

UCSF

UC San Francisco Previously Published Works

Title

A Fully Automated Approach to Spike Sorting

Permalink

<https://escholarship.org/uc/item/0cj4d194>

Journal

Neuron, 95(6)

ISSN

0896-6273

Authors

Chung, Jason E
Magland, Jeremy F
Barnett, Alex H
[et al.](#)

Publication Date

2017-09-01

DOI

10.1016/j.neuron.2017.08.030

Peer reviewed



HHS Public Access

Author manuscript

Neuron. Author manuscript; available in PMC 2018 September 13.

Published in final edited form as:

Neuron. 2017 September 13; 95(6): 1381–1394.e6. doi:10.1016/j.neuron.2017.08.030.

A fully automated approach to spike sorting

Jason E. Chung^{2,*}, Jeremy F. Magland^{1,*}, Alex H. Barnett^{1,3}, Vanessa M. Tolosa^{4,5}, Angela C. Tooker⁴, Kye Y. Lee⁴, Kedar G. Shah^{4,6}, Sarah H. Felix⁴, Loren M. Frank^{2,7,8}, and Leslie F. Greengard¹

¹Center for Computational Biology, Flatiron Institute, 162 Fifth Avenue, New York, NY 10010 and Courant Institute, NYU, New York, NY 10012, USA

²Neuroscience Graduate Program, Kavli Institute for Fundamental Neuroscience and Department of Physiology, University of California San Francisco, CA 94158, USA

³Department of Mathematics, Dartmouth College, Hanover, NH 03755, USA

⁴Center for Micro- and Nano-Technology, Lawrence Livermore National Laboratory, Livermore, CA 94550, USA

⁷Howard Hughes Medical Institute

Summary

Understanding the detailed dynamics of neuronal networks will require the simultaneous measurement of spike trains from hundreds of neurons (or more). Currently, approaches to extracting spike times and labels from raw data are time consuming, lack standardization and involve manual intervention, making it difficult to maintain data provenance and assess the quality of scientific results. Here, we describe an automated clustering approach and associated software package that addresses these problems and provides novel cluster quality metrics. We show that our approach has accuracy comparable to or exceeding that achieved using manual or semi-manual techniques with desktop CPU runtimes faster than acquisition time for up to hundreds of electrodes. Moreover, a single choice of parameters in the algorithm is effective for a variety of electrode geometries and across multiple brain regions. This algorithm has the potential to enable reproducible and automated spike sorting of larger scale recordings than is currently possible.

ETOC

Lead contact information: Loren M. Frank Ph.D., Professor, Howard Hughes Medical Institute, Kavli Institute for Fundamental Neuroscience, Department of Physiology – Box 0444, Sandler Neuroscience Building Rm 514F, San Francisco, California 94143, Tel: 415-502-7357, loren@phy.ucsf.edu.

⁵Current address: Neuralink Corp., San Francisco, CA 94107, USA.

⁶Current address: Verily Life Sciences, 269 E Grand Ave, South San Francisco, CA 94080, USA

⁸Lead Contact

*These authors contributed equally

Author Contributions

J.M., A.B. and L.G. developed the methodology. J.M. and A.B. wrote the software. J.C. collected and curated tetrode and polymer probe datasets. J.M., J.C., and A.B. analyzed the data. V.T., A.T., K.L., K.S., and S.F. developed and fabricated the polymer probes. J.M., J.C., A.B., L.F., and L.G. conceptualized the project and wrote the manuscript.

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Magland et. al. present MountainSort, a new fully automatic spike sorting package with a powerful GUI. MountainSort has accuracy comparable to current methods and runtimes faster than real-time, enabling automatic and reproducible spike sorting for high-density extracellular recordings.

Introduction

Advances in our understanding of how populations of neurons represent and process information has been enabled by tightly packed multi-electrode arrays that allow for isolation of large numbers of simultaneously recorded neurons (Csicsvari et al., 2003; Gray et al., 1995; Lopez et al., 2016; McNaughton et al., 1983). Data collected from these devices comprise multiple channels of continuously sampled extracellular voltages. A key step in making these data interpretable is spike sorting, the process of detecting spiking events and assigning those events to single units corresponding to putative individual neurons (Einevoll et al., 2012; Harris et al., 2016; Lewicki, 1998; Muller, 1996; Quiroga, 2012).

A number of challenges make spike sorting more complex than clustering in many other disciplines. First, there is no simple noise model. The background signal arises from the combinations of multiple complex signals, including small spikes from hundreds of distant neurons, and can contain electrical noise mixed with true neural signals (Einevoll et al., 2012). Second, variation in waveform shapes for a given cell can be highly non-Gaussian and skewed, particularly when bursting occurs (Harris et al., 2001; Quirk et al., 2001; Quirk and Wilson, 1999) or when the cell positions drift over time relative to the physical electrode (Bar-Hillel et al., 2006; Calabrese and Paninski, 2011; Chestek et al., 2007; Emondi et al., 2004). Multiple neurons may fire simultaneously, leading to time-overlapping spike signals, and while this may not occur frequently in some brain areas or with small electrode arrays, for large arrays sampling tens to hundreds of neurons, individual spikes will often time-overlap with other events (Ekanadham et al., 2014; Franke et al., 2010).

The majority of spike sorting algorithms comprise a sequence of steps such as bandpass filtering, spatial whitening, detection of threshold-crossing events, and clustering based on voltage waveform shape in a suitable feature space; this generally involves one or more manual processing steps (Einevoll et al., 2012; Lewicki, 1998; Marre et al., 2012; Quiroga, 2012). At one extreme, the clustering itself is performed by a human operator, viewing the event features in two-dimensional projections, and drawing cluster boundaries with the assistance of specialized user interfaces (Xclust, M.A. Wilson; MClust, A.D. Redish; Offline Sorter, Plexon). In other situations, clustering is automated, but the user must curate the results by selecting which clusters to reject, merge, or even split (Hill et al., 2011; Kadir et al., 2014; Rossant et al., 2016). There also exist post-processing steps that resolve overlapping spikes (Ekanadham et al., 2014; Franke et al., 2015; Pachitariu et al., 2016; Pillow et al., 2013), and algorithms based on independent component analysis (ICA) that do not explicitly involve clustering (Takahashi et al., 2002). Presently, despite many available packages and proposed algorithms, no generally adopted software packages offer fully automated sorting that can take in the raw time series data and output spike times and identities without the expectation of further curation.

From the standpoint of efficient and reproducible science, any human intervention has disadvantages. Manual sorting can have error rates in excess of 20% (Wood et al., 2004) and there is substantial variability in labeling across different sorting sessions (Harris et al., 2000; Pedreira et al., 2012; Rossant et al., 2016). Furthermore, the human spike sorter could never keep up with the increasing volume of data arising from increasingly large electrode arrays applied over increasingly long durations (Berenyi et al., 2014; Dhawale AK, 2015; Du et al., 2011; Lopez et al., 2016; Santhanam et al., 2007; Shobe et al., 2015).

Although fully-automated spike sorting has been of interest for many years (Abeles and Goldstein, 1977), and despite prior efforts to automate sorting algorithms (Calabrese and Paninski, 2011; Carlson et al., 2014; Einevoll et al., 2012; Ekanadham et al., 2014; Franke et al., 2010; Franke et al., 2015; Kadir et al., 2014; Lewicki, 1998; Marre et al., 2012; Pillow et al., 2013; Quiroga et al., 2004; Rodriguez and Laio, 2014; Swindale and Spacek, 2014; Takekawa et al., 2012), the majority of laboratories still rely heavily on manual intervention. In this work, we set out to develop a fully automated spike sorting algorithm having error rates that are comparable to or lower than those of existing manual and semi-manual approaches, and with runtimes faster than acquisition times. We introduce MountainSort, a novel spike sorting algorithm and open-source software suite of processing, visualization, and curation tools.

Results

A central goal of our overall spike sorting strategy has been to develop a single algorithm that can be applied to data from different brain regions without the need for brain-region specific models or parameters. This requires that the procedure be insensitive to differences in dataset properties such as non-Gaussian cluster distributions, electrode densities, and firing rates. Further, selecting parameters such as thresholds and regularization constants is time-consuming and can call into question the objectivity of results. Therefore, our guiding philosophy is to minimize both the number of user-defined parameters and the number of modeling assumptions, while maintaining high spike sorting accuracy and efficiency.

To satisfy these requirements we developed the spike sorting pipeline shown in Fig. 1A. This involves preprocessing, sorting on electrode neighborhoods, consolidation across those neighborhoods, fitting, derivation of cluster metrics, and automated annotation based on those metrics. The last step replaces manual curation (deciding on which clusters to accept), but importantly does not involve corrections to clustering as with other approaches. Details of all processing steps are provided in Methods. Here we give an overview of the most critical steps.

Clustering of neural events

At the heart of our sorting pipeline is a new, efficient, nonparametric, density-based clustering algorithm termed ISO-SPLIT (Fig. 1B), used to sort spike events based on their representations in a low-dimensional feature space. The algorithm makes only two general assumptions about cluster distributions in this space. First, we assume that each cluster arises from a density function which, when projected onto any line, is unimodal, having a single region of highest density. Second, we assume that any two distinct clusters may be

separated by a hyperplane, in the neighborhood of which there is a relatively lower density. In our experience, the unimodality hypothesis appears to hold for the large majority of neurons taken from a variety of brain areas — this assumption is also implicit in most neural clustering methods, even those that do not assume a Gaussian shape (Fee et al., 1996; Quiroga et al., 2004; Vargas-Irwin and Donoghue, 2007). In a minority of cases, we have observed a multimodal cluster distribution, reflecting more complex firing properties of a single neuron. Our strategy for handling this challenging scenario is discussed below.

Technical details for ISO-SPLIT are provided in Methods. The algorithm comprises a series of nonparametric statistical tests for unimodality and makes no assumptions about the shapes of clusters aside from having unimodal one-dimensional projections. It therefore involves few adjustable parameters. Essentially one needs only to specify a statistical threshold for rejecting the null-hypothesis of unimodality; the clustering output is largely insensitive to this threshold due to the test being repeated at every iteration. The method is also insensitive to the initialization (see Methods). We use the same set of parameters for all examples in this study. Moreover, the algorithm needs no *a priori* information about the expected number of clusters nor the expected cluster densities.

Sorting large electrode arrays

As indicated in Fig. 1, sorting is first performed independently on electrode neighborhoods (one neighborhood per electrode) based on the geometric layout of the array. Redundant clusters are then removed during the next phase entitled “consolidation across neighborhoods” (Fig. S1). There are a number of advantages of using such a consolidation approach rather than a more error-prone merging procedure (see Methods). Importantly, this approach allows scaling to very large electrode arrays as the neighborhood sizes will remain roughly constant.

Identification of putative “single unit”, “noise”, “non-isolated”, and “bursting” clusters

As part of the overall automation, we developed metrics for determining which clusters are sufficiently isolated from noise and other clusters to be included in the final output. In this way only sufficiently isolated clusters are selected for downstream analysis. Our quality assessment categorizes the clusters into three groups: “single unit”, “noise”, and “non-isolated”. In addition, we automatically identify (based on timing information) clear cases where the events of a bursting unit are split into multiple unimodal clusters (see Methods). Such decisions have traditionally been handled via case-by-case curation by manual operators. In contrast, our strategy only requires the operator to set thresholds on cluster quality metrics, as defined below. These metrics can be adjusted based on the type of analysis that will be done. Furthermore, the metrics can also be exported alongside the event times and labels, allowing analyses sensitive to unit isolation and noise contamination to be repeated with different thresholds or weighting criteria.

As stated, a central goal of our approach is to minimize the number of modeling assumptions. Currently available metrics (Harris et al., 2001; Schmitzer-Torbert et al., 2005; Schmitzer-Torbert and Redish, 2004), while useful, make assumptions about an underlying noise model. Here, we introduce two new metrics that make no such assumptions and are

specifically suited to spike sorting: isolation and noise overlap. We also use a measure of cluster signal-to-noise (SNR) to exclude clusters contaminated by artifacts, and employ a timing criterion to flag bursting situations.

Isolation—The isolation metric quantifies how well separated (in feature space) the cluster is from other nearby clusters. Clusters that are not well separated from others would be expected to have high false positive and false negative rates due to mixing with overlapping clusters. This quantity is calculated in a nonparametric way based on nearest-neighbor classification.

Noise overlap—Noise overlap estimates the fraction of “noise events” in a cluster, i.e., above-threshold events not associated with true firings of this or any of the other clustered units. A large noise overlap implies a high false positive rate. The procedure first empirically computes the expected waveform shape for noise events that have by chance crossed the detection threshold. It assesses the extent of feature space overlap between the cluster and a set of randomly selected noise clips after correcting for this expected noise waveform shape.

