Real Time Algorithms for Sharp Wave Ripple Detection

Ankit Sethi[1] and Caleb Kemere[1,2]

***Abstract*— Neural activity during sharp wave ripples (SWR), short bursts of co-ordinated oscillatory activity in the CA1 region of the rodent hippocampus, is implicated in a variety of memory functions from consolidation to recall. Detection of these events in an algorithmic framework, has thus far relied on simple thresholding techniques with heuristically derived parameters. This study is an investigation into testing and improving the current methods for detection of SWR events in neural recordings. We propose and profile methods to reduce latency in ripple detection. Proposed algorithms are tested on simulated ripple data. The findings show that simple real-time algorithms can improve upon existing power thresholding methods and can detect ripple activity with latencies in the range of 10-20 ms.**

I. INTRODUCTION

When rodents are asleep or awake but not actively engaged in exploration, large amplitude excursions occur in the local field potential (LFP) recorded in the *stratum radiatum* of area CA1 of the hippocampus [1]. These events, known as sharp waves, are associated with fast oscillations in the range of 150-250 Hz [2] [3]. These oscillations, known as Sharp Wave Ripples or simply ripples, are at a higher frequency than other LFP bands such as theta or gamma bands and last for about 50-150 ms [4]. Numerous studies [4] [5] [6] have established the existence of a relationship between SWRs and memory processes. Specifically, the reactivation of patterns of neural activity present during behavior - "replay" - occurs on a compressed timescale during SWRs, and the dynamics of this replay changes during the process of learning. The exact nature of the relationship between SWRs and memory is not comprehensively understood, but it has been shown [7] [8] [10] that the disruption of SWRs by electrical stimulation of the hippocampus impairs the performance of rats in memory tasks. This has been interpreted to imply that disruption of replay prevents the consolidation of short term memories for long terms storage as well as the recall of memories which guide behavior. Studies like these which aim to causally investigate the role of ripples require real time low-latency detection of SWRs. The motivation of this work is to identify optimal algorithms for SWR detection.

A variety of approaches have been used in previous studies involving real-time ripple detection. In [7] [9], simple power measurements were employed in a custom sliding time window followed by a thresholding. One study [8] used an analog version of the same technique. Most recently, in [10] a heuristic envelope estimation is done on the signal followed by a dynamically updated threshold. This study analyzes the performance of these algorithms in simulations and also two new algorithms: one based on an alternate method for digital envelope estimation [11] and one using the CUSUM method for event detection [12]. In order to specifically target the neural activity which occurs during SWR, disruption requires the latency of detection to be as minimal as possible. This implies tradeoffs in the choice of thresholds, window sizes, and other algorithmic parameters. In addition, to achieve continuous real-time operation, the computational requirements of detection algorithms are limited by the processing power of the data processing system. Section II introduces the theoretical framework for the study, including a) the SWR model used for simulated testing b) brief theory of the algorithms under study and c) a description of the testing process, Section III describes a hardware framework that has been set-up for ripple detection, Section IV presents results, and Section V concludes with suggestions for fast real time implementation.

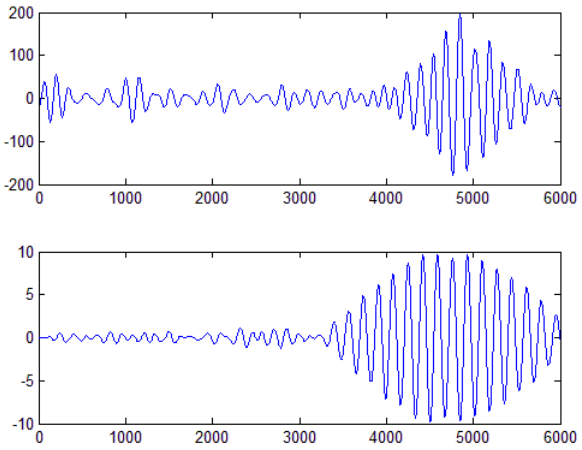

Fig. 1. Recorded (top) vs. Simulated (bottom) Ripple

II. THEORY

A. Sharp Wave Ripple Model

A SWR event is modelled functionally as a sinusoid with amplitude A and frequency f_c (150 Hz $\leq f_c \leq$ 250 Hz), that is amplitude modulated by a second sinusoid with amplitude $A_m = 1$ and $f_m = 1/2t_r$, where t_r is the duration of a ripple event. Gaussian $(N(0,1))$ white is generated and filtered to produce pink noise $(S(f) \propto 1/f)$. The standard deviation of the noise σ is used in conjunction with a user specfied SNR to calculate $A = 10^{SNR/20} * \sqrt{2}\sigma$. The sampling rate

