Proceedings of the 26th Annual International Conference of the IEEE EMBS
San Francisco, CA, USA • September 1-5, 2004

An Extensible Infrastructure for Fully Automated Spike Sorting during Online Experiments

Gopal Santhanam[1], Maneesh Sahani[2], Stephen I. Ryu[1,3], and Krishna V. Shenoy[1,4]
[1]Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA
[2]Department of Physiology, University of California, San Francisco, 94143 CA, USA
[3]Department of Neurosurgery, Stanford University School of Medicine, Stanford, CA, 94305, USA
[4]Neurosciences Program, Stanford University School of Medicine, Stanford, CA, 94305, USA

Abstract— **When recording extracellular neural activity, it is often necessary to distinguish action potentials arising from distinct cells near the the electrode tip, a process commonly referred to as "spike sorting." In a number of experiments, notably those that involve direct neuroprosthetic control of an effector, this cell-by-cell classification of the incoming signal must be achieved in real time. Several commercial offerings are available for this task, but all of these require some manual supervision *per electrode*, making each scheme cumbersome with large electrode counts. We present a new infrastructure that leverages existing unsupervised algorithms to sort and subsequently implement the resulting signal classification rules for each electrode using a commercially available Cerebus Neural Signal Processor. We demonstrate an implementation of this infrastructure to classify signals from a cortical electrode array, using a probabilistic clustering algorithm (described elsewhere). The data were collected from a rhesus monkey performing a delayed center-out reach task. We used both sorted and unsorted (thresholded) action potentials from an array implanted in pre-motor cortex to "predict" the reach target, a common decoding operation in neuroprosthetic research. The use of sorted spikes led to an improvement in decoding accuracy of between 3.6 and 6.4%.**

Keywords— **spike sorting, extracellular, multi-unit, unsupervised classification, real-time, neural prosthetics.**

I. Introduction

In systems neuroscience, electrophysiology experiments are traditionally conducted with the goal of understanding how neurons participate in the context of stimulus driven tasks. When investigating the response properties of a neuron, the emission of an action potential ("spike") is usually the signal of interest and the remainder of the waveform is noise. The procedure of spike sorting is to infer the times at which one or more neurons emit spikes by examining the voltage deflections on a set of recording electrodes. A good review of the challenges associated with this problem can be found in [1].

Recently, there has been a push for implanting large numbers of immovable electrodes (100s) for both neuroscience and neuroprosthetic research. The electrodes' locations are fixed and there is little flexibility to increase the signal-to-noise ratio after implantation. Hence, implantable electrodes are manufactured with only moderately high impedances (e.g., 200–500 kΩ) to ensure recordings from at least one neuron, though in practice they typically record from two or more. Sophisticated spike sorting algorithms exist for training and classifying multiple clusters ("units") in low signal-to-noise situations [2], [3], but none of these have been applied across high electrode counts under real-time classification constraints.

Spike sorting can lead to greater information extraction from the brain in the context of online neuroprosthetic development. For example, in the extreme case, it would be highly detrimental to lump two neurons with opposite response properties together. However, there has been a tendency to shy away from spike sorting in this area of research, where overall decoding performance is of primary interest. One recent study recognizes the importance of spike sorting but argues that the task is impractical for large electrode counts given that sorting provides only an incremental performance gain [4]. Some studies sort units on small numbers of electrodes [5], [6], but do so in a semi-automated fashion. Not surprisingly, there can be wide variability in the number of neurons and spikes detected when several different individuals are asked to manually spike sort an identical raw data stream [7].

In this work, we describe a system that allows for fully automated spike sorting during online experiments. Our architecture facilitates the use of any advanced clustering algorithm and provides a distributed framework for processing a batch of electrodes in parallel. The clusters from each electrode are transformed for real-time sorting by a 128 channel window discrimination classifier. We also analyze the neural data in the context of a behavioral task to demonstrate the benefits of spike sorting and to espouse the value of our infrastructure.