The noise overlap and isolation metrics vary between 0 and 1, and in a sense, represent the fraction of points that overlap either with another cluster (isolation metric) or with the noise cluster (noise overlap metric). However, they should *not* be interpreted as a direct estimate of the misclassification rate, but should rather be considered to be predictive of this quantity. Indeed, due to the way they are computed, these values will depend on factors such as the dimensionality of the feature space and the noise properties of the underlying data. Therefore, the annotation thresholds should be chosen to suit the application. With that said, in this study we used the same sorting parameters and annotation thresholds for all analyses.

Cluster SNR—Depending on the nature of signal contamination in the dataset, some clusters may consist primarily of high amplitude artefactual signals such as those that arise from movement, muscle, or other non-neural sources. In this case, the variation among event voltage clips will be large compared with clusters that correspond to neural units. To automatically exclude such clusters we compute cluster SNR, defined as the peak absolute amplitude of the average waveform divided by the peak standard deviation. The latter is defined as the standard deviation of the aligned clips in the cluster, taken at the channel and time sample where this quantity is largest.

Bursting units—While events corresponding to a single unit almost always form a cluster that is well approximated by a unimodal distribution, there are instances where the underlying distribution is multimodal. From our data we see that this most often occurs when the first spike in a burst has a different shape and higher amplitude than subsequent spikes in the burst. Our sorting algorithm by design will separate these events into two or more clusters. Fortunately, such clusters can be readily identified using event timing information, as the smaller spikes will always occur within a short time after the first spike in the burst. In the case of our hippocampal data this time window is on the order of 15 ms (Harris et al., 2001). In Methods, we describe the criteria used to label a cluster as having a “bursting parent” cluster.

Comparison to manual clustering for a tetrode dataset

If automated spike sorting is to be useful, it should provide cluster labels with accuracy comparable to or exceeding existing standards. We therefore began with a comparison of our automated approach to manual sorting for a dataset that poses serious spike sorting challenges: tetrode recordings from the CA1 region of rat hippocampus (Fig. 2A). The pyramidal cells in CA1 are densely packed, and thus large numbers of cells can be detected on the same electrode. Furthermore, extracellular recordings from single CA1 neurons show substantial waveform variability as a result of bursting and other history-dependent effects (Harris et al., 2001; Quirk et al., 2001). We chose a dataset with some electrode drift (45-minute recording session) and some artifact contamination resulting from animal movement. The data were derived from a novel exposure to a spatial environment where we expected to see neurons changing their firing rates substantially over time (Frank et al., 2004; Wilson and McNaughton, 1993). The standard in the field for such datasets is either fully manual clustering, or semi-automatic clustering where the algorithm over-clusters the data leaving the user to manually merge clusters or redraw cluster boundaries (Rossant et al., 2016).

Three different manual operators clustered the dataset using drawn polygons across several different 2D projections (see Methods). As expected (Harris et al., 2000), while there were clusters that all three operators identified, there was variability across operators, both in which clusters were sufficiently isolated to merit inclusion, as well as in the placement of the boundaries separating clusters, resulting in a range of unmatched events in each cluster (Fig. 2B). MountainSort was then run; results of the comparison are shown in the confusion matrices of Fig. 2C. A confusion matrix (or contingency table (Zaki and Meira, 2014)) summarizes the consistency between two sortings of the same data by showing the pairwise counts (Fig. S2A). The entry a_{ij} represents the number of events that were classified as i in the first sorting and as j in the second. To handle the arbitrary ordering of labels, the rows and columns are permuted to maximize the sum of the diagonal entries (Kuhn, 1955). A purely diagonal matrix corresponds to perfect agreement. For compact visualization, 11 of the 24 automatically sorted MountainSort clusters with the best match to the manual sorting results are shown.

As shown in Fig. 2C, all 6 clusters that are identified in 2 or more manual clusterings are also identified by MountainSort (MountainSort labels 23, 15, 24, 5, 14/16, 28). For example, MountainSort cluster 24 corresponds to clusters 3, 4, and 3 in the first, second and third manual sortings, respectively, with more than 97% of the manual events also detected by MountainSort. At the same time, MountainSort identifies a large number of events as part of these clusters that are not included in the manual sortings (Fig. 2C). That is not surprising given that our approach to manual clustering aims to minimize the mistaken inclusion of incorrect events at the expense of missing true events; this approach was chosen because false positives can lead to incorrect inferences about correlated activity (Quirk and Wilson, 1999).

Are these additional events likely to be true spiking events associated with the cell? Ground truth is not available for this dataset, or for the vast majority of other datasets, but in this case we can take advantage of the well-known “place fields” of hippocampal neurons (O’Keefe and Dostrovsky, 1971) to infer the accuracy of the sorting. We therefore examined

the animal's location at the times the spikes were detected. If the additional events are correctly classified, then they should congregate in the same location as the bulk of the events. This is indeed the case. As shown in Fig. 2D and S2B, events detected by MountainSort but not by manual operators have spatial distributions very similar to those of the jointly detected or individually detected events, suggesting that these additional events are likely to be correctly assigned to this cluster.

MountainSort also identifies a large number of clusters that were not identified by the manual sorters. For the tetrode data featured in Fig. 2 we used the following metric thresholds: noise overlap <0.03 , isolation >0.95 , firing rate >0.1 Hz, SNR >1.5 , although identical classifications for this dataset would result from using only noise overlap and firing rate. After automated curation and one automated bursting-related cluster merge (below), this resulted in the identification of 24 putative single-units (Fig. 3A). Importantly, these isolated clusters have few, if any, refractory period violations even though MountainSort does not use time information for clustering decisions (Fig. 3B). Furthermore, 22 of the 24 putative single-units have spatially restricted firing properties, consistent with expected behavior from hippocampal CA1 principal cells (Fig. 3C). MountainSort clusters 12 and 21 do not have spatially restricted firing properties and are relatively low amplitude, however, both have few refractory period violations and also have noise overlap scores that are below threshold (0.01 and 0.02, respectively; noise overlap threshold = 0.03). Further, the high firing rates of these units suggest that they are likely to correspond to one of the subtypes of inhibitory interneurons that can be found near the CA1 cell layer (Freund and Buzsaki, 1996).

We further evaluated the quality of the isolation of these additional units by identifying the cluster pairs which had the most similar waveforms, as quantified by isolation score, and asking whether there was evidence of contamination or low-quality clustering in those pairs. In this dataset, clusters 7 and 9 have the lowest isolation score of 0.96, and clusters 6 and 4 have the next lowest, an isolation score of 0.97. Despite these lower isolation scores, both pairs of clusters have noticeable waveform differences, clear separation in the principal component space, and a difference in spatial firing preferences (Fig. S3A, B). This suggests that these clusters all represent well isolated single units that likely correspond to single neurons.

Overall, we found that an isolation score <0.99 could reflect either activity originating from at least two different neurons (Fig. S3A, B) or activity from a single, bursting neuron with substantial amplitude variation and a multimodal cluster distribution. One such bursting cluster pair was automatically identified in this dataset (Fig. S3C, D cluster labels 14, 16; isolation = 0.97). Examination of their respective waveform shapes, spatial firing properties (Fig. S3C), and cross-correlation (Fig. S3D), suggests that the two clusters come from a bursting neuron, with the higher amplitude cluster often spiking before the lower-amplitude cluster. Indeed, after an automatic merge of MountainSort clusters 14 and 16 based on an identification of burst pairs (see Methods), we find strong similarity to manually identified clusters (Manual 1 cluster 5, manual 2 cluster 2, manual 3 cluster 5, Fig. S2B). Here we note that automatically joining these two clusters required adding assumptions about relative spike timing and amplitudes into the post-clustering automated annotation stage.

Nonetheless, while the merge is done in an automated fashion, the software maintains a record of that annotation alongside the original cluster assignments. This makes it straightforward for other scientists to assess all annotations made during sorting.

Five clusters were tagged as overlapping with noise (using noise overlap > 0.03). Visual inspection of the events in these clusters revealed four of the five having of broad and symmetric waveforms (Fig. S3E), characteristic of what one might expect from the summation of activity from many distant neurons crossing the event detection threshold. Furthermore, all five of these clusters have events falling in the refractory period, suggesting that the noise overlap measure effectively identifies clusters with events that should indeed be considered noise (Fig. S3F). In summary, the findings above indicate that MountainSort can produce high quality automatic sorting of tetrode datasets.

Finally, it is worth noting that only one of the isolated units was able to be identified on the basis of the individual channel sorting. This was MountainSort cluster 21. When using a single channel instead of all 4 channels, we found a noise overlap of 0.052 vs. 0.021, Isolation 0.98 vs. 0.99, and SNR 5.05 vs. 3.40. Interestingly, this is an interneuron of relatively low amplitude (mean peak height of $\sim 110 \mu\text{v}$). This suggests that single channels are insufficient to isolate single neurons, at least in hippocampal area CA1. We believe this would be true for any sorting algorithm.

Sorting of multi-contact electrode arrays

While tetrodes remain in use across many laboratories, new, high-density multielectrode arrays offer the ability to record from much larger ensembles of neurons (Berenyi et al., 2014; Du et al., 2011; Rios et al., 2016). MountainSort has a number of features designed specifically for such arrays. To demonstrate and evaluate these features, we applied our algorithm to 7 hours of data from a 16-channel, polymer probe (Fig. 4A) (Tooker et al., 2014; Tooker et al., 2013) dataset with challenges similar to those of the tetrode dataset used in Figs. 2 and 3. Although this array was placed in a different brain region (prelimbic cortex), had four times as many channels as the tetrode recording shown above, and contained a full seven hours of continuous recording, we used identical parameters for the clustering pipeline and for the quality metric thresholds.

Recall that MountainSort independently applies spike sorting on small electrode neighborhoods, and then consolidates across all channels. In the case of the 16-channel probe, each local neighborhood consisted of up to 7 electrodes. As a consequence, the feature space in which each electrode's clustering was done was derived from a different, sometimes non-overlapping, set of electrodes. This is a notable difference with the tetrode dataset where every channel was included in every neighborhood.

Applying MountainSort to this dataset resulted in the identification of 37 putative single-units (Fig. 4B). Importantly, as in Fig. 3, the putative single-units have few, if any, refractory period violations (Fig. 4C). Putative noise clusters (noise overlap > 0.03) are shown in Fig. S4. After removing clusters with high noise overlap, there was only one cluster pair flagged as non-isolated (isolation < 0.95). This was the one identified bursting cell pair, MountainSort ID 32 and 33, 0.91 isolation (Fig. S4C). The cluster pair with the next highest

overlap was cluster pair 30 and 31, 0.97 isolation, a pair identified as putative single-units during automated annotation (Fig. S4D). These results demonstrate that MountainSort can be applied to a range of datasets without the need for parameter adjustments.

Comparison with other sorting algorithms

MountainSort aims to provide a fully automated spike sorting pipeline in the sense that it takes as input a raw timeseries and generates a set of well isolated clusters. Most other software packages provide only a degree of automation by producing a set of clusters requiring further curation. This is oftentimes done using manual means: the expectation is that users will discard, merge, and sometimes even split clusters before using the results for downstream analyses. Setting aside the issue of human intervention, we compared MountainSort (MS) with two other spike sorting packages: KiloSort (KS) (Pachitariu et al., 2016) and Spyking Circus (SC) (Yger et al., 2016). The three algorithms were applied to (a) real data from our laboratory (the tetrode dataset described above), (b) a publicly available extracellular dataset with known ground truth (128-channel silicon polytrode together with a juxtacellular ground-truth measurement) (Neto et al., 2016), and (c) simulated data obtained from superimposing synthetic waveforms on background signal taken from a real dataset. Each of the three software packages has parameters that can be modified to optimize performance for different applications. However, as our objective is to create a spike sorting environment that is well suited to a diverse set of problems without requiring parameter tuning, we used the recommended set of parameters in each of the three packages (i.e., default values or the settings used in the examples distributed with the software). Importantly, for each algorithm we used the same parameters in all three experimental settings.

Confusion matrices comparing MS with KS and SC on the hippocampal dataset are shown in Fig. 5. For visualization purposes, the obviously invalid clusters (based on autocorrelograms) were manually excluded from KS and SC prior to assembling the matrices. These matrices make it clear that the three algorithms find many of the same units, but also highlight a number of clusters where the algorithms produce different results. These include one cluster that MS identified but KS did not (MS 12), and 9 clusters that MS identified but KS merged into other clusters (MS 18, 7, 23, 25, 29, 11, 10, 27, 17), 7 clusters that MS identified but SC did not (MS 5, 27, 25, 12, 9, 8, 7), and 7 clusters that MS identified but SC merged into other clusters (MS 10, 23, 18, 17, 16, 11, 4). Taken together, neither KS nor SC identified MS 12, and both KS and SC merged MS 18, 17, 11, 10, and 23 into other clusters. The 7 clusters found by KS but not by MS (appearing on the right side of Fig. 5A) correspond to low amplitude clusters rejected in the automated annotation stage of MountainSort for having a high noise overlap score.