A. Sethi and C. Kemere ({as84,caleb.kemere}@rice.edu) are with the [1]Department of Electrical and Computer Engineering, Rice University and [2]Department of Neuroscience, Baylor College of Medicine.

978-1-4244-7929-0/14/$26.00 ©2014 European Union

f_s^h used is 30,000 Hz to match the sampling rate of current hardware acquisition systems. The SWR is then bandpass filtered (150 Hz - 250 Hz) and downsampled to a working sampling rate $f_s = 1500$ Hz. Figure 1 shows a comparison between an example ripple recorded from a rat hippocampus and a simulated ripple.

B. Detection Algorithms

This section examines algorithms used in past literature as well as two new methods. In previous studies and for all the algorithms presented below, LFP data are first bandpass filtered to the ripple band, typically 150 Hz - 250 Hz. Implementations for filtering have varied, but we use a 4th order IIR Butterworth filter.

1) Sliding Power Window Thresholding (PWT): It is assumed that a preliminary noise estimation has been performed to obtain $\hat{\mu}$ and $\hat{\sigma}$, i.e. mean and standard variation of the background noise. This is implemented prior to start of detection by averaging over a large number of continuously acquired samples or, alternatively, averaging over a large number of randomly selected segments of the signal. The RMS averaged power in a sliding time window of duration t_w is calculated and compared to a threshold chosen as $\mu + K\sigma$, where K is usually set in the range of 3-5. Previously, t_w has been set to values ranging from 10ms to 100ms. To encourage low latency and reduced memory usage, it is fixed at 4 ms in the simulation, which corresponds to the length of a half-cycle ripple at 250 Hz.

2) Heuristic Envelope Based Thresholding (HBT): As developed in [10], this algorithm works by calculating the envelope of the filtered signal, while comparing it to an iteratively updated threshold. Their algorithm is reproduced here for convenience. The smoothened estimate of the signal is given by:

$$v_{est}(n) = v_{est}(n-1) + g(n-1)\left(|x(n)| - v_{est}(n-1)\right) \tag{1}$$

where v_{est} is the smoothed estimate, $g(n)$ is an adaptive gain, and $x(n)$ is the filtered SWR . The gain increases and decreases accordingly as a rising or falling slope is detected:

$$g(n) = \begin{cases} 0.2 & |x(n)| < v_{est}(n-1) \\ \frac{1}{20}\{\sum_{i=1}^{19} g(n-i) + 1.2\} & |x(n)| > v_{est}(n-1). \end{cases} \tag{2}$$

The mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ of the noise are calculated using an iterative update algorithm for $N_{smooth} = 10000$ samples before commencing with detection:

$$\hat{\mu}(n) = \frac{\mu(n-1)\left(N_{smooth} - 1\right) + |x(n)|}{N_{smooth}} \tag{3}$$

$$\hat{\sigma}(n) = \frac{\left(|x(n)| - \mu(n-1)\,|\,- \sigma(n-1)\right)}{N_{smooth}} + \sigma(n-1). \tag{4}$$

A SWR is detected when the envelope estimate $v_{est}(n)$ exceeds $\hat{\mu} + K\hat{\sigma}$. In [10], values of K in the range of 4-6 were used.

3) Envelope Detection Filter Thresholding (EDF): A low latency method that requires only current and previous sample of $\{x(n)\}$ was developed in [11]. It's performance for feasibility for ripple detection was studied. The envelope estimate is given by:

$$v(n) = \sqrt{x(n)^2 + \left(\frac{x(n-1)}{\sin\omega_0} - \frac{x(n)}{\tan\omega_0}\right)^2} \tag{5}$$

where $\omega_0 = 2\pi f_c / f_s$. Instead of estimating f_c online, we set $f_c = 150Hz$ (the assumed minimum ripple frequency). Simulations showed that this does not effect the estimate of the envelope significantly and avoids the complications involved in also estimating the ripple frequency. This also permits an offline calculation of ω_0. Noise estimation was done as in previous algorithms and the estimate was thresholded to detect a ripple event.

