KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Real-time detection of voltage patterns in the brain

Tomas Fiers

Thesis submitted for the degree of
Master of Science in
Biomedical Engineering

Thesis supervisors:
Prof. dr. ir. A. Bertrand
Prof. dr. F. Kloosterman

Assessors:
Prof. dr. ir. R. Puers
Dr. eng. J. Couto

Mentors:
Ir. J. Wouters
Dott. D. Ciliberti

Academic year 2018 – 2019

Abstract – 30 Nov

Contents

Abbreviations

BPF Band-pass filter. See chapter 4.

CA1 "Cornu Ammonis", subregion 1. Region in the hippocampus where voltages are recorded from (see **????**).

CA3 "Cornu Ammonis", subregion 3. Region in the hippocampus (see **????**). CA3 sends many axons (called "Schafer collaterals") to CA1.

FIR Finite impulse response. A linear filter whose output is a convolution of the input signal with some kernel.

GEVal Generalized eigenvalue. See section 5.2.

GEVec Generalized eigenvector. See section 5.2.

IIR Infinite impulse response. A linear filter whose output at each timestep is a function of both input samples and preceding output samples.

IQR Interquartile range. A measure of the spread of a set of one-dimensional values, that is robust to outliers. Difference between the 75th and the 25th data percentile.

KDE Kernel density estimate.

LFP Local field potential. The extracellular electric potential (see **??**).

RMS Root-mean-square. $\sqrt{\langle x_t^2 \rangle}$ for a signal x_t.

RNN Recurrent neural network. See chapter 6.

SNR Signal-to-noise ratio. See section 5.2.

SOTA State of the art. The algorithm currently used for SWR detection, namely an online single channel band-pass filter.

SWR Sharp wave-ripple. The pattern in the LFP that we want to detect in real-time. See **??**.

Symbols

Notation

y	Scalars are denoted in lowercase italic.
$\mathbf{z}$	Vectors are denoted in lowercase boldface.
$\mathbf{A}$	Matrices are denoted in uppercase boldface.
$\langle \cdot \rangle$	Time-average of a signal.
$\lvert \cdot \rvert$	Number of elements in a set; Magnitude.
$\odot$	Elementwise multiplication. ("Hadamard product").
$\sigma(\cdot)$	Sigmoid 'squashing' function, $\mathbb{R} \to (0, 1)$. $\sigma(x) = \frac{1}{1+\exp(-x)}$.
$\tanh(\cdot)$	Hyperbolic tangent, $\mathbb{R} \to (-1, 1)$. $\tanh(x) = 2\ \sigma(x) - 1$.

Signals

$\mathbf{z}_t$	Digitized LFP sample at discrete time step t. $\mathbf{z}_t \in \mathbb{R}^C$, with C the number of channels (i.e. the number of electrodes simultaneously recorded from). Input to an SWR detection algorithm.
o_t	Output signal of an SWR detection algorithm, $\in \mathbb{R}$.
n_t	'Envelope'. Transformation of o_t, so that it is constrained to $\mathbb{R}^+$. Should be high when the corresponding input sample $\mathbf{z}_t$ is part of an SWR segment, and low when it is not. $n_t = \lvert o_t \rvert$ for online linear filters; $n_t = \sigma(o_t)$ for the RNN's of chapter 6.
y_t	Binary target signal, used when training data-driven SWR detection algorithms. We define $y_t = 1$ when the corresponding input sample $\mathbf{z}_t$ is part of an SWR segment, and $y_t = 0$ when it is not.

Measures & parameters

T Detection threshold applied to the envelope n_t. $T \in [\min n_t,\ \max n_t]$. Each threshold T yields a different P-value, R-value, F_1-value, etc.

P Precision. Also known as positive predictive value. The fraction of correct detections, out of all detections.

R Recall. Also known as sensitivity, hit rate, or true positive rate. The fraction of detected reference SWR segments, out of all reference SWR segments.

F_β F-score: weighted harmonic mean of recall and precision. $F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P+R}$. Measures detection performance "for a user who attaches β times as much importance to recall as to precision." [1]

F_1 F-score where recall and precision are weighted equally.

f_s Sampling frequency of a signal.

Chapter 1

Introduction

1.1 Closed-loop brain-computer interfaces

1.2 Sharp wave-ripples

Description

Scientific importance

Biophysics

Closed-loop technology

1.3 Problem statement

1.4 Thesis overview

Chapter 2

Offline labelling of SWR segments

To quantify the performance of a sharp wave-ripple (SWR) detection algorithm, we need an evaluation or 'test' recording, annotated with the segments of time when actual SWR events were present. This section describes how such annotations can be made, and how our data specifically was annotated.

SWR's are an empirical phenomenon of hippocampal area CA1, 'defined' by what their voltage traces look like. In other words, there is no ground truth available to know when SWR's occur. Scientists looking to annotate their LFP recordings with SWR segments therefore have to rely either on judgement calls by human labellers, or on an automated, offline SWR detection algorithm. The former could be considered more subjective, and is definitely more labour intensive than the latter – especially if multiple scientists are consulted to obtain a consensus labelling. Most studies use an automated, offline algorithm to detect SWR segments (see appendix A.1 for relevant quotes from a collection of such studies).

'Offline' here means that SWR detection happens after the recording has been completed, and that there are thus no real-time constraints on the detection algorithm. This means that 1) there are no hard bounds on algorithm execution time, and 2) that the algorithm can use information 'from the future': when deciding whether a recording sample $\mathbf{z}_t$ belongs to an SWR segment, it can consider samples $\mathbf{z}_{t_f}$ that occured after $\mathbf{z}_t$ (i.e. $t_f > t$), instead of considering only past samples $\mathbf{z}_{t_p}$ (where $t_p \leq t$).

2.1 Overview of offline ripple detection

The main steps of the offline SWR detection algorithm that most studies use – such as the ones cited in appendix A.1, and the one of this thesis – can be summarized as follows:

1. Use a single channel of input data; namely from an electrode in the pyramidal cell layer of CA1, where the ripple part of SWR's is most strongly present. (We will denote this voltage signal with z_t);
2. Band-pass filter the recording to retain only 'ripple' frequencies. (We will denote the filter output as o_t);

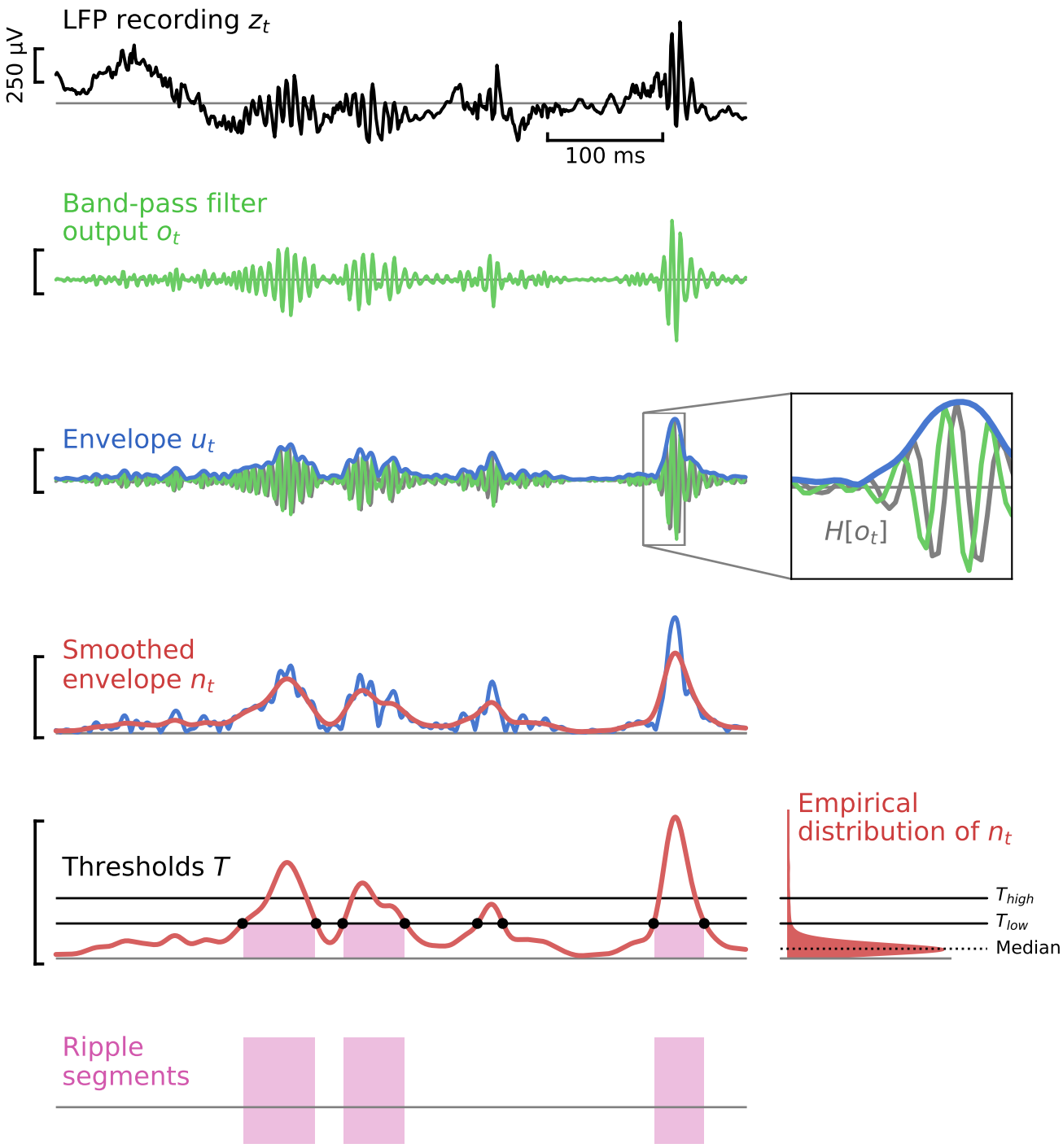


Figure 2.1: **Steps for automated, offline SWR labelling**. See text for details. Each vertical scalebar indicates the same voltage range. $H[o_t]$ denotes the Hilbert transform of o_t. Note its phase lag of 90° with respect to o_t. In the second panel from the bottom, note the two threshold crossings of T_{low} (marked with black dots) that did not result in a ripple segment, because the second threshold T_{high} was not reached. The distribution of the envelope n_t was estimated using the entire dataset (and not just the displayed fragment).

3. Obtain the envelope of the band-pass filtered signal. (We will denote this envelope with n_t);
4. Calculate a 'high' and a 'low' threshold (T_{high} and T_{low}) to apply to the envelope n_t, based on summary statistics of n_t and two custom multipliers (α_{high} and α_{low});
5. Define ripple events as times when the envelope crosses the high threshold (i.e. $n_t > T_{\text{high}}$);
6. Define the start and end time of each such ripple as the closest times where the envelope falls back below the lower threshold (i.e. $n_t < T_{\text{low}}$).

Figure 2.1 visualizes each step, as applied to a fragment of our dataset. Note that this procedure only detects ripples, and not sharp waves.[1]

2.2 Validating parameter choices

Some of the above steps have free parameters (such as the ripple frequency band used for filtering, or the threshold multipliers α_{high} and α_{low}). If not mentioned otherwise, our parameter choices were made as follows.

In a first pass of the offline detection algorithm, ripples were detected using a very broad-band filter and a low detection threshold. This ensured that all 'true' SWR's were included in the detected events set (in addition to many spurious detections).

Next, five neuroscientists were independently asked to decide for each detected event whether it was a sharp wave-ripple or not. This was done through a custom-made web app, an example screen of which is shown in fig. D.1. Note that the labellers could factor multiple recording channels into their decision, including stratum radiatum channels displaying sharp wave activity. Only the events that were labelled as an SWR by at least three neuroscientists were retained.

Finally, when setting the parameters of the eventual offline ripple detection algorithm, the algorithm's output was compared to the decisions made by the neuroscientists. The parameters were then adjusted until the output of the algorithm matched the neuroscientists' decisions reasonably well.

The following subsections describe the detection steps in more detail, and compare our choices of methods and parameters to those made in the literature.

2.3 Band-pass filter design

The wideband voltage signal x_t was band-pass filtered between 100 and 200 Hz, using a linear time-invariant filter with zero output lag.

Many other studies use a higher left bound, of about 140–150 Hz (see table A.1). Using such a high bound in our dataset however resulted in ripples that went undetected (even

[1] Although interestingly, the sharp wave part of sharp wave-ripples was discovered before the ripple part [2, p. 1].

at low detection thresholds), although they were convicingly marked as SWR's by the neuroscientists.

The band-pass filter was designed using the windowed-sinc method, with a Kaiser window (using SciPy's `firwin` and `kaiserord` functions) [3], [4]. The transition width was chosen to be 10% of the bandwidth (i.e. 10 Hz), and the attenuation to be 40 dB, resulting in an FIR filter of order 150 (at a 1000 Hz sampling frequency).