Based on autocorrelograms and place maps, we believe the clusters found by MS and not by the other algorithms to be valid units, including MS12, a putative interneuron. Clusters in the KS or SC sorting which comprise events from multiple MS clusters (Fig. 5) could be the result of a MS false split or a KS or SC false merge. For simplicity, attention was focused on the cases where KS and SC had the same or overlapping MS cluster subsets (KS 48, SC 6; KS 8, SC 23; KS 2, SC 2). Despite KS and SC having a degree of agreement, the MS

clusters showed appreciable differences in PCA projections, waveforms, and spatial firing properties (Fig. S5), suggesting that the MS clusters were more likely to correspond to well isolated single units.

We then applied MS, KS, and SC to a publicly available 128-channel dataset with independent juxtacellular firing information for one of the cells (Neto et al., 2016). This dataset is one of ten datasets in the repository exhibiting varying levels of sorting difficulty. We found that only one of these featured a (ground-truthed) cell with sufficiently high amplitude-to-noise ratio to perform accurate sorting using any of the techniques. MountainSort identified this high amplitude unit with very high accuracy (>99%; 24 false positives and 32 false negatives out of 4895 true events) and appropriately marked it as a bursting pair. KiloSort also split the unit into two pieces with >99% accuracy, but has no mechanism to report bursting pairs. Spyking Circus split the true cluster into two pieces, but the larger portion (60% of events) were merged with a low amplitude cluster that included many false positives. Instructions for running these algorithms on this publicly available dataset are provided in the software repository for MountainSort.

In addition, we applied the algorithms to a publicly available tetrode recording from a rat hippocampus (Harris et al., 2000) with a juxtacellular ground truth channel. MountainSort found the unit with 10% false positives and 11% false negatives. Spyking Circus had only 6% false positives but 21% false negatives. KiloSort could not be run (in the current version of the software) because the channel count was too low.

Fig. 6 shows that MountainSort has a consistently higher accuracy score than the other two algorithms, particularly when the number of detectable clusters is high (Fig. 6C). We note, however, that because we chose the default/recommended parameters for the three algorithms, these results do not preclude the possibility that KS and SC could do better if parameters were adjusted.

Assessment of computational efficiency

MountainSort was also able to cluster all of these datasets much faster than real-time. We carried out timing tests on a Linux workstation with 192 GB RAM and 20 physical processors with hyper-threading capability (although not all cores were used in the experiments, as indicated). The 46-minute, 4-channel dataset used in Figs. 2 and 3 had a total MountainSort runtime of 40 seconds (including 22 seconds preprocessing) when run using 16 threads. This is around 70 times faster than real time. The 6.6-hour, 16-channel dataset used in Fig. 4 involved over 7 million detected events and had a total MountainSort runtime of 805 seconds (including 336 seconds preprocessing) using 16 threads. This is 30 times real time. The 128-channel publicly available dataset with juxtacellular ground truth (Neto et al., 2016) was sorted in 249 seconds (including 128 seconds preprocessing) using 40 threads. This represents 2.5 times real time.

To assess how processing times scale with the duration of recording and number of events, we processed subsets of the 7-hour, 16-channel probe dataset which contained ~1.1 million events per hour. We found that processing times scaled roughly linearly with the data duration, with a preprocessing time of ~1 minute per hour of the recording and a sorting

time of ~1.3 minutes per hour (Fig. 7A). We also wanted to assess the efficiency on machines with fewer processing cores. We varied the number of logical cores between 4 and 20 for processing a 4-hour subset of the same 16-channel data. We note that even though the processing is fastest when using 16 or more threads, the processing speed is still much faster than real time when restricting to only 4 logical cores. Taken together, and even assuming linear scaling with the number of electrodes (sublinear scaling is expected due to parallelization if we also increase the number of cores) our results suggest that 320 channels from the same electrode array could be sorted in real time on a single machine; this is on the order of the GPU rate reported in Pachitariu et al. (2016) for KiloSort.

Discussion

The MountainSort software suite provides fully automated spike sorting from electrode arrays of varied sizes and geometries in multiple brain regions, with accuracies comparable to or exceeding existing standards, and computational times much faster than acquisition times on non-specialized hardware. Importantly, MountainSort is also a fully functional package providing an intuitive graphical user interface with the ability to visualize subsets of selected clusters and annotation tools for exporting the data (see the online source code repository for documentation and tutorials). This stands in contrast to many previously developed clustering algorithms, where it would be difficult for users to test out algorithms on their own data. Thus, it is relatively straight-forward to incorporate MountainSort into a laboratory's data processing pipeline.

To our knowledge, this is the first time that a fully-automated spike sorting approach has been demonstrated to have comparable error rates to existing manual and semi-manual standards. Moreover, MountainSort sorted a dataset from the hippocampus better than humans, despite the presence of both complex spike bursts and noise events that arise from muscle and movement-related artifacts. Indeed, MountainSort identified many more putative single units from a hippocampal CA1 tetrode than manual sorters (24 vs 5–10). We note here that there are principled reasons to expect that MountainSort would outperform manual sorting. Typical manual sorting, such as in this study, is done in a static feature space of 4 to 16 dimensions (such as amplitude on channel 1, peak-to-trough ratio on channel 3, etc.), computed from filtered waveforms. Moreover, manual sorting usually involves drawing polygons in two dimensional projections. In contrast, MountainSort operates in a 10-dimensional space corresponding to the first 10 principal components (PCs), computed from spatially-whitened waveforms (effectively it is more than 10 dimensions when considering the branch method, see Methods). Moreover, each of the 10 dimensions does not correspond to only one channel at a time, as with some other clustering approaches, but instead each is a PC across all of the channels of the neighborhood. Further aiding in separation is the recomputation of the PCs when doing cluster comparisons. Due to these differences, it is highly likely that MountainSort can more reliably separate clusters than could a human operator, the current gold standard for tetrode spike sorting.

Comparison of MountainSort to two other recently released packages, KiloSort and Spyking Circus, suggests that MountainSort also performs better than these other packages. When applied to the hippocampal dataset, all three algorithms identified a subset of well isolated

units, but MountainSort found units that were not identified by Spyking Circus, and both KiloSort and Spyking Circus merged units with distinct PCA features and spatial firing patterns that MountainSort separated. Comparisons on simulated data revealed the same trends: MountainSort consistently identified units with greater accuracy than the other algorithms.

These comparisons highlight two other features of MountainSort. First, these comparisons were based on using the default parameter values for all three algorithms. While it is possible that different parameters could have improved the performance of KiloSort and Spyking Circus, one of our major goals was to develop an algorithm with essentially no free parameters, and our success in sorting data from multiple brain regions as well as multiple simulated dataset using the same parameters illustrates the power of this approach. Second, while these comparisons used the fully automated output from MountainSort, manual curation remained necessary for both KiloSort and Spyking Circus. It is important to note that these algorithms were not designed to be fully automatic, so the need for manual curation is not surprising, but it remains the case that, at the time of this publication, only MountainSort is successful in a fully automated mode.

Automation provides clear benefits for reproducibility among sortings of the same data and comparability across datasets. This is accomplished using novel isolation and noise overlap metrics. By contrast, commonly used cluster metrics, such as isolation distance (Harris et al., 2001) and L-ratio (Schmitzer-Torbert et al., 2005; Schmitzer-Torbert and Redish, 2004), utilize Mahalanobis distance calculations, which are dependent upon the dimensionality of the selected feature space, making it difficult to compare cluster quality between datasets. Furthermore, the commonly-used L-ratio metric, among others (Hill et al., 2011), makes the assumption of a multivariate normal cluster distributions, which can be problematic because this is often not valid as in cases of burst firing or electrode drift. By taking into account dataset-specific noise and nearest neighbor distances, our noise overlap and isolation metrics provide a nonparametric means to compare cluster quality across datasets.

This ability to compare quality across datasets allow these cluster quality metrics, combined with the annotation and provenance strategy built in to MountainSort, to be propagated to downstream analyses. Maintaining access to all clusters and their cluster quality metrics, including those that likely correspond to multiunit spiking, has a number of important advantages. First, thresholds could be established by examining how known properties of a given set of units (such as the number and size of hippocampal place fields) vary as a function of cluster quality (Harris et al., 2016). Second, simple changes in inclusion thresholds can be used for direct assessment of whether cluster quality influences a specific finding. Cluster quality could also be used to weight the contribution of each unit to a given analysis, thereby ensuring that the best isolated clusters have a proportionally greater influence on the findings. Third, there are cases where including multiunit spiking improves the quality of the results, as is the case for clusterless decoding of animal position from unsorted hippocampal spiking activity (Deng et al., 2015; Kloosterman et al., 2013). Inclusion of all clusters regardless of quality, greatly simplifies the application of these techniques to the data.

The combination of automation with a minimal set of parameters and model independence also has particular advantages. Existing spike sorting pipelines (Hill et al., 2011; Kadir et al., 2014; Pachitariu et al., 2016; Rossant et al., 2016; Yger et al., 2016) typically depend upon numerous adjustable parameters, operator judgment, and even allow for cluster boundaries to be redrawn, significantly increasing potential inter-operator variability. Automation that relies on a single, common set of parameters eliminates these issues. Further, our algorithm is also effectively model-free, making only the assumption that clusters have unimodal density projections. In contrast, existing sorting algorithms that rely on a template matching step (Franke et al., 2015; Vargas-Irwin and Donoghue, 2007; Zhang et al., 2004) (i.e., minimizing the L2-norm of the error for a given model) implicitly assume a multivariate Gaussian noise model, resulting in increased variability in output and operator curation when this assumption is not valid. MountainSort's model independence offers versatility, allowing it to deal gracefully with non-Gaussian waveform variation and to be used across recording conditions and locations.

While MountainSort's assumption of unimodality works well even in the case of the hippocampal cells (with non-Gaussian cluster distributions) we applied it to, there are still cases where bursting results in multiple unimodal clusters that should be merged. MountainSort therefore involves a post-processing step for automatically detecting clusters that are bursting pairs. The software also provides annotation tools to allow the user to indicate that he or she believes that the clusters should be merged, but importantly the original automatic clustering is also preserved in the sorting output, allowing for identification of all user decisions. This occasional need for manual merging could be further mitigated by using additional post-clustering analyses beyond our current relatively simple automated bursting pair criteria (Harris et al., 2001; Quirk et al., 2001; Quirk and Wilson, 1999).

MountainSort also runs quickly, even on large datasets. Clustering is always performed on local electrode neighborhoods, so clustering time theoretically scales linearly with the number of channels. Importantly, the clustering stage, along with most other stages of the processing, are implemented as parallel computations that can be run simultaneously on multiple CPU cores. As a result, the total time required for sorting and metric computation was much shorter than the recording times for the same data on standard computer hardware for both tetrodes (70× real-time) and a 16-channel polymer probe (30× real-time). These speed-ups are in line with other approaches (Pachitariu et al., 2016; Yger et al., 2016) where strategies such as hardware acceleration have brought computational times down significantly. However, these alternative approaches lack an automated curation component. As computers become increasingly powerful, the time spent by manual operators remains the largest bottleneck to spike sorting. With the elimination of manual curation, total sorting time becomes faster than acquisition time, and we can explore further directions that would otherwise be infeasible. One is the ability to run the clustering algorithm multiple times, allowing for cross-validation in the absence of ground truth (Barnett et al., 2016). Such stability metrics ensure that clustering results are not overly sensitive to realistic noise perturbations.

While MountainSort offers substantial advantages over current manual and semi-automatic clustering methods, additional work will be required to fully address two remaining challenges: overlapping spikes and drift. Currently, as with KlustaKwik (Kadir et al., 2014; Rossant et al., 2016), it has no problem with coincident spiking events where the two neurons are sufficiently separated in space (i.e., induce peak signal on different channels). However, neither package solves the harder problem of resolving spike waveforms that significantly overlap on the same electrode. Current solutions to that problem rely on model-based frameworks (Ekanadham et al., 2014; Franke et al., 2010; Pachitariu et al., 2016; Pillow et al., 2013), which, as discussed, include strong noise model assumptions. These approaches have been shown to be successful when applied to in-vitro datasets, and it may be possible to adapt these approaches as a post-sorting step to identify events that are best explained as the superposition of two or more spikes.

MountainSort naturally handles some level of drift, where there are systematic changes in waveforms thought to result from movement of electrodes relative to the tissue or perhaps due to glial cells migrating along electrode shanks (Bar-Hillel et al., 2006; Calabrese and Paninski, 2011; Chestek et al., 2007; Emondi et al., 2004). We expect that MountainSort will correctly identify cluster boundaries as long as the resultant time-collapsed cluster is unimodal and does not overlap with another cluster. Errors are expected when these time-collapsed clusters overlap with one another, however. That said, because MountainSort can be run on overlapping sections of data, it is conceptually straightforward to include augmentations that link clusters across time slices using segmentation fusion based algorithms (Dhawale AK, 2015). Cases where clusters drift in and out of noise or other clusters are more problematic, both for MountainSort and for all other current approaches. Here the new cluster metrics we have developed can be helpful, as clusters that drift into other clusters or the noise will tend to have poor isolation and noise overlap scores, allowing identification of times the cluster is sufficiently uncontaminated.

