Turkish Journal of Electrical Engineering and Computer Sciences

Volume 25 | Number 4 Article 3

1-1-2017

Taller Peaks: an improved spike detection algorithm that simultaneously reduces type I and type II errors for Wave_clus

MURAT OKATAN

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 2583 – 2594

doi:10.3906/elk-1606-400

Taller Peaks: an improved spike detection algorithm that simultaneously reduces type I and type II errors for Wave_clus

Murat OKATAN*
Department of Biomedical Engineering, Faculty of Technology, Cumhuriyet University, Sivas, Turkey

Received: 28.06.2016 • **Accepted/Published Online:** 20.09.2016 • **Final Version:** 30.07.2017

Abstract: Wave_clus is an unsupervised spike detection and sorting algorithm that has been used in dozens of experimental studies as a spike sorting tool. It is often used as a benchmark for comparing the performance of new spike sorting algorithms. For these reasons, the spike detection performance of Wave_clus is important for both experimental and computational studies that involve spike sorting. Two measures of spike detection performance are the number of false positive detections (type I error) and the number of missed spikes (type II error). Here, a new spike detection algorithm is proposed that reduces the number of misses and false positives of Wave_clus in a widely used simulated data set across the entire range of commonly used detection thresholds. The algorithm accepts a spike if its amplitude is larger than the amplitude of its two immediate neighbors, where an immediate neighbor is the nearest peak of the same polarity within ± 1 refractory period. The simultaneous reduction that is achieved in the number of false positives and misses is important for experimental and computational studies that use Wave_clus as a spike sorting tool or as a benchmark. A software patch that incorporates the algorithm into Wave_clus as an optional spike detection algorithm is provided.

Key words: Biomedical signal processing, spike sorting, spike train

1. Introduction

In vivo extracellular recordings of neuronal activity provide a wealth of information about the information processing that takes place in biological neural networks [1]. Spiking activity of individual neurons is extracted from such recordings through a process called spike sorting [2]. A variety of tools have been developed for this purpose, such as the Offline Sorter (Plexon Inc., Dallas, TX, USA), SpikeSort 3D (Neuralynx, Inc., Bozeman, MT, USA), KlustaKwik [3], Spike2 (Cambridge Electronic Design Limited, Cambridge, UK), Wave_clus [4], MClust (A David Redish, University of Minnesota, Minneapolis, MN, USA), OpenSorter and OpenEx (Tucker Davis Technologies Inc., Alachua, FL, USA), and AutoSort (DataWave Technologies, Loveland, CO, USA) to name a few. These tools perform spike detection using a variety of algorithms, including thresholding signal energy functions and amplitude thresholding. For spike sorting, these tools allow the user to perform manual, semiautomatic, or unsupervised sorting using algorithms such as template matching, k-means, expectation maximization, valley seeking, and superparamagnetic clustering. The clustering can be performed in a variety of feature spaces spanned by features such as peak or valley amplitude, principal components, or wavelet coefficients. In addition to these commercial or widely used tools, new algorithms for performing spike sorting continue to be developed [5–12].

*Correspondence: muratokatan@cumhuriyet.edu.tr

Among this wide collection of tools and algorithms, Wave_clus has been used as an open source spike sorting tool in dozens of experimental [13–19] and methodological studies [20], and also as a benchmark for new spike sorting algorithms [7,8,11,12,21]. Considered as a state-of-the-art spike sorting algorithm [11], Wave_clus has been reported to be the second most highly cited spike sorting algorithm in the literature [21].

A simulated data set introduced by Quiroga et al. [4] has been used in several studies as a basis for comparing the performance of various spike sorting algorithms [5–11,21,22]. Here, this data set is used for examining how Wave_clus' spike detection algorithm misses spikes or detects false positives. It is discovered that Wave_clus treats certain identifiable nonspikes as valid spikes (false positives) and that any subsequent actual spikes that happen to be within a refractory period of these false positives are ignored by the algorithm (misses). The characteristics of the identifiable false positives are explained and an algorithm that successfully rejects them is presented. The proposed algorithm and the spike detection algorithm of Wave_clus are compared in terms of their spike detection performance. A software patch that integrates the code of the proposed algorithm into the Wave_clus software as an optional spike detection algorithm is provided at scicrunch.org under RRID: SCR_014652. In this way, the proposed algorithm can be readily integrated into the original Wave_clus software to improve the latter's spike detection performance.

Because Wave_clus is widely used as a spike sorting tool in experimental studies and as a benchmark in computational studies, improving its spike detection performance is expected to improve the accuracy of the experimental data analyses that are performed using Wave_clus and raise the bar for alternative methods that are tested against it.

2. Materials and methods

This section explains the proposed algorithm and the spike detection algorithm of Wave_clus, along with the method that is used for measuring the performance of these algorithms. All computations were performed in MATLAB (R2015a, MathWorks, Inc., USA) under a 64-bit Windows 8.1 Single Language (2013) operating system on a laptop computer with 6 GB RAM and 2.60 GHz Intel Core i5-3230M CPU.