II. Methods

A. Basic Platform

A cornerstone of our system is the *Cerebus 128 Channel Data Acquisition System* (Cyberkinetics, Inc.). We chose to use the Cerebus system because its architecture allows for easy interfacing with our design. First, the Cerebus "front-end" amplifies the incoming signals, applies an anti-aliasing

This work was supported by the NDSEG Fellowship (G.S.), NIH grant NS-10414 (M.S.), the Coleman Fund (M.S.), the Christopher Reeve Paralysis Foundation (S.I.R. and K.V.S.), and the following awards to K.V.S.: the NSF Center for Neuromorphic Systems Engineering at Caltech, ONR, Whitaker Foundation, Center for Integrated Systems at Stanford, Sloan Foundation, and Burroughs Wellcome Fund Career Award in the Biomedical Sciences. Please address correspondences to gopals@stanford.edu.

0-7803-8439-3/04/$20.00©2004 IEEE

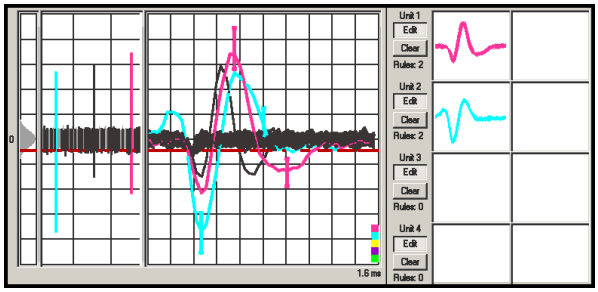

Fig. 1. A screenshot of the Cerebus user interface. Two units are sorted while a third was left unsorted. The remainder of the waveforms are from noise crossing the trigger. The operator sets the trigger threshold (red horizontal line) and places hoops to classify incoming waveforms. The NSP can classify a spike with a round-trip latency of 1–1.5 ms.

filter, and digitally samples each channel (electrode) at 30 kHz. The digitized output is transmitted via a fiber optic link to the Cerebus Neural Signal Processor (NSP).

The NSP can filter the incoming data stream for spike extraction. We chose a fourth order high-pass Butterworth filter with a cut-off frequency of 250 Hz. The NSP compares the filtered data in real-time against a simple threshold trigger – if the trigger is tripped, a 1.6 ms "spike snippet" is sampled. Next, the NSP compares the spike snippet against several sets of time-amplitude window discriminators ("hoops"). Each set of hoops can be used to classify a unit – if a spike waveform passes through all of the active hoops for a specific unit it is classified with that unit number. There can be up to 4 hoops per unit and 5 units per electrode channel. Snippets that do not satisfy any hoops are tagged as unclassified. The spike snippets, with their classification numbers, are broadcast over a private UDP network. The NSP can optionally broadcast the electrodes' 30 kHz raw data onto the network as well.

A desktop PC runs a graphical user interface (GUI) under Microsoft Windows. The GUI can configure the NSP via the UDP network, including modifying the threshold levels and hoops for online classification. Additionally, the GUI receives the spike snippets and plots each snippet, color coded by classification number. A human operator would ordinarily determine the best sets of hoops for each channel by examining the past history of spike snippets. This is known as the training phase. Fig. 1 is a screenshot of the user interface for one particular electrode.

Other commercial online spike sorting products offer more advanced visualization tools, but all of these existing approaches are only semi-automated in that they require human assistance during the training phase to learn a set of sorting parameters. Given these human-specified parameters, the systems then trivially capture and classify all new waveforms.

B. New Architecture

Our approach is to leverage the data acquisition and classification capabilities of the Cerebus system, while automating

