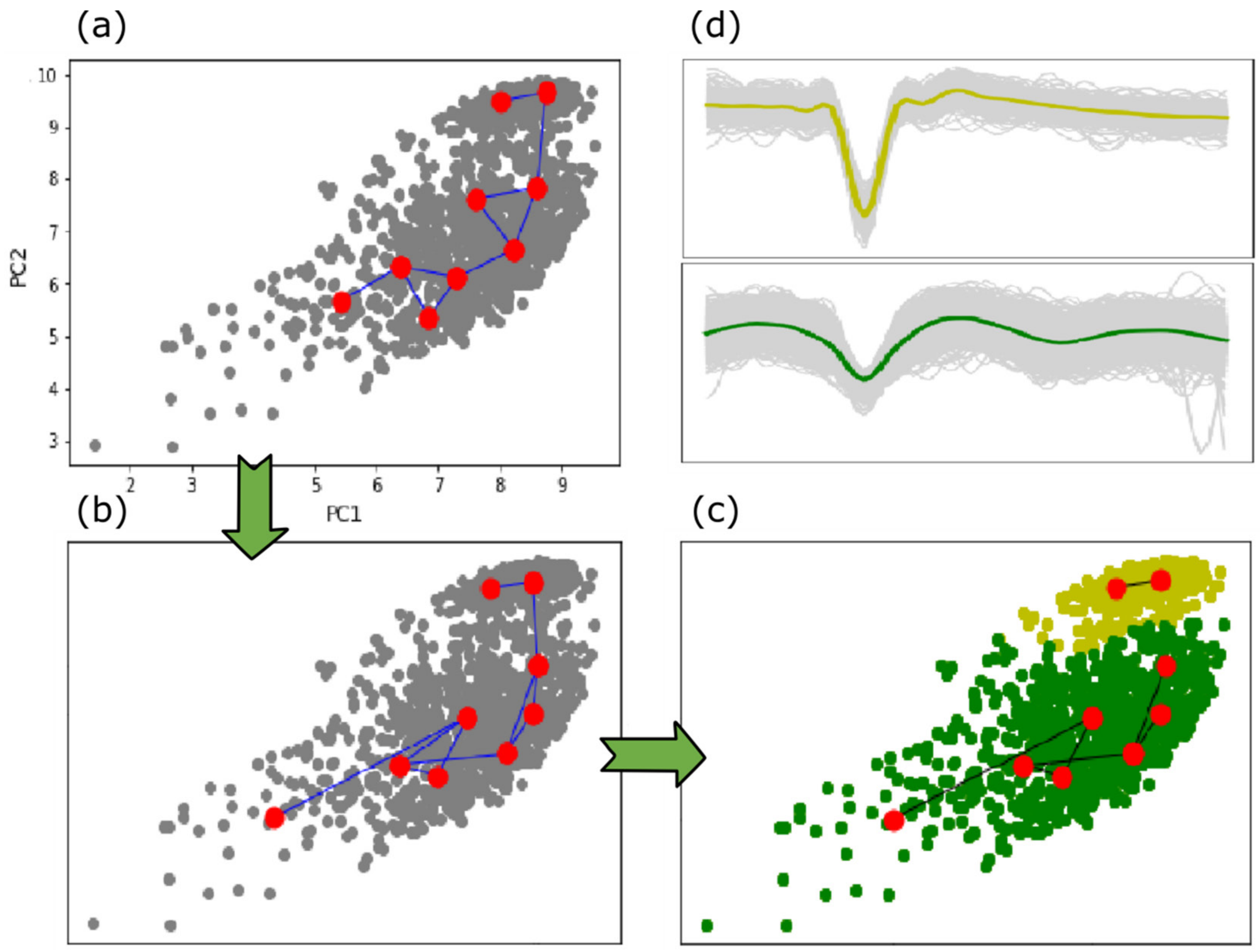


Figure 6. Pre-recorded real neural spikes were correctly sorted by the EGNG algorithm. (a) EGNG nodes were connected together as a single large EGNG group at the beginning of the sort. (b) The EGNG nodes were pulled towards the areas with higher concentrations of neural spikes. (c) The EGNG edge between the two neural groups was pruned to correctly separate the neural spikes into two EGNG clusters. (d) Shows the temporal waveforms and the waveform averages of the two clusters in (c).

3.4. Neural spikes recorded from an anesthetized gerbil correctly classified by the EGNG algorithm

Pre-recorded neural spikes measured from the fiber tract of the fifth cranial nerve of an anesthetized gerbil were used to demonstrate the sorting capability of the EGNG algorithm for real neural data. The temporal shapes of the recorded neural spikes have similar structures of a prominent negative peak, making the ground truth determination of the classification challenging, and the classification accuracy therefore cannot be calculated. The EGNG algorithm, however, determined that there were two separable clusters, with one cluster having larger negative peaks than those of the other cluster, as shown in figures 6(d) and (e). At the beginning of the sort, EGNG nodes were all connected together forming a large EGNG cluster covering the entire neural data, as shown in both figures 6(a) and (b). The EGNG algorithm, however, took advantage of the fact that the density of neural points were more concentrated in the cluster centers than the perimeters, and the EGNG nodes were then separated at the boundary and finally broken into two disconnected clusters, as shown in figure 6(c). It is worth mentioning that the lower cluster fired about twice as often compared to the upper cluster and also neural spikes of the lower cluster had much worse signal-to-noise ratio than those of the upper clusters. Given that the similar temporal structures, these neural spikes could have been difficult to be sorted, but EGNG handled them well and was able to separate them into two closely packed clusters.

3.5. EGNG Classification of publically available neural data sets

Figure 7 shows the results of the EGNG classification results for both the hc-1 (a) and NCHybrid136 (d) data sets, and both of the data sets resulted in two EGNG clusters respectively. The corresponding temporal waveforms and the calculated waveform averages of the clusters were also plotted in figures 7(b) and (c) for the hc-1 data set, and in figures 7(e) and (f) for the NCHybrid136 data set. The waveform averages showed distinctive differences between the different clusters of neural spikes.