2.1. Data set used

The spike detection performance of the algorithms considered here is assessed using a simulated extracellular recording data set in which the occurrence time of spikes is known. These data were generated by Quiroga et al. using a database of 594 average action potential waveforms of different shapes, collected from the cortex and the basal ganglia [4]. The simulated recordings consist of spike trains superimposed on background noise. The latter is generated through the superposition of action potential waveforms that are randomly selected from the database, where the superposition is performed with randomized amplitudes and occurrence times. The spiking activity in each simulated recording is generated by selecting three different action potential waveforms from the database and adding each to the background noise as Poisson events with a frequency of 20 Hz. Thus, each simulated recording contains the simultaneous spiking activity of three distinct neurons, with an overall firing rate of about 60 Hz. A 2-ms refractory period is enforced for the spike train of each distinct waveform. The spike amplitudes in each simulated recording are set to 1 unit. For each triplet of action potential waveforms, four different simulated recordings are generated by setting the standard deviation of the background noise to 0.05, 0.1, 0.15, and 0.2 units, respectively. Four different waveform triplets are used, resulting in a total of 16 simulated extracellular recordings. The sampling rate and the duration of each simulated recording are 24 kHz and 60 s. These data were downloaded along with the Wave_clus software package from www.vis.caltech.edu/∼rodri [4].

2.2. Spike detection algorithm of Wave_clus

Wave_clus is a computer program for detecting spike waveforms found in extracellular neural recordings and sorting them using the superparamagnetic clustering method [4]. The scope of the present study is limited to the presorting spike detection performance of Wave_clus. Type I and type II errors of presorting spike detection affect the accuracy of the subsequent spike sorting analysis.

In analyzing the spike content of extracellular recording data, which is represented by the variable x in Wave_clus, two separate bandpass-filtered versions of the recording are generated by Wave_clus. One filtered signal, *xf_detect*, is used for spike detection, whereas the other, *xf*, is used for spike sorting. Spike timestamps are determined in *xf_detect*, and the waveforms are extracted from *xf* at those timestamps for subsequent sorting. The detection threshold is computed using the robust median estimator of the standard deviation [4,23]:

$$\hat{\sigma} = median\left(|xf\_detect|\right)\left(\Phi^{-1}\left(0.75\right)\right)^{-1} \tag{1}$$

Here, $\Phi^{-1}(\cdot)$ is the inverse of the standard normal cumulative distribution. The spike detection threshold, *thr*, is then determined using Eq. (2), where *stdmin* is a user-defined parameter, which is set to 4 [4].

$$thr = stdmin \times \hat{\sigma} \tag{2}$$

Wave_clus provides the user with the choice of using a positive (*thr*) or negative (*–thr*) spike detection threshold, or both at the same time. The results of Quiroga et al. that are considered here were obtained using the positive threshold (see the first table in [4]). Whereas the simultaneous use of positive and negative thresholds reduces the number of misses by about a factor of two, it increases the number of false positives by about a factor of eight in the present data set (results not shown). Perhaps because of this reason, either the positive or the negative threshold alone is used in virtually all studies that use Wave_clus for spike sorting [6,13,24]. Because a positive threshold may be used instead of a negative one simply by using the negative of x as the input data, the algorithms considered here will be explained under the assumption that the positive threshold is used for spike detection.

Wave_clus starts by finding all samples in *xf_detect* that exceed *thr* and then processes each suprathreshold sample in the order of increasing timestamp to perform the following operations:

1. If the sample is within the refractory period of the last spike, then continue to the next sample.
2. Otherwise, find the timestamp of the peak value of *xf* in the interval that starts with the current sample and that lasts half a refractory period.
3. Accept the peak found in Step 2 as a spike and go to Step 1 to process the next suprathreshold sample.

Initially the time of the last spike is taken as 0. After all spikes are found using the algorithm described above, spikes that exceed a threshold called *thrmax* are considered to be noise and rejected:

$$thrmax = stdmax \times \hat{\sigma} \tag{3}$$

Here, *stdmax* is another user-defined parameter. The results of Quiroga et al. [4] could be replicated here by setting the value of this parameter to 27 and the refractory period to 2 ms, or 48 samples.

2.3. Proposed spike detection algorithm: the Taller Peaks algorithm

This section describes the operation of the proposed algorithm by citing the associated code lines in the relevant MATLAB .m files. The proposed algorithm starts by finding all local peaks in *xf* that exceed *thr* in *xf_detect* (line 15 in taller_peaks.m). These peaks form the set of candidate spikes. Next, for each peak, the temporally next peak is rejected if it occurs within a refractory period of the current peak and its amplitude is smaller (lines 22–29 in taller_peaks.m). Finally, for each remaining peak, the temporally previous peak is rejected if it occurs within a refractory period of the current peak and its amplitude is smaller (lines 32–40 in taller_peaks.m). Because the taller peaks survive this pruning, this algorithm is henceforth referred to for convenience as the Taller Peaks algorithm. The surviving peaks that exceed *thrmax* are considered noise and rejected by Wave_clus (performed in Wave_clus' amp_detect.m after the insertion point of taller_peaks.m). The remaining peaks are accepted as spikes.