The filter was applied bidirectionally, resulting in a zero-lag output, and a total attenuation of 80 dB. Figure 2.1 shows an example of the filter output in green.

(We cannot easily compare our filter design method with the literature, as most studies do not mention anything about filter design besides the frequency band used).

2.4 Envelope calculation

Next, the instantaneous envelope u_t of the filter output o_t was calculated using its Hilbert transform[2] $H[o_t]$:

$$u_t = \sqrt{o_t^2 + H[o_t]^2}, \tag{2.1}$$

The Hilbert transform delays each frequency component of a signal by 90° (see the gray signal in fig. 2.1) [6]. This means that the envelope of a narrowband signal such as o_t can be easily obtained as the magnitude of the so called 'analytic signal' $o_t + jH[o_t]$, as in eq. (2.1).[3]

After calculating u_t, a final, smoothed envelope n_t was obtained by convolving u_t with a Gaussian kernel (σ = 7.5 ms, support radius of 4σ). Compare the blue (u_t) and the red signal (n_t) in fig. 2.1.

Other studies (such as [7] and [8]) use a "root-mean-square" approach to calculate the envelope of the filter output o_t. In these studies, presumably, the squared signal o_t^2 is smoothed using some kernel (of unspecified type and bandwidth) to obtain the "mean-square" signal of which the square root is taken.

[2] To be precise, H is a discrete approximation to the continuous Hilbert transform $\mathcal{H}$. There are multiple ways to make a discrete approximation to $\mathcal{H}$ [5]. Most of them make use of the discrete Fourier transform, which makes computing H an efficient operation. We used SciPy's `hilbert` function (which, confusingly, does not return the Hilbert transform of its input, but rather its analytical signal), and zero-padded the input signal to the nearest power of 2 or 3 to benefit from the speedup brought by the fast Fourier transform algorithm.

[3] To see why, consider a local approximation of the narrowband signal o_t by a sinusoid $a\cos\omega t$. Its Hilbert transform is then $a\sin\omega t$. From eq. (2.1), the envelope u_t will be locally approximated by the magnitude of the original signal: $u_t = \sqrt{(a\cos\omega t)^2 + (a\sin\omega t)^2} = |a|$.

2.5 Threshold calculation

The two detection thresholds were calculated as follows:

$$T_{\text{high}} = \alpha_{\text{high}} \times \widetilde{n_t} \tag{2.2}$$

$$T_{\text{low}} = \alpha_{\text{low}} \times \widetilde{n_t} \tag{2.3}$$

where $\widetilde{n_t}$ denotes the median of the smoothed envelope n_t. As per the procedure described in section 2.2, we set $\alpha_{\text{high}} = 6.2$ and $\alpha_{\text{low}} = 3.6$. At a median envelope magnitude $\widetilde{n_t} = 17.0\,\mu\text{V}$, this results in thresholds $T_{\text{high}} = 105.4\,\mu\text{V}$ and $T_{\text{low}} = 61.2\,\mu\text{V}$.

Most studies calculate thresholds as follows: $T_{\text{high}} = \overline{n_t} + \beta_{\text{high}} \times \text{std}(n_t)$. Here $\overline{n_t}$ denotes the mean of the envelope, $\text{std}(n_t)$ denotes its standard deviation, and β_{high} is a custom multiplier analogous to α_{high}. For the non-negative, assymetric distributions of envelope signals, using both a measure of center and a measure of spread to define thresholds seems unnecessary (see the distribution of n_t in fig. 2.1), which is why we chose to use only one measure (namely the median).

The detection multiplier β_{high} varies wildly between studies: from 1, over 3, 4, and 5, up until 7 ([7]–[11], respectively).[4] It is clear that such different thresholds will give very different sensitivity-precision trade-offs for SWR detection (the lower thresholds yielding more false positive detections, and the higher thresholds yielding more missed true SWR events).

To compare our thresholds to those in the literature, we calculate the β multipliers corresponding to our chosen α values. Given that our dataset has a mean envelope magnitude of 22.3 µV and a standard deviation of 22.9 µV, we find $\beta_{\text{high}} = 3.63$ and $\beta_{\text{low}} = 1.70$. These values fall near the center of those reported in the literature (see appendix A.1).

2.6 Segment post-processing

Finally, two add-hoc rules were applied to the automatically detected ripple segments. First, segments with only a small gap between them (of less than 10 ms) were joined together. Then, segments of too short a duration (less than 25 ms) were eliminated.

This step is rarely done in other studies. An exception is e.g. [9], were segments shorter than 15 ms were eliminated.

[4] Threshold multipliers cannot be compared precisely. Imagine two recordings with equally powerful ripples. Both recordings will then need an equal threshold T_{high} to detect the same types of ripples. When one of the recordings has a different background 'noise' level or a different ripple incidence rate, it will have different $\overline{n_t}$ and $\text{std}(n_t)$ values. This means that β_{high} needs to change to maintain an equal threshold T_{high}.

Chapter 3

Performance quantification of online SWR detectors

Chapter 2 described how we annotated our dataset with 'reference' SWR segments, using an offline algorithm. These reference segments can now be used to benchmark the performance of an online SWR detection algorithm. This section describes how this is done.

3.1 Generating online detections

Each online SWR detection algorithm studied in this thesis generates an output envelope n_t, where samples n_t of larger magnitude denote a higher belief that the corresponding input sample $\mathbf{z}_t$ is part of an SWR event. (See fig. 5.2A for three example output envelopes).

These output envelopes n_t can be converted to a discrete set of detection times by applying a threshold T to the envelope.[1] Additionally, we require a certain minimum duration between online detections (which we will call the "lockout time", L). The set D of detection times t_d (with $d = 0, 1, ..$) is then:

$$D = \{\, t_d \mid \; n_{t_d} > T \quad \text{and} \quad t_d > t_{d-1} + L \,\}. \tag{3.1}$$

In our analysis, we chose a lockout time L based on SWR durations. Specifically, we set L to the 25-th percentile of the durations of all SWR segments detected in our dataset by the offline algorithm, resulting in a lockout time $L = 34$ ms.

The threshold T is generally set by the user of the online SWR detection algorithm (see the discussion in the next section).

[1] In general, this threshold can be adaptive (based on a moving average of signal statistics, for example), and we should therefore write "T_t" to be precise. In this thesis we only evaluate online detectors using constant thresholds T.

3.2 Quantifying detector accuracy

In the following, we denote reference SWR segments as the closed intervals $s_r = [t_r^{\text{start}},\ t_r^{\text{end}}]$ (with $r = 0, 1, ..$). The set of all reference segments is denoted by $S = \{s_r\}$.

To benchmark the performance of an online SWR detection algorithm (at a certain threshold T), we compare its detections D with the offline reference segments S, and divide both sets into two subsets each: namely detected versus undetected reference segments, and correct versus incorrect detections. Formally:

$$D^{\text{correct}} = \{t_d \mid \exists s_r : t_d \in s_r\} \tag{3.2}$$

$$D^{\text{incorrect}} = D \setminus D^{\text{correct}} \tag{3.3}$$

$$S^{\text{detected}} = \{s_r \mid \exists t_d : t_d \in s_r\} \tag{3.4}$$

$$S^{\text{undetected}} = S \setminus S^{\text{detected}}, \tag{3.5}$$

i.e. correct online detections are contained within a reference segment, and detected reference segments contain at least one online detection. Figure 5.2A shows some example correct and incorrect detections (green and red triangles, respectively).

Next, the number of detected reference segments is counted, and compared to the total number of reference segments. This fraction is the recall R of the SWR detector (also known as sensitivity, hit rate, or true positive rate):

$$R = \frac{\left|S^{\text{detected}}\right|}{|S|} = \frac{\text{\# of detected reference segments}}{\text{total \# of reference segments}} \tag{3.6}$$

Similarly, the number of correct detections is counted, and compared to the total number of detections. This fraction is the precision P of the SWR detector (also known as the positive predictive value):

$$P = \frac{\left|D^{\text{correct}}\right|}{|D|} = \frac{\text{\# of correct detections}}{\text{total \# of detections}} \tag{3.7}$$

Its opposite is the false discovery rate (FDR), which counts the number of so called "false positive" detections:

$$FDR = \frac{\left|D^{\text{incorrect}}\right|}{|D|} = \frac{\text{\# of incorrect detections}}{\text{total \# of detections}} = 1 - P \tag{3.8}$$

Each detection threshold yields a different (P, R)-combination. This is often a trade-off: higher thresholds yield more precise detectors (less false positive detections), at the cost of a lower sensitivity (more missed reference SWR segments). Conversely, lower thresholds generally yield more sensitive but also less precise SWR detectors.

This is why the performance of a detection algorithm is often given for a *range* of thresholds, instead of for just a single threshold: different users of the SWR detection algorithm may prefer different trade-offs between latency and precision. (One user might require a minimal amount of false detections, for example, while another may

want to detect as many SWR events as possible, without minding false detections). When precision and recall are plotted against each other for a range of different thresholds, the result is a so called "PR-curve" (see the lower-left panel of fig. 5.3 for an example). Generally, higher quality detectors have PR-curves that draw nearer to the top-right $(1, 1)$-corner.

The precision P and recall R for a given threshold can be summarized in a single number: the so called F-score. It is defined as:

$$F_\beta = \frac{(1+\beta^2)PR}{\beta^2 P + R}. \tag{3.9}$$

It is the mean of recall and precision (more precisely: their harmonic mean, as precision and recall are ratio's), with recall weighted by β^2. It measures detection performance "for a user who attaches β times as much importance to recall as to precision." [1]

For example, the F_1-score weighs recall and precision equally. When we want to summarize the accuracy of a detector in a single number, we will often choose the *maximum* F_1-score of the detector (over the entire threshold range) as this number.

A note on terminology: we will refer to the precision and sensitivity performance of a detector often simply as the 'accuracy' of the detector. This is a shorthand, and does not refer to the technical definition of accuracy.[2]

3.3 Quantifying detector latency

Besides counting how many SWR segments were detected, we are also interested in how early they were detected. For each detected reference segment $s_r = [t_r^{\text{start}},\ t_r^{\text{end}}]$, we note the first online detection t_d contained within it (reference segments may contain multiple online detections).

The absolute detection latency of this SWR segment is then defined as:

$$L_r^{\text{abs}} = t_d - t_r^{\text{start}} \tag{3.10}$$

We also define a detection latency relative to the duration of the SWR segment in question:

$$L_r^{\text{rel}} = \frac{L_r^{\text{abs}}}{t_r^{\text{end}} - t_r^{\text{start}}} \tag{3.11}$$

Whereas an SWR detector yields only a single precision and a single recall value for a given threshold, it yields two entire sets of absolute and relative detection latencies

[2] Technically, accuracy is the sum of true positive and true negative cases, divided by the total number of cases. This concept does does not readily apply to SWR detection, as there is no straightforward definition of "true negative" cases here. Only false positives ($D^{\text{incorrect}}$), true positives (D^{correct} and S^{detected}), and false negatives ($S^{\text{undetected}}$) are clearly defined.
Note that there are two types of "true positives": correct detections $\in D^{\text{correct}}$, and detected reference segments $\in S^{\text{detected}}$ (with $\left|D^{\text{correct}}\right| \geq \left|S^{\text{detected}}\right|$, because one reference segment may contain multiple detections).

(namely an absolute and a relative latency value for each detected reference segment). This is why we often summarize latency performance using the median of one of these sets. When not otherwise specified, the latency distribution (or its median) corresponding to the threshold with maximal F_1-score is reported.

Chapter 4

Current online SWR detection

This chapter discusses the current state-of-the-art method in online SWR detection. This method is very similar to the procedure for offline ripple detection described in chapter 2: a single recording channel that contains ripples is selected; it is band-pass filtered through the ripple band; and apply a detection threshold is applied to the envelope of the filter output.

The difference is that the SWR detection must now happen under real-time constraints. No future samples can be used when deciding whether the current recording sample belongs to an SWR event. This means that the used band-pass filter (and in some studies, also the envelope estimation method) must necessarily introduce latency.

We first discuss the performance of the online band-pass filters used in the literature, and then shortly discuss online estimation of the filter output envelope.

4.1 Replicating existing filters

We selected three previously used online ripple filters and analyzed their performance. The first two filters are described in the literature. (We gathered a representative sample of original research papers that use online ripple detection – see appendix A.2. Most of these papers do not provide sufficient information to replicate the used filter. The two selected filters are from the two papers that did provide sufficient information.) The third analyzed filter is the default online ripple filter in *Falcon*. (*'Falcon'* is an open-source software package for closed-loop neuroscience, developed and used in the Kloosterman lab, and described in Ciliberti et al. 2017 [12]).

The original analyzed filters use different passbands and sampling frequencies. When we 'replicate' the original filters to apply them to our data, we use a standardized sampling frequency f_s of 1000 Hz. When designing the replica filters, we try to match the gain- and group delay-profiles of the original filters.