Taken together, our findings demonstrate that MountainSort is a sensible and efficient automated solution to the spike sorting problem. With a combination of minimal assumptions, fast run times, and a powerful graphical user interface, MountainSort can greatly shorten the time required for spike sorting while increasing the reproducibility and transparency of the process. With one electrode channel neighborhood allocated per available logical core, MountainSort is naturally extensible to distributed computing and large electrode arrays. Our approach is compatible with future refinements and extensions for resolving overlapping spikes and tracking units in the presence of drift.

CONTACT FOR REAGENT AND RESOURCE SHARING

Further information and requests for resources will be fulfilled by the lead contact, Loren Frank (loren@phy.ucsf.edu).

EXPERIMENTAL MODEL AND SUBJECT DETAILS

Rat

Both tetrode and 16-channel probe datasets were collected from a male long-evans rat (RRID: RGD_2308852), roughly 2 months after implantation of tetrode microdrive and polymer multielectrode arrays. At the time of data collection, the rat was ~8 months of age weighing ~525 g, and was fed standard rat chow (LabDiet 5001) in addition to sweetened evaporated milk for reward during behavioral performance. The rat was ordered from Charles River Laboratories at a weight of 350 g and ~3 months of age.

METHOD DETAILS

Throughout our sorting pipeline (Fig. S1A) we adhere to the objective of minimizing the number of parameters and the number of assumptions made. We also aim for full automation.

For preprocessing we used a bandpass filter implemented via Fast Fourier Transform (FFT) convolution followed by a heuristic procedure for masking out high voltage artifacts: we simply removed chunks of data with amplitudes outside of the expected range by replacing these data by zeros. We then applied a (nonparametric) spatial whitening filter to the entire time series matrix (details below) which we found to be crucial for separating nearby clusters. Both procedures were parallelized by dividing the dataset into time chunks (overlapping chunks in the case of the bandpass filter).

Sorting is initially performed separately on neighborhoods (one for each electrode channel) using information only from the central electrode and its neighbors (Fig. S1B). The neighborhood patch is determined by the geometric layout and a single user-specified distance parameter, an electrode adjacency radius. For the tetrode dataset, the neighborhood comprised all four channels, but the sorting was still performed four times, with each channel being considered as the central channel on its respective local sorting. In other words, sorting was done four times in a feature space derived from all four channels, each time only with the events detected on one of the four channels. For the 16-channel probe, neighborhoods of size 6–7 were used for all but two channels. The feature space for each electrode's sorting was derived from a different electrode neighborhood, an important difference from the tetrode dataset. The local sorting steps are event detection, clustering, and fitting as described below. In general, using larger electrode neighborhoods increases computation time, and we have observed that using more than 6–10 electrodes per neighborhood does not substantially influence sorting results for these examples.

In a second pass, the local sorting results are consolidated across electrodes to resolve duplications. By removing redundancies rather than merging clusters, we avoid a host of problematic scenarios. This is described in more detail below. Finally, per-cluster metrics are computed and automated annotation is applied so that clusters are labeled as “single unit”, “noise”, and “non-isolated”.

Filtering and spatial whitening

We bandpass filter (600 Hz to 6000 Hz) each channel in a standard fashion via the FFT applied to overlapping chunks of time. As the final stage of preprocessing, spatial whitening removes correlations among channels that are not due to the neuronal signals of interest. We have found this to be crucial in separating nearby clusters. Let Y be the $M \times N$ signal matrix. The whitened signal is given by

$$\tilde{Y} = UV^T$$

where

$$Y = USV^T$$

is the singular value decomposition. This is sometimes known as ZCA (zero-phase component analysis) whitening. The new data has an identity covariance matrix and can be calculated by applying the following channel mixing matrix at each time sample:

$$W_{mixing} = US^{-1}U^T$$

This procedure is parallelized by splitting the dataset into time chunks and computing the covariance matrix in the first pass through the data and applying the mixing matrix in a second pass.

Event detection

As described above, event detection is performed independently on each electrode (as part of its local neighborhood sorting) using the preprocessed (whitened) data. With closely spaced electrodes, it is expected that the same event will be flagged multiple times on different channels, but the subsequent cluster consolidation and fitting stages resolve this redundancy. An event is flagged at time sample t_0 whenever the following two criteria are met:

$$\begin{aligned} |Y(t_0)| &> \mu \\ |Y(t_0)| &> |Y(t_1)| \text{ for all } |t_0 - t_1| \leq \tau \end{aligned}$$

where Y is the preprocessed signal, μ is a detection threshold in units of standard deviations away from the mean, and τ is a detection interval specifying the minimum allowable time difference (in samples) between two events on the same channel. For our datasets we used $\mu = 3$ and $\tau = 10$ which corresponds to 0.33 ms at our 30 kHz sampling rate.

While the choice of these parameters is somewhat arbitrary, the detection threshold can be chosen to be quite low (in order to capture low amplitude firing events), with the caveat that the lower the threshold, the greater the computation time. Similarly, with low thresholds, the

noise clusters (ultimately discarded) will be larger. Our experience has been that the final sorting results are independent of this choice, provided that it is low enough.

Feature extraction and the branch method

Clustering is performed separately in each electrode neighborhood (typically 5–10 electrodes are used, regardless of total array size). Event clips are extracted, PCA features are computed, and then ISO-SPLIT clustering (described below) is applied in n -dimensional feature space, where n is typically 10. This has the potential to yield in an incorrect cluster merging due to the relatively low dimensionality of the feature space. Therefore, assuming that more than one cluster is found, we recompute the PCA features from the original clips for each initial cluster separately. Clustering is then applied to each of these and the procedure is repeated until no further cluster splitting occurs. In this way, we increase the sensitivity of distinguishing between distinct but nearby clusters. We call this the *branch method*.

At each stage of the branch clustering we use principal component analysis (PCA) features. For each detected time sample t_0 , a $M \times T$ clip centered at t_0 is extracted from the preprocessed data. Here M is the number of channels constituting the neighborhood of the electrode of interest, and T is the clip size in samples. We used $T = 50$ throughout. These clips are considered as MT -dimensional vectors and are mapped into the n -dimensional feature space corresponding to the first n principal components. We used $n = 10$ throughout, regardless of the neighborhood size. Again, we note that the effective size of the feature space is larger due to the branch method in which event features are computed recursively.

Clustering via ISO-SPLIT

Clustering takes as input a set of points (n -component vectors) and assigns to each point an integer cluster label. Our efficient, nonparametric clustering method termed ISO-SPLIT is at the heart of the sorting pipeline. It is a density-based method that does not require a tunable scale parameter, nor does it need an *a priori* estimate for the number of clusters. Instead a statistical test is applied on one-dimensional projections at each iteration (Magland and Barnett, 2016). This crucial one-dimensional kernel operation (ISO-CUT) is described in a separate section below.

The algorithm is initiated with a fine parcellation (over-clustering) and then the points are redistributed between clusters in successive iterations until convergence (Fig. 1B). While in principle the results can depend on the initial parcellation (or clustering), in practice when the parcellation is fine enough, they do not. In fact, a deterministic (albeit more computationally intense) procedure would be obtained by initially assigning each point to its own cluster.

Each iteration then involves pairwise comparisons of nearby clusters. Let A and B be two clusters to be compared. First the points of $A \cup B$ are projected onto a line of optimal discrimination between the two sets. A simple choice, although not always ideal, is to use the line connecting the centroids of A and B . In our implementation, we select an optimal line based on the centroids and the empirical covariance matrices of the sets. Next a statistical test is performed to determine whether this one-dimensional sample is unimodal,

or alternatively has two or more modes. This is the ISO-CUT procedure described below. If the unimodality hypothesis is rejected, the data points are redistributed between A and B according to the optimal cut point as determined by ISO-CUT. Otherwise, if unimodality is accepted, then the two clusters are merged.

In order to avoid infinite loops, the algorithm keeps track of which cluster pairs have already been compared. Once all pairs of remaining clusters have been compared, the algorithm is deemed to have converged.

ISO-CUT

The ISO-CUT algorithm (the kernel operation in ISO-SPLIT) is a non-parametric method for testing whether a one-dimensional sample arises from a unimodal distribution and for determining an optimal cut point for clustering. It is similar to the Hartigan dip test for unimodality (Hartigan and Hartigan, 1985) but there are important differences. First, Hartigan's test is only a criterion for accepting/rejecting the unimodality hypothesis, whereas ISO-CUT also returns an optimal cut point. Second, ISO-CUT uses a maximum-likelihood unimodal approximation to the data, which can be evaluated extremely efficiently using a modified version of isotonic regression. Third, ISO-CUT handles a crucial situation where the Hartigan test fails -- that is when a sparse cluster is adjacent to a very dense cluster. This situation arises often in spike sorting as cells can have vastly different firing rates.

Here we present a higher-level description of the algorithm and refer the reader to the Fig. S1 and the open source MATLAB and C++ implementations contained in the software itself for the lower-level implementation details. The first step of ISO-CUT is to sort the (assumed distinct) one-dimensional data points so we have $x_1 < x_2 < \dots < x_n$. Next the maximum-likelihood unimodal fit to the sample is computed. One can show that the density function for the fit is piecewise constant with endpoints at the data points, and that the sequence of values may be obtained using a modified version of isotonic regression, which we call up-down isotonic regression, applied to the reciprocal of the spacings $s_i^{-1} = (x_{i+1} - x_i)^{-1}$. A modified Kolmogorov-Smirnov (KS) statistic (Massey, 1951) is then computed to quantify the closeness of the empirical cumulative distribution function to the cumulative distribution function of the unimodal approximation (the modification is needed to handle the important case of sparse clusters on the periphery). If this unitless quantity is above the threshold (we used a value of 1 throughout) then unimodality is rejected.

In the case where the unimodality hypothesis is rejected, ISO-SPLIT returns a cut point, or an optimal cut point between two modes at a point of minimal density (note that the distribution may have more than two modes). Again, we aim to avoid density estimates in selecting such a cut point. Therefore, we employ the non-parametric isotonic regression once again by applying down-up isotonic regressions to the residuals $s_i^{-1} - t_i^{-1}$ where t_i is the estimate of the spacings based on the unimodal approximation. The cut point is taken at the deepest minimum of this sequence within a range of interest. Thus, the optimal cut point is obtained in a non-parametric manner without any density estimates.

Cluster consolidation across electrodes

Just as ISO-SPLIT is at the heart of the local clustering, our cluster consolidation strategy (Fig. S1B) is the central step for handling the multi-electrode arrays in a nonparametric manner. Since spike sorting is applied separately on each electrode (along with its neighbors), the same neuron is likely to be identified multiple times. In fact, it is expected that the same neural unit will appear on several channels, depending on the density of the electrode array. It is therefore necessary to merge the results across all channels into a single sorting output.

There are several problems with comparing pairs of clusters for potential merging. First, depending on the size and density of the electrode array, and the number of neurons detected on each channel, this could pose a computational challenge. Second, there is a problem with non-transitivity of merge decisions. For example, we could determine that clusters A and B (on different channels) should be merged, and that B and C should be merged, but that A and C should not be merged.

No matter whether merge decisions are based on average waveform shape similarity, coincident firing times, or cluster overlap in feature space, there are other fundamental problems with making such pairwise merge decisions. Consider, for example, the case where clustering on channel 1 is accurate, whereas on channel 2 there is a false merge. Then the corresponding clusters may fail to match up, or the incorrect results of channel 2 may supersede the accuracy on channel 1. This can easily happen when neurons are closer to the first channel but still give a detectable, but small, signal on the second. A myriad of problematic situations like these can arise.

We therefore propose a different approach – an alternative to merging – which we refer to as *cluster consolidation*. The assumption is that in the majority of cases, a neuron will fire with largest signal on a single primary channel, with the peak signal being lower on neighboring channels. Therefore we retain a cluster $C_{m,i}$ (the i^{th} cluster identified on channel m) if its peak average signal on channel m is sufficiently greater than its peak average signal on every other channel. Otherwise it is considered redundant and is discarded. More precisely, cluster $C_{m,i}$ is retained if for every channel number $m' \neq m$,

$$\max_t |W_{m,i}(m, t)| > \eta \max_t |W_{m,i}(m', t)|$$

where $W_{m,i}(m', t)$ is the average spike waveform for cluster $C_{m,i}$ on channel m' and clip sample t , and $\eta < 1$ is a constant. The buffer parameter η (0.9 in this study) is included in order to minimize the chances that the same cluster is eliminated on every channel, and therefore excluded altogether.