4) CUSUM Algorithm Thresholding: The classic technique in sequential analysis [12] was adapted for ripple detection. Low computational complexity, proven history of use in change-point detection, and that fact that using a likelihood function as a parameter is popular but not necessary, helped to motivate this choice. The usual algorithm proceeds by assuming a binary hypothesis and calculating the log-likelihood:

$$LL(n) = \ln\left(\frac{p\left(x(n), \theta_1\right)}{p\left(x(n), \theta_0\right)}\right). \tag{6}$$

Here, the amplitude of the ripple is not known a priori. Also, depending on the depth of the electrodes and their neighborhood, the amplitudes of successive events may exhibit a high degree of variation. In the absence of a p.d.f. that can model the amplitude of a ripple event, a Gaussian p.d.f. that models the null hypothesis, i.e. "no ripple present", is used with a constant offset to make it work with the rest of the update steps. In the usual scheme, a decision function is updated at every sample index as:

$$G(n) = \{G(n-1) + LL(n)\}^+ \tag{7}$$

where $\{z\}^+ = sup(z, 0)$ and $G(0) = 0$. The log-likelihood is replaced with the following term:

$$V(n) = \left(\frac{x(n) - \hat{\mu}}{\hat{\sigma}}\right)^2 - k^2 \tag{8}$$

where k is a constant whose value shall be discussed shortly. This offset is necessary because the logarithm of the Gaussian p.d.f. is always non-negative, whereas the CUSUM algorithm relies on the log-likelihood to be negative for substantive periods, i.e. when there is only noise present. Substituting this to get a modified update rule:

$$G(n) = \{G(n-1) + V(n)\}^+. \tag{9}$$

Whereas earlier $G(n)$ starts accruing for sample indexes where $LL(n) > 0$, under the modified rule it starts accruing once $V(n) > 0$, i.e. when a sample is larger than k s.d. above the estimated mean of the background noise. The decision function is compared to a user-determined threshold, h. A

simple, semi-automated method to get a minimum value of h would be

$$h = \left(\frac{f_s}{2f_c}\right)(m^2 - k^2) \tag{10}$$

which represents the value that $G(n)$ should *at least* add up to, at the end of a ripple half-cycle. The first term on the R.H.S. equals the number of samples in a ripple half-cycle. m represents the average signal level over a half-cycle and k represents the signal level over which a sample is more likely to be a ripple than noise. Here, k, m are multiples of $\hat{\sigma}$. In practice, f_c may be set to 250Hz and m set as $k + 1$. The parameter k should be set between 1-2 to ensure low latency detection. The threshold can be calculated offline. For testing purposes, $k = 2$ and $m = 3$ were used. The noise parameters $\hat{\mu}$, $\hat{\sigma}$ are estimated as previously.

C. Testing

Each algorithm was tested on 500 simulated "ripple events" - here defined as 100 ms of noise followed by a 100 ms long ripple. To test for false positives half of these cases had no actual ripple waveform. They were evaluated for detection latency and the computational cost. Recorded data was used for manual testing to visually confirm efficacy, but could not be used for quantifying performance due to absence of gold standard data. The simulations were done at a "normal" SNR of 8 dB and a "low" SNR of 0 dB.

III. ONLINE IMPLEMENTATION

Open-ephys (`http://open-ephys.org`) is a set of collaborative, open-sourced tools for extracellular recordings with an emphasis on high quality multichannel data acquisition. The open-ephys acquisition board is used in conjunction with its open-source GUI to display, record and save spike activity and LFP data. Pulse Pal (`https://sites.google.com/site/pulsepalwiki/home`) is an open source pulse stimulator designed to deliver precisely timed control signals for neural stimulation. The open-ephys GUI offers a module that interfaces with the Pulse Pal through its API for seamless integration. A ripple detection module for open-ephys can be quickly developed in C++ and added to the open-ephys GUI. Since it is open-source, it is easy to rapidly implement any of the above or a combination of algorithms for use during behavioral experiments.

Figure 2 outlines the flow of data and events using the open-ephys and Pulse Pal hardware.