2.4. Performance measurement method

This section explains the performance measurement method that is used for counting the number of missed spikes and false positive detections. The timestamps of the spikes that are contained in the simulated extracellular recordings are found in the variable *spike_times* in each data file. The cross-correlogram of the spike trains associated with these timestamps and the event trains detected by Wave_clus reveal that the detected events tend to lag behind the actual spikes. Based on this observation, it is assumed that the spike with timestamp n is missed by Wave_clus if no event is detected in an interval $I^+(n) = [n+a,\ n+b]$, $0 < a < b$. Otherwise the spike is not considered missed, and all of the detected events that fall in $I^+(n)$ are labeled so that they are not matched with any other spike. The results of Quiroga et al. [4] could be replicated here by using $a = 12$ samples (0.5 ms) and $b = 42$ samples (1.75 ms), where the values of a and b have been determined by trial and error. Similarly, it is assumed that the detected event with timestamp n is a false positive if no spike timestamp exists in the interval $I^-(n) = [n\text{-}b,\ n\text{-}a]$, using the same values as above for a and b. Otherwise the event is not considered a false positive, and all the spikes that fall in $I^-(n)$ are labeled so that they are not matched with any other event.

The number of missed spikes and false positives is determined only for nonoverlapping spikes for clarity. The index for nonoverlapping spikes is found in the variable *spike_class* in each data file.

The same criteria are used for measuring the spike detection performance of both algorithms.

2.5. Performance as a function of detection threshold

Spike detection thresholds are usually set to a value between $3\hat{\sigma}$ and $5\hat{\sigma}$ [4,12,25]. Here, the multiplier of $\hat{\sigma}$ is selected depending on the researcher's preferences. Recently a pair of spike detection thresholds, called truncation thresholds, were introduced and these thresholds are computed using an entirely data-driven method that does not depend on the researcher's preferences [26]. Briefly, truncation thresholds are a pair of voltage values that are defined on the basis that they are as far away from each other as possible and that the samples that are contained between them obey a truncated probability distribution according to the Kolmogorov–Smirnov test. Truncation thresholds have been computed for the data files used here under the truncated normal distribution as the noise model, and the average positive truncation threshold is found to be approximately $1.78\hat{\sigma}$. Therefore, to compare the spike detection performance of the two algorithms at different detection thresholds, including the positive truncation threshold, the numbers of false positive detections and missed spikes are determined at thresholds between $\hat{\sigma}$ and $5\hat{\sigma}$ with steps of $\hat{\sigma}/10$, where $\hat{\sigma}$ is defined in Eq. (1). The

total numbers of false positives and missed spikes are obtained separately for Wave_clus and the Taller Peaks algorithm. The percentage decrease achieved by using the Taller Peaks algorithm instead of Wave_clus' native spike detection algorithm is plotted separately for false positives and missed spikes using Eq. (4).

$$Reduction\,(\%) = \frac{Num_{Wave\_clus} - Num_{Taller\,Peaks}}{Num_{Wave\_clus}} \times 100 \tag{4}$$

Here, Num_{Wave_clus} and $Num_{Taller\,Peaks}$ denote the total number of missed spikes or false positives for the indicated algorithm at a given threshold across all 16 data files.

2.6. Difference in computation time

To determine whether the two algorithms differ in terms of computation time, the average time it takes for the algorithms to process the data files used here is measured using the tic and toc functions of MATLAB. This analysis is performed at all thresholds considered here.

2.7. Performance as a function of refractory period duration

The analyses explained in Sections 2.5 and 2.6 are performed for refractory period durations of 2.5 ms, 50/24 ms, 2 ms, 46/24 ms, 1.5 ms, 1 ms, and 0.5 ms, where the values 50/24 ms and 46/24 ms are used to probe the change in performance in the immediate vicinity of 2 ms.

3. Results

The comparison of the two algorithms starts by graphically examining some of the waveforms that are falsely classified as spikes by Wave_clus (Figures 1A and 1B). These waveforms also illustrate how Wave_clus misses subsequent spikes that occur within the refractory period. It is explained that the Taller Peaks algorithm successfully rejects those waveforms and claims the missed spikes. Next, Wave_clus and the Taller Peaks algorithm are compared in terms of their spike detection performance (Table; Figures 2A and 2B). The two algorithms are also compared in terms of their computation time (Figure 2C).

3.1. Valid spikes that are missed due to preventable false positives

Figures 1A and 1B show two instances of how a preventable false positive causes a subsequent spike to be missed. False positives may be due to noise (Figure 1A) or suprathreshold parts of action potential waveforms (Figure 1B). In either case, Wave_clus ignores any spike that occurs within the next refractory period (2 ms in Figures 1A and 1B). This may also cause subsequent spikes to be missed, as shown in Figures 1A and 1B. Note that the amplitude of the false positives shown in Figures 1A and 1B is smaller than the amplitude of their immediate neighbor(s), where an immediate neighbor is defined here as the nearest peak of the same polarity that is within ± 1 refractory period. The Taller Peaks algorithm rejects peaks whose amplitude is smaller than the amplitude of their immediate neighbors. As a result, false positives such as those shown in Figures 1A and 1B are prevented and smaller numbers of false positives and missed spikes are achieved by the Taller Peaks algorithm (Table; Figures 2A and 2B).