3.6. Comparing EGNG to other common clustering algorithms

In this section, the EGNG algorithm was compared to four other commonly used clustering algorithms (DBSCAN,

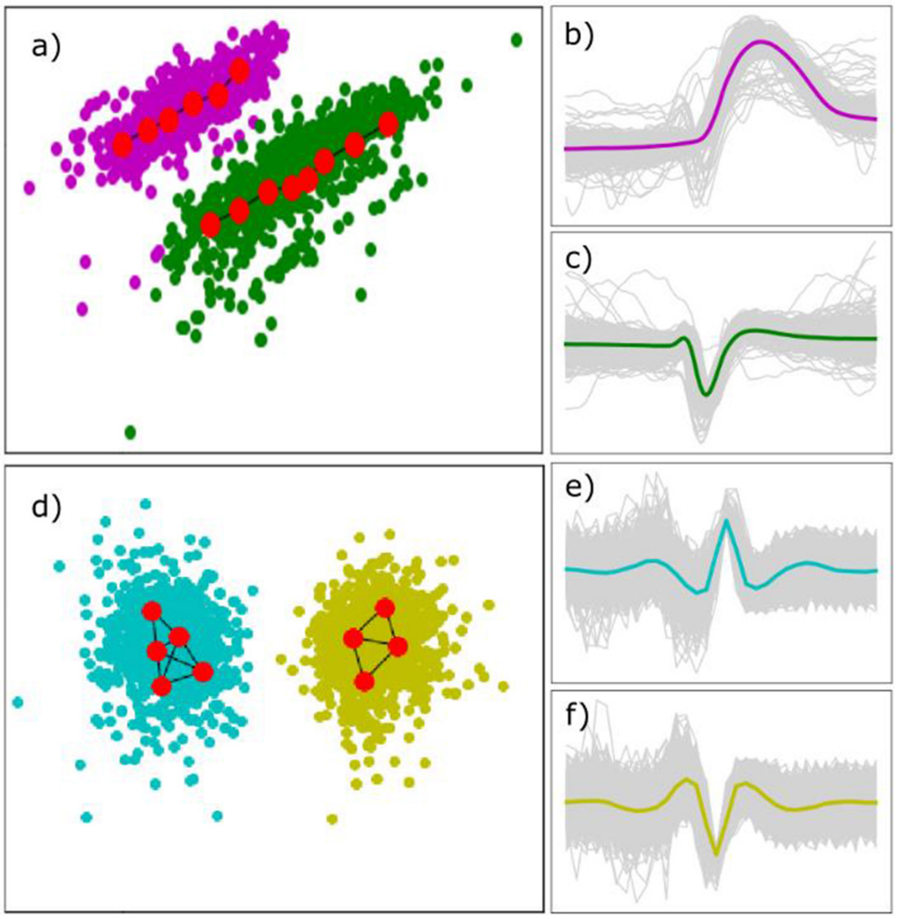


Figure 7. EGNG algorithm was used to classify publically available neural data sets. (a) Neural data obtained from the hc-1 and (d) from NCHybrid136 data sets were classified separately with off-line EGNG. (b) and (c) Show the temporal waveforms and the waveform averages of the two clusters of (a). (e) and (f) Show the temporal waveforms and the waveform averages of the two clusters of (d).

K-means, EM and SPC) for sorting accuracy. The sorted results are displayed in figure 8. Four datasets (noisy moon data, five simulated clusters, three anisotropic Gaussian clusters, pre-recorded real neural spikes from a gerbil) were submitted to the four algorithms (the above three and EGNG) for the comparison. The true positive (*tp*), the false negative (*fn*) and the false positive (*fp*) were calculated based on the sorted results to obtain the F1-scores for the four sorting algorithms for quantitative comparison. The F1-score is defined as [44, 45]

$$F_1 = 2 \times \left(\frac{precision \times recall}{precision + recall}\right)$$

where

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}.$$

The F1-score has a value between 0 and 1 in which 1 indicates a 100% correctness for the sort. The means and the standard derivations of the F1-scores evaluated from 10 repeated runs for the four sorting methods are listed in table 2. The results indicate that the EGNG algorithm could sort correctly for all the four datasets with high F1-scores, while the other four algorithms had problems sorting at least one of the data sets. This is largely due to the fact that EGNG algorithm can handle closely spaced and non-Gaussian clusters. The convergence time of the EGNG algorithm was also compared to the other four clustering algorithms using the five simulated clusters dataset on a 16 GB, 3.4 HHz Intel i7 computer. Due to the algorithmic simplicity of EGNG, EGNG had the shortest convergence time among the five algorithms, as shown in table 3. It is also worth mentioning that EGNG was written in the Python programming language. Python is an interpreting computer language that is notoriously slow in handling for and while loops. In contrast, the other algorithms were highly optimized with C language in the Scikit-learn library, and EGNG may converge faster if properly optimized.

3.7. Convergence time and clustering accuracy against increased EGNG dimensions

The convergence time and clustering accuracy were also evaluated as the dimensions of the EGNG nodes and the neural data were increased. The benchmarks were calculated on the same i7 computer using the same data set of figure 5(a). As shown in table 4, the convergence time was approximately increased by a factor of 2 as the EGNG node and data dimensions were