IV. RESULTS

Figure 3 and 4 show a comparison of the algorithms at. All plots have been downsampled by a factor of 4 for clarity. All four algorithms show similar trends in variation with the False Positive Rate(FPR). The CUSUM algorithm is the method that yields the lowest latency overall. The standard deviation of latency in each case suggests also that the CUSUM algorithm is the most closely centered around the mean, while the HBT, EDF and PWT algorithms have increasingly greater amounts of latency variation. This suggests that the CUSUM is a good choice an algorithm for fast detection with a low variation in detection time. The EDF algorithm improves upon the PWT somewhat while the HBT displays intermediate performance. The Miss Rate (MR) plot shows a big difference between the characteristics of the CUSUM algorithm and the other three. In the previous plot also it is seen that the latency mean for the CUSUM algorithm has the highest slope near the 0 FPR point. These two facts indicate a robustness to variation in the threshold set. This is an important property since threshold setting is a choice that is usually made with limited information and a certain amount of trial-and-error. Similar behavior is observed in the low SNR (0 dB) case. In the case of a noisy electrode, one may expect the 0 FPR latency to increase by 20 ms. Computational overheads of each algorithm have been tabulated in Table I.

V. CONCLUSIONS

The CUSUM algorithm is found to be the most robust method among the techniques investigated. There is however, an absence of a rigorous theoretical method for choosing a threshold. The HBT algorithm has been previously used for

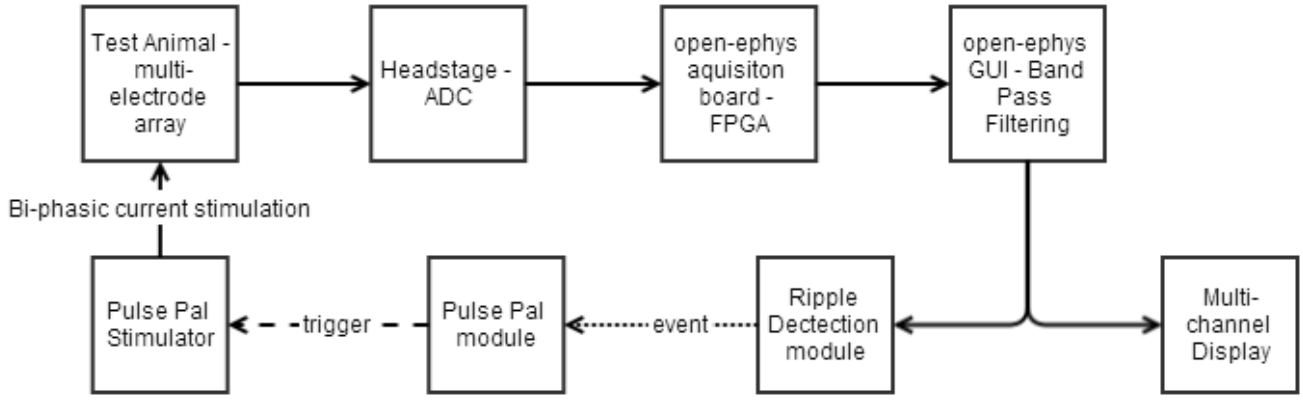

Fig. 2. Flow diagram for ripple detection

TABLE I

COMPUTATIONAL COST

Algorithm	Samples in Memory	Multiply/Add ops. (per time index)
PWT	$t_w * f_s$	1/2
HBT	20	2/4
EDF	2	4/2
CUSUM	1	2/2

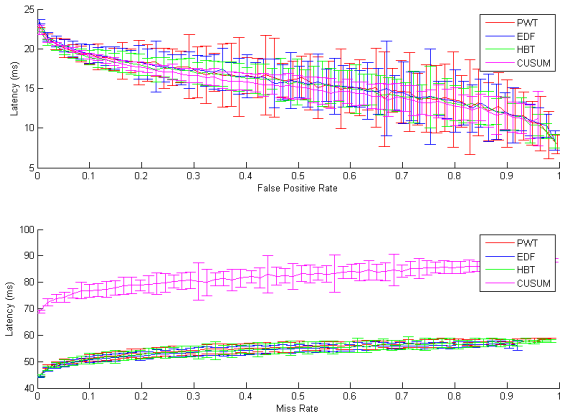

Fig. 3. Ripple detection latency vs. false positive rate/miss rate (8dB)