3.2. Comparing the spike detection performance of the two algorithms

Wave_clus and the Taller Peaks algorithm are first compared in terms of their spike detection performance using the method described in Section 2.4. In this analysis, the spike detection threshold is set to $4\,\hat{\sigma}$ and the

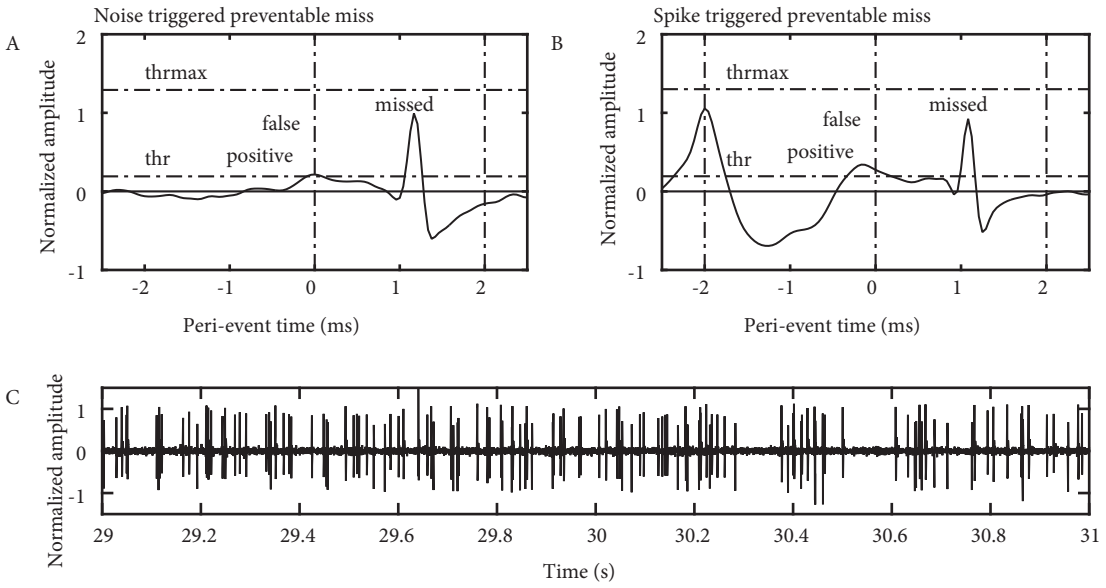

Figure 1. Spikes missed due to preventable false positives. (A) Noise fluctuation that exceeds the detection threshold (thr) is detected as a spike by Wave_clus in file C_Difficult1_noise005.mat at sample number 1201358 (false positive). Wave_clus ignores any spikes that occur within the next refractory period, which is 2 ms in this figure (Step 1 in the algorithm). This causes an action potential that happens to occur in that time window to be missed (missed). The Taller Peaks algorithm eliminates the false positive since its amplitude is smaller than its immediate neighbor's. The true spike survives the pruning and is detected by the Taller Peaks algorithm. (B) A postaction potential rebound is detected as a spike by Wave_clus in file C_Easy1_noise005.mat at sample number 1438158 (false positive). An action potential that happens to occur within the next refractory period is ignored by the algorithm (missed). The Taller Peaks algorithm eliminates the false positive since its amplitude is smaller than its immediate neighbor's. The true spikes survive the pruning and are detected by the Taller Peaks algorithm. Note that the false positive is detected by Wave_clus as soon as the 2 ms refractory period triggered by the preceding spike expires (Steps 1 and 2 of Wave_clus algorithm). (C) The central 2 s window showing xf for the file C_Easy1_noise005.mat is provided here to illustrate an example of a spike train waveform.

duration of the refractory period is set to 2 ms. The number of false positives and missed spikes is shown in the Table for Wave_clus. The software that generates these results from the underlying publicly available data set is available at scicrunch.org under RRID: SCR_014652. These numbers are equal to or slightly lower than the results reported by Quiroga et al. [4].

The same performance measurement method is used for determining the number of false positive detections and missed spikes for the Taller Peaks algorithm. Compared to Wave_clus, the Taller Peaks algorithm misses fewer spikes in five cases (Table). It misses more spikes than Wave_clus in only one case (Example 3 [0.20]), and the difference is only 1 spike. Overall, the Taller Peaks algorithm misses about 1.22% fewer spikes than Wave_clus.

The performance increase achieved by the Taller Peaks algorithm in the number of false positives is more substantial. As shown in the Table, compared to Wave_clus, the Taller Peaks algorithm detects substantially fewer false positives in eight cases. It detects more false positives than Wave_clus in only one case (Example 3 [0.20]), and the difference is only 1 spike. Overall, the Taller Peaks algorithm detects 64.56% fewer false positives than Wave_clus.