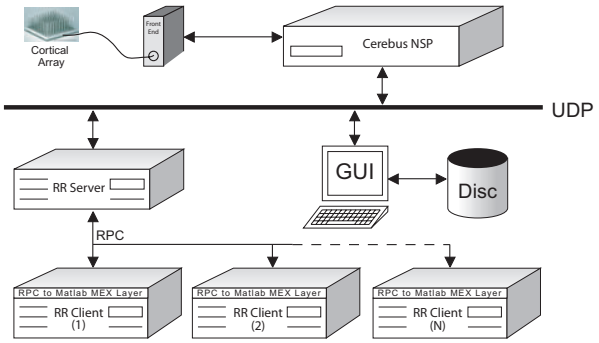

Fig. 2. The system diagram of our "RR" architecture. The Cerebus "front-end" collects raw data from the set of electrodes and interfaces with the Cerebus NSP as usual. The GUI is now relegated to a monitoring role. A second PC, running RTAI Linux (a real-time variant of Linux) is also on the UDP interface – we dub this the "RR server." It can receive data from the NSP as well as manipulate the NSP's configuration. The RR server communicates with data processing clients on a separate network interface. These clients train on the data and manipulate the Cerebus NSP parameters by using the RR server as a proxy. We also wrote a Matlab (Mathworks, Inc.) MEX interface for communication with the RR server; this allows for easy integration of clustering algorithms that are written in Matlab.

the training phase. A block diagram of the system is given in Fig. 2. First, the RR server configures the NSP to broadcast the 30 kHz data stream from all active electrodes. The collection time contain a sufficient number of neural events for the training algorithm. The RR server buffers data from all electrodes in memory.

After collection is finished, an RR client can request a specific electrode's data from the RR server through a remote procedure call (UNIX `rpcgen`). The client processes the data with the algorithm of choice (as detailed in the following sections), identifying the units present on an electrode. There are typically several computational clients communicating with the server on a TCP/IP network. Each electrode or group of electrodes can be farmed out to one of these clients for parallel processing. This is a key feature since parallelization can dramatically reduce the overall time to train the spike sorter across all of the electrodes. We used generic Pentium 4, 3.0 GHz computers with 2 GB of RAM for the RR server and the three accompanying RR clients.

Once an electrode's data is processed, the client uses the clustering information to generate hoops for online classification by the NSP. The sorting clients relay the new threshold level and hoops to the NSP via the RR server. The NSP subsequently classifies all incoming neural events based on these hoops.

C. Spike Clustering Algorithm

We use methods described in [2] to identify the shapes of action potentials associated with different cells in the recording, prior to on-line classification of spikes using the hoops of the Cerebus NSP. This training algorithm was run in Matlab and interfaced with the RR server using compiled MEX functions. We summarize the algorithm here, but refer the reader to [2] for more details. The objective of the

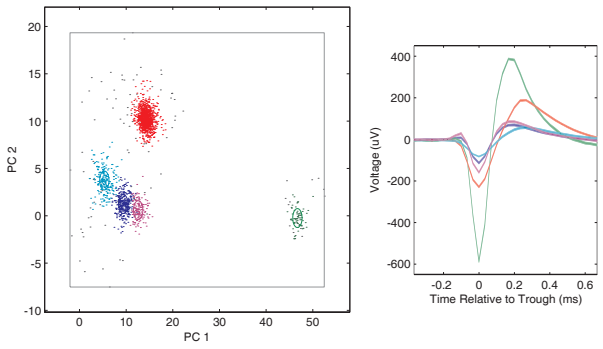

Fig. 3. Clustering results from electrode `G20040117.22`. Projections into the 2-dimensional principal subspace (after peak alignment and noise whitening) are shown (left panel). Median waveforms for each cluster demonstrate the difference in unit shapes in the temporal domain (right panel). The false positive and miss rates for four clusters are each less than 5% when examining the *a posteriori* cluster assignment probabilities in the training set. However, only two of the units could have been reasonably sorted using hand-positioned hoops. Note that the pre-processing of snippets described in the text is essential to cell identification; conventional principal components estimated from unprocessed data do not reveal the differences between the three lower-amplitude action-potential shapes.

algorithm is to estimate the number of sources (neurons) that contribute to the observed signal, and to characterize the distribution of action-potential shapes that each source produces.

The data are first high-pass filtered to eliminate local-field fluctuations, and a threshold is chosen relative to the RMS of the filtered signal. A snippet is sampled around each threshold crossing, but snippets that do not match a predefined shape heuristic are discarded. The remaining snippets are shifted in time to align their peaks. The recorded signal is also sampled at times where the RMS-derived threshold was not exceeded, so as to build an estimate of the covariance matrix of the noise. The extracted snippets are noise-whitened and the principal components of the transformed ensemble estimated by a fitting technique that is robust to outliers. Finally, the snippets are projected into the corresponding 4-dimensional principal subspace, where a mixture model is fit to the data by maximum-likelihood, using a "relaxation" variant of Expectation-Maximization that reduces the chances of converging to local maxima. The particular relaxation scheme employed allows model selection to be integrated into the fitting procedure, thus automatically identifying the number of cells.