Ego-Stengel et al. 2009

In one of the first ripple disruption studies [13], an analog band-pass filter is used. An 8th-order Butterworth high-pass filter at 100 Hz is combined with an 8th-order

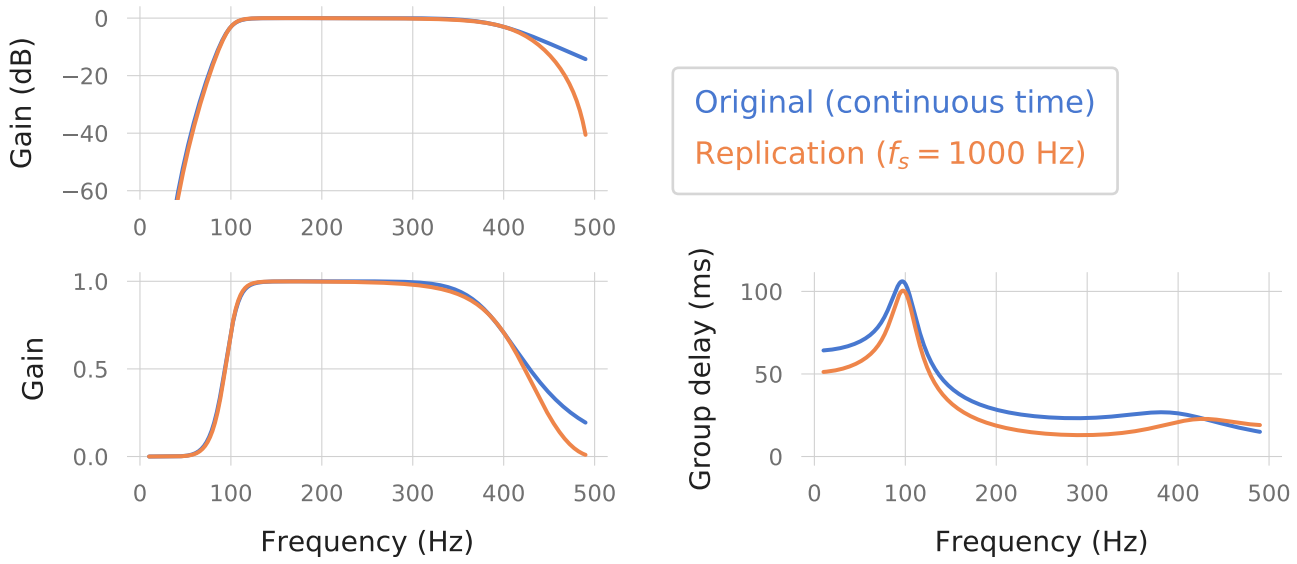


Figure 4.1: **Online ripple filter from Ego-Stengel et al. 2009**. The group delay of each filter is calculated as $-\,\mathrm{d}\phi/\mathrm{d}f$, where ϕ is the unwrapped phase response of the filter. (The derivative is calculated numerically using a Savitsky-Golay filter, with 5 samples per window, cubic polynomials, and $\Delta f \approx 0.05$ Hz). This method is also used for subsequent group delay plots.

Butterworth low-pass filter at 400 Hz. We approximate this continuous-time filter by the product of two discrete-time Butterworth filters at $f_s = 1000$ Hz: an 8th-order high-pass filter at 100 Hz, and a 2nd-order low-pass filter at 400 Hz.[1] Its frequency response is compared with that of the original filter in fig. 4.1.

Dutta et al. 2018

In this recent analysis of an online ripple detection system [9], a 30-tap FIR filter at $f_s = 3000$ Hz is used. In the open source code of this study[2], we find that the filter is designed using the windowed-sinc method, with a Hamming window. We approximate this filter at a sampling frequency of 1000 Hz with an 11-tap FIR filter, designed using the same method, window type, and passband. The correspondence is good (see fig. 4.2). Note the low attenuation outside the passband, and the constant, relatively low group-delay.

Falcon (Ciliberti et al. 2017)

The default ripple filter in *Falcon* is a Type II Chebyshev filter, with a passband of ≈ 130–283 Hz, edge widths of ≈ 10 Hz, a minimum attenuation in the stopbands of 40 dB, and

[1] Note that we do not choose the literal discretization of the continuous-time filter (namely the combination of an 8th-order high-pass and an 8th-order low-pass Butterworth filter, discretized using e.g. the zero-order-hold method or Tustin's bilinear transform). Around 400 Hz, this discretization has too high a peak in group delay and too steep a loss in gain for its frequency response profiles to match the original filter well.

[2] `https://github.com/shayokdutta/RippleDetectionAnalysis/blob/master/DataAnalysisScripts/ripple_filtering.py`

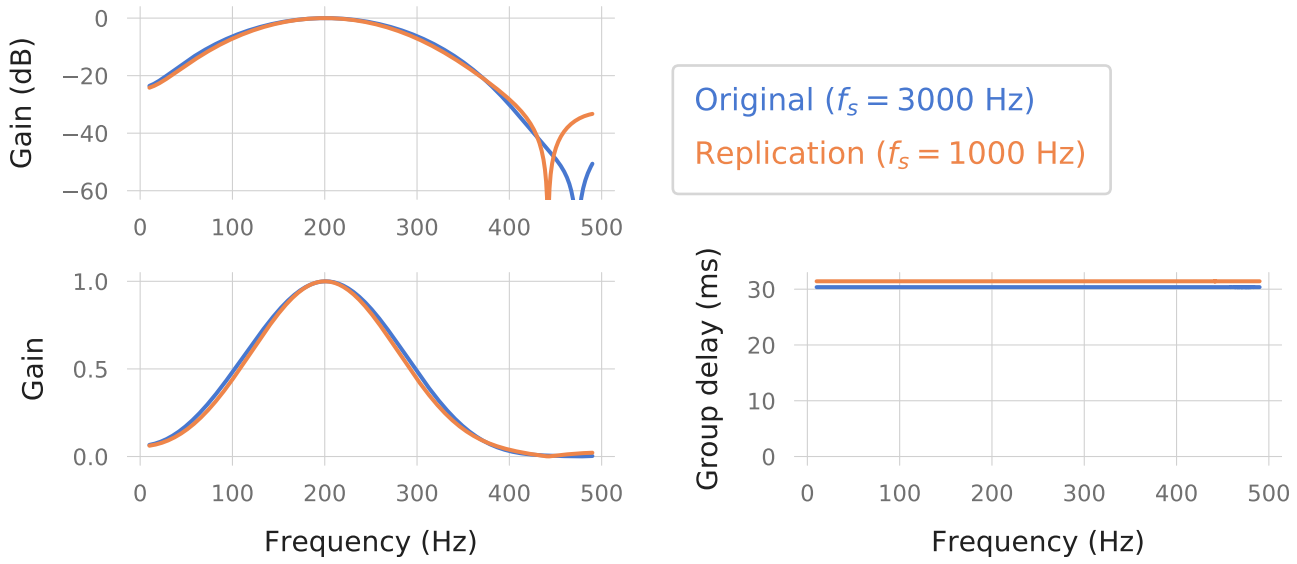


Figure 4.2: **Online ripple filter from Dutta et al. 2018**.

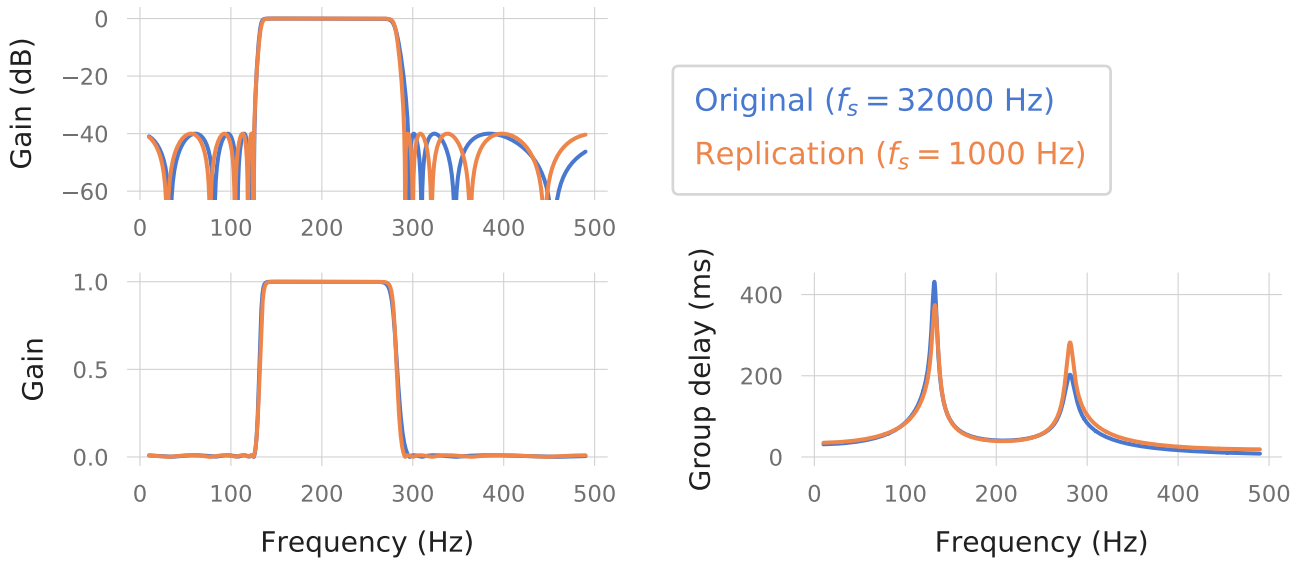


Figure 4.3: **Default online ripple filter in Falcon**.

a maximum attenuation in the passband of 1 dB. The original filter operates on data with a sampling rate of 32 kHz. We can closely approximate this filter at $f_s = 1000$ Hz using a 10th-order (21-taps) Type II Chebyshev filter, with the same design parameters. Its frequency response is shown in fig. 4.3. Note the trade-off between filter quality and group delay apparent in these and the previous plots: sharp passband edges correspond to high group delays, and vice-versa.

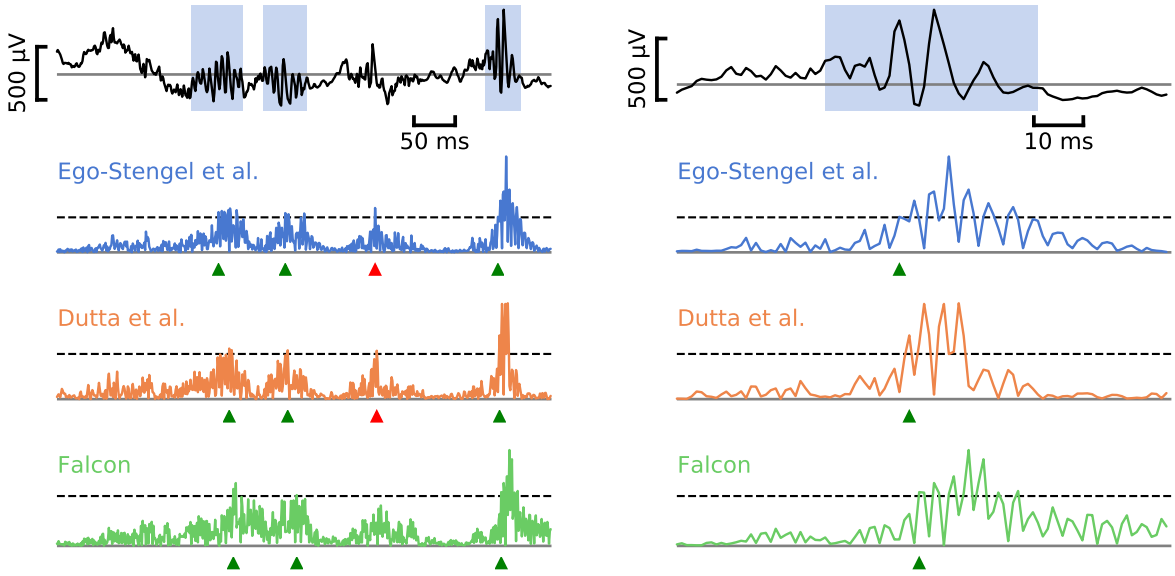


Figure 4.4: **Rectified outputs of online ripple filters**. *Left*: 600 ms of input and output data. *Right*: zoom in on the last ripple of the left panel.