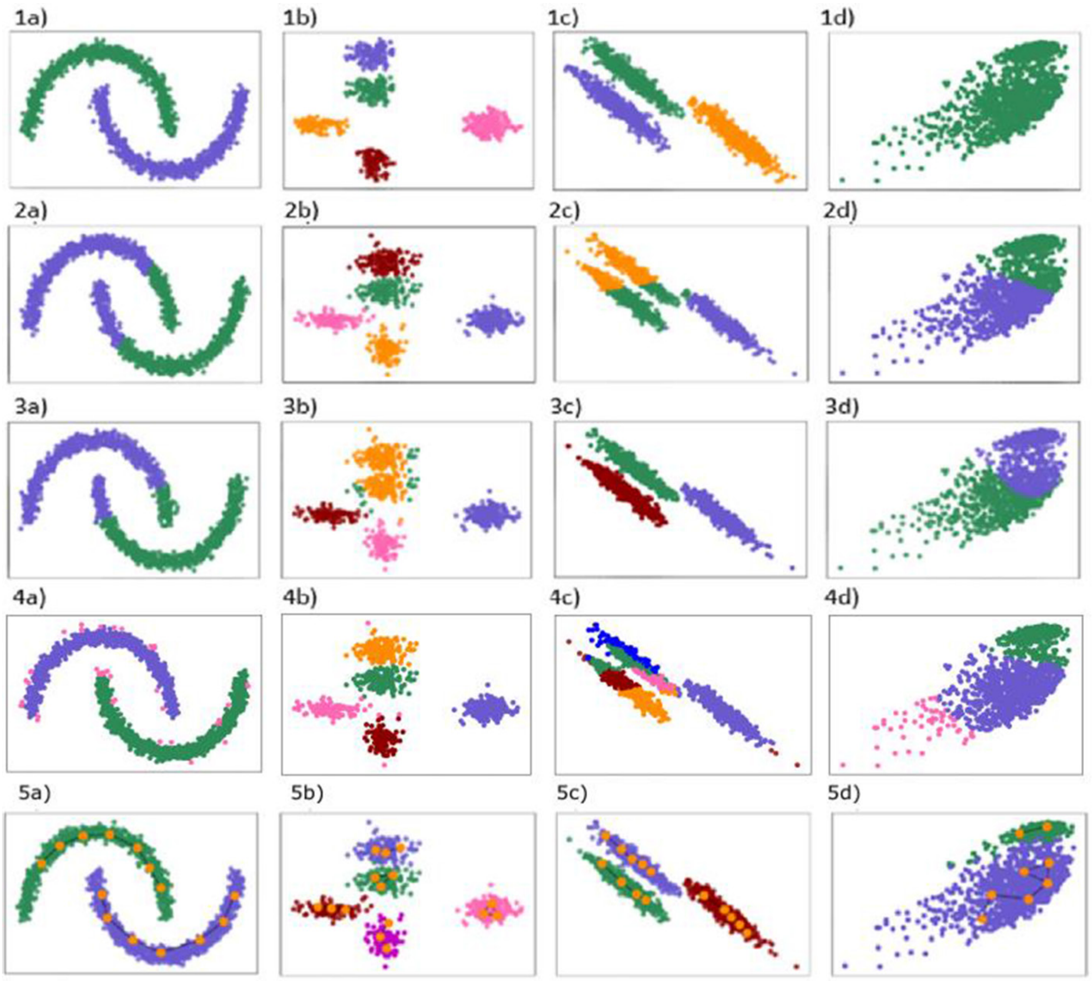


Figure 8. Comparison of sorting accuracy between three commonly used classification algorithms to EGNG. The four sorting algorithms are (1) density-based spatial clustering of applications with Noise (DBSCAN), (2) *K*-means, (3) expectation maximization (EM), (4) super-paramagnetic clustering (SPC) and (5) EGNG. Four different datasets were used to test the algorithms and they are (a) noisy moon data, (b) synthetic five cluster groups with Gaussian distribution, (c) synthetic three cluster groups with anisotropic Gaussian distribution, and (d) pre-recorded real neural spikes from a gerbil. The EGNG algorithm can correctly classify all four datasets while the other three algorithms had trouble classifying at least one of the data sets.

Table 2. Means and standard derivation of F1-scores comparing the five sorting algorithms in which EGNG algorithm has the highest F1-score for the four datasets.

	Data 1 (moon data)	Data 2 (5-clusters)	Data 3 (anisotropic)	Data 4 (real data)
DBNSCAN	1.0 ± 0	$0.923 \pm 3.53 \times 10^{-8}$	$0.996 \pm 3.18 \times 10^{-8}$	$0.227 \pm 5.49 \times 10^{-9}$
EM	$0.855 \pm 1.36 \times 10^{-6}$	$0.766 \pm 2.62 \times 10^{-8}$	1.0 ± 0	$0.720 \pm 2.80 \times 10^{-5}$
K-means	$0.754 \pm 4.88 \times 10^{-6}$	$0.604 \pm 2.34 \times 10^{-8}$	$0.827 \pm 2.73 \times 10^{-8}$	$0.812 \pm 2.12 \times 10^{-6}$
SPC	$0.986 \pm 3.43 \times 10^{-5}$	$0.985 \pm 3.12 \times 10^{-5}$	$0.493 \pm 1.13 \times 10^{-5}$	$0.8536 \pm 1.32 \times 10^{-5}$
EGNG	1.0 ± 0	$0.987 \pm 4.21 \times 10^{-8}$	$0.998 \pm 3.21 \times 10^{-6}$	$0.974 \pm 5.31 \times 10^{-5}$

increased from 2 to 5. In addition, the clustering accuracy was unchanged as the dimension was increased.

3.8. On-line EGNG for real-time streaming neural spike classification

The true power of the EGNG algorithm rests on its ability to rapidly and adaptively classify streaming neural spikes in real-time. Figure 9 shows the process of how EGNG clusters were developing through analyzing streaming neural spikes in real-time. At the beginning of the clustering, as shown in figure 9(a), only a few initial neural points entered the algorithm and a few EGNG nodes were created interconnected as one EGNG cluster to cover them. Since the full distribution of the neural points has not been learned by the EGNG nodes at this early stage, these early neural points were classified to belong to a first EGNG cluster. As more neural points were streamed into the algorithm, more EGNG nodes were created and separated into EGNG clusters covering each of the neural clusters better based on the EGNG clustering process.

Table 3. Convergence time in seconds comparing EGNG to the four other clustering algorithms in classifying data 2 (5-clusters), in which EGNG has the shortest convergence time.

	Convergence time (s) of data 2 (5-clusters)
DBNSCAN	6.013
EM	5.621
K-means	8.351
SPC	6.241
EGNG	5.408

Incoming neural points were then classified based on the EGNG cluster distribution at the time, as shown in figures 9(b) to (d). As more and more neural points were streamed into the clustering process and analyzed, the full neural distribution was then learned by the EGNG nodes, and the EGNG clusters became stable. Subsequently incoming neural points were classified correctly, as shown in figures 9(e) and (f). In these figures, colors were used to represent the assigned cluster of the neural point at the time the point was analyzed. It is noted that some early neural points were not correctly classified at the beginning of the clustering because the neural distribution was not yet fully learned, and the algorithm had to rely on only a few neural points to build the full picture of the neural distribution, resulting in some early classification errors. As more neural points streamed in, the full neural distribution was more thoroughly learned by the EGNG clusters and the subsequent classification was correct. This learning period is brief and only requires the first tens of neural spikes, as demonstrated in figure 9. This initial error should not pose a significant problem for data analysis in a real-time closed-loop setting, especially for longer recordings. It is purposely to show that the neural points were not reanalyzed to allow rapid classification for immediate downstream data analysis.

3.9. EGNG clusters adapt to neural distribution changes

In addition to the ability to learn the neural distribution rapidly in real-time, another major advantage of EGNG is its flexibility to adapt to neural distribution changes for continual and correct neural clustering, as well as the ability to differentiate actual neural spikes from noise. Figure 10(1(a)) shows the EGNG algorithm has learned the neural distribution and correctly covered it with three EGNG clusters. Figure 10(1(b)) shows how new neural points were streamed in to the left side of the third cluster (yellow) located at the lower part of the figure. These new neural points represented a situation as it might occur during long-term electrophysiological recordings when the implanted electrode moved slightly, resulting in a temporal shape change of the neural spikes and a location shift for the neural points in the vector space. As more neural points were streamed in to the left of the cluster, the EGNG nodes of the cluster were shifted towards these new neural points to the left, as displayed in figure 10(1(c)). The movements of EGNG clusters allow the neural points to be continuously and correctly classified, even when the physiological conditions are altered.