Fig. 3 shows the results on a two minute segment of neural data.

D. Hoop Design for Online Classification

Given the mixture model derived by the spike-clustering algorithm, each action-potential snippet can be assigned to the cell from which it is most likely to have originated. However, this operation cannot be carried out on the standard Cerebus NSP hardware. We propose a novel method that uses the probabilistic assignments from the training set to generate hoops for each cell so that the Cerebus NSP can classify new snippets in real-time[1]:

1) Choose the cluster whose waveforms have the highest power about their peak.
2) Given the set of snippets for this cluster, for each time point consider a hoop whose amplitude window encompasses a fixed multiple of the interquartile range of snippet samples at that time point. Center the windows about the median voltage at the respective time point. This non-parametric metric minimizes the effect of outliers in a given class.
3) Select the hoop from those considered at all time points that minimizes the false positive rate from other neural events in the data stream. Continue this process until there are no false positives remaining or the four available hoops are exhausted.
4) Remove all events that have been correctly classified by this set of hoops. Since the hoop selection is non-optimal and is not as robust as the original clustering, there can be many unclassified neural events remaining for this cluster (i.e., misses). These events continue to remain in the training data since they will impact the hoop selection for other clusters.
5) Repeat steps until all clusters have been assigned hoops.

Although our process of choosing hoops is not optimal, it is a computationally-efficient greedy algorithm. It implements an intuitive heuristic for setting hoops from a set of tagged waveforms.

We added an extra heuristic to reduce the leakage of false positives into legitimate classifications. We used the first set of hoops to extract mostly unsortable activity that crosses threshold. Four hoops are placed at equispaced time points shortly after the threshold crossing. Their amplitude windows are twice the threshold level of that channel, centered about zero volts. We call this the "hash unit." The NSP classifies units in a prioritized fashion and all classifications are mutually exclusive. Hence, the hash unit can reduce the false positive rate at the expense of miscatagorizing other spikes into the hash unit.

Fig. 4 shows the median waveforms of each unit along with the hoop settings. This is the final result from the clustering and hoop design process.

E. Data Collection and Analysis

We analyzed data from a rhesus monkey trained to perform delayed center-out reaches to visual targets presented on a fronto-parallel screen. The monkey started each trial by touching a central target. After 250–500 ms, a peripheral target appeared on the screen. Following a 200–750 ms "plan period," the monkey was instructed to reach to the

[1]Note that since the NSP does not perform any snippet alignment before classifying, all training spike snippets are locked to NSP threshold crossings for hoop design.

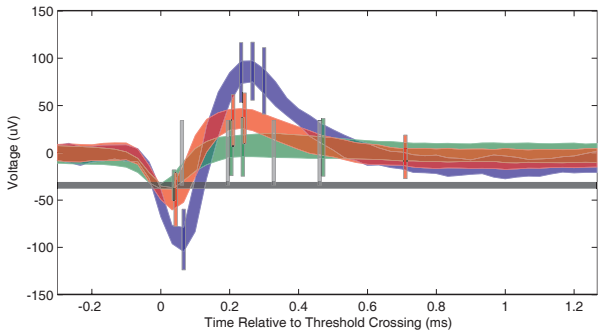

Fig. 4. Threshold and hoop design for three clusters on electrode `G20040202.14`. Waveforms are bounded by 1.5 times the interquartile range, centered about the median. Hoop positions are graphed with a slight jitter along the x-axis to provide visibility when hoops overlap. Features of note include: the hash unit (gray hoops) captures most of the green multi-unit cluster; the red unit registers ∼10% false positives due to the green unit and ∼20% misses due to the hash unit.