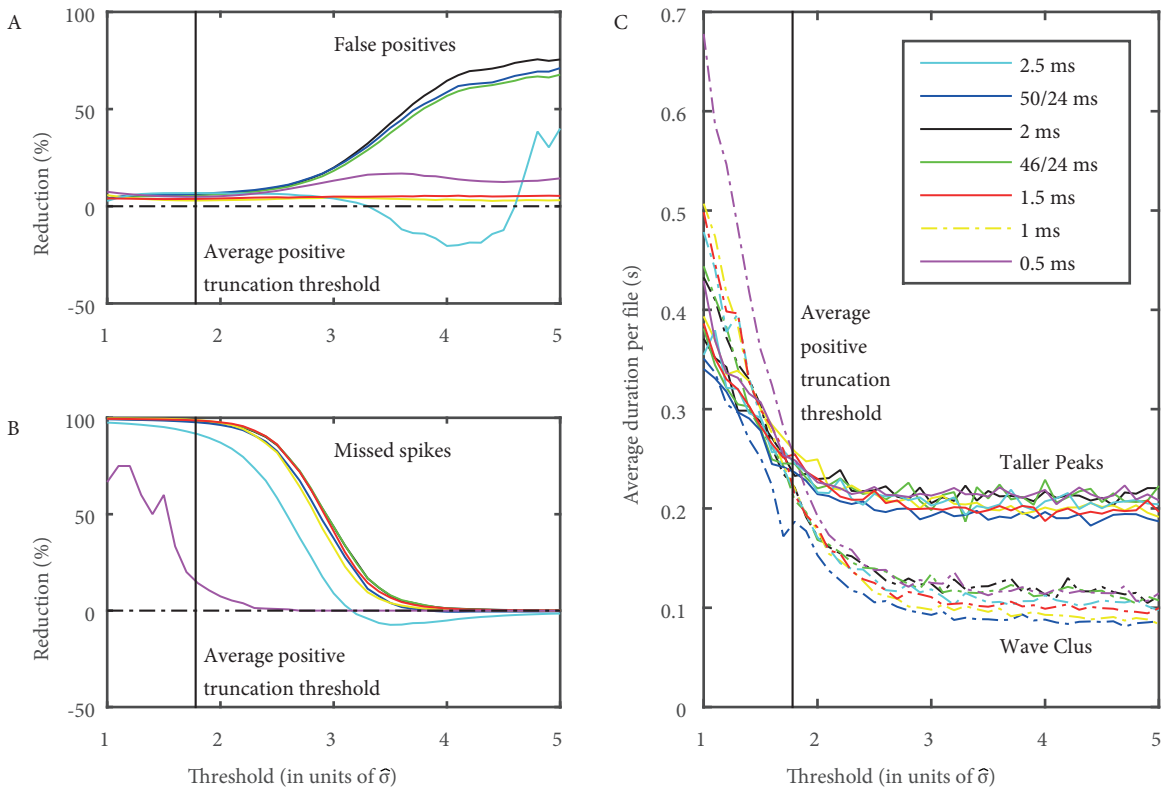

Figure 2. Dependence of performance improvement and computation time on threshold and refractory period. (A) The Taller Peaks algorithm detects fewer false positives than Wave_clus at all thresholds tested, except for a refractory period of 2.5 ms, where it detects more false positives for a range of threshold values between about $3\hat{\sigma}$ and $4.5\hat{\sigma}$. Performance improvement is highest for refractory periods that are near 2 ms. (B) The Taller Peaks algorithm misses fewer spikes than Wave_clus at all thresholds tested when the refractory period is smaller than or equal to 2 ms. Performance improvement is highest for refractory periods that are near 1–2 ms. For refractory periods that exceed 2 ms, the performance of the Taller Peaks algorithm is lower than that of Wave_clus' native spike detection algorithm at high thresholds. (A, B) The curves are obtained using Eq. (4). The vertical line indicates the average positive truncation threshold across all data files. (C) Below the average positive truncation threshold, both algorithms take approximately the same time to complete spike detection. The computation times quickly diverge above the average positive truncation threshold: the Taller Peaks algorithm takes a longer time than the spike detection algorithm of Wave_clus. The durations tend to decrease with increasing threshold, since there are fewer suprathreshold peaks and samples at higher thresholds. The same color code applies to parts (A), (B), and (C).

The Taller Peaks algorithm misses one more spike than Wave_clus in 'Example 3 [0.20]', because the missed spike is not taller than its immediate neighbor (not shown). The latter, which is due to noise, survives the pruning performed by the Taller Peaks algorithm and constitutes the single additional false positive that is detected by this algorithm in the same case.

3.3. Dependence of spike detection performance on detection threshold and refractory period duration

By performing the analysis in Section 3.2 for thresholds between $\hat{\sigma}$ and $5\hat{\sigma}$, as explained in Sections 2.5 and 2.7, the curves in Figures 2A and 2B are obtained. Figures 2A and 2B show that the performance improvement

Table. Comparison of the spike detection performance of Wave_clus and the Taller Peaks algorithm. The cases where the Taller Peaks algorithm performs better than Wave_clus are shown in boldface. The case where it performs poorer is shown with thick borders.