Ideally each cluster will be represented exactly once after consolidation. However, sometimes a neuron may yield a very similar peak signal on two adjacent electrodes and may therefore be retained in more than one neighborhood. While this depends on the choice of η and the density of the electrodes, this is a relatively rare occurrence for our datasets.

The goal of the second pass is to remove redundant clusters that survived the first pass of the consolidation (Fig. S1). As stated, we need to be careful about non-transitive merge criteria. To be clear, both passes involve only discarding rather than merging redundant clusters. The first step of the second pass is to sort the clusters in order of absolute peak amplitude (of the average spike waveform); if two clusters are determined to be duplicates, then we discard the one with lower absolute peak amplitude. Moving through the sorted list, each cluster is compared with all preceding clusters. Simply, if it is determined that it matches a previous cluster, then it is discarded.

The criteria for two clusters to match are three-fold: (a) they must have been detected on different channels, otherwise presumably they would not be separated by the clustering; (b) their peak amplitudes must be relatively close (within 30% for this paper), otherwise one would have been discarded in the first pass; (c) they must have a significant number of simultaneous (up to a time deviation threshold) firings (>50% coincident events for this paper).

Eliminating redundant events via fitting

While cluster consolidation removes redundant clusters, it can still happen that the same event may be redundantly flagged in more than one neighborhood. For example, if event a is correctly included in cluster A on channel neighborhood 1, but happens to be incorrectly assigned to cluster B on channel neighborhood 2, then, since A and B are not redundantly associated with the same unit, they will both survive the consolidation stage. However, it is important that each event be included exactly once in the final output. Therefore, the following procedure is used in a final pass to eliminate such redundant events.

In this fitting stage (a multi-pass procedure) we consider each detected event as a candidate. We begin with an empty set of accepted events, and add candidates during multiple sweeps through the data according to whether they reduce the L_2 -norm of the residual signal. Let Y be the $M \times N$ signal matrix (recall M is the number of channels and N the number of time samples). On the j^{th} pass, let Y_j be the residual signal (initialized by $Y_1 = Y$). We first compute an L_2 -norm reduction score for each candidate event that has not yet been included as follows:

$$S_{j,i} = \sum_{t=1}^T \sum_{m=1}^M \left(Y_j \left(m, t_0 - \frac{T}{2} + t \right) - W_{k_i}(m, t) \right)^2 - Y_j \left(m, t_0 - \frac{T}{2} + t \right)^2$$

where (t_i, k_i) is the time sample and label for the i^{th} candidate event, $W_k(m, t)$ is the representative waveform for the k^{th} cluster, and T is the clip size (assumed even). The i^{th} event is then included if

$$S_i > 0 \text{ and } S_i > S_j \text{ for all } j \neq i, |i-j| \leq T_0$$

That is, the L_2 -norm is reduced by including the event, and the amount by which it is reduced is greater than that of any overlapping candidate event. The waveforms for the

newly accepted events are then subtracted from Y_j to obtain a new residual Y_{j+1} and the procedure is repeated until no further changes occur in a pass. To speed things up we keep track of which scores need to be recomputed on subsequent passes.

Partially overlapping events are thus handled properly, assuming that they are at least somewhat separated in either time or space. Note also that the requirement for the score to be positive has the benefit of removing outliers to some extent. Crucially, in contrast to model-based approaches, (Ekanadham et al., 2014; Franke et al., 2015; Pachitariu et al., 2016; Pillow et al., 2013) which use L_2 fitting to update waveform shapes, firing times, and amplitudes, in our scheme L_2 fitting is only used to *eliminate duplicate events*. In particular, this preserves labels chosen by our non-Gaussian clustering method.

Isolation and noise overlap metrics

Here we describe the post-processing step that classifies “single unit”, “noise”, and “non-isolated” clusters based on an objective set of criteria, as described above. We have elected *not* to use any firing time information (e.g., dips in cross-correlograms) to make these decisions, so as to leave such information as an independent validation metric. Adhering to our overall approach, we also do not want our metrics to depend on additional assumptions about cluster distributions. Here we define the isolation and noise overlap methods which control automated cluster selection.

First, we describe the isolation metric for assessing how well a cluster

$$A = \{a_1, \dots, a_n\}$$

is separated from the other clusters. Let A and B be two distinct clusters. For each x in $A \cup B$, let $n_1(x), \dots, n_k(x)$ be the k nearest neighbors of x in $A \cup B$. Let ρ be the membership function so that $\rho(A) = 1$ and $\rho(B) = 2$. Then we define the k -nearest neighbor overlap between A and B to be

$$m_{overlap}(A, B) = \frac{1}{n} \sum_{j=1}^n \frac{\#\{x \in A \cup B : \rho(n_j(x)) = \rho(x)\}}{\#(A \cup B)}$$

which is the fraction of the nearest neighbors that are classified consistently with their parent point¹. The isolation metric is then given as

$$m_{isolation}(A) = 1 - \min_{clusters B} m_{overlap}(A, B)$$

where the minimum is taken over all other clusters.

¹Often the number of points in one cluster will be much larger than in the other. This can lead to an artificially high overlap metric because the likelihood of misclassification depends not only on the degree of separation but also on the relative sizes of the clusters. Therefore we only sample N random points from each of the two clusters where N is the minimum size of the two clusters. To reduce computation time we also require that N is at most 500.

We take a somewhat different approach for noise overlap. We define the noise overlap of cluster A to be the overlap metric for A with a “fake” cluster

$$B = \{b_1, \dots, b_n\}$$

comprised of the same number of random noise events (clips extracted from the original data at purely random times). However, the difficulty in comparing these two clusters is that the event clips of A have a bias in that they were selected for being (perhaps by chance) super-threshold. Therefore, we need to first remove this bias by subtracting out a multiple of the expected noise waveform Z to obtain adjusted clips prior to computing the overlap. We use the following weighted average for the expected noise waveform:

$$Z(m, t) = \frac{\sum_{i=1}^n b_i(m_0, t_0) b_i(m, t)}{\sum_{i=1}^n b_i(m_0, t_0)}$$

here the weight $b_i(m_0, t_0)$ is the value of the spike at its central channel and central time sample. We then define the noise overlap to be the overlap metric applied to the sets of event clips after projecting out the dimension defined by the expected noise shape:

$$m_{noise}(A) = m_{overlap}(\tilde{A}, \tilde{B})$$

Automatic identification of bursting clusters

We use two criteria (waveform similarity and relative timing) to flag a pair of clusters as belonging to the same unit, with one cluster A being labeled as the bursting parent of a second cluster B . First, the average spike waveforms must be similar to one another up to scaling. Here we require that the correlation across time and channels be greater than 0.8 (an adjustable threshold). Second, the cross-correlogram between clusters A and B must have a significant asymmetry within a chosen time window (± 15 ms in our analyses) with events in B tending to occur a short time after events in A . This was quantified by comparing n_{after} , the number of B events occurring within 15 ms *after* events in A , to n_{before} , the number of B events occurring within 15 ms *before* events in A . If n_{after} is significantly greater than $2n_{before}$ ($p < 0.001$, assuming a Poisson distribution), and the correlation criterion is met, then we flag (A, B) as a bursting pair. We note, however, that the output of MountainSort retains the identity of the individual clusters associated with each bursting pair. This maintains data provenance and allows for alternative criteria to be applied at a later date if so desired.

Manual clustering

Data were bandpass filtered 600–6000 Hz and then thresholded at 60 μ v for event detection. Individual units (putative single neurons) were identified by drawing cluster boundaries in two-dimensional projections of peak amplitude, the first two principal components of each channel, peak to trough ratio, or spike width as variables (MatClust, M.P.K.). Only well-

isolated neurons with stable spike waveform amplitudes were clustered. Clusters that were cut off at spike threshold were not selected.

Simulations using synthetic data

The datasets used in Fig. 6 were generated by superimposing synthetic waveforms on background signal taken from a real dataset. We wanted the background signal to include realistic noise, correlations between channels, and low-amplitude signals from distant neurons. We therefore started with the hippocampal dataset used in this paper, automatically removed time segments that included detectable (relatively high amplitude) spike events and then pieced together the remaining time segments using an overlap region and apodization (a filtering approach to smooth the signal) at the edges of the segments to avoid inflating the noise level on the overlaps.

The shapes of the superimposed spike waveforms were not derived from the sorted dataset because we did not want to bias the results toward whichever sorting algorithm used to obtain those shapes. We also wanted to easily control the number of simulated units and the distribution of peak amplitudes and shapes. Therefore, we created a formula that generated realistic spike shapes dependent on a rise time, peak amplitude, decay time and recovery time on each of the four channels. We generated units with differing firing rates (randomly selected between 0.5 and 3 spikes per second), different spike waveform shapes, and peak amplitudes ranging from zero to a maximum of 20 standard deviations above the mean, with more units at lower amplitude as reflected in real data.

In the first of the three simulated datasets, 15 units were simulated with around 8 having high enough amplitude to be detected (Fig. 6A and 6D). For the second dataset (Fig. 6B), 30 units were simulated with around 15 detectable. For the third (Fig. 6C), 60 units were simulated with around 30 detectable. The MATLAB scripts for generating these may be found in the source code repository.

Computational efficiency measurements

We conducted all processing on a Dell Linux workstation with 192 GB RAM and 40 logical processing cores (Intel Xeon processor 2.8 GHz), although not all cores were used in the experiments, as indicated in the text. All algorithms were implemented as custom C++ programs.

In vivo datasets

A total of four 16-channel polymer shanks were targeted to Prelimbic cortex (+ 2.4 to 2.6 AP, \pm 1.1 ML, DV -3.4 , 7° from sagittal). The tetrode was constructed from four 12.5 μ m nichrome wires (California Fine Wire) spun together, with each of the four wires electroplated with gold to an impedance of \sim 250 k Ω at 1 kHz. The tetrode was one of sixteen in individual cannulae whose centroid was targeted -3.6 AP, \pm 2.4 ML, and then adjusted to CA1 cell layer based on LFP and spiking activity.

For the tetrode dataset, the animal was run for 45-minutes on the first exposure to the W-track continuous spatial alternation behavior (Frank et al., 2000; Kim and Frank, 2009).

The 16-channel polymer probe dataset was collected during a 7-hr period while the animal was in a 13" x 13" walled rest box.

All data were sampled and saved at 30 KHz on a 320-channel modular headstage using Trodes software (SpikeGadgets llc.).

DATA AND SOFTWARE AVAILABILITY

The MountainSort clustering software, as well as the code used to generate the simulated datasets is publically available at: <https://github.com/magland/mountainlab>. The tetrode dataset is available at: <http://dx.doi.org/10.17632/kmmndvycx8.1>. The polymer probe dataset is available at: <http://dx.doi.org/10.17632/j2mfvnsz2t.1> (part 1/3), <http://dx.doi.org/10.17632/ssb8s4766s.1> (part 2/3), and <http://dx.doi.org/10.17632/9mgpgn7fsw.1> (part 3 of 3).

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

The Flatiron Institute is a division of the Simons Foundation. This work was also supported by NINDS grant U01NS090537 to L.M.F. and V.T. and NIMH grant F30MH109292 to J.E.C. We thank Kevin Fan for assistance with data collection, Demetris Roumis and Marielena Sosa for manual clustering, and Witold Wysota for programming support. We are grateful for helpful discussions with E.J. Chichilnisky, Eero Simoncelli, György Buzsáki, Adrien Peyrache, Carl Schoonover, and Andrew Fink.