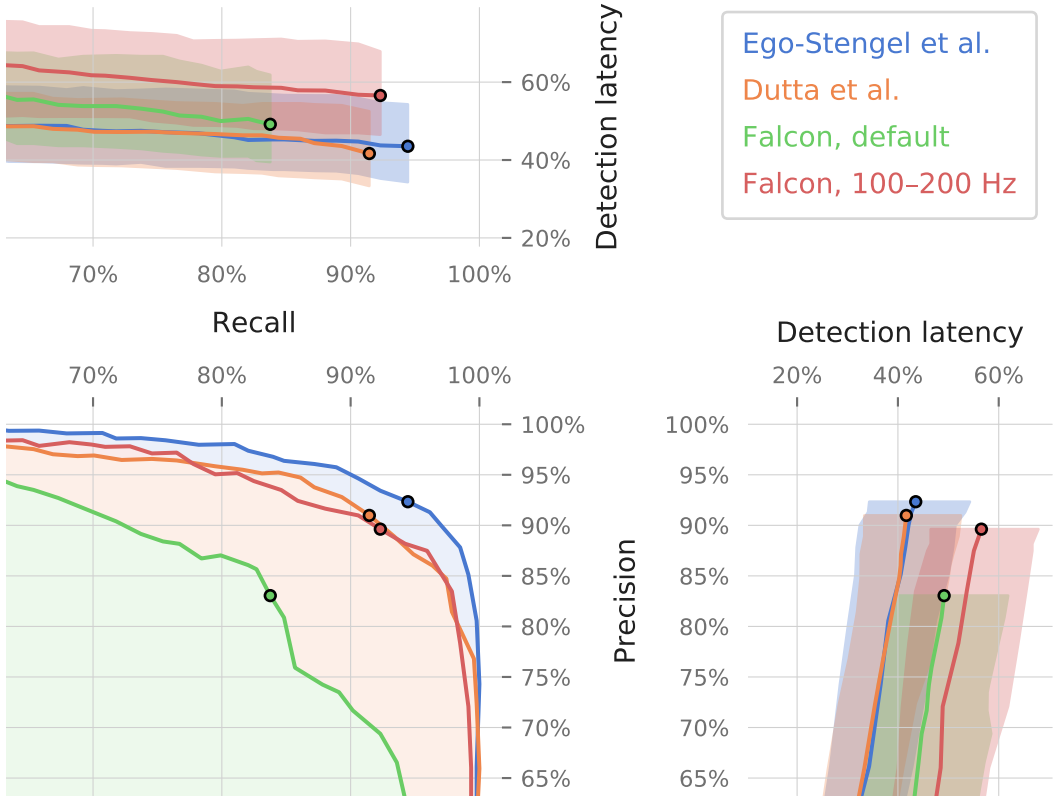


Figure 4.5: .

4.2 SWR detection performance

4.3 The best linear ripple filter

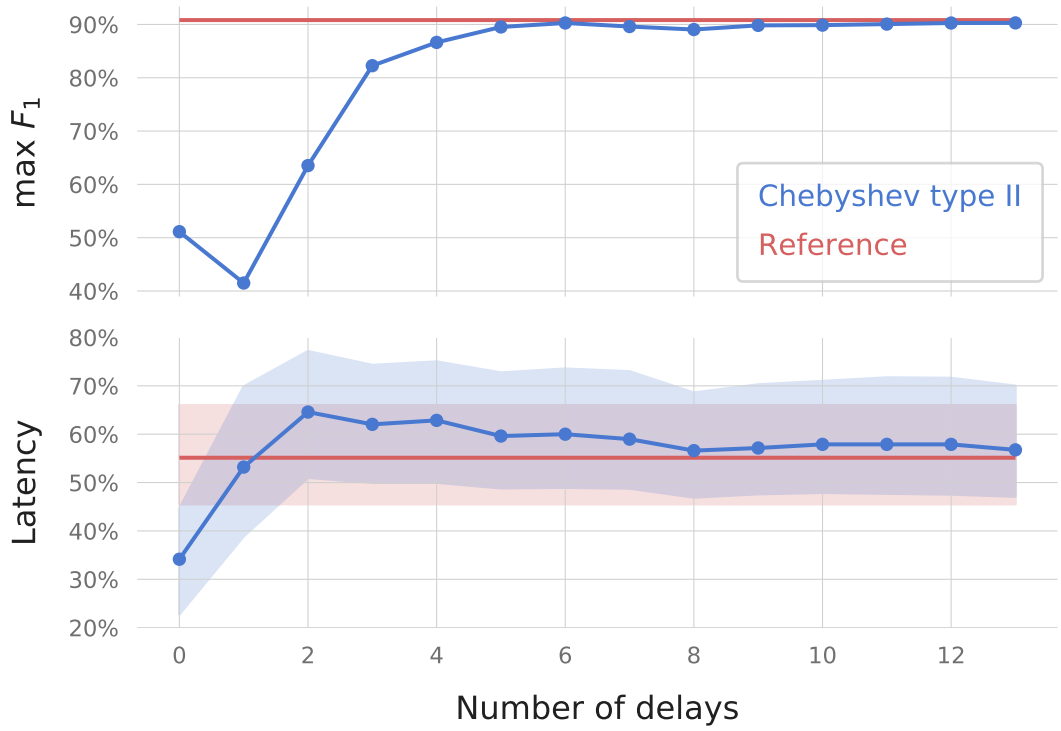


Figure 4.6: **Performance of Chebyshev type 2 IIR-filters**, for different filter orders. The minimum stop-band attenuation was set to 40 dB.

4.4 Online envelope estimation

The real-time constraint also means that we can no longer use the Hilbert transform to calculate the envelope n_t of the band-pass filter output o_t. In practice, the band-pass filter output is often simply rectified (i.e. $n_t = |o_t|$).

Extensions of simple rectification have also been used. In Jadhav et al. 2012 [14], the online envelope n_t is an exponentially weighted moving average (EWMA) of $|o_t|$, with a variable gain. In their update equation, a slightly higher weight is used for $|o_t|$ whenever $n_t < |o_t|$ (see appendix A.2). In Dutta et al. 2018 [9], $|o_t|$ is smoothed with a 50 Hz low-pass FIR filter.

Both methods add latency. The EWMA method of Jadhav et al. delayed the envelope by about 2 milliseconds when applied to our recording. The FIR low-pass filter of Dutta et al. has a constant group delay of about 5 milliseconds.

Smoothing $|o_t|$ yields a visually more pleasing envelope. This is not a requirement for online SWR detectors however. Smoothing may also prevent spurious detections whenever the unsmoothed signal contains outliers (strong peaks in magnitude that do not correspond to a peak in the underlying 'real' signal). Our recording (and the band-pass filtered version of it) do not seem to contain such outliers however. Because smoothing adds latency, we do not smooth our online envelopes; all online envelopes in this thesis are calculated as $n_t = |o_t|$.

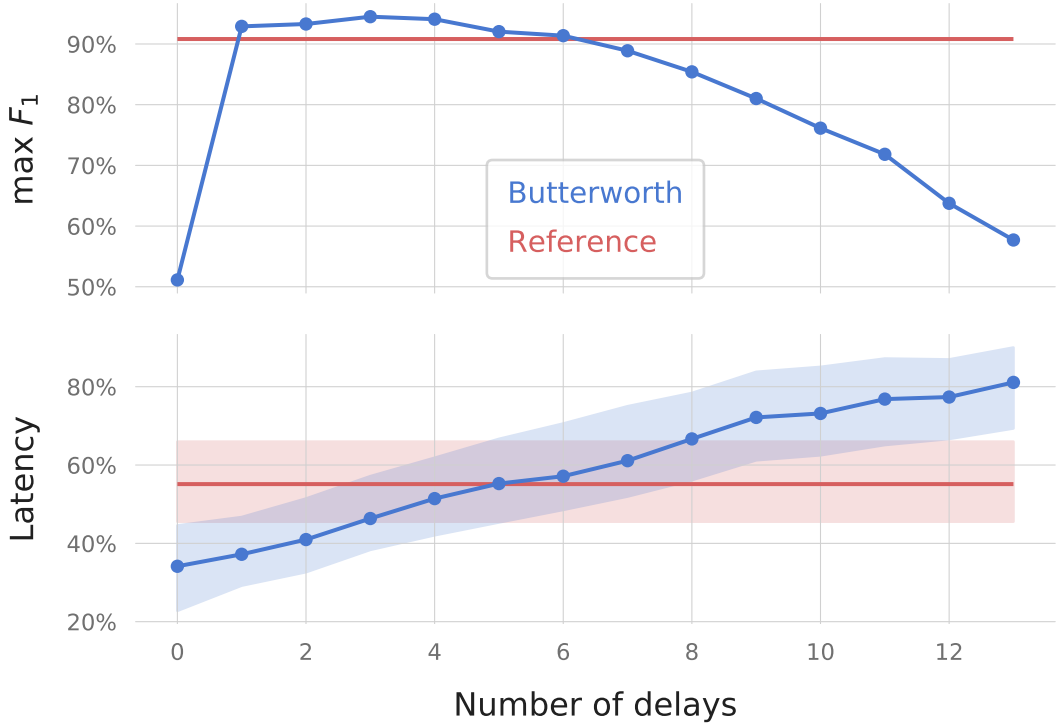


Figure 4.7: **Performance of Butterworth IIR-filters**, for different filter orders.

Chapter 5

Multi-channel linear filtering – 26 Oct

5.1 Data-driven algorithms

In this and the following chapter, we describe *supervised*, or data-driven SWR detection algorithms: they require training data $\mathbf{z}_t^{\text{train}}$, and an associated labelling y_t^{train} which marks the presence of an SWR event in $\mathbf{z}_t^{\text{train}}$, for every discrete time sample t. We arbitrarily define $y_t \in \{0, 1\}$, with $y_t = 1$ when the corresponding input sample $\mathbf{z}_t$ is part of an SWR segment, and $y_t = 0$ when it is not.

The problem of obtaining such a labelling y_t for some recording data $\mathbf{z}_t$ is the topic of chapter 2. Training labels can be obtained either by human expert labellers, or by using an automated *offline* SWR detection algorithm, where we assume that the automated labelling corresponds well to a supposed human expert labelling. In this thesis, we use the automated labelling method of chapter 2 to generate target labellings y_t^{train}.

Before a supervised algorithm can be used for real-time detection, its parameters have to be 'tuned'. This is done using a training dataset $(\mathbf{z}_t^{\text{train}}, y_t^{\text{train}})$, during the so called *training phase*. Parameters are changed such that the algorithm's output o_t for an input $\mathbf{z}_t^{\text{train}}$ matches the target labelling y_t^{train} well. Sections 5.2 and 6.2 describe how this tuning can be done for two concrete detection algorithms.

The hope is that the trained algorithm also performs well on input data $\mathbf{z}_t^{\text{test}}$ not part of the training set. That is, that the algorithm has good *generalization performance*. When this is not the case and the algorithm is tuned so that it only performs well on the training data, we say that the algorithm has been *overfit*. Often, so called *regularization* methods exist to discourage overfitting on the training data. Some example regularization methods are discussed in sections 5.5 and 6.3.

5.2 Linear signal-to-noise maximisation

In this chapter, we search for a linear combination of channels that yields an output signal o_t useful for sharp wave-ripple detection. More precisely, we search for a vector

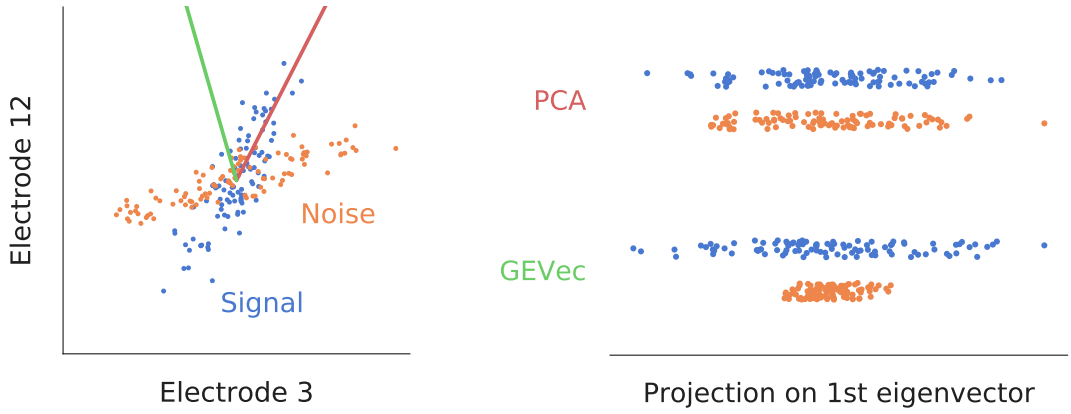


Figure 5.1: **Linear signal-to-noise maximisation**. Toy example to illustrate the generalized eigenvector approach to signal detection. *Left*: multi-channel time-series data plotted in 'phase space' (meaning without time axis), with blue dots representing samples where the signal was present, and orange dots representing samples where it was not. Actually toy data drawn from two 2-dimensional Gaussian distributions with different covariance matrices. Red vector: first eigenvector of the signal covariance matrix (also known as the first principal component). Green vector: first generalized eigenvector of the signal and noise covariance matrices. *Right*: Projection of both data sets on both the ordinary eigenvector ("PCA") and the generalized eigenvector ("GEVec"). The ratio of the projected signal data variance versus the projected noise data variance is maximised for the GEVec case.