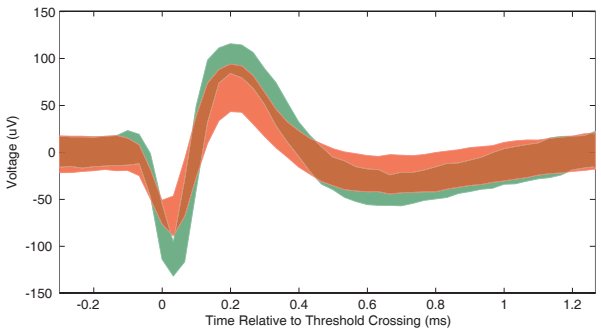

Fig. 5. Waveforms from two clusters with a bound of two times the interquartile range, centered at the median. These units are easily separated by the clustering algorithm, with low false positive and miss rates. While the median shapes are distinct, a hoop-based classifier struggles with the data due to the spread of the waveforms. Hoops placed for the green unit capture 26.7% false positives from the red unit even though clustering algorithm estimates false positives at less than 5%. Data were taken from electrode `G20040312.21`.

target. The animal received a liquid reward after holding the peripheral target for 200 ms. On randomly-interleaved trials the monkey was shown the peripheral target but not cued to reach. We recorded arm position (Polaris, Northern Digital, Inc.) and neural data from a 100 electrode chronically implanted electrode array (Cyberkinetics, Inc., impedances nominally 200–500 kΩ). Animal protocols were approved by the Stanford University Institutional Animal Care and Use Committee.

After testing and verifying the entire RR system, we investigated the benefits of sorting by running analyses to ascertain how well target location can be estimated from plan period spike rates for a single trial. Given a particular target location, the distribution of spike rates for each trial was modeled as a multivariate Gaussian. We employed maximum likelihood methods (similar to [8]) to determine the highest probability target location for a given trial. Either sorted data or threshold crossings were input into the estimator. All dimensions were taken to be independent. We obtained classification percentages for each day's session through leave-one-out cross-validation.

III. Results and Discussion

A. Clustering and Classification

The two key parameters for our algorithm were the threshold level and hoop extent; these were set to 3.5 times the RMS of the filtered data and 3.73 times the interquartile range, respectively. The parameters were empirically determined to provide adequate results.

The traditional problem with testing spike sorting algorithms on real neural data is that there is no measure for the ground truth. As an alternative, the training algorithm uses the *a posteriori* probability densities to calculate a false positive and miss probability for each cluster. The cluster is said to be well-isolated if each type of misclassification probability is under 5%. For our `G20040202`, `G20040312`, and `G20040330` data sets, there were 62, 40, and 41 units that fit this criteria, respectively.

We then asked if these units are still well-isolated when classified with hoops. We computed the false positive and miss rates for hoop classification by comparing against the initial clustering results. Since the original clustering algorithm is taken to be the ground truth, we should only consider neurons that were previously deemed well-clustered. For the same three data sets, 46, 33, and 25 units had false positive and miss rates less than 5% when sorted with hoops. However, this comparison does not include the effect of misclassification against known noise. If we lift this exemption, the noise heavily influences the misclassification rates and many fewer hoop classifications satisfy our goodness criteria.

Ultimately, the hoop-based classifier performed well but did not achieve exceptional results. There will either be extraneous noise sorted with legitimate units or a loss of spikes into the hash unit. For example, our hoop-based system is unable to reliably sort 5 units on an electrode as shown in Fig. 3. Nevertheless, the overall sorting performance was assessed to be qualitatively equivalent to human selection of the hoops; often an individual may feel he is selecting acceptable hoops, but he is unable to fully appreciate the underlying clustering of the data. Fig. 5 provides an extreme limit case where hoop-based sorting breaks down.

Our infrastructure is highly effective in terms of training time. The clustering algorithm takes approximately 20 seconds per electrode, and we sorted 96 electrodes in 10 minutes with three RR clients. This is at least as fast as human-assisted training, but the strength of our architecture is its scalability and repeatability for very large electrode counts. Training time can be reduced by simply adding more RR clients.

B. Target Location Estimation

Next, we performed a target estimation analysis to verify that spike sorting provides greater information extraction.