References

- Abeles M, Goldstein MH. Multispike Train Analysis. *Proceedings of the IEEE*. 1977; 65:762–773.
- Bar-Hillel A, Spiro A, Stark E. Spike sorting: Bayesian clustering of non-stationary data. *J Neurosci Methods*. 2006; 157:303–316. [PubMed: 16828167]
- Barnett AH, Magland JF, Greengard LF. Validation of neural spike sorting algorithms without ground-truth information. *J Neurosci Methods*. 2016; 264:65–77. [PubMed: 26930629]
- Berenyi A, Somogyvari Z, Nagy AJ, Roux L, Long JD, Fujisawa S, Stark E, Leonardo A, Harris TD, Buzsaki G. Large-scale, high-density (up to 512 channels) recording of local circuits in behaving animals. *J Neurophysiol*. 2014; 111:1132–1149. [PubMed: 24353300]
- Calabrese A, Paninski L. Kalman filter mixture model for spike sorting of non-stationary data. *J Neurosci Methods*. 2011; 196:159–169. [PubMed: 21182868]
- Carlson DE, Vogelstein JT, Qisong W, Wenzhao L, Mingyuan Z, Stoetzner CR, Kipke D, Weber D, Dunson DB, Carin L. Multichannel electrophysiological spike sorting via joint dictionary learning and mixture modeling. *IEEE Trans Biomed Eng*. 2014; 61:41–54. [PubMed: 23912463]
- Chestek CA, Batista AP, Santhanam G, Yu BM, Afshar A, Cunningham JP, Gilja V, Ryu SI, Churchland MM, Shenoy KV. Single-neuron stability during repeated reaching in macaque premotor cortex. *J Neurosci*. 2007; 27:10742–10750. [PubMed: 17913908]
- Csicsvari J, Henze DA, Jamieson B, Harris KD, Sirota A, Bartho P, Wise KD, Buzsaki G. Massively parallel recording of unit and local field potentials with silicon-based electrodes. *J Neurophysiol*. 2003; 90:1314–1323. [PubMed: 12904510]
- Deng X, Liu DF, Kay K, Frank LM, Eden UT. Clusterless Decoding of Position from Multiunit Activity Using a Marked Point Process Filter. *Neural Comput*. 2015; 27:1438–1460. [PubMed: 25973549]

- Dhawale AK, PR, Kopelowitz E, Normand V, Wolff S, Olveczky B. Automated long-term recording and analysis of neural activity in behaving animals. *bioRxiv*. 2015:033266.
- Du J, Blanche TJ, Harrison RR, Lester HA, Masmanidis SC. Multiplexed, high density electrophysiology with nanofabricated neural probes. *PLoS One*. 2011; 6:e26204. [PubMed: 22022568]
- Einevoll GT, Franke F, Hagen E, Pouzat C, Harris KD. Towards reliable spike-train recordings from thousands of neurons with multielectrodes. *Curr Opin Neurobiol*. 2012; 22:11–17. [PubMed: 22023727]
- Ekanadham C, Tranchina D, Simoncelli EP. A unified framework and method for automatic neural spike identification. *J Neurosci Methods*. 2014; 222:47–55. [PubMed: 24184059]
- Emondi AA, Rebrik SP, Kurgansky AV, Miller KD. Tracking neurons recorded from tetrodes across time. *J Neurosci Methods*. 2004; 135:95–105. [PubMed: 15020094]
- Fee MS, Mitra PP, Kleinfeld D. Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-Gaussian variability. *J Neurosci Methods*. 1996; 69:175–188. [PubMed: 8946321]
- Frank LM, Brown EN, Wilson M. Trajectory encoding in the hippocampus and entorhinal cortex. *Neuron*. 2000; 27:169–178. [PubMed: 10939340]
- Frank LM, Stanley GB, Brown EN. Hippocampal plasticity across multiple days of exposure to novel environments. *J Neurosci*. 2004; 24:7681–7689. [PubMed: 15342735]
- Franke F, Natora M, Boucsein C, Munk MH, Obermayer K. An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes. *J Comput Neurosci*. 2010; 29:127–148. [PubMed: 19499318]
- Franke F, Quiroga R, Hierlemann A, Obermayer K. Bayes optimal template matching for spike sorting - combining fisher discriminant analysis with optimal filtering. *J Comput Neurosci*. 2015; 38:439–459. [PubMed: 25652689]
- Freund TF, Buzsaki G. Interneurons of the hippocampus. *Hippocampus*. 1996; 6:347–470. [PubMed: 8915675]
- Gray CM, Maldonado PE, Wilson M, McNaughton B. Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex. *J Neurosci Methods*. 1995; 63:43–54. [PubMed: 8788047]
- Harris KD, Henze DA, Csicsvari J, Hirase H, Buzsaki G. Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements. *J Neurophysiol*. 2000; 84:401–414. [PubMed: 10899214]
- Harris KD, Hirase H, Leinekugel X, Henze DA, Buzsaki G. Temporal interaction between single spikes and complex spike bursts in hippocampal pyramidal cells. *Neuron*. 2001; 32:141–149. [PubMed: 11604145]
- Harris KD, Quiroga RQ, Freeman J, Smith SL. Improving data quality in neuronal population recordings. *Nat Neurosci*. 2016; 19:1165–1174. [PubMed: 27571195]
- Hartigan JA, Hartigan P. The dip test of unimodality. *The Annals of Statistics*. 1985:70–84.
- Hill DN, Mehta SB, Kleinfeld D. Quality metrics to accompany spike sorting of extracellular signals. *J Neurosci*. 2011; 31:8699–8705. [PubMed: 21677152]
- Kadir SN, Goodman DFM, Harris KD. High-Dimensional Cluster Analysis with the Masked EM Algorithm. *Neural Comput*. 2014; 26:2379–2394. [PubMed: 25149694]
- Kim SM, Frank LM. Hippocampal lesions impair rapid learning of a continuous spatial alternation task. *PLoS ONE*. 2009; 4:e5494. [PubMed: 19424438]
- Kloosterman F, Layton SP, Chen Z, Wilson MA. Bayesian Decoding using Unsorted Spikes in the Rat Hippocampus. *J Neurophysiol*. 2013
- Kuhn HW. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*. 1955; 2:83–97.
- Lewicki MS. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network*. 1998; 9:R53–78. [PubMed: 10221571]
- Lopez, CM., Mitra, S., Putzeys, J., Raducanu, B., Ballini, M., Andrei, A., Severi, S., Welkenhuysen, M., Van Hoof, C., Musa, S., Yazicioglu, RF. A 966-electrode neural probe with 384 configurable

- channels in 0.13?? m SOI CMOS. 2016 IEEE International Solid-State Circuits Conference; 2016. p. 392-393.
- Magland JF, Barnett AH. Unimodal clustering using isotonic regression: ISO-SPLIT. 2016 arXiv, 1508.04841v04842.
- Marre O, Amodei D, Deshmukh N, Sadeghi K, Soo F, Holy TE, Berry MJ 2nd. Mapping a complete neural population in the retina. *J Neurosci*. 2012; 32:14859–14873. [PubMed: 23100409]
- Massey F. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*. 1951; 46:68–78.
- McNaughton BL, O’Keefe J, Barnes CA. The stereotrode: a new technique for simultaneous isolation of several single units in the central nervous system from multiple unit records. *J Neurosci Methods*. 1983; 8:391–397. [PubMed: 6621101]
- Muller R. A quarter of a century of place cells. *Neuron*. 1996; 17:813–822. [PubMed: 8938115]
- Neto JP, Lopes G, Frazao J, Nogueira J, Lacerda P, Baiao P, Aarts A, Andrei A, Musa S, Fortunato E, et al. Validating silicon polytrodes with paired juxtacellular recordings: method and dataset. *J Neurophysiol*. 2016; 116:892–903. [PubMed: 27306671]
- O’Keefe J, Dostrovsky J. The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain Res*. 1971; 34:171–175. [PubMed: 5124915]
- Pachitariu M, Steinmetz N, Kadir S, Carandini M, Harris K. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *bioRxiv*. 2016:061481.
- Pedreira C, Martinez J, Ison MJ, Quian Quiroga R. How many neurons can we see with current spike sorting algorithms? *J Neurosci Methods*. 2012; 211:58–65. [PubMed: 22841630]
- Pillow JW, Shlens J, Chichilnisky EJ, Simoncelli EP. A model-based spike sorting algorithm for removing correlation artifacts in multi-neuron recordings. *PLoS One*. 2013; 8:e62123. [PubMed: 23671583]
- Quirk MC, Blum KI, Wilson MA. Experience-dependent changes in extracellular spike amplitude may reflect regulation of dendritic action potential back-propagation in rat hippocampal pyramidal cells. *J Neurosci*. 2001; 21:240–248. [PubMed: 11150341]
- Quirk MC, Wilson MA. Interaction between spike waveform classification and temporal sequence detection. *J Neurosci Methods*. 1999; 94:41–52. [PubMed: 10638814]
- Quiroga RQ. Spike sorting. *Curr Biol*. 2012; 22:R45–46. [PubMed: 22280903]
- Quiroga RQ, Nadasdy Z, Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Comput*. 2004; 16:1661–1687. [PubMed: 15228749]
- Rios G, Lubenov EV, Chi D, Roukes ML, Siapas AG. Nanofabricated Neural Probes for Dense 3-D Recordings of Brain Activity. *Nano letters*. 2016
- Rodriguez A, Laio A. Machine learning. Clustering by fast search and find of density peaks. *Science*. 2014; 344:1492–1496. [PubMed: 24970081]
- Rossant C, Kadir SN, Goodman DF, Schulman J, Hunter ML, Saleem AB, Grosmark A, Belluscio M, Denfield GH, Ecker AS, et al. Spike sorting for large, dense electrode arrays. *Nat Neurosci*. 2016
- Santhanam G, Linderman MD, Gilja V, Afshar A, Ryu SI, Meng TH, Shenoy KV. HermesB: a continuous neural recording system for freely behaving primates. *IEEE Trans Biomed Eng*. 2007; 54:2037–2050. [PubMed: 18018699]
- Schmitzer-Torbert N, Jackson J, Henze D, Harris K, Redish AD. Quantitative measures of cluster quality for use in extracellular recordings. *Neuroscience*. 2005; 131:1–11. [PubMed: 15680687]
- Schmitzer-Torbert N, Redish AD. Neuronal activity in the rodent dorsal striatum in sequential navigation: separation of spatial and reward responses on the multiple T task. *J Neurophysiol*. 2004; 91:2259–2272. [PubMed: 14736863]
- Shobe JL, Claar LD, Parhami S, Bakhurin KI, Masmanidis SC. Brain activity mapping at multiple scales with silicon microprobes containing 1,024 electrodes. *J Neurophysiol*. 2015; 114:2043–2052. [PubMed: 26133801]
- Swindale NV, Spacek MA. Spike sorting for polytrodes: a divide and conquer approach. *Front Syst Neurosci*. 2014; 8:6. [PubMed: 24574979]
- Takahashi S, Sakurai Y, Tsukada M, Anzai Y. Classification of neuronal activities from tetrode recordings using independent component analysis. *Neurocomputing*. 2002; 49:289–298.

- Takekawa T, Isomura Y, Fukai T. Spike sorting of heterogeneous neuron types by multimodality-weighted PCA and explicit robust variational Bayes. *Front Neuroinform.* 2012; 6:5. [PubMed: 22448159]
- Tooker A, Liu D, Anderson EB, Felix S, Shah KG, Lee KY, Chung JE, Pannu S, Frank L, Tolosa V. Towards a large-scale recording system: demonstration of polymer-based penetrating array for chronic neural recording. *Conf Proc IEEE Eng Med Biol Soc.* 2014; 2014:6830–6833. [PubMed: 25571565]
- Tooker, A., Tolosa, V., Shah, K., Sheth, H., Felix, S., Delima, T., Pannu, S. Optimization of multi-layer metal neural probe design. 34th Annual International Conference of the IEEE EMBS; San Diego, CA. 2013.
- Vargas-Irwin C, Donoghue JP. Automated spike sorting using density grid contour clustering and subtractive waveform decomposition. *J Neurosci Methods.* 2007; 164:1–18. [PubMed: 17512603]
- Wilson MA, McNaughton BL. Dynamics of the hippocampal ensemble code for space. *Science.* 1993; 261:1055–1058. [PubMed: 8351520]
- Wood F, Black MJ, Vargas-Irwin C, Fellows M, Donoghue JP. On the variability of manual spike sorting. *IEEE Trans Biomed Eng.* 2004; 51:912–918. [PubMed: 15188858]
- Yger P, Spampinato G, Esposito E, Lefebvre B, Deny S, Gardella C, Stimberg M, Jetter F, Zeck G, Picaud S, et al. Fast and accurate spike sorting in vitro and in vivo for up to thousands of electrodes. *bioRxiv.* 2016:067843.
- Zaki, MJ., Meira, WJ. *Data mining and analysis: fundamental concepts and algorithms.* New York, NY: Cambridge University Press; 2014.
- Zhang PM, Wu JY, Zhou Y, Liang PJ, Yuan JQ. Spike sorting based on automatic template reconstruction with a partial solution to the overlapping problem. *J Neurosci Methods.* 2004; 135:55–65. [PubMed: 15020089]

Highlights

- MountainSort is a new fully-automated spike sorting algorithm and software package.
- Accuracy is comparable to or greater than that of existing methods.
- Desktop computation speeds are $\sim 70/30/2.5x$ real-time for 4/16/128 electrodes.
- The software enables reproducible spike sorting for large datasets.