$\mathbf{w} \in \mathbb{R}^C$ in channel (or electrode) space to project the samples $\mathbf{z}_t \in \mathbb{R}^C$ on, so that the output signal

$$o_t = \mathbf{w}^T \mathbf{z}_t \tag{5.1}$$

has high variance (or power) during SWR events, and low variance outside them.[1] This principle is illustrated with a two-dimensional toy dataset in fig. 5.1. We can then detect SWR events using threshold crossings of the envelope of o_t, as discussed in chapter 3.

The next two sections describe how this vector $\mathbf{w}$ can be found.

The optimisation problem

Suppose all training samples $\mathbf{z}_t^{\text{train}}$ are gathered and divided over two data matrices $\mathbf{S} \in \mathbb{R}^{C \times N_S}$ and $\mathbf{N} \in \mathbb{R}^{C \times N_N}$, where $\mathbf{S}$ (for 'signal') contains all N_S samples of $\mathbf{z}_t^{\text{train}}$ where an SWR is present, and $\mathbf{N}$ (for 'noise') contains all N_N other samples. (These matrices can be easily constructed by concatenating segments from $\mathbf{z}_t^{\text{train}}$).

[1] We assume that the input signals are zero-mean, such that the power P of the output signal equals its variance: $P_o = \langle o_t^2 \rangle = \langle (o_t - \mu_o)^2 \rangle = \text{Var}(o_t)$ when $\mu_o = 0$, which is the case for zero-mean input channels: $\mu_o = \langle o_t \rangle = \langle \mathbf{w}^T \mathbf{z}_t \rangle = \sum_i w_i \langle z_{t,i} \rangle = 0$ when $\langle z_{t,i} \rangle = 0$ for all channels i.
This zero-mean assumption is reasonably well fulfilled for the analysed recording: the sample values of a 10-second moving average of the recording are near zero (median -0.02 µV, IQR 0.13 µV. In comparison, the RMS-value of the recording is 210 µV).
Input data that is not zero-mean can be readily transformed to be so (even in an online setting), by subtracting a (moving) average from the input signal.

Equation (5.1) then becomes, in vector notation:

$$\mathbf{o}_S = \mathbf{w}^T\mathbf{S}$$
$$\mathbf{o}_N = \mathbf{w}^T\mathbf{N},$$

where each element of the row vectors $\mathbf{o}_S$ and $\mathbf{o}_N$ is a filtered sample of $\mathbf{S}$ and $\mathbf{N}$, respectively. Figure 5.1 (right) shows the distribution of the values in two example data vectors $\mathbf{o}_S$ and $\mathbf{o}_N$.

We want to find the weight vector $\widehat{\mathbf{w}}$ that maximises the variance of $\mathbf{o}_S$ versus the variance of $\mathbf{o}_N$, i.e.

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} \frac{\mathrm{Var}(\mathbf{o}_S)}{\mathrm{Var}(\mathbf{o}_N)} \tag{5.2}$$

$$= \arg\max_{\mathbf{w}} \frac{\frac{1}{N_S}\mathbf{o}_S\mathbf{o}_S^T}{\frac{1}{N_N}\mathbf{o}_N\mathbf{o}_N^T}$$

$$= \arg\max_{\mathbf{w}} \frac{\frac{1}{N_S}\mathbf{w}^T\mathbf{S}\mathbf{S}^T\mathbf{w}}{\frac{1}{N_N}\mathbf{w}^T\mathbf{N}\mathbf{N}^T\mathbf{w}}$$

In this last equation, we recognize the empirical covariance matrices $\mathbf{R}_{SS}$ and $\mathbf{R}_{NN}$, which are defined as:

$$\mathbf{R}_{SS} = \frac{1}{N_S}\mathbf{S}\mathbf{S}^T \tag{5.3}$$
$$\mathbf{R}_{NN} = \frac{1}{N_N}\mathbf{N}\mathbf{N}^T \tag{5.4}$$

$\mathbf{R}_{SS} \in \mathbb{R}^{C\times C}$ and $\mathbf{R}_{NN} \in \mathbb{R}^{C\times C}$ are symmetric matrices, where each diagonal element yields the variance of a channel, and each off-diagonal element yields the covariance between a pair of channels.

The condition for the optimal weight vector, eq. (5.2), is thus equivalent to:

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T\mathbf{R}_{SS}\mathbf{w}}{\mathbf{w}^T\mathbf{R}_{NN}\mathbf{w}} \tag{5.5}$$

In appendix C, we show that the solution $\widehat{\mathbf{w}}$ to this optimisation problem is the first so called "generalized eigenvector" of $(\mathbf{R}_{SS}, \mathbf{R}_{NN})$.

The generalized eigenproblem

An arbitrarily scaled vector $\mathbf{w}_i$ is a *generalized eigenvector* (GEVec) for the ordered matrix pair $(\mathbf{R}_{SS}, \mathbf{R}_{NN})$ when the following holds:

$$\mathbf{R}_{SS}\mathbf{w}_i = \lambda_i\mathbf{R}_{NN}\mathbf{w}_i, \tag{5.6}$$

for some scalar λ_i, which is called the *generalized eigenvalue* (GEVal) corresponding to $\mathbf{w}_i$. The largest scalar λ_1 for which eq. (5.6) holds is the 'first' GEVal, and as mentioned before, the corresponding GEVec $\mathbf{w}_1$ is the solution $\widehat{\mathbf{w}}$ to eq. (5.5).

Since the 1960's, numerically stable algorithms exist that solve the generalized eigenproblem eq. (5.6) [15]. A specialized algorithm is applicable when the input matrices are symmetric – as is the case for $\mathbf{R}_{SS}$ and $\mathbf{R}_{NN}$. This algorithm (based on a Cholesky factorization and the classical QR-algorithm for ordinary eigenproblems) is implemented in the LAPACK software package (as `ssygv` and `dsygv`), and can be easily applied using e.g. the `eig` function from MATLAB, or the `eigh` function from SciPy's `linalg` module.

5.3 Result for SWR detection

We divided the 34-minute long LFP recording into two datasets. The first 60% was used as training data, to calculate the covariance matrices $\mathbf{R}_{SS}$ and $\mathbf{R}_{NN}$, and to calculate from these the optimal linear combination of channels $\widehat{\mathbf{w}}$, as described in the preceding sections. The remaining 40% was used to evaluate this filter $\widehat{\mathbf{w}}$, and to compare it to the state-of-the-art method (the single-channel online band-pass filter).[2]

Figure 5.2A shows an excerpt of the test input signal, and the corresponding filter output envelopes (blue for state-of-the art method, orange for GEVec-based multichannel method. Filter outputs o_t are rectified to obtain envelopes $n_t = |o_t|$). Additional excerpts are shown in fig. D.2. The elements of $\widehat{\mathbf{w}}$ (i.e. the filter weights) are visualized in fig. 5.2B.

It is clear that the GEVec filter output indeed has high power during SWR events, as promised by the theoretical derivation. The filter weights and signal excerpts reveal that the GEVec output is composed mainly of a few channels in the stratum radiatum (channels 4-5-6 here), where they pick up the sharp waves. However, the filter output envelope is also high for sharp wave-like activity on these channels, without or with only very weak ripple activity in the higher channels: see figs. D.2a, D.2b and D.2d. This results in false positive detections.

We also notice some premature detections (figs. D.2a and D.2c), where the sharp wave – and thus also the GEVec filter output – already has high power before the corresponding ripple has started. The GEVec detection then happens before the start of the reference SWR-segment, which is based on ripple power. These early detections thus count (arguably unfairly so) as false positives, under the evaluation scheme that we use.

This effect, where the sharp wave is discernible before the ripple, results in faster detections when the detection *does* fall within the reference segment. When all 468 SWR events from the test set are analysed, we find a large improvement in detection latency: at thresholds where both methods detect 80% of these reference SWR segments, the median absolute detection latency drops from 24 ms for the state-of-the-art online band-pass filter to 12 ms for the GEVec-based filter. The relative detection latency drops by 32.5 percentage points, from 58.1% to 25.6%. Similar latency improvements are found for other recall and precision values: see fig. 5.3.

This strong improvement in detection latency trades off with an increase in false positives, as was already observed qualitatively. At the aforementioned sensitivity of 80%, the

[2] The 60-40 division was chosen arbitrarily – under the constraint that there are sufficient amounts of both training and test data.

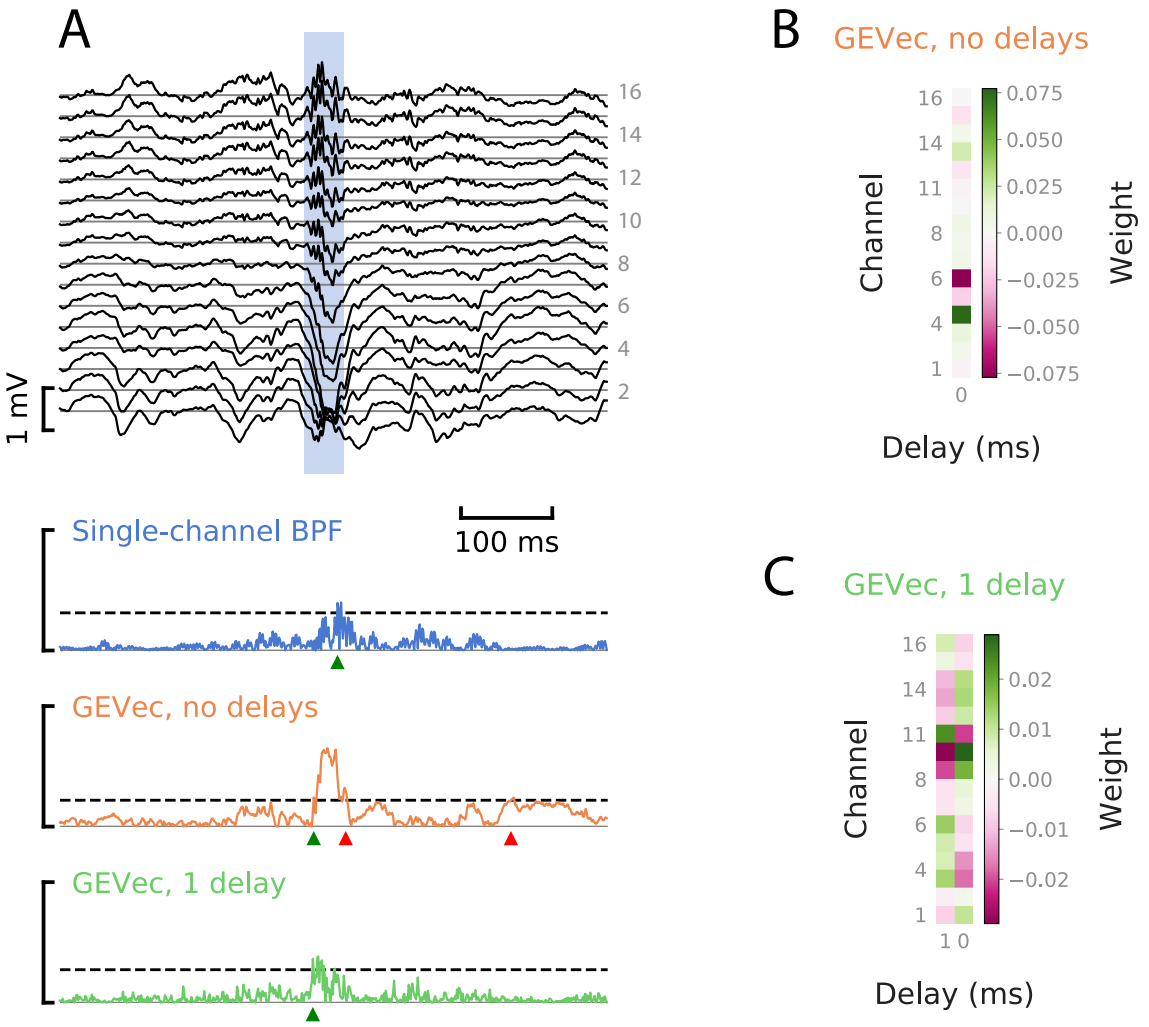


Figure 5.2: **Linear, SNR-maximising combinations of electrodes**.
A. Example input and output signals. *Top*: multi-channel LFP, $\mathbf{z}_t$. Light-blue vertical band: a reference SWR segment. *Bottom*: output envelopes n_t, for different filtering algorithms. Dashed horizontal lines: detection thresholds, chosen so that each algorithm reaches a recall value of 80%. Green triangles: correct detections. Red triangles: incorrect detections. Brackets indicate envelope range (min, max) over the entire test set.
B. Generalized eigenvector $\widehat{\mathbf{w}}$ (i.e. the weights of the multichannel filter), for a purely spatial filter.
C. Generalized eigenvector $\widehat{\mathbf{w}}$ for a spatiotemporal filter with a one-sample delay.

