A Pulse-Based Feature Extractor for Spike Sorting Neural Signals

Christy L. Rogers, John G. Harris, Jose C. Principe, and Justin C. Sanchez

Abstract—Spike sorting is often required for analyzing neural recordings to isolate the activity of single neurons. Wave shape analysis in the spike sorting procedure provides a means to detect spikes while minimizing the influence of false alarms. As neural recording techniques allow for recording hundreds of electrodes, power is too limited in neural implants for current spike sorting algorithms. Even with spike sorting at the back-end where more power is available, the bandwidth is too limited to transmit enough information for current spike sorting techniques. A low-power pulse-based feature extractor presented in the paper is a solution to the bandwidth bottleneck. It reduces the bandwidth of the neural signal by several orders of magnitude while preserving enough information for spike sorting.

I. INTRODUCTION

Many neuroscience applications are devoted to the analysis of spike trains, which reflect the firing of individual neurons. One application is brain-machine interfaces (BMI) [1] which, for example, may be used to control robotic devices based on extracted information from neural recordings. Current instrumentation technology and surgical procedures allow for recording from hundreds of electrodes at once where each electrode contains spikes from multiple neurons. Thus, neuroscientists must detect the spikes on each electrode and then perform spike sorting to classify the originating neuron for each spike. The bottleneck is how to transfer the large bandwidth data streams (one channel is typically sampled at \(25KHz\), 16-bits) without requiring the subject to be tethered with wires from the electrode to a signal processing unit. The difficulty in data reduction lies in the requirements of small size and low power for implanted circuitry. The size is constrained due to space on the subject and low power is necessary because of the difficulty of charging or changing implanted batteries and power dissipation over 80 mW/cm\(^2\) has been reported to cause general tissue damage [2].

Neural signal data reduction is a classical problem in neuroscience with many proposed algorithms in the literature. Popular data reduction methods include spike detection followed by different options to reduce the data. One option is to wirelessly transmit a clip of the raw waveform surrounding the spike for spike sorting outside the subject where power and size constraints are less stringent [3]. Another option is to extract and send the features themselves [4], but how to transmit enough high-quality features wirelessly at low-power is problematic. The least bandwidth intensive options are to only transmit spike times [5], [6] or binned spike counts but these options do not allow for spike sorting.

Our lab previously proposed to encode neural signals in a biphasic pulse train (amenable to low-power wireless transmission) for reconstruction on the back-end and traditional spike sorting methods applied [7]. The proposed pulse-based feature extractor provides further data reduction by using a reduced pulse train directly for spike sorting instead of first reconstructing the signal. The proposed method offers an intriguing compromise between spike sorting error and bandwidth/power. Our results suggest this pulse-based feature extraction method reduces the required power and bandwidth compared to traditional methods.

Section II provides an overview of the pulse-based feature extractor which is used for spike sorting as explained in Section III. Matlab spike sorting simulation results using our feature extractor compared to a traditional spike sorting software, Spike2, are discussed in Section IV.

II. PULSE-BASED FEATURE EXTRACTION

A. Algorithm

Instead of transmitting the raw neural waveform, the pulse-based feature extractor encodes information about each spike in a biphasic pulse train in order to reduce the bandwidth required to transmit the spike trains. The encoding scheme uses pulses based on an area per time threshold to represent neural spikes while the noise is mostly disregarded. Data reduction is significant since spike occurrences are sparse. A block diagram for the biphasic encoding system is shown in Fig. 1. If the output of the integrator minus the leak, \(z(t)\), reaches the positive threshold of the comparator, \(\theta^+\), the output of that comparator raises and resets the integrator after a short delay, \(\tau\), in the feedback loop. Similarly, if the \(z(t)\), reaches the negative threshold, \(-\theta\), the output of that comparator raises and also resets the integrator. The leak value sets the cutoff frequency for the low-pass filter formed with the integrator and efficiently filters the noise. This leak value along with the proper threshold settings then allows the system to dedicate most of the pulses to represent the neural spikes. The timing of two consecutive pulses must satisfy the following equation:

\[
\frac{1}{C} \int_{t_i+\tau}^{t_{i+1}} x(\Delta) e^{-\frac{\Delta - t_{i+1}}{RC}} \, d\Delta = \theta_i
\]  

(1)

where \(\theta_i \in \{-\theta, \theta\}\) and \(C\) is related to the integration capacitor and the \(R\) is related to the leak value.