Example number (noise level)	Wave_clus		Taller Peaks	
	Misses	False positives	Misses	False positives
Example 1 [0.05]	16	705	**0**	**172**
[0.10]	2	56	**0**	**30**
[0.15]	145	14	145	**11**
[0.20]	714	10	**712**	**8**
Example 2 [0.05]	0	0	0	0
[0.10]	0	2	0	2
[0.15]	10	1	10	1
[0.20]	376	5	376	5
Example 3 [0.05]	1	63	**0**	**59**
[0.10]	0	10	0	**8**
[0.15]	8	6	**7**	**4**
[0.20]	184	2	**185**	**3**
Example 4 [0.05]	0	1	0	1
[0.10]	0	5	0	**4**
[0.15]	3	4	3	4
[0.20]	262	2	262	2
Total	1721	886	**1700**	**314**

achieved by the Taller Peaks algorithm depends on both the spike detection threshold and the refractory period duration used. The performance improvement is maximal at a refractory period duration of 2 ms. The Taller Peaks algorithm performs better than Wave_clus at all thresholds tested if the refractory period does not exceed 2 ms. At low thresholds, Wave_clus misses a lot more spikes than the Taller Peaks algorithm. For a refractory period of 2 ms, Wave_clus misses about 22% of all spikes at the lowest threshold tested, and virtually every spike that is missed by Wave_clus is captured by the Taller Peaks algorithm. At the same time, the Taller Peaks algorithm detects fewer false positives. When the positive truncation threshold is used as the detection threshold, using the Taller Peaks algorithm in place of Wave_clus' native spike detection algorithm reduces the number of missed spikes and false positives by about 98% (refractory periods near 1–2 ms) and 6% (all refractory periods tested), respectively. At the highest thresholds tested, both algorithms miss about the same number of spikes, yet the Taller Peaks algorithm detects fewer false positives. For the unusually long refractory period duration of 2.5 ms, the Taller Peaks algorithm detects more false positives for threshold values between about $3\hat{\sigma}$ and $4.5\hat{\sigma}$, and it misses more spikes for threshold values above about $3\hat{\sigma}$.

3.4. Computation time

To determine the computational cost of the Taller Peaks algorithm's higher spike detection performance, the average time it takes to process the data files considered here is measured for each algorithm at all thresholds and refractory periods tested (Figure 2C). Interestingly, the computation times of the two algorithms are about the same when the threshold is at or below the average positive truncation threshold, but the computation times quickly diverge above the average positive truncation threshold. For both algorithms, the computation time tends to decrease with increasing threshold. At high thresholds, the Taller Peaks algorithm takes about twice the time Wave_clus takes to detect spikes.

4. Discussion

Spike sorting is an indispensable step in the investigation of the information encoded in the spike trains of individual neurons when such activity is recorded extracellularly. For this reason, spike sorting has been and continues to be a central problem in computational neuroscience. Freely available spike sorting software and data sets, such as Wave_clus and its associated simulated data set, are valuable resources that help the field of computational neuroscience move forward [27].

The present study used Wave_clus and the data set generated by Quiroga et al. [4] to address the question of how Wave_clus misses spikes or detects false positives. The answer to this question led to an improvement of Wave_clus' spike detection algorithm. It is found that Wave_clus treats some preventable false positive events as spikes and that this also causes some spikes to be missed (Figures 1A and 1B). The improvement of the spike detection algorithm is therefore directly based on rejecting those identifiable false positives. The latter are identifiable because their amplitude is smaller than the amplitude of their immediate neighbor (Figures 1A and 1B). Therefore, the proposed improvement to the spike detection algorithm of Wave_clus has been to reject a spike candidate if its amplitude is smaller than the amplitude of one of its immediate neighbors. Because taller peaks survive this pruning, the proposed algorithm is called the Taller Peaks algorithm. If a false positive and one or both of its immediate neighbors have the same amplitude, then the algorithm cannot discriminate between them based on their amplitudes and they survive the pruning. In the intact Wave_clus, however, a false positive causes a subsequent immediate neighbor to be deleted regardless of the amplitude comparison, thereby resulting in a higher number of missed spikes.

The Taller Peaks algorithm outperforms Wave_clus over the entire range of spike detection thresholds that are used in the literature and for refractory periods not exceeding 2 ms (Table; Figures 2A and 2B). Optimal performance improvement is obtained for a refractory period of 2 ms (Figures 2A and 2B). This is consistent with the fact that the simulated spike trains analyzed here were generated with a refractory period of 2 ms. Because the simulated data that are used here are quite realistic in terms of the noise and the spike waveforms they contain, the Taller Peaks algorithm is expected to outperform Wave_clus in real data analysis situations as well. The software that is made available here makes it possible to switch between the Taller Peaks algorithm and Wave_clus' native spike detection algorithm to assess the differential impact of using one algorithm versus the other for any data. One real data analysis application that could be used to assess the performance of the algorithm is spike train decoding [14,25]. If the Taller Peaks algorithm reduces both type I and type II errors of Wave_clus in real data, then quantities that are decoded using spikes sorted with the Taller Peaks algorithm would be expected to be more accurate. This could be tested by replicating an experimental study with and without turning the Taller Peaks algorithm on, and then comparing the results.

The Taller Peaks algorithm misses substantially fewer spikes and detects a lot fewer false positives than Wave_clus in the Example 1 data set compared to other data sets (Table). The reason for this is that a spike waveform that is present in that data set has a suprathreshold postaction potential rebound, which is shown in Figure 1B. Those rebounds trigger false positives and missed spikes, as shown in the same figure. Because these are successfully prevented by the Taller Peaks algorithm, the performance improvement is substantially higher in the Example 1 data set. Spike waveforms with such rebounds are not found in the other data sets considered here.