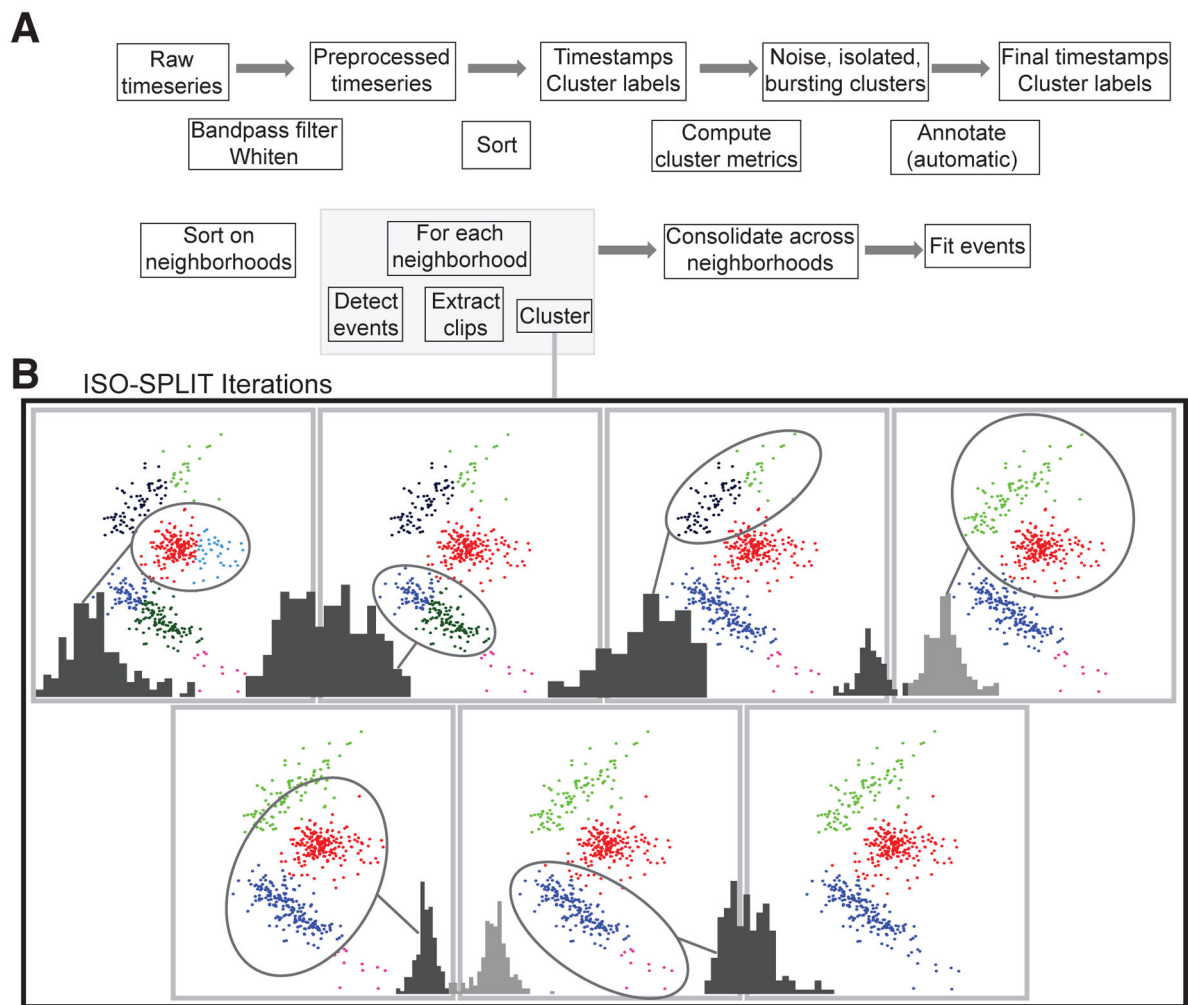


Figure 1. Overview of the fully-automated MountainSort processing pipeline

(A) Flow diagram. After preprocessing, sorting is performed on individual electrode neighborhoods. Clusters are then consolidated across neighborhoods (see also Fig. S1). Clusters are either accepted or rejected in the automatic annotation phase based on the computed cluster metrics. (B) Illustration of the final 6 iterations of the ISO-SPLIT clustering algorithm for a synthetically generated set of points. At each iteration, two clusters are compared using a one-dimensional projection of the union of the two clusters (shown in the histograms at each iteration). The ISO-CUT procedure is applied to the projected data to determine whether the two clusters should be merged (single color in the histogram) or whether the points should be redistributed according to an optimal cut point (two colors in the histogram).

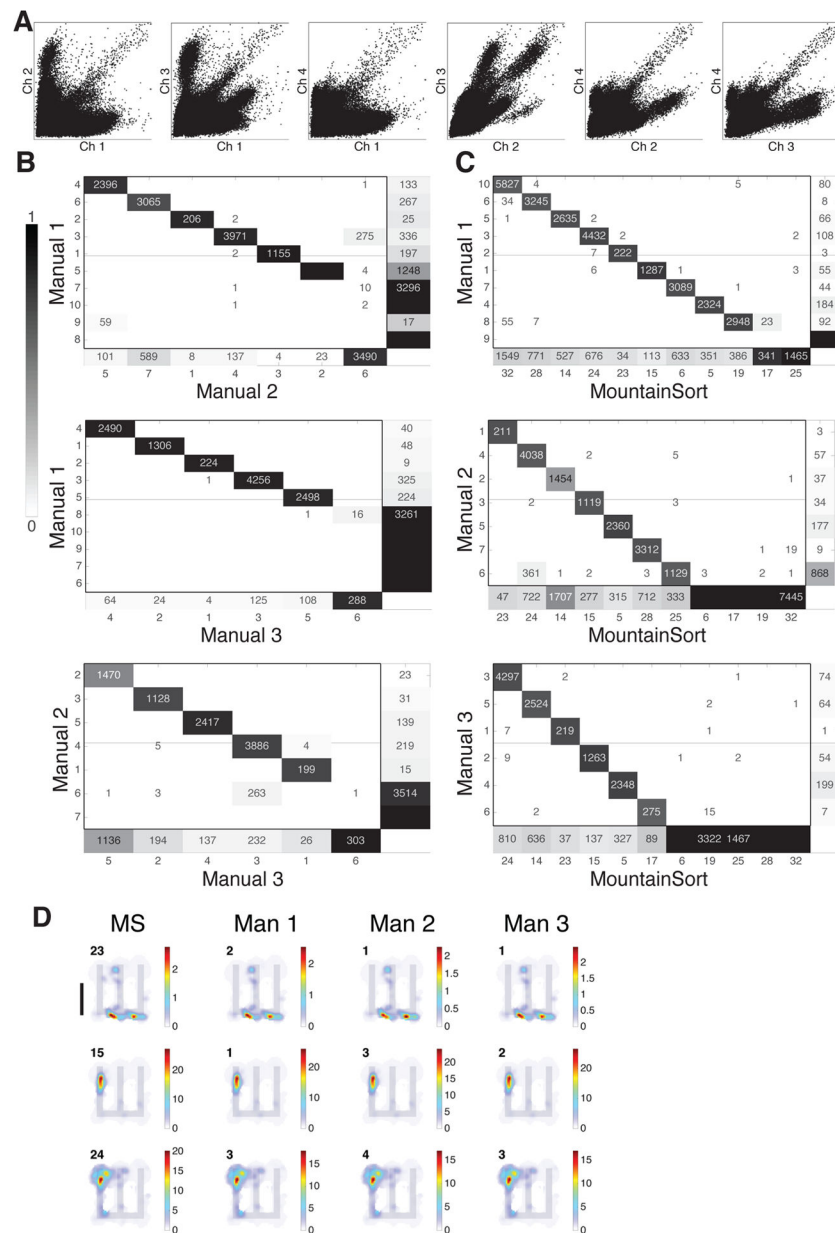


Figure 2. Fully-automated MountainSort produces clusters comparable to manual operators (A) Two-dimensional peak amplitude projections of the data used for manual sorting. Axes show values between 0 and 500 μV . (B) Pairwise confusion matrices (fig. S2A) for the three manual sortings. For each matrix, the numbers in the leftmost column and bottommost row correspond to cluster ID number for the respective manual sorter. Shading corresponds to the proportion of each column-labeled cluster that is classified into each row-labeled cluster. For the top matrix, this is the proportion of events in each manual 2 cluster that match the corresponding manual 1 cluster. Each number within the matrix corresponds to the absolute number of matching events. The final column corresponds to the number of events found in the row-labeled clustering not found in the column-labeled clustering. For the top matrix, this corresponds to the events in manual 1 clusters that are not found in any manual 2

clusters. Similarly, the second row from the bottom corresponds to the number of events found in the manual 2 clusters not found in any manual 1 clusters. (C) As in (B), except for purpose of compact visualization, only the MountainSort clusters which correspond to one or more manual clusters are shown. (D) Occupancy-normalized spatial firing rate color maps for three clusters corresponding across MountainSort (MS) and manual operators. See also Fig. S2. Track outline is shown in gray. Note that the track has no walls, and we used the animal's head for position tracking. As the animal often looked out over the edge of the track, many of the positions shown are outside the track outline. Scalebar corresponds to 50 cm. Note that colorbar scale varies across clusters.

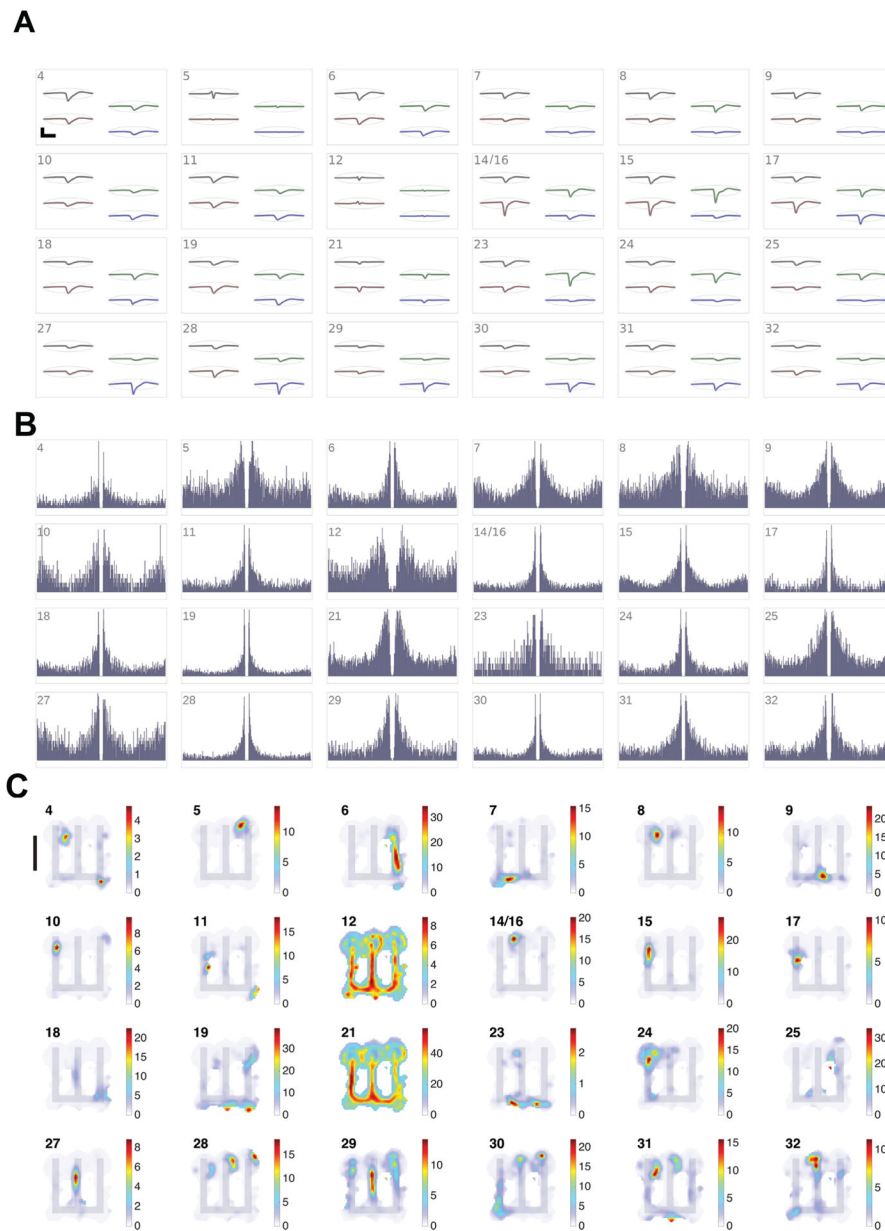


Figure 3. All MountainSort-identified putative single-unit clusters from the hippocampal CA1 tetrode dataset

(A) Average waveforms (bandpass filtered 300 – 6000 Hz) for the putative single-unit clusters as determined using metric thresholds: noise overlap <0.03 , isolation >0.95 , firing rate >0.05 Hz. MountainSort cluster ID is inset. Scale corresponds to $250 \mu\text{v}$ and 1 ms. (B) Autocorrelograms for the corresponding clusters; X-axis range is ± 100 ms, normalized Y-axis range. (C) Occupancy-normalized spatial firing rate maps for the inset MountainSort cluster ID. Track outline is shown in gray. Scalebar corresponds to 50 cm.