Table 4. Changes of the convergence time and the clustering accuracy off-line as the dimension of the EGNG nodes and the neural data were increased from 2 to 5.

EGNG node dimension	Convergence time (s)	Clustering accuracy
2	10.88	0.971
3	17.6	0.968
4	20.8	0.974
5	23.97	0.973

Electrode movement can also result in picking up of neural spikes from a neuron that was previously too far away from the electrode to yield a useful signal. In this situation, EGNG can dynamically create new EGNG clusters to add to the existing EGNG clusters for correct clustering. Figure 10(2(a)) shows three stable EGNG clusters covering the neural distribution. Next, additional neural points were streamed in at a location away from the existing clusters (top right), as shown in figure 10(2(b)). These remote neural points caused the algorithm to generate new EGNG nodes and move towards them. In this manner, a new EGNG cluster was created and neural points at the vicinity were classified as belonging to the new cluster, as shown in figure 10(2(c)).

Finally, electrode movement might result in the loss of neural spikes from a neuron. In this case, the EGNG cluster covering these neural spikes should be removed. Figures 10(3(a)–(c)) demonstrate this situation by only streaming in neural points belonging to two EGNG clusters, leaving the third cluster (lower right) without new neural points to strengthen it. As a result, the third EGNG cluster without neural spikes was removed from the distribution, leaving only the first two EGNG clusters to continue for the classification.

EGNG can easily integrate with noise rejection to eliminate accidental noisy spikes from genuine neural spikes. Neural spikes from the same neuron tend to cluster close to one another, but noisy spikes are scattered all over the place. If the Euclidian distance of a point to its nearest EGNG node is longer than the noise rejection threshold (d_n), the point is rejected as noise and is not classified. In figure 10(4(b)), all new points are colored in grey; the points closer to a cluster are colored with the respective clusters and the points that are classified as noise are colored in purple and rejected.

3.10. Hardware implementation of EGNG to sort streaming neural data in real-time

A hardware implementation of the EGNG algorithm was developed for an FPGA chip to demonstrate the real-time spike clustering capability of EGNG. An FPGA development broad (Arty A7-35T, Xilinx, CA) containing an FPGA chip (xc7a35ticsg324-1L, Xilinx, CA) running on a 20 MHz system clock was used for the implementation. The EGNG code for the FPGA hardware was written using Verilog, which is a hardware description language, and was compiled and implemented on the FPGA using the Vivado design suite (Xilinx, CA).

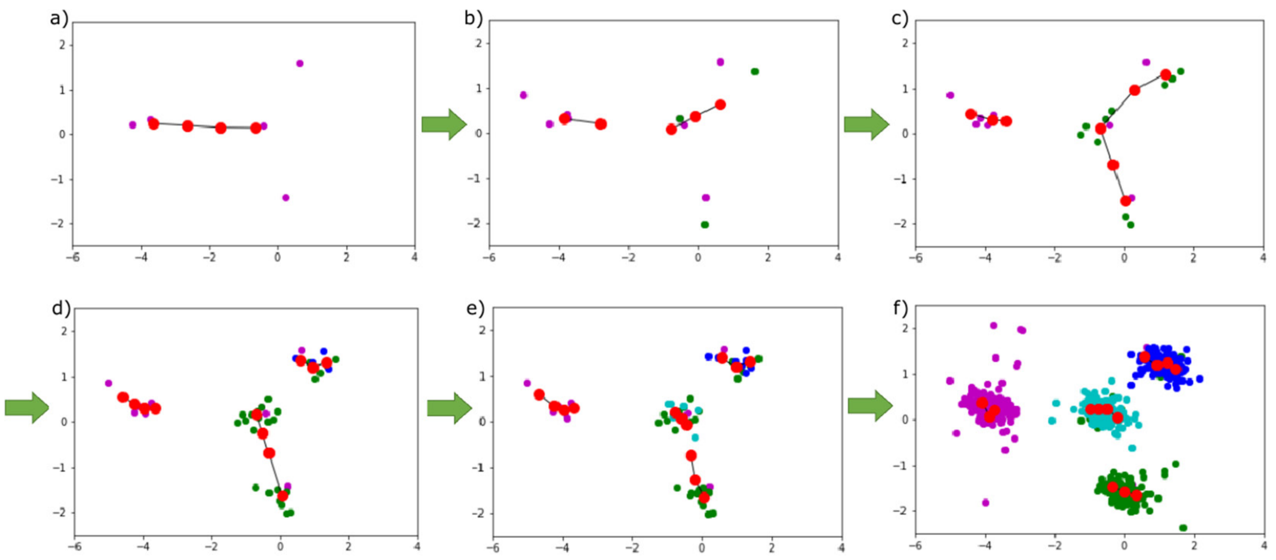


Figure 9. On-line EGNG algorithm to sort streaming neural spikes. (a) At the beginning of the sort, a few neural points were streamed into the system and several connected EGNG nodes were created forming a single EGNG cluster. Since only one EGNG cluster was formed, all the neural points were classified to the cluster (purple). (b) As more neural points were streamed in, the EGNG nodes broke into two clusters and the new incoming neural points will be classified based on the shortest Euclidian distances to the two clusters (cluster 1—purple, left; cluster 2—green, right). (c) As more neural points came in, the new neural points continued to be classified to the two EGNG clusters. (d) The EGNG nodes were further broken into three groups and new incoming neural points were classified into three cluster groups (cluster 1—purple, left; cluster 2—green, below; cluster 3—blue, right). (e) The EGNG nodes were broken into four groups as the data distribution became more apparent as more data were analyzed. New incoming data were classified into four clusters (cluster 1—purple, left; cluster 2—green, below; cluster 3—blue, right; cluster 4—cyan, center). (f) The distribution of the neural points were completely learned by the algorithm and the EGNG clusters became stable. New incoming neural data were correctly classified by the four stable EGNG clusters.