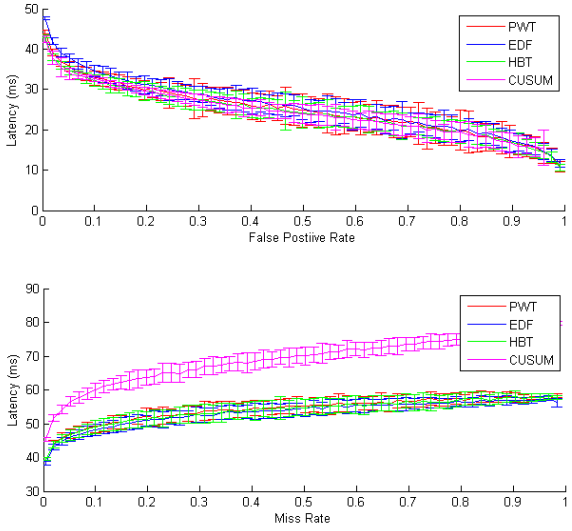

Fig. 4. Ripple detection latency vs. false positive rate/miss rate (0dB)

ripple interruption and it is superior to power window methods although it also relies upon a semi-automated method for choosing the threshold. The EDF algorithm is a very simple, efficient technique that offers reasonable improvement over, or at worst, comparable performance to, the PWT algorithm.

REFERENCES

[1] G. Buzski, L. Lai-Wo S., and C. H. Vanderwolf, Cellular bases of hippocampal EEG in the behaving rat, Brain Res. Rev., vol. 6, no. 2, pp. 139171, Oct. 1983.

[2] G. Buzsaki, Z. Horvath, R. Urioste, J. Hetke, and K. Wise, High-frequency network oscillation in the hippocampus, Sci. , vol. 256 , no. 5059 , pp. 10251027, May 1992.

[3] J. Csicsvari, H. Hirase, A. Mamiya, and G. Buzski, Ensemble Patterns of Hippocampal CA3-CA1 Neurons during Sharp WaveAssociated Population Events, Neuron, vol. 28, no. 2, pp. 585594, Nov. 2000.

[4] J. J. Chrobak and G. Buzski, High-Frequency Oscillations in the Output Networks of the HippocampalEntorhinal Axis of the Freely Behaving Rat, J. Neurosci. , vol. 16 , no. 9 , pp. 30563066, May 1996.

[5] G. Buzski, Two-stage model of memory trace formation: A role for noisy brain states, Neuroscience, vol. 31, no. 3, pp. 551570, Jan. 1989.

[6] M. A. Wilson and B. L. McNaughton, Reactivation of hippocampal ensemble memories during sleep, Sci. , vol. 265 , no. 5172 , pp. 676679, Jul. 1994.

[7] G. Girardeau, K. Benchenane, S. I. Wiener, G. Buzski, and M. B. Zugaro, Selective suppression of hippocampal ripples impairs spatial memory., Nat. Neurosci., vol. 12, no. 10, pp. 12223, Oct. 2009.

[8] V. Ego-Stengel and M. A. Wilson, Disruption of ripple-associated hippocampal activity during rest impairs spatial learning in the rat, Hippocampus, vol. 20, no. 1, pp. 110, Jan. 2010.

[9] M. S. Nokia, M. Penttonen, and J. Wikgren, Hippocampal ripple-contingent training accelerates trace eyeblink conditioning and retards extinction in rabbits., J. Neurosci., vol. 30, no. 34, pp. 1148692, Aug. 2010.

[10] S. P. Jadhav, C. Kemere, P. W. German, and L. M. Frank, Awake Hippocampal Sharp-Wave Ripples Support Spatial Memory, Sci. , vol. 336 , no. 6087 , pp. 14541458, Jun. 2012.

[11] C. Fritsch and A. Iba, Filter for Real-Time Operation, IEEE Trans. Instrum. Meas., vol. 48, no. 6, pp. 12871293, 1999.

[12] P. Granjon, The CUSUM algorithm a small review, class notes for "Decision and Change Detection", Grenoble Institute of Technology, Jun. 2012.