The front end of the feature extractor generates and transmits pulse train features for spike sorting with a trade-off between spike sorting performance and communication...
bandwidth. In the pulse-based feature extractor three parameters determine the bandwidth: leak value, threshold magnitude, and integration time constant. As this system is originally intended for analog VLSI hardware, the integrator is simulated as a capacitor with a value of 10 pF. The leak value and threshold values can be adjusted after fabrication and their effect on bandwidth and spike sorting performance is explored. Increases in the leakage and threshold values both decrease the bandwidth as shown in Fig. 2. While more than one leakage and threshold combination will give the same bandwidth, they do not necessarily provide the same sorting error as they do not preserve the same information. Fig. 3 shows sorting error (described in Section IV) versus minimum bandwidth for leakage and threshold combinations at three different SNRs. The plots show the inverse relationship between bandwidth and sorting error.

Once the pulse trains have been transmitted, a classifier can performs spike sorting outside the body where power issues are not so critical. The encoded pulses for each spike serve as a spike signature and a pulse-based spike sorting algorithm is used to classify the spike. The classifier would be trained once in the initial setup and then could be periodically retrained by sending short segments of the raw waveforms (by adjusting the leak and threshold values) from one electrode at a time.

B. Circuitry

The feature extraction algorithm was originally intended for low-power analog circuitry so it can be implanted as part of a neural recording system but the algorithm can also be implemented on a low power digital chip if implantation is not required. The feature extractor circuit, shown in Fig. 4, takes a current input and encodes the neural signal’s shape in a biphasic pulse train using a leaky integrate and fire (LIF) neuron, a simple extension (such as adding a resistor in parallel to the integrator capacitor for the leakiness) of the biphasic IF neuron [7]. The LIF neuron integrates the signal and then produces a positive pulse when the integrated signal rises above one threshold and a negative pulse when it falls below a second threshold. The leakiness of the LIF sets an area per time threshold which filters out noise while preserving the spikes. This allows the noise in the signal to only trigger an occasional stray pulse, and thus keeps the bandwidth and power consumption even lower. The slightly simpler IF neuron circuit consumes an average of 100 μW of power [7] and the LIF extension should consume only slightly more power.

III. SPIKE SORTING ALGORITHM

Spike sorting is a classical problem in neuroscience, with many proposed algorithms in the literature [8]. Popular spike
sorting methods, such as matched filters and PCA, are too computationally intensive for implanted devices. Thus, the spike sorter is pushed to the back-end where power and area constraints are less stringent.

For our feature extractor the signals are biphasic pulse trains. While spike train distortion metrics have been studied extensively in various fields, many methods are computationally complex and far from real time such as the edit distance [9]. Another idea is to low-pass filter the pulse trains so more traditional signal processing can be applied [10]. The spike sorter used for the pulse-based feature extractor similarly convolves the pulse train with a Gaussian function, where the \( \sigma \) determines if the detector is more of a coincidence detector (\( \sigma \) much smaller than the interpulse interval) or a pulse count detector (large \( \sigma \)). A Gaussian function was chosen over an exponential as it is more concentrated around the peak allowing the \( \sigma \) to better control the detector type. Once the pulse train is convolved with the Gaussian it is then compared to each user defined neuron template. The template with the lowest MSE is a match unless it exceeds the maximum allowed MSE and then it is considered noise.

IV. RESULTS

A. Data

The pulse-based feature extractor algorithm was tested with neural recordings from Bionic's 128-channel hardware neural signal simulator. The use of a neural signal simulator allows the ground truths, the time of each spike and which neuron it came from, to be known. The neural simulator outputs a repeated 11 s pattern of spikes from three different action potentials with amplitudes of 100 \( \mu V \) - 150 \( \mu V \) and a width of 1 ms. The interspike interval is 1 s for 10 s and then reduces to 10 ms for 1 s of burst firing. To increase the number of neurons on one channel the reference was chosen as another channel instead of ground. The referenced channel was carefully chosen to be a 5 ms delayed version of the first channel. In this manner, the simulated neural signal contains spikes from six different neurons with no superimposed spikes which are not addressed in this work since they are problematic for all spike sorting algorithms.

The UF bioamplifier [11], with a gain of 100, was used to amplify the neural simulator output. The amplified signal was then digitized at \( \sim 24.4 \) KHz and 34.6 s were captured with a digital logic analyzer. The average spike firing rate for the data set is 19 Hz. The signal's SNR is about 30 dB. A portion of the signal during bursting with all six neural spikes is shown in Fig. 5(A).