Figure 11(a) illustrates the overall design of the EGNG hardware implementation. The FPGA design was separated into several hardware blocks: (1) 16 interconnected EGNG nodes, (2) a sorting network block, (3) an edge update block, (4) a connected components block, and (5) a label matcher block. As a neural spike was streamed into the FPGA, the position of the neural spike in the vector space was sent to all EGNG nodes simultaneously, and the Euclidian distances between the neural spikes to all the EGNG nodes were calculated in parallel with a single clock cycle. Based on the calculated Euclidian distances, the sorting network block then determined the closest two EGNG nodes (S_1 and S_2), which were then used to update the positions and insert parameters of all the EGNG nodes (step 2 and 3). If the number of iterations had exceeded the value of λ, the sorting step was repeated with the insert parameters of the EGNG node, and a new EGNG node was inserted (step 4). The edge update block then created new EGNG edges and deleted old ones, and condensed all the EGNG edges into a single vector representing a triangular matrix from which connected components could be determined (step 5). The edge vector was passed into the connected components block, which then determined the clusters of EGNG nodes connected by EGNG edges. The resulting clusters were sent to the label matcher block that compared the most recently identified clusters to the previous set of labeled clusters in order to ensure that cluster labels for each group of connected EGNG nodes remained consistent. Finally, the input neural spike was classified according to the EGNG cluster containing the S_1 node, and a cluster label was assigned to the neural spike at the output.

Figures 11(c)–(h) show snapshots of the EGNG clustering with the FPGA hardware using the same artificial data set used to generate figure 9 in software clustering as inputs. A movie showing the detailed clustering process was also included in the supplementary information. The hardware EGNG clustering followed the same evolution as that in the software classification to separate and form EGNG clusters, and finally resulted in four well-separated EGNG clusters, demonstrating the same clustering capability in both hardware and software. Figure 11(i) compares the results of the hardware and software EGNG clustering and shows identical clustering number assignments, except in the early stage of the process when the four clusters were not yet formed in the hardware implementation. A normalized confusion matrix was also constructed to demonstrate the clustering agreement between the hardware and software implementation, as shown in figure 11(b). Agreement values between 0.96 to 1.00 were obtained, and the agreements can further be improved by using more neural spikes as the disagreements mainly occurred in the beginning of the clustering.

The worse-case clustering latency between the time when the neural spike was sent to the FPGA and when the clustering label assignment was determined was 3.10 μs (62 clock cycles), allowing a minimum of 322 580 neural spikes to be clustered per second. However, several of the FPGA modules were not optimized and future improvements can be made to

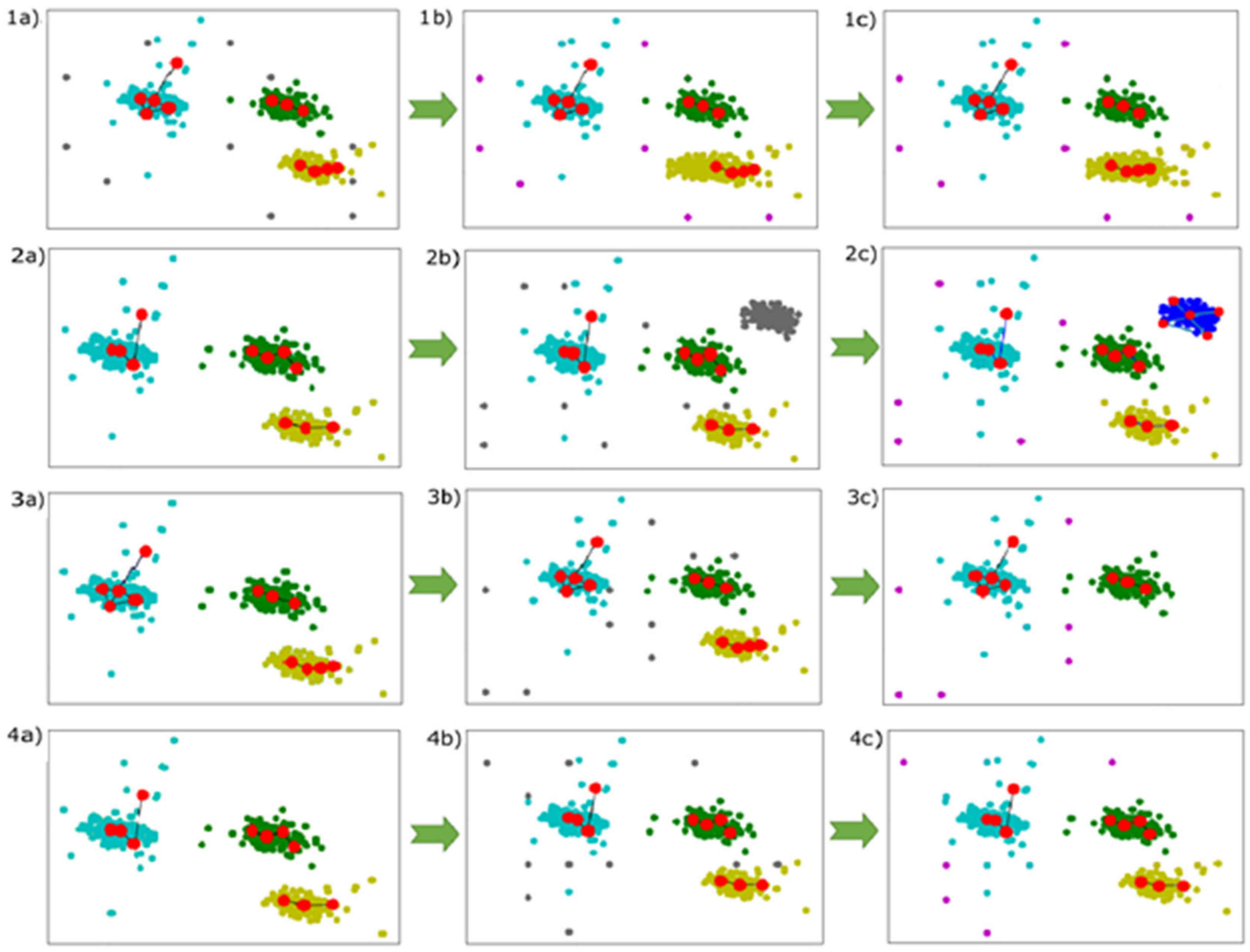


Figure 10. The EGNG algorithm is highly adaptive to changes in neural distribution. (1(a)–(c)) Moving one of the EGNG clusters: Three stable EGNG clusters were initially established and new neural points were added to the left side of the cluster in below; (c) the EGNG nodes were moved to the left to adapt to the movement of the cluster. (2(a)–(c)) Addition of a new EGNG cluster: new neural points (grey) were added to the upper left side to simulate a new neuron being recorded and a new EGNG cluster was created to correctly classify the new neural spikes. (3(a)–(c)) Deletion of obsolete clusters: no neural points near the EGNG cluster were input into the algorithm to simulate neural spikes from a neuron which can no longer be picked up by the electrode, and the EGNG cluster was then removed from the distribution. (4(a)–(c)) The points located near to a cluster are colored with the respective clusters and the points colored in purple, were classified as noise and rejected. Note: grey points represent new neural points entering the algorithm and purple points were incoming neural points classified as noise due to their far distance to any of the EGNG clusters.