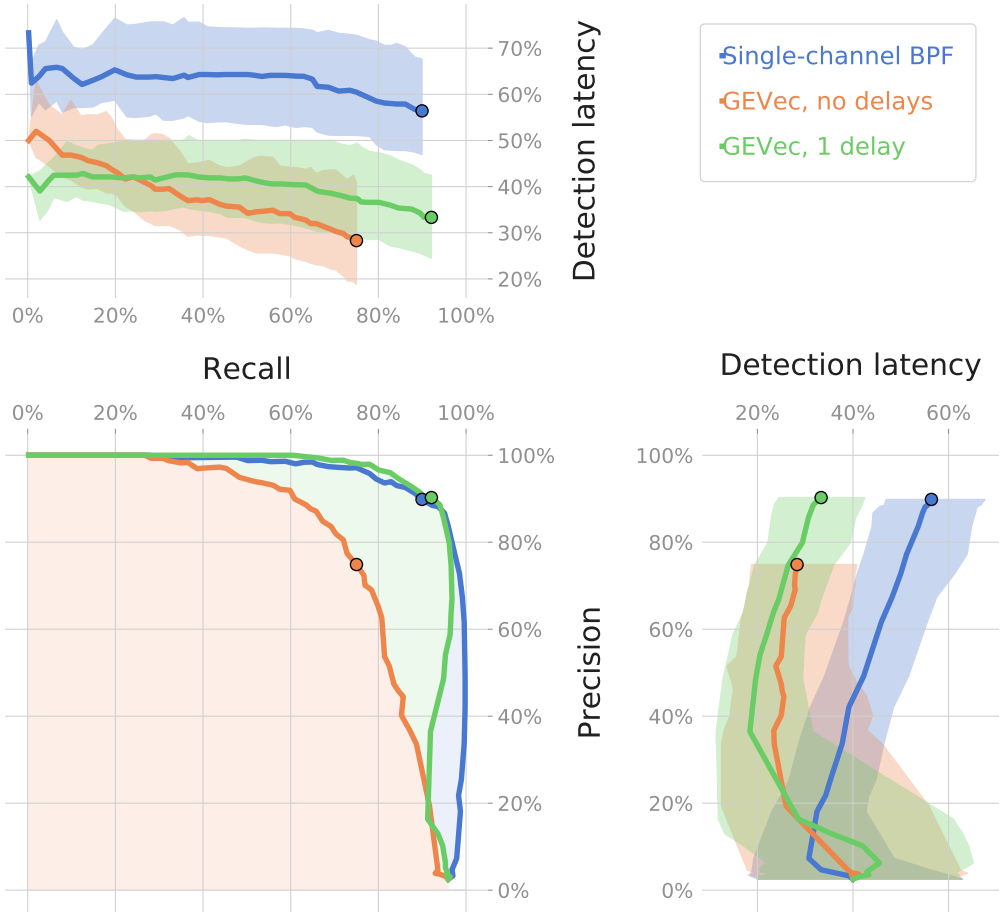


Figure 5.3: **Sensitivity, precision & latency tradeoffs,** for different linear filters & for a range of thresholds.
Each threshold setting for an algorithm corresponds to a point on the precision-recall curve in the bottom-left panel, and to a distribution of relative detection latencies. The median and interquartile range of this distribution are plotted in the top-left or bottom-right panel (as a point of the bold curve, and a slice of the shaded band, respectively).
These latency distribution plots are divided over two panels so that their entire range can be clearly visualized, both for the low recall – high precision regime as for the high recall – low precision regime. The cutoff is made at the point where recall equals precision (AKA the $\max F_1$ point), marked with shaded black circles.

state-of-the-art method has a precision of 94%, whereas the GEVec-based method has a precision of only 63% (i.e. more than a third of detected events are classified as false positives). This strong decrease in precision is true over the entire PR-curve: see fig. 5.3.

5.4 Combining space and time

It is hardly surprising that the GEVec-based algorithm as described above cannot discern ripple activity (which is by definition a temporal pattern), as the algorithm is a purely *spatial* filter: at each timestep t, only current information from the different channels is used in calculating the output o_t, without incorporating temporal information from previous timesteps $t_p < t$.

The GEVec method can be easily adapted to also incorporate temporal information however, by defining a vector $\mathbf{z}_t^{\text{stack}} \in \mathbb{R}^{CP}$ which consists of stacked sample vectors (each consisting of C channels) from P different timesteps $t_p \leq t$. The linear weights $\mathbf{w}^{\text{stack}} \in \mathbb{R}^{CP}$ used to obtain the output signal $o_t = (\mathbf{w}^{\text{stack}})^T \mathbf{z}_t^{\text{stack}}$ are then calculated analogously to the purely spatial filter, i.e. as the first generalized eigenvector of the ordered pair $(\mathbf{R}_{SS}^{\text{stack}}, \mathbf{R}_{NN}^{\text{stack}})$, with both covariance matrices $\in \mathbb{R}^{CP \times CP}$.

Adding just one such delayed time step (i.e. $P = 2$) yields a major performance improvement (see fig. 5.3): the precision-recall curve shoots up to (and even slightly exceeds) the PR-curve of the state-of-the-art algorithm, while the latency improvements of the 'no delay' GEVec algorithm are mostly retained: at a sensitivity of 80%, the median absolute latency for the one delay GEVec filter is 15 ms, which is 9 ms faster than the state-of-the-art method (and 3 ms slower than the no delay GEVec filter). The relative latency is 36.6%: 21.5 percentage-points lower than the state-of-the-art (and 11 pp. higher than the no delay GEVec filter).

At this 80% recall mark, the one delay GEVec filter attains a precision of 97%, a 3% increase over the state-of-the-art. The full precision-recall-latency tradeoff and algorithm comparison is shown in fig. 5.3. Note that for very low thresholds, the PR-curve of the one-delay GEVec method is no longer concave: decreasing the threshold further yields more (not less) missed reference SWR segments. Figure 5.2C shows the GEVec $\mathbf{w}^{\text{stack}}$. Figure 5.2A and fig. D.2 show example output envelopes (green traces).

These results, particularly the visualized weights in fig. 5.2C, indicate that this one-delay GEVec filter utilizes spatiotemporal information about both the sharp wave and the ripple.

Choosing the number of delays

Adding more delays (fig. 5.4) improves detection accuracy even further – up to a peak *max* F_1 of 93% at about eleven delays. (This corresponds to eleven milliseconds, or approximately half a ripple phase). Detection latency also increases with increasing number of delays, but only slightly, always staying well below the state-of-the-art latency. Like the *max* F_1 score, latency stagnates after about eleven delays. Further, we note that the latency distributions of the GEVec-based detectors have a lower spread than those of the state-of-the-art detector.

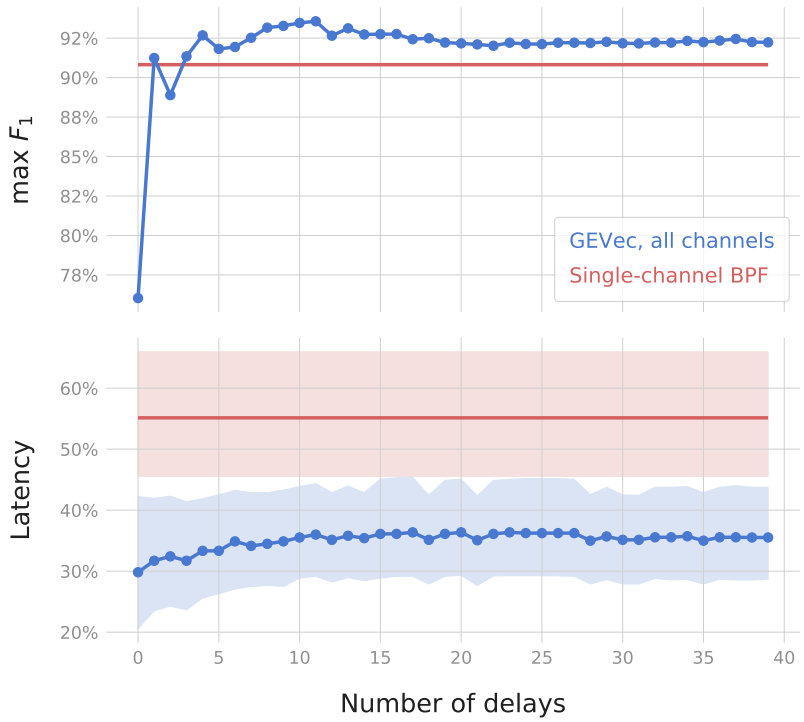


Figure 5.4: **Performance of the GEVec-based SWR detector, for different delay line lengths**. At the chosen 1000 Hz sampling rate, each delay corresponds to 1 ms. The red baseline is the state-of-the-art SWR detector. Detection latency is specified as a fraction of the duration of the corresponding SWR event, and is evaluated at the threshold where each detector reaches its maximum F_1-score. In the latency panel, bold lines and shaded areas indicate the median and the interquartile range of the latency distributions, respectively.

There is thus a slight tradeoff to be made when choosing the number of delays for a GEVec-based detector: using more delays yields detectors that are more accurate, but also slightly slower. Given that the decrease in speed is minor (about five percentage-points), it is reasonable to choose the amount of delays that maximizes detection accuracy. In this analysis, this optimal point is reached at an eleven milliseconds-long delay line.

Selecting channels

Many CA1 LFP recordings are not made with multichannel probes, but rather with one or more tetrodes. It is therefore relevant to ask how GEVec-based detectors perform on these types of recordings. We can approximate this setting with our current multichannel probe recording, by only using one or a few channels.

Each such selection of input channels, in combination with a certain number of delays, yields a different GEVec-based SWR detector. We evaluate and compare these detectors using the test data set, analyzing both accuracy (fig. 5.5) and latency (fig. 5.6).

We note some general trends. Using no extra delays results in low accuracy detectors, no matter which channels are included (top row of fig. 5.5). For most input channel

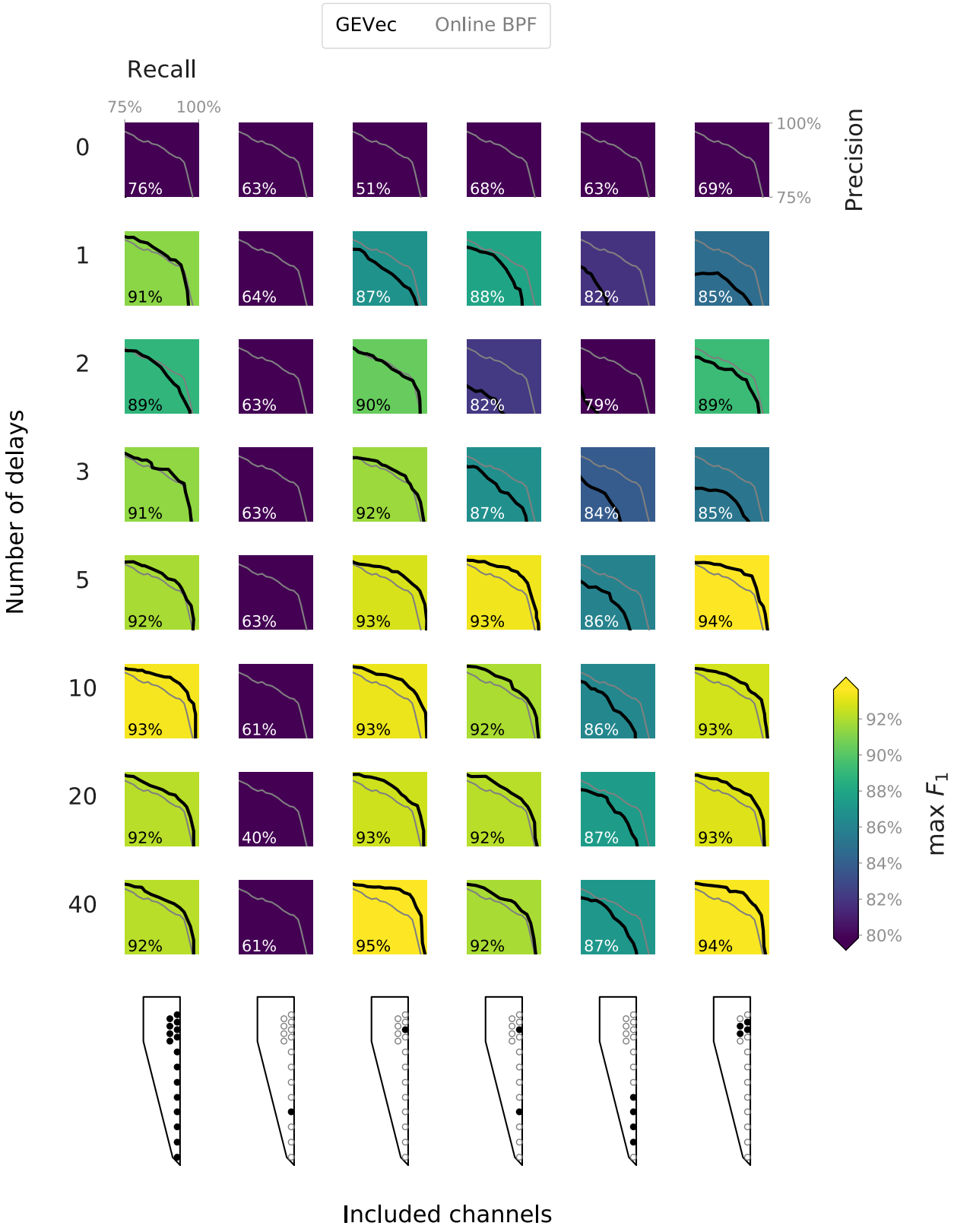


Figure 5.5: **Accuracy of GEVec-based SWR detectors,** for different combinations of active input channels and number of delays. Active input channels are marked in solid black on a schematic of the probe tip. (Height of schematic: 687 µm). The percentage and background color of each panel indicate the maximum F_1 value obtained for that GEVec-based SWR detector. Each panel includes the same baseline from the state-of-the-art SWR detector, in gray.