B. Spike2

Spike2, a popular commercial program that can spike sort offline, is used as a comparison to the feature extractor's spike sorting performance. Spike2 first performs crude spike detection by capturing windows around events that cross a user defined threshold(s). Then, spike sorting is performed with a combination of template matching and a PCA based cluster cutting. This process requires the user to select many parameters during the template setup such as the number of templates and allowable variation within the template. Spike2 provides the user with an interactive visual display to assist in setting the spike sorting parameters. The parameters were set by an expert in the field with the same procedures used in typical experiments.

C. Matlab Simulations

Matlab was used to simulate the pulse-based feature extractor and its spike sorter before moving to silicon. The LIF was first set with a threshold and leakage value such that its spike sorting error was similar to Spike2's which resulted in a bandwidth of 455 pulses/s. Fig. 5(B) shows examples of the biphasic output for spikes from two different neurons. The regions between spikes did not have any pulses. The biphasic output was then spike sorted with the results shown in Table I along with the results from Spike2. The results are divided into each neuron class and the percent correctly classified (true positives, \( tp \)) and the false positives (fp) which are spikes incorrectly classified as from that neuron. The best case is 100% \( tp \) and 0% fp. The percent error is calculated with equation 2.

\[
\text{\% error} = \frac{\sum \text{missed spikes} + \sum \text{false positives}}{\sum \text{total number spikes}} \times 100 \%
\]  

At 455 pulses/s the feature extractor had 6.8% error compared with Spike2 which had 6.1% error. While maintaining a similar classification error to traditional sorting with Spike2, the feature extractor requires much less bandwidth with only 455 pulses/s compared to 300 Kbps for a traditional 25 KHz sampled signal at only 12-bits. 1 pulse/s is equivalent to 1bps. UF's biphasic output for reconstruction on the back-end would require 71.9 Kbps/s. The pulse-based feature extractor can reduce its bandwidth even further if more sorting error can be tolerated or increase its bandwidth to lessen sorting errors. The two are inversely related.

At 680 pulses/s the feature extractor actually outperforms Spike2 for this data set. More data simulations need to be performed across different SNRs and data sets to see if the trend continues, but one possible explanation is the feature extractor preserves the important information in distinguishing between spikes while losing extraneous information.

Neuron 2, the smallest spike, is one of the hardest to classify for both sorters. It is more poorly classified with the feature extractor at the lower bandwidth, 455 pulses/s, because there are not enough pulses to properly represent the spike. Increasing the bandwidth to 680 pulses/s reduces the sorting error by increasing the pulses across all spikes but at the cost of over representing the larger spikes.

D. Current Work

Currently we are working toward using real neural data to test the feature extractor; however, as others in this area have stated, it is difficult to test a spike sorting algorithm with real data sets because the ground truths are not known and must be determined through existing spike sorting procedures. Thus, the new algorithm is penalized for its own errors as well as for the ground truth errors. Therefore, some groups
have solely focused on developing realistic simulated data where the ground truths are known with complete accuracy.

To further reduce the bandwidth of the feature extractor a refractory period for the pulse output and an adaptive threshold are being studied [12]. The refractory period would set a maximum firing rate and the adaptive threshold would keep the pulse rate from saturating and information being lost, similarly to the biological neuron. The adaptive threshold helps to even out the number of pulses for different amplitude spikes to keep more similar amounts of information for all spikes allowing more uniform sorting performance across neurons without such a significant increase in bandwidth.

V. CONCLUSION

UF’s feature extractor dramatically reduces the required wireless bandwidth while remaining low-power, \( \sim 100 \mu W \), without sacrificing the ability to spike sort. The preliminary simulation results show the pulse-based feature extractor preserves enough information to allow for similar spike sorting results as Spike2 with over two magnitudes of order less bandwidth, 455 pulses/s compared to 300 Kbps. Further bandwidth reductions can be obtained if the increased sorting error can be tolerated. Therefore, the pulse-based feature extractor appears to be a good candidate to solve the data reduction problem while still allowing for spike sorting.

ACKNOWLEDGMENT

Supported by University of Florida Presidential Fellowship and DARPA sponsor grant #N66001-02-C-8022.

TABLE I

<table>
<thead>
<tr>
<th>Spike Sorting Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Extractor (455 pulses/s)</td>
</tr>
<tr>
<td>neuron</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>avg</td>
</tr>
</tbody>
</table>

REFERENCES