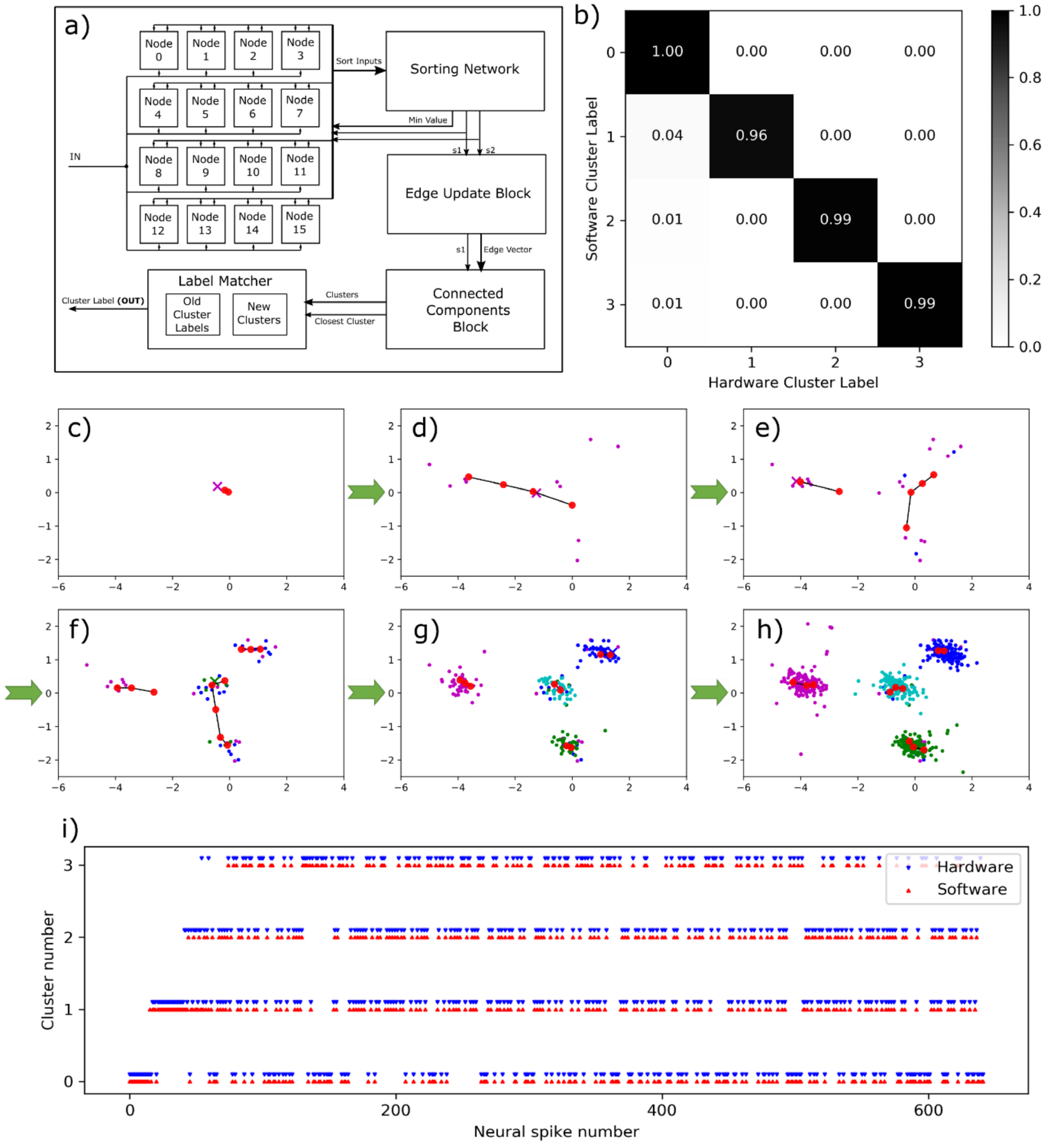
further reduce the clustering latency and increase the number of neural spikes that can be handled by the FPGA hardware.

4. Discussion

EGNG is a robust spike clustering algorithm that correctly classifies both pre-recorded and streamed neural spikes in off-line and real-time situations. The advantages of EGNG include (1) rapid classification for streamed neural spikes, (2) self-adjustment to cluster movements for continual correct clustering, (3) addition and deletion of new and obsolete EGNG clusters as the electrode move closer to or away from neurons, (4) no assumption for the clusters to be Gaussian distributed, (5) no need to retain neural data in system memory, (6) automatic determination of the neural cluster number, and (7) compatibility with parallel computation using digital electronic technology.

In order to design a spike clustering algorithm that is suitable for long-term neural recordings and for system miniaturization using digital electronic technology, several guidelines were followed during the design. First, EGNG was designed to be computationally lightweight. This was accomplished through an approach in which the calculations are mostly limited to the computation of the Euclidean distance between two points and the minimum Euclidean distance between two nodes. Second, because the neural distribution is learned by EGNG clusters, neural points are not required to be retained in the system memory once processed, saving a significant amount of system resources. Finally, EGNG clusters were designed to be highly adaptive to neural distributional changes, which is important for long-term recordings during which electrode movements are difficult to avoid.

The EGNG algorithm performs very well when classifying pre-recorded neural spikes, even though the true strength and usefulness of EGNG are in classifying streaming neural



Software \ Hardware	0	1	2	3
0	1.00	0.00	0.00	0.00
1	0.04	0.96	0.00	0.00
2	0.01	0.00	0.99	0.00
3	0.01	0.00	0.00	0.99

Figure 11. EGNG implementation on an FPGA to demonstrate hardware real-time neural spike classification capability. (a) Schematic diagram of the FPGA implementation with 16 EGNG nodes simultaneously computing the Euclidian distances between the input neural spike data (IN) and each of the EGNG nodes. A sorting network was built to determine S_1 and S_2 nodes. An edge update block was built to update EGNG edges creation and pruning, and a connected components block was built to find clusters of connected EGNG nodes. A label matcher block was used to maintain consistency between clusters and cluster labels. Finally, the neural spike data was classified for the output; (b) normalized confusion matrix comparing the classification results between the software and hardware EGNG. (c)–(h) Real-time neural spike classification using the FPGA hardware with the same input data of figure 9. The evaluation of the EGNG nodes closely followed that of figure 9. (i) Semi-raster plot comparing the classification between the software and hardware implementations.

spikes in real-time. The algorithm does not require the neural clusters to be Gaussian distributed, which allows classifying neural spikes contaminated with non-traditional noises. The algorithm does this by conforming and self-connecting the EGNG nodes into various shapes to best cover the neural distribution. This capability is generally lacking from other spike clustering algorithms, including EM, *K*-mean, SPC and DBSCAN.