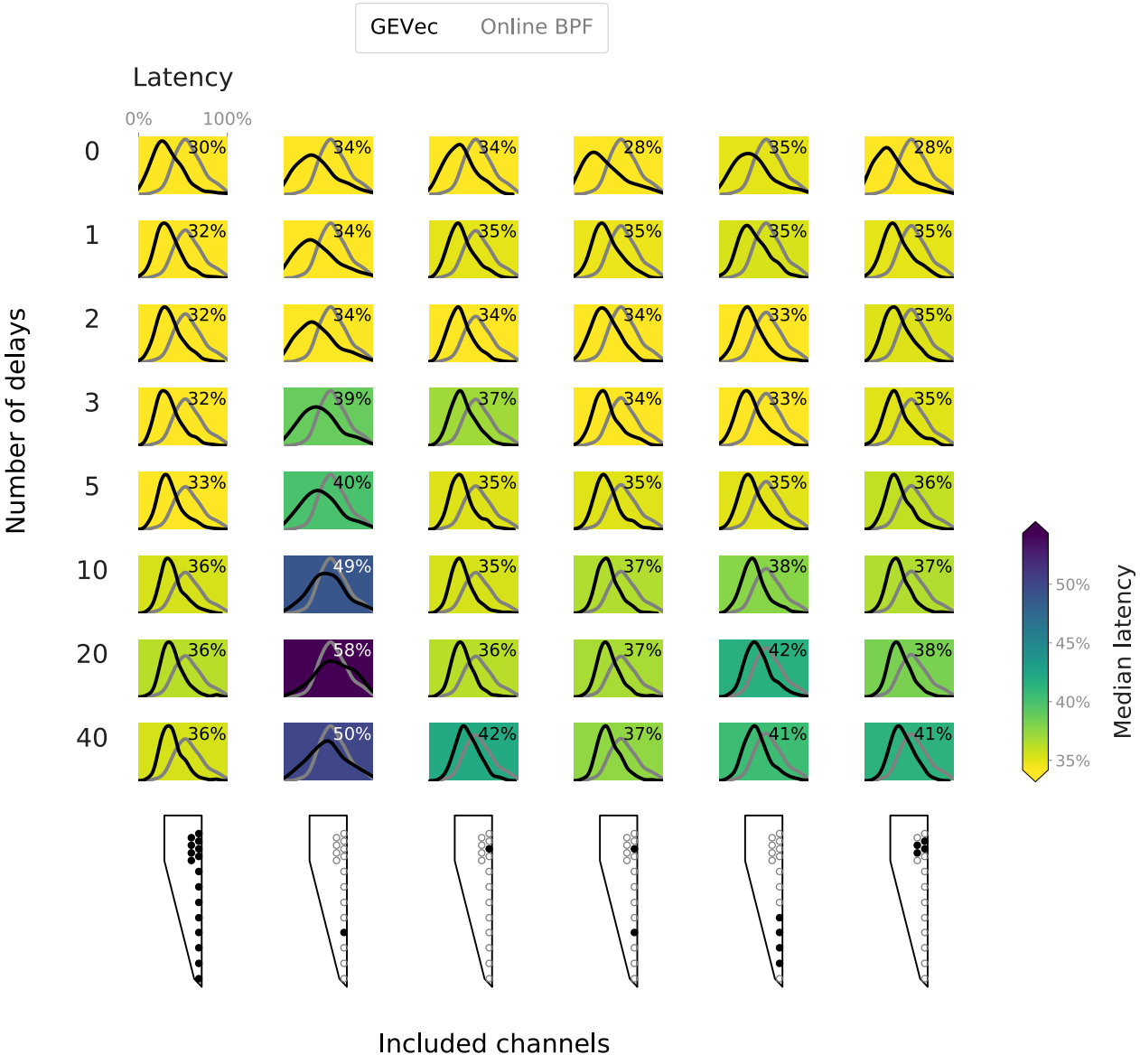


Figure 5.6: **Latency of GEVec-based SWR detectors**. See fig. 5.5 for legend.

combinations, the tradeoff in number of delays observed in the previous section is preserved: using more delays increases accuracy (up to a certain point), while also increasing detection latency.

The weight visualizations of fig. 5.2 show a relatively high weight for the stratum radiatum / sharp wave channels. Using only one such channel as input results in low accuracy detectors however, no matter how many delays are used (second column of figs. 5.5 and 5.6). Using four such channels (second to last column) results in detectors with an accuracy approaching, but not reaching, that of the state-of-the art ripple-based detector when a sufficient number of delays is used. Interestingly, this decent accuracy is reached despite none of these four channels (numbers 2 to 5) displaying any noticeable ripple activity (fig. 5.2A).

Conversely, using one channel from the pyramidal cell layer (where ripples are observed) yields GEVec-based detectors on par with those that use all channels as input, both regarding accuracy and latency (third and first column of figs. 5.5 and 5.6).

This one channel GEVec-based detector thus also outperforms the "state-of-the-art" detector based on a band-pass filter. The state-of-the-art detector also uses only one channel as input, and is also a linear filter. This suggests that either 1) finding linear filter weights through 'machine learning' (the GEVec technique) yields better results than manual filter design; or 2) that the band-pass filter used as "state-of-the-art" detector was not designed optimally in the first place.

Adding a stratum radiatum / sharp wave channel to the single pyramidal cell layer channel (fourth column of figs. 5.5 and 5.6), or using a cluster of channels in the pyramidal cell layer (last column) does not improve performance over the one channel case.

5.5 Regularization

Chapter 6

Nonlinear signal detection

Chapter 7

Discussion

7.1 Comparing detectors

7.2 Further work

Chapter 8

Conclusions

Appendices

Appendix A

SWR detection in the literature

We gathered a representative sample of research papers that discuss sharp wave-ripples (and/or other hippocampal oscillations). Different studies define these oscillations with different frequency bands – these differences are listed in table A.1. Appendices A.1 and A.2 quote papers that use SWR detection; specifically their description of the SWR detection procedure. Emphasis is added.

Source	Theta (Hz)	High gamma (Hz)	Ripple (Hz)
Nádasdy et al. 1999 [7]			150 – 250
Csicsvari et al. 2000 [8]			80 – 250
Behrens et al. 2005 [10]			40 – 400
O'Keefe 2007 [16]	6 – 12	30 – 100	100 – 200
Girardeau et al. 2009 [17]			100 – 200
Ego-Stengel et al. 2009 [13]			100 – 400
Jadhav et al. 2012 [14], online			100 – 400
Jadhav et al. 2012 [14], offline			150 – 250
Buzsáki 2015 [2]	6 – 10	100+	110 – 200
Colgin 2016 [18]	6 – 12	60 – 100	150 – 200
Sadowski et al. 2016 [11]			120 – 250
Talakoub et al. 2016 [19]			80 – 150
Eichenbaum 2017 [20]	4 – 12	80 – 140	
Dutta et al. 2018 [9]			150 – 250
Ólafsdóttir et al. 2018 [21]	6 – 12		140 – 250
`fklab`	6 – 12	60 – 140	140 – 225
L2 recording	5 – 10		100 – 200

Table A.1: **Frequency bands of hippocampal LFP events**, according to different sources (both original research papers and literature reviews). When the source does not make a distinction between high and low gamma, the full gamma range is given. Note that Behrens et al. considered artificially induced SWR's in ex-vivo hippocampus slices. Talakoub et al. studied macaque monkeys; other primary research papers studied rats. '`fklab`' refers to the default frequency bands used in the data-analysis software used in the Kloosterman lab. The last row refers to the dataset analysed in this thesis.

A.1 Offline detection algorithms

Nádasdy et al. 1999

> "For the extraction of sharp-wave (SPW) ripple events during sleep, the wide-band recorded data were bandpass filtered digitally (**150–250 Hz**). The power (**root mean square**) of the filtered signal was calculated, and the beginning, peak, and end of individual ripple episodes were determined. The threshold for ripple detection was set to **7 SDs above the background mean** power (Csicsvari et al., 1999 [22])." [7]

Csicsvari et al. 2000

> "*Detection of SPW-Associated Fast Ripples*: The procedures described here were identical to those described earlier (Csicsvari et al., 1999b [23]). The wide-band (1–5 kHz) recorded data was digitally band-pass filtered (**80–250 Hz**), and the power (**root-mean-square**) of the filtered signal was calculated for each electrode. The mean and standard deviation (SD) of the power signal were calculated to determine the detection threshold. Oscillatory epochs with a power of **one** or more **SD above the mean** were detected. The beginning and the end of oscillatory epochs were marked at points where the power fell below **0.5 SD**. Theta periods, detected by using the theta-delta power ratio (Csicsvari et al., 1999a [22]), were excluded from the analysis." [8]

Behrens et al. 2005

> "For ripple detection, raw data were filtered with a Spike 2 software band-pass filter of **40–400 Hz** (threshold: **4–6** times the s.d. of **eventless baseline noise**). For sharp wave detection, recordings were low-pass filtered at 20 Hz." [10]

Girardeau et al. 2009

> "Offline ripple detection was performed by band-pass filtering (**100–200 Hz**), **squaring and normalizing**, then thresholding the field potential recorded in CA1 pyramidal layer. Ripples were defined as events peaking at **>5 standard deviations** and lasting **<100 ms**." [17]

Jadhav et al. 2012

> "SWRs were detected during post-hoc analysis as described previously (11, 23). Raw LFPs recorded from the tetrodes used for online SWR detection were filtered between **150 – 250 Hz** and the SWR envelope was determined using a **Hilbert transform**. The envelope was **smoothed with a Gaussian with a s.d. of 4 ms and a width of 32 ms**. SWRs were defined as contiguous periods when the smoothed SWR envelope stayed **above 3 s.d.** of the mean [sic] for **at least 15 ms** on at least one tetrode." [14]

Sadowski et al. 2016

> "Ripples were detected offline in the LFP recorded on one CA1 channel. Raw LFP signal was filtered between **120 and 250 Hz**, and deflections in the ripple **power envelope** greater than **5 SDs from the mean** were classified as ripple events. Ripple start times were defined locally as when ripple power exceeded **2 SDs**. Samples of raw LFP and detected ripple times were compared manually to verify detection fidelity." [11]

Dutta et al. 2018

> "Post-recording, ripple events were defined on tetrodes that displayed characteristics of the CA1 area of the hippocampus. Specifically, the recorded LFP in one of the channels of the selected tetrode (same one subject to online detection for our realtime analysis) first had a digital reference subtracted away. This signal was then LFP band filtered with a 400 Hz low-pass infinite impulse response (IIR) filter (from Trodes). Afterwards the signal was decimated and ripple band filtered (**150–250 Hz**) with a **25 tap** finite impulse response (**FIR**) filter. Ripple band filtering was done using a forward and a time-reversed path, resulting in a net **zero group delay** (time shift from filtering). The instantaneous power of the ripple band filtered signal was then calculated via a **Hilbert Transform** and further **smoothened with a Gaussian kernel** with a **4 ms standard deviation**. Ripple events were detected as times when z-score of the smoothened power signal signal exceeded a threshold of **3 z-units** for **at least 15 ms**. The canonical ripple epochs were defined as the time points from which the processed signal **returned down to the mean before and after threshold crossings** [24], [25].
>
> In cases when multiple electrodes (typically channels on different tetrodes) are available for ripple detection, a different canonical definition is required. Ripples were initially defined using as above for each electrode. A canonical multichannel ripple was defined as one which is simultaneously detected on each electrode (two in our analysis). The multichannel ripple epoch is defined as the union of the detected single-channel ripple epochs, i.e., the start of the earliest ripple detected and to end with bound of last ripple detected. As such, we obtain a conservative ripple detection latency estimate while covering the entire span of the time the LFP is in a high ripple band power state. We reanalyzed our data with the canonical ripples being defined on different channels and tetrodes with a 300 tap bandpass FIR filter allowing 1% "ripple" in the passband with -30 dB suppression in the stopband but our results and subsequent conclusions remained consistent." [9]

A.2 Online detection algorithms

Girardeau et al. 2009

> "The onset of SPW-Rs was detected online by filtering the signal in the ripple-band and thresholding it. [..] Brain signals were preamplified (..), acquired and **digitized** using two synchronized Power1401 systems (CED, Cambridge, UK). [..] In both cases (test and control) the number of stimulations was **limited to 5 per second**." [17]