TABLE I
DECODING PERFORMANCE IMPROVEMENT DUE TO SORTING

Data Set	# of Tgts,Elec.	Unsorted Perf.	Sorted Perf.
G20040329	8,36	64.6%	70.7%
G20040330	8,35	63.3%	69.7%
G20040413	16,35	74.8%	79.9%
G20040417	8,48	90.4%	94.0%
G20040421	8,42	83.8%	89.1%

For each day's data, we excluded electrodes that did not have two or more clustered units as determined by our training algorithm. For our task, the estimation performance asymptotes as the number of electrodes is increased, even if the additional electrodes have only unsortable neural activity. To illustrate the the benefits of spike sorting we biased the simulations by considering only sortable electrodes. We suggest that this biasing would not be necessary for a more challenging task (see [4] where more performance was gained by spike sorting).

Spiking rate was calculated for each unit (or electrode for the unsorted simulations) in a 150 to 350 ms window following the peripheral target presentation. The results of the maximum-likelihood estimator are summarized in Table I. We found a performance increase between 3.6 and 6.4% when using spike sorted units for classification. The increase was dependent on the following parameters: model training size, spike integration window, and electrodes dropped. Searching this entire space of parameters is intractable. However, we can confidently report that in the various scenarios that we tested spike sorting resulted in at least the same or better estimation performance. On two occasions, we compared the automated sorting architecture and hand-optimized hoop locations; the two methods were nearly equivalent in performance.

IV. CONCLUSION

We demonstrated that fully automated spike sorting for laboratory experiments involving hundreds of neural electrodes is practical with present-day technology. Our architecture facilitates use of unsupervised clustering algorithms for configuring existing real-time spike classifiers. Furthermore, we also demonstrated that the performance of a reach target estimator is improved when using sorted information as opposed to threshold crossings. While the performance improvement is not stellar, it is gained with little expense. The infrastructure, once installed, is trivial to run before every day's experiment, and it is extensible past the point where rapid, consistent, human-assisted sorting of hundreds of electrodes becomes untenable. Furthermore, the training stage is truly quantifiable and can serve as a more robust daily record of the neural implant's stability.

Our current architecture is designed to exploit the real-time classification capability of the Cerebus NSP. However, spike-shapes can be more accurately sorted using projection techniques similar to those used in the clustering algorithm described here. To this end, we have preliminary plans for a revised architecture. It will implement real-time classification with more sophisticated algorithms while still using off-the-shelf equipment for data acquisition.

ACKNOWLEDGMENTS

We thank Byron Yu for assisting with data collection and Missy Howard for expert surgical assistance and veterinary care. We also thank Afsheen Afshar and Aaron Batista for valuable editorial comments on this manuscript.

REFERENCES

[1] M.S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network: Comput. Neural Syst.*, vol. 9, no. 4, pp. 53–77, Nov. 1998.

[2] M. Sahani, *Latent Variable Models for Neural Data Analysis*, Ph.D. thesis, Computation and Neural Systems. California Institute of Technology, 1999.

[3] S. Shoham, M.R. Fellows, and R.A. Normann, "Robust, automatic spike sorting using mixtures of multivariate t-distributions," *Journal of Neuroscience Methods*, vol. 127, pp. 111–122, 2003.

[4] J.M. Carmena et al., "Learning to control a brain machine interface for reaching and grasping by primates," *PLoS Biology*, vol. 1, pp. 193–208, Nov. 2003.

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

[6] D.M. Taylor, S.I. Helms-Tillery, and A.B. Schwartz, "Direct cortical control of 3d neuroprosthetic devices," *Science*, vol. 296, no. 3, pp. 1829–1832, June 2002.

[7] F. Wood, M.J. Black, C. Vargas-Irwin, M. Fellows, and J.P. Donoghue, "On the variability of manual spike sorting," *IEEE Transactions on Biomedical Engineering, special issue on Brain Machine Interfaces*, 2004, in press.

[8] K.V. Shenoy et al., "Neural prosthetic control signals from plan activity," *NeuroReport*, vol. 14, pp. 591–596, 2003.