Another advantage of the EGNG algorithm that is different from those of the current generation of spike clustering algorithms is its ability to adapt to real-time changes of the neural distribution. If the clustering is not adapted as the electrode drifts away from its initial position, the classification may become unstable, leading to decoding problems downstream. EGNG may help maintain the classification for a longer period of time as the algorithm continues to follow the changes in the neural distribution. In contrast, the current generation of neural decoding routines are based on an unchanging neural distribution and will require new developments in order to take advantage of the dynamic sensing of new and obsolete clusters. In a boarder sense, EGNG can be considered as a template matching technique that uses EGNG nodes and edges to build the templates (EGNG clusters) in the vector space, and that has built-in capability to adapt to changes of the neural distribution.

It is recognized that some of the early neural spikes were not classified correctly in the real-time mode. This is due to the fact that the algorithm learns from the data as it comes in, and does not have enough information to deduce the full picture of the entire neural distribution during the initial phase of a real time clustering process. This situation, however, significantly improves as more neural spikes are streamed in. The algorithm continuously updates itself to better match the full neural distribution, making the classifications for latter incoming spikes more accurate. This process can be understood as a Bayesian learning process in which the EGNG clusters are continuously updated with new information of the incoming neural spikes. Through this process, an increasingly better inference of the neural distribution can be estimated, yielding to increasingly better classification predictions. This adaptability is manifested by allowing the algorithm to move, add and delete EGNG clusters to better match the neural distribution as the electrophysiological condition changes.

The EGNG algorithm is designed to take advantage of the computational parallelism in modern digital electronics for rapid spike clustering. As demonstrated in our FPGA implementation, 16 EGNG nodes were implemented and each node was an individual calculator computing the Euclidian distance between the EGNG node and the input neural point. Although our current implementation only has 16 EGNG nodes, it is relatively simple to replicate the basic building block of an EGNG node to create more nodes and retain the parallel computation structure. Additional acceleration was achieved through use of parallel comparators to determine the EGNG nodes with the shortest and the second shortest Euclidian distances. Even with the additional hardware modules to manage the EGNG node and edge updates, the entire classification took less than 62 clock cycles, and the classification latency for our current system was less than a hundred microseconds, allowing rapid signal analysis downstream for closed-loop neural control.

We have previously developed an analog integrated circuit (IC) combining a high-sensitive neural amplifier with an adjustable optical light source driver with a physical dimension of $2 \times 3\,\text{mm}^2$ [21]. Using this IC, we have demonstrated experimentally that action potentials from an anesthetized gerbil can be measured and at the same time optogenetically inhibited. We plan to also integrate the EGNG algorithm onto this IC. The goal is to develop a miniaturized and low-power mixed signal IC that can perform neural recording, neural data analysis, and optogenetic stimulation/inhibition without the need of a desktop computer in the near future.

The EGNG algorithm was only discussed and demonstrated to classify neural spikes recorded from a single electrode at this point. This does not, however, imply that the algorithm can only be used to classify neural spikes recorded from a single source. The algorithm can be extended for use with for multiple electrodes by simply adding an additional dimension in the vector space to represent the recording site number. In this manner, neural points recorded from different electrodes are presented in their respective planes, and EGNG nodes and edges can be extended for the additional dimension to classify the neural spikes accordingly. This extension is particularly applicable for electrodes with a small number of recording sites, such as in tetrode recording. For electrodes with high density of recording sites, additional development of the algorithm is needed to avoid a significant increase in hardware resources which would prevent system miniaturization. We are currently working on refining the algorithm and hardware implementation techniques to better handle higher electrode density recording.

Acknowledgments

The authors would like to acknowledge the support by the Science and Technology Development Fund of Macau (FDCT) under Grants 093/2015/A3, 088/2016/A2 and the University of Macau under Grants MYRG2014-00010-AMSV, MYRG2015-00178-AMSV, MYRG2016-00157-AMSV, MYRG2018-00146-AMSV, MYRG2015-00178-AMSV & MYRG2016-00157-AMSV. The authors would also want to thank Drs. Elizabeth McCullagh and Chang Hao Chen for helpful discussions.

Supplements

A supplementary document can be downloaded including the pseudo-codes for both on-line and real-time EGNG algorithms, additional evaluation of the algorithm using split data set, selection of parameter values and performance evaluation against increased EGNG node number and dimension.

Three movies were included in the supplementary information. The first two movies demonstrate the EGNG algorithm in off-line and real-time modes. The first video is the movie version of figure 1 showing the generation, movement and deletion of EGNG nodes and edges to cover the pre-recorded neural distribution and classify the neural points according to the EGNG clusters. The second video is the movie version of figure 9 demonstrating how the EGNG algorithm adapts and classifies neural spikes as the neural spikes were streaming in to the algorithm in real-time. The third video is the extended movie version of figure 11, demonstrating real-time EGNG clustering with the FPGA hardware.