An important question concerning the performance of the Taller Peaks algorithm is whether it can detect spikes in bursts. Because the present data set consists of Poisson spike trains, this issue could not be directly addressed in the present analysis. However, both Wave_clus' intact spike detection algorithm and the Taller Peaks algorithm should detect bursting spikes as long as the refractory period is set such that consecutive spikes

in a burst do not become immediate neighbors. Since even the intraburst interspike intervals are not expected to be shorter than 1–2 ms, the performance improvement achieved by the Taller Peaks algorithm would be expected to apply to bursting spike detection as well (Figures 2A and 2B). Setting the refractory period to even lower values, however, would decrease the performance improvement, as shown by the curves that are obtained for a refractory period of 0.5 ms in Figures 2A and 2B.

Unlike the biophysical spiking threshold, which depends on such activity characteristics as the rate of depolarization, firing rate, and short-term synaptic depression [28–30], the spike detection threshold used in amplitude thresholding is a fixed threshold. Making the latter depend on local activity characteristics of extracellular recording data may further improve spike detection accuracy.

When the data are grouped per noise level, the Taller Peaks algorithm is found to miss fewer spikes and detect fewer false positives than Wave_clus in each group (Table). The Taller Peaks algorithm missed one more spike than Wave_clus in only one instance (Example 3 [0.2]), and that was due to a noise-triggered immediate neighbor whose amplitude was larger than the amplitude of the missed spike. Because this was in one of the cases with the highest noise level, this suggests that the performance of the Taller Peaks algorithm may be further improved at high noise levels by considering the waveform features of the immediate neighbors before deciding to reject a suprathreshold event as noise.

The improved spike detection performance of the Taller Peaks algorithm comes at a computational cost that depends on the spike detection threshold, which is reasonable since the latter controls the amount of suprathreshold data processed. Below or at the average positive truncation threshold, the Taller Peaks algorithm takes about as much time as Wave_clus for spike detection (Figure 2C), but it misses at least 98% fewer spikes (for refractory periods near 1–2 ms) and detects approximately 4%–6% fewer false positives than Wave_clus in that threshold range (for all refractory periods tested) (Figures 2A and 2B). Samples that are below the average positive truncation threshold can safely be assumed to represent noise since they obey the noise distribution according to the Kolmogorov–Smirnov test ($P \geq 0.05$) [26]. Therefore, a detection threshold that is below the average positive truncation threshold is expected to be of little practical use. Above the average positive truncation threshold, the computation times quickly diverge with increasing threshold (Figure 2C). There is no apparent reason why the computation times should sharply differ below and above the positive truncation threshold, other than the explanation that truncation thresholds serve as effective boundaries between the signal and the noise in these recordings [26].

Because samples that exceed the positive truncation threshold are not likely to reflect noise, using the positive truncation threshold as the spike detection threshold may maximize the amount of spike information extracted from extracellular recordings. At that threshold value, the number of spikes missed by the Taller Peaks algorithm is only about 2% of the number missed by Wave_clus (Figure 2B; for refractory periods near 1–2 ms) at no additional computational cost (Figure 2C).

A spike validation method proposed in an earlier study bears similarities to the Taller Peaks algorithm [31] and was used in combination with Wave_clus to prevent duplicate detection of multiphasic spikes [32]. However, that algorithm has more rules than the Taller Peaks algorithm [31], and a quantitative analysis of its spike detection performance does not seem to have been documented in the literature.

Overall, the results provided here suggest that the Taller Peaks algorithm is likely to increase the spike detection and classification performance of Wave_clus in real data analysis applications. This improvement is expected to benefit both experimental and computational studies that use Wave_clus as a data analysis tool or benchmark.

Acknowledgment

This work was supported by the Scientific Research Project Fund of Cumhuriyet University (Project No. TEKNO-002).

References

[1] Buzsáki G. Large-scale recording of neuronal ensembles. Nat Neurosci 2004; 7: 446-451.

[2] Lewicki MS. A review of methods for spike sorting: the detection and classification of neural action potentials. Network 1998; 9: R53-R78.

[3] Harris KD, Henze DA, Csicsvari J, Hirase H, Buzsáki G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. J Neurophysiol 2000; 84: 401-414.

[4] Quiroga RQ, Nadasdy Z, Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural Comput 2004; 16: 1661-1687.

[5] Paraskevopoulou SE, Wu D, Eftekhar A, Constandinou TG. Hierarchical adaptive means (HAM) clustering for hardware-efficient, unsupervised and real-time spike sorting. J Neurosci Meth 2014; 235: 145-156.

[6] Nguyen T, Khosravi A, Creighton D, Nahavandi S. Spike sorting using locality preserving projection with gap statistics and landmark-based spectral clustering. J Neurosci Meth 2014; 238: 43-53.

[7] Adamos DA, Laskaris NA, Kosmidis EK, Theophilidis G. NASS: An empirical approach to spike sorting with overlap resolution based on a hybrid noise-assisted methodology. J Neurosci Meth 2010; 190: 129-142.