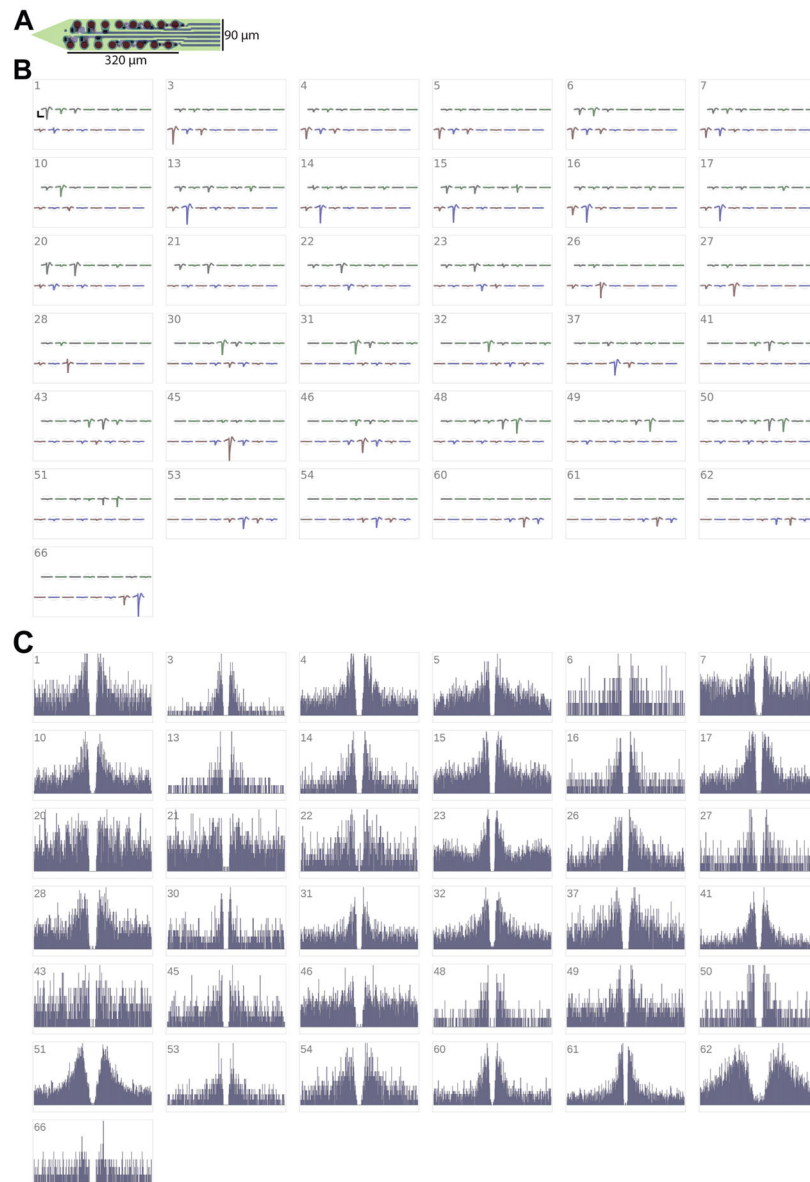


Figure 4. MountainSort identification of well-isolated units for 16-channel probe
 (A) Geometric layout of polymer probe. Contact size 20 μm, horizontal contact edge-to-edge 20 μm, vertical contact edge-to-edge 38 μm. (B) Average waveforms filtered 300 – 6000 Hz, of putative single-units with waveforms organized according to (A), with cluster identification number inset. Scalebar corresponds to 100 μv and 2 ms. (C) Corresponding autocorrelograms. X-axis scale 100 ms, normalized Y-axis scale.

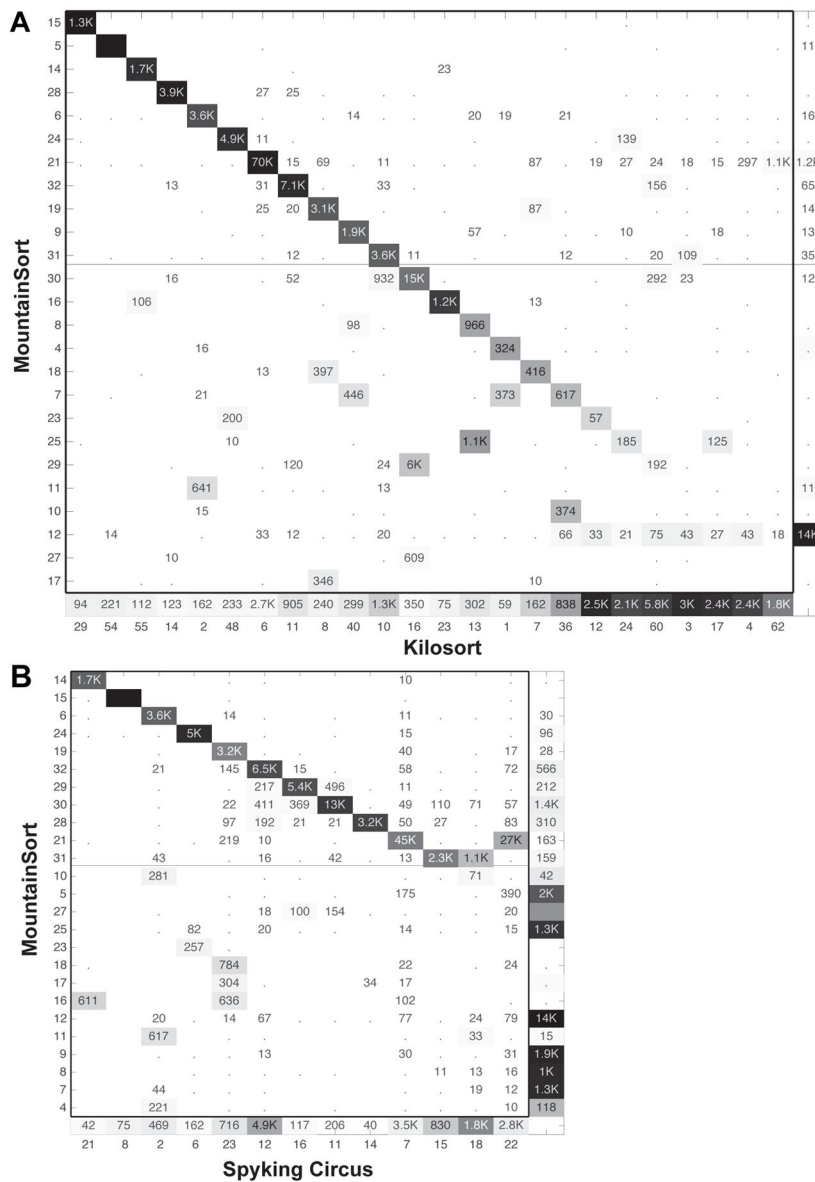


Figure 5. Confusion matrices comparing MountainSort with (A) Kilosort or (B) Spyking Circus applied to the hippocampal dataset of Fig. 3
 See Fig. S2 for details on interpreting confusion matrices. For readability, entries with fewer than 10 events are marked with a period. MountainSort is fully automated, whereas a number of obviously invalid clusters needed to be manually excluded from the KiloSort and Spyking Circus runs before assembling the confusion matrix.

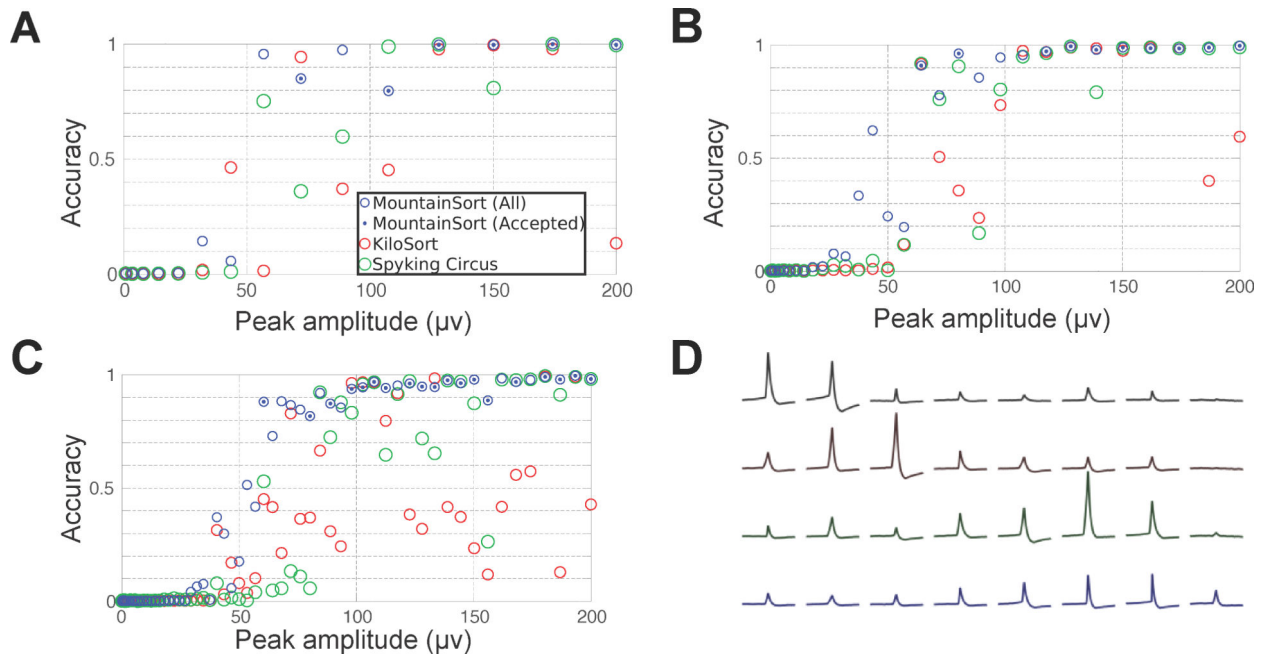


Figure 6. Comparison among MountainSort, KiloSort, and Spyking Circus using three simulated datasets

(A,B,C) Synthetically generated spike waveforms were superimposed at random times on background signal taken from a real dataset (see Methods). Accuracies for each simulated cluster (see Methods) are plotted against the peak amplitude of the waveform. Clusters not matching any true units are not shown. Clusters automatically accepted by MountainSort (using the isolation and noise overlap metrics) are marked using a filled-in blue circle. (D) Synthetic waveforms for the eight clusters with largest peak amplitudes for the simulated dataset shown in (A).

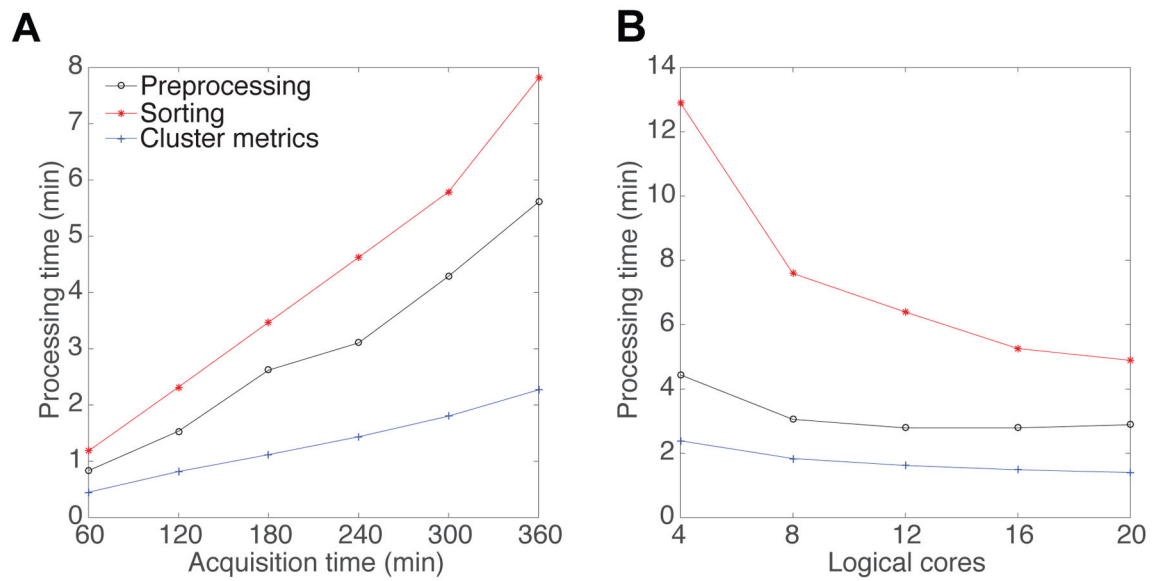


Figure 7. Computational efficiency by processing stage. All plots are from the 16-channel polymer probe dataset from Fig. 4

(A) Processing time versus acquisition time. Each hour of recording contains about 2 million events. (B) Processing time versus number of logical cores used (i.e., processing threads).