ORCID iDs

Tim C Lei https://orcid.org/0000-0002-3797-1092

References

[1] Pashaie R, Baumgartner R, Richner T J, Brodnick S K, Azimipour M, Eliceiri K W and Williams J C 2015 Closed-loop optogenetic brain interface *IEEE Trans. Biomed. Eng.* **62** 2327–37
[2] Arsiero M, Lüscher H R and Giugliano M 2007 Real-time closed-loop electrophysiology: towards new frontiers in *in vitro* investigations in the neurosciences *Arch. Italiennes Biol.* **145** 193–209
[3] Edward E S, Kouzani A Z and Tye S J 2018 Towards miniaturized closed-loop optogenetic stimulation devices *J. Neural Eng.* **15** 021002
[4] Grosenick L, Marshel J H and Deisseroth K 2015 Closed-loop and activity-guided optogenetic control *Neuron* **86** 106–39
[5] Rosin B, Slovik M, Mitelman R, Rivlin-Etzion M, Haber S N, Israel Z, Vaadia E and Bergman H 2011 Closed-loop deep brain stimulation is superior in ameliorating parkinsonism *Neuron* **72** 370–84
[6] Williams M 2007 Electrophysiological Techniques *Curr. Protoc. Pharmacol.* **39** 10–2
[7] Hubel D 1957 Tungsten microelectrode for recording from single units *Science* **25** 549–50
[8] Cuevas J 2014 Electrophysiological recording techniques *Ref. Module Biomed. Res.* 1–7
[9] Humphrey D R and Schmidt E M 1990 Extracellular single-unit recording methods *Neurophysiological Techniques* vol 15, ed A A Boulton *et al* (New York: Humana Press) pp 1–64
[10] Chen C H, Pun S H, Mak P U, Vai M I, Klug A and Lei T C 2014 Circuit models and experimental noise measurements of micropipette amplifiers for extracellular neural recordings from live animals *Biomed Res. Int.* **2014** 135026
[11] Rossant C *et al* 2016 Spike sorting for large, dense electrode arrays *Nat. Neurosci.* **19** 634–41
[12] Rey H G, Pedreir A C and Quiroga R Q 2015 Past, present and future of spike sorting techniques *Brain Res. Bull.* **119** 106–17
[13] Kadir S N, Goodman D F and Harris K 2014 High-dimensional cluster analysis with themasked EM algorithm *Neural Comput.* **26** 2379–94
[14] Jun J, Mitelut C, Lai C, Gratiy S L, Anastassiou C A and Harris T D 2017 Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction (bioRxiv:101030)
[15] Chung J E, Magland J F, Barnett A H, Tolosa V M, Tooker A C, Lee K Y, Shah K G, Felix S H, Frank L M and Greengard L F 2017 A fully automated approach to spike sorting *Neuron* **95** 1381–94
[16] Blatt M, Wiseman S and Domany E 1997 Data clustering using a model granular magnet *Neural Comput.* **9** 1805–42
[17] Rutishauser U, Schuman E M and Mamelak A N 2006 Online detection and sorting of extracellularly recorded action potentials in human medial temporal lobe recordings, *in vivo J. Neurosci. Methods* **154** 204–24
[18] Mishra J and Gazzaley A 2015 Closed-loop cognition: the next frontier arrives *Trends Cogn. Sci.* **19** 242–3
[19] Roth E, Sponberg S and Cowan N J 2014 A comparative approach to closed-loop computation *Curr. Opin. Neurobiol.* **25** 54–62
[20] Newman J P, Fong M, Millard D C, Whitmire C J, Stanley G B and Potter S M 2015 Optogenetic feedback control of neural activity *Elife* **4** e07192
[21] Chen C H, McCullagh E A, Pun S H, Mak P U, Vai M I, Mak P I, Klug A and Lei T C 2017 An integrated circuit for simultaneous extracellular electrophysiology recording and optogenetic neural manipulation *IEEE Trans. Biomed. Eng.* **64** 557–68
[22] Fenno L, Yizhar O and Deisseroth K 2011 The development and application of optogenetics *Annu. Rev. Neurosci.* **34** 389–412
[23] Deisseroth K 2011 Optogenetics *Nat. Methods* **8** 26–9
[24] Quiroga R 2007 Spike sorting *Scholarpedia* **2** 3583
[25] Fraley C and Raftery A E 2002 Model-based clustering, discriminant analysis, and density estimation *J. Am. Stat. Assoc.* **97** 611–31
[26] Quiroga R Q, Nadasdy Z and Ben-Shaul Y 2004 Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering *Neural Comput.* **16** 1661–87
[27] Vaibhav K, Gibson S and Marković D 2013 A 75 μW, 16-channel neural spike-sorting processor with unsupervised clustering *IEEE J. Solid-State Circuits* **48** 2230–8
[28] Luan S, Williams I, Maslik M, Liu Y, De Carvalho F, Jackson A, Quiroga R Q and Constandinou T G 2018 Compact standalone platform for neural recording with real-time spike sorting and data logging *J. Neural Eng.* **15** 046014
[29] Park J, Kim G and Jung S-D 2017 A 128-Channel FPGA-based real-time spike-sorting bidirectional closed-loop neural interface system *IEEE Trans. Neural Syst. Rehabil. Eng.* **25** 2227–38
[30] Kuon I, Tessier R and Rose J 2007 FPGA architecture: survey and challenges *Found. Trends® Electron. Des. Autom.* **2** 135–253
[31] Fee M S, Mitra P P and Kleinfeld D 1996 Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-Gaussian variability *J. Neurosci. Methods* **69** 175–88
[32] Liu J and Newsome W T 2006 Local field potential in cortical area MT: stimulus tuning and behavioral correlations *J. Neurosci.* **26** 7779–90
[33] Fritzke B 1995 A growing neural gas learns topologies *Adv. Neural Inf. Process. Syst.* **7** 625–32
[34] Yadav J and Sharma M 2013 A review of *K*-mean algorithm *Int. J. Eng. Trends Technol. (IJETT)* **4** 2972–6
[35] Doya K, Ishii S, Pouget A and Rao Rajesh P N 2007 *Bayesian Brain—Probabilistic Approaches to Neural Coding* (Cambridge, MA: MIT Press)
[36] Qin A K and Suganthan P N 2004 Robust growing neural gas algorithm with application in cluster analysis *Neural Netw.* **17** 1135–48
[37] Pedregosa F *et al* 2012 Scikit-learn: Machine Learning in Python *J. Mach. Learn. Res.* **12** 2825–30
[38] Henze D A, Harris K D, Borhegyi Z, Csicsvari J, Mamiya A, Hirase H, Sirota A and Buzsáki G 2009 Simultaneous intracellular and extracellular recordings from hippocampus region CA1 of anesthetized rats CRCNS.org. (https://doi.org/10.6080/K02Z13FP)
[39] Henze D A, Borhegyi Z, Csicsvari J, Mamiya A, Harris K D and Buzsáki G 2000 Intracellular features predicted by extracellular recordings in the hippocampus *in vivo J. Neurophysiol.* **84** 390–400
[40] Hybrid analysis for spike sorting, NCHybrid136 data set, https://github.com/klusta-team/hybrid_analysis
[41] Ester M, Kriegel H, Sander J and Xu X 1996 A density-based algorithm for discovering clusters in large spatial databases with noise *Computer* **96** 226–31
[42] Moon T K 1996 The expectation-maximization algorithm *IEEE Signal Processing* **13** 47–60
[43] Wave_Clus spike sorting (https://www2.le.ac.uk/centres/csn/research-2/spike-sorting)
[44] Sasaki Y 2007 The truth of the F-measure *Teach. Tutor Mater.* **1** 1–5
[45] Powers D M W 2011 Evaluation: from precision, recall and f-measure to roc, informedness, markedness & correlation *J. Mach. Learn. Technol.* **2** 37–63