[8] Yuan Y, Yang C, Si J. The M-Sorter: an automatic and robust spike detection and classification system. J Neurosci Meth 2012; 210: 281-290.

[9] Dai M, Hu S, Qi J. Spike sorting based on radial basis function network with overlap decomposition. Comput Math Appl 2011; 62: 2736-2742.

[10] Yang Z, Zhao Q, Liu W. Neural signal classification using a simplified feature set with nonparametric clustering. Neurocomputing 2009; 73: 412-422.

[11] Herbst JA, Gammeter S, Ferrero D, Hahnloser RHR. Spike sorting with hidden Markov models. J Neurosci Meth 2008; 174: 126-134.

[12] Vargas-Irwin C, Donoghue JP. Automated spike sorting using density grid contour clustering and subtractive waveform decomposition. J Neurosci Meth 2007; 164: 1-18.

[13] Whitmire CJ, Waiblinger C, Schwarz C, Stanley GB. Information coding through adaptive gating of synchronized thalamic bursting. Cell Rep 2016; 14: 795-807.

[14] Ossmy O, Fried I, Mukamel R. Decoding speech perception from single cell activity in humans. NeuroImage 2015; 117: 151-159.

[15] Huang L, Liu Y, Li M, Hu D. Hemodynamic and electrophysiological spontaneous low-frequency oscillations in the cortex: directional influences revealed by Granger causality. NeuroImage 2014; 85: 810-822.

[16] Murugan M, Harward S, Scharff C, Mooney R. Diminished FoxP2 levels affect dopaminergic modulation of corticostriatal signaling important to song variability. Neuron 2013; 80: 1464-1476.

[17] Pagano RL, Fonoff ET, Dale CS, Ballester G, Teixeira MJ, Britto LRG. Motor cortex stimulation inhibits thalamic sensory neurons and enhances activity of PAG neurons: possible pathways for antinociception. Pain 2012; 153: 2359-2369.

[18] Lim BK, Cho S, Sumbre G, Poo M. Region-specific contribution of ephrin-B and Wnt signaling to receptive field plasticity in developing optic tectum. Neuron 2010; 65: 899-911.

[19] Kraskov A, Dancause N, Quallo MM, Shepherd S, Lemon RN. Corticospinal neurons in macaque ventral premotor cortex with mirror properties: a potential mechanism for action suppression? Neuron 2009; 64: 922-930.

[20] Mahmud M, Bertoldo A, Girardi S, Maschietto M, Vassanelli S. SigMate: a Matlab-based automated tool for extracellular neuronal signal processing and analysis. J Neurosci Meth 2012; 207: 97-112.

[21] Wild J, Prekopcsak Z, Sieger T, Novak D, Jech R. Performance comparison of extracellular spike sorting algorithms for single-channel recordings. J Neurosci Meth 2012; 203: 369-376.

[22] Kapucu FE, Mäkinen ME, Tanskanen JMA, Ylä-Outinen L, Narkilahti S, Hyttinen JAK. Joint analysis of extracellular spike waveforms and neuronal network bursts. J Neurosci Meth 2016; 259: 143-155.

[23] Donoho D, Johnstone IM. Ideal spatial adaptation by wavelet shrinkage. Biometrika 1994; 81: 425-455.

[24] Rey HG, Pedreira C, Quiroga RQ. Past, present and future of spike sorting techniques. Brain Res Bull 2015; 119: 106-117.

[25] Todorova S, Sadtler P, Batista A, Chase S, Ventura V. To sort or not to sort: the impact of spike-sorting on neural decoding performance. J Neural Eng 2014; 11: 056005.

[26] Okatan M, Kocatürk M. Truncation thresholds: a pair of spike detection thresholds computed using truncated probability distributions. Turk J Elec Eng & Comp Sci 2017; 25: 1436-1447.

[27] Smith LS. Why sharing matters for electrophysiological data analysis. Brain Res Bull 2015; 119: 145-149.

[28] Azouz R, Gray CM. Dynamic spike threshold reveals a mechanism for synaptic coincidence detection in cortical neurons in vivo. P Natl Acad Sci USA 2000; 97: 8110-8115.

[29] Ozer M, Uzuntarla M, Perc M, Graham LJ. Spike latency and jitter of neuronal membrane patches with stochastic Hodgkin-Huxley channels. J Theor Biol 2009; 261: 83-92.

[30] Uzuntarla M, Ozer M, Ileri U, Calim A, Torres JJ. Effects of dynamic synapses on noise-delayed response latency of a single neuron. Phys Rev E 2015; 92: 062710.

[31] Wagenaar D, DeMarse TB, Potter SM. MeaBench: a toolset for multi-electrode data acquisition and on-line analysis. In: IEEE EMBS 2005 International Conference on Neural Engineering; 16–19 March 2005; Washington, DC, USA. New York, NY, USA: IEEE. Pp. 518-521.

[32] Mok SY, Nadasdy Z, Lim YM, Goh SY. Ultra-slow oscillations in cortical networks in vitro. Neuroscience 2012; 206: 17-24.