Ego-Stengel et al. 2009

> "We selected one tetrode in CA1, for which the LFP signal exhibited ripple events of large amplitude, for online ripple detection. The LFP was amplified and filtered online in the ripple band by an **8th-order Butterworth lowpass filter at 400 Hz** followed by an **8th-order Butterworth highpass filter at 100 Hz** (KrohnHite 3384 analog filters, total gain 10,000). A threshold-crossing detector (FHC Window Discriminator) was used to generate TTL pulses when the ripple amplitude exceeded a value adjusted manually by the experimenter on the first experimental day for each rat (0.1 ± 0.02 mV). These pulses triggered isolated stimulation units via a computer- controlled burst generator with a preset 1-ms delay [..] A **2-s recovery period** was forced after any stimulation burst before the next stimulation could be triggered." [13]

Jadhav et al. 2012

> "We disrupted awake hippocampal SWRs [..] with the use of an online feedback system similar to that used in previous studies that disrupted SWRs during post-behavior sleep [13], [17]. SWRs in CA1 were detected by monitoring power in the ripple band simultaneously across multiple tetrodes. [..] This [detection-triggered stimulation] terminated the ripple oscillation **within 25 ms of SWR onset** and transiently inhibited CA1 spiking [..].
> We recorded continuous local field potentials (LFP, filtered 0.5-400 Hz and sampled at 1500 Hz) from all tetrodes (one channel was chosen from each tetrode for LFP recording). [..]
> *Real-time detection algorithm*. Field potential signals from the 5-6 tetrodes chosen for online detection were broadly filtered in the ripple band (**20 tap band-pass IIR filter, 100-400 Hz**). In order to establish a disruption threshold, we calculated smoothed values of the mean and s.d. of the absolute value of the filtered LFP signal on each tetrode being used for detection using an iterative procedure:[1]
>
> $$\mu_t^{\text{est}} = \mu_{t-1}^{\text{est}} \frac{N^{\text{smooth}} - 1}{N^{\text{smooth}}} + \frac{|o_t|}{N^{\text{smooth}}}$$
> $$\sigma_t^{\text{est}} = \sigma_{t-1}^{\text{est}} \frac{N^{\text{smooth}} - 1}{N^{\text{smooth}}} + \frac{\left||o_t| - \mu_{t-1}^{\text{est}}\right|}{N^{\text{smooth}}}$$
>
> Here μ^{est} and σ^{est} are the estimated mean and s.d. of the absolute value of the filtered LFP, o, and N^{smooth} is the number of samples for smoothing (typically 10000). We allowed these estimates to stabilize before each run session. To generate a smoothened

estimate of the envelope (n^{est}) of the filtered LFP, we used the following iterative estimator:[2]

$$n_t^{\text{est}} = (1 - g_{t-1})n_{t-1}^{\text{est}} + g_{t-1}|o_t|$$

To allow for rapid detection of increases in power, we used a larger gain g for periods when the envelope was increasing: when the envelope was decreasing ($|o| \leq n^{\text{est}}$), $g = 0.2$ when the envelope was increasing, we used a moving average of the last 19 values of g and 1.2:

$$g_t = \begin{cases} 0.2, & \text{if} \quad n_{t-1}^{\text{est}} \geq |o| \\ \langle g_{t-20}, g_{t-19}, \, .. \, , g_{t-1}, 1.2 \rangle \, , & \text{otherwise} \end{cases}$$

The threshold for disruption was set to **4-6 s.d. above the mean**. To prevent false-positives, vHC stimulation was triggered only when the smoothed LFP envelope exceeded threshold on at least 2 tetrodes. Stimulation rate was limited to a maximum of 4 Hz by enforcing a **lock-out period of 250 ms** after each stimulation event." [14]

Talakoub et al. 2016

"Local field potentials were [..] sampled at **32 kHz**. [..] Selecting the electrode channel with the highest-amplitude ripple activity [..]. In this study, we set the threshold to **6-sd** of the band activity, a value similar to rodent interruption studies. The 6-sd threshold was estimated based on the average and variance of the ripple band from earlier recordings which were consistent over days in these experiments (9.5 ± 0.1 μV mean ± SEM).
[..] the signals were bandpass filtered using a custom-designed finite impulse response (**FIR**) filter with **512 taps** (**16 ms delay**, which is about one cycle of ripple activity).[3] Activities slower than **80 Hz** or faster than **150 Hz** are suppressed more than 20 dB (Fig. [..]). " [19]

Dutta et al. 2018

"Like our canonical ripple detections, the realtime or online detection algorithm is comprised of single channel and multichannel modalities. The single channel case performs realtime reference subtraction from the LFP (filtered in the same way as the offline case) and **decimation from 30 kHz to 3 kHz** as in the canonical detection case. However, the difference between online and canonical detections begins from ripple band filtering. The decimated signal is **filtered to the ripple band with a 30 tap FIR filter**. In order to perform a realtime instantaneous power estimation and smoothing, the realtime algorithm computes the **absolute value of the ripple band filtered signal** and **further filters it by a 33 tap 50 Hz low-pass FIR** (instead of a Hilbert transform followed by Gaussian kernel smoothing). These filters cause an intrinsic sample delay from the offline case (≈10.167 ms in our case). It is worth noting

[3] An FIR filter has a constant group delay of (# taps $- 1)/(2f_s)$. A 512 tap FIR filter at $f_s = 32$ kHz should therefore have a group delay of 8 ms (and not the reported 16 ms). Or, conversely, an FIR filter with a group delay of 16 ms should have ≈ 1024 taps at $f_s = 32$ kHz (and not the reported 512 taps).

that the number of filter taps as well as filter types were determined by analyzing algorithmic delay and detection accuracy based on metrics described in the Data Analysis subsection. To normalize detection thresholds in the realtime case, the mean and standard deviation of the smoothed envelope are estimated over a 20 minute training period. In two ≈90 minute sleep box recording sessions, when we sampled 20 minute time intervals at random (N=1000), the resulting mean and standard deviation were within 5% of the values for the entire sessions. This length of time and subsequent error in parameter estimation likely depends on the behavioral and/or sleep state of the animal — in our experimental recordings, animals were contained in a sleep box. realtime detections are then triggered when the envelope crosses a threshold defined as standard deviations above the mean (threshold = $\alpha \cdot \sigma + \mu$) or α z-units. Following a detection, there is a **200 ms lockout period** where we ignore any further threshold crossings (i.e., to avoid stimulation artifacts). **Additionally**, we impose a hard **limit on the number of detections per second** (set to **three** during the experiments in this work).” [9]

Appendix B

Data description

Appendix C

Generalized eigenvectors maximize signal-to-noise

In this appendix, we show that the weight vector that maximizes the ratio of signal to noise variance (i.e. the solution $\widehat{\mathbf{w}}$ to eq. (5.5)) is equivalent to the first generalized eigenvector $\mathbf{w}_1$ of the ordered pair of covariance matrices $(\mathbf{R}_{SS}, \mathbf{R}_{NN})$ (as defined in section 5.2).

This ratio of variances in eq. (5.5) is a quotient of quadratic forms, namely the so called "generalized Rayleigh quotient" of $(\mathbf{R}_{SS}, \mathbf{R}_{NN})$.

Formally, the generalized Rayleigh quotient of a non-zero vector $\mathbf{w} \in \mathbb{R}^N$ and the ordered, symmetric matrix pair $(\mathbf{A}, \mathbf{B})$ is the scalar $r(\mathbf{w})$ defined as:

$$r(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{A} \mathbf{w}}{\mathbf{w}^T \mathbf{B} \mathbf{w}} \tag{C.1}$$

We must then prove the following:

C.1 Theorem

The generalized eigenvector $\mathbf{w}_1$ corresponding to the largest generalized eigenvalue λ_1 of $(\mathbf{A}, \mathbf{B})$, is also the vector $\widehat{\mathbf{w}}$ that maximises the generalized Rayleigh quotient $r(\mathbf{w})$ of $(\mathbf{A}, \mathbf{B})$.

C.2 Proof

As a first step, we will show that if $\widehat{\mathbf{w}}$ is the maximum of $r(\mathbf{w})$, that it is indeed an eigenvector of $(\mathbf{A}, \mathbf{B})$. In the second step, we will show that the largest eigenvalue λ_1 of $(\mathbf{A}, \mathbf{B})$ corresponds to the maximum of $r(\mathbf{w})$.

If $\widehat{\mathbf{w}}$ is a maximum of $r(\mathbf{w})$, then

$$\boldsymbol{\nabla} r(\widehat{\mathbf{w}}) = \mathbf{0}. \tag{C.2}$$

Working out the partial derivatives that comprise the gradient of $r(\mathbf{w})$, we find:

$$\boldsymbol{\nabla} r(\mathbf{w}) = \frac{2\mathbf{A}\mathbf{w}\big(\mathbf{w}^T\mathbf{B}\mathbf{w}\big) - 2\mathbf{B}\mathbf{w}\big(\mathbf{w}^T\mathbf{A}\mathbf{w}\big)}{\big(\mathbf{w}^T\mathbf{B}\mathbf{w}\big)^2}$$

With eq. (C.2), we then have the following condition for our maximising vector $\widehat{\mathbf{w}}$:

$$2\mathbf{A}\widehat{\mathbf{w}}\big(\widehat{\mathbf{w}}^T\mathbf{B}\widehat{\mathbf{w}}\big) = 2\mathbf{B}\widehat{\mathbf{w}}\big(\widehat{\mathbf{w}}^T\mathbf{A}\widehat{\mathbf{w}}\big)$$

or

$$\mathbf{A}\widehat{\mathbf{w}} = \frac{\widehat{\mathbf{w}}^T\mathbf{A}\widehat{\mathbf{w}}}{\widehat{\mathbf{w}}^T\mathbf{B}\widehat{\mathbf{w}}}\,\mathbf{B}\widehat{\mathbf{w}}$$

$$\mathbf{A}\widehat{\mathbf{w}} = r(\widehat{\mathbf{w}})\,\mathbf{B}\widehat{\mathbf{w}}$$

This is the generalized eigenvalue/eigenvector definition (eq. (5.6)) for $\mathbf{w}_i = \widehat{\mathbf{w}}$ and $\lambda_i = r(\widehat{\mathbf{w}})$.

We have thus shown that if $\widehat{\mathbf{w}}$ is a maximum of $r(\mathbf{w})$, that it is an eigenvector of $(\mathbf{A}, \mathbf{B})$, with $r(\widehat{\mathbf{w}})$ its corresponding eigenvalue.

As the second step, we now show that $r(\widehat{\mathbf{w}})$ is the *largest* eigenvalue of $(\mathbf{A}, \mathbf{B})$. We follow the reasoning of Trefethen and Bau, who prove a related result for the ordinary Rayleigh quotient [26, p. 204].

We will rewrite the generalized Rayleigh quotient $r(\mathbf{w})$ by writing the arbitrary vector $\mathbf{w}$ as a linear combination of the generalized eigenvectors $\mathbf{w}_i$ of $(\mathbf{A}, \mathbf{B})$: $\mathbf{w} = \sum_i c_i\mathbf{w}_i$. Then:

$$r(\mathbf{w}) = \frac{(\sum_i c_i\mathbf{w}_i)^T\mathbf{A}(\sum_i c_i\mathbf{w}_i)}{(\sum_i c_i\mathbf{w}_i)^T\mathbf{B}(\sum_i c_i\mathbf{w}_i)}$$

$$= \frac{\sum_i c_i^2\mathbf{w}_i^T\mathbf{A}\mathbf{w}_i}{\sum_i c_i^2\mathbf{w}_i^T\mathbf{B}\mathbf{w}_i}$$

$$= \frac{\sum_i c_i^2\lambda_i\mathbf{w}_i^T\mathbf{B}\mathbf{w}_i}{\sum_i c_i^2\mathbf{w}_i^T\mathbf{B}\mathbf{w}_i}.$$

Generalized eigenvectors are defined up to a scaling factor. We may therefore define our $\mathbf{w}_i$ to be scaled such that $\mathbf{w}_i^T\mathbf{B}\mathbf{w}_i = 1$. We then have:

$$r(\mathbf{w}) = \frac{\sum_i c_i^2\lambda_i}{\sum_i c_i^2}.$$

Each generalized Rayleigh quotient is thus a convex combination of generalized eigenvalues λ_i. The maximum of a convex combination of one-dimensional points is obtained in the largest of these points. If λ_1 is thus the largest generalized eigenvalue of $(\mathbf{A}, \mathbf{B})$, then $\max r(\mathbf{w}) = \lambda_1$.

We have thus shown that $\arg\max\, r(\mathbf{w}) = \mathbf{w}_1$, where $\mathbf{w}_1$ is an eigenvector of $(\mathbf{A}, \mathbf{B})$, and that its corresponding eigenvalue $\lambda_1 = \max r(\mathbf{w})$ is the largest of the eigenvalues of $(\mathbf{A}, \mathbf{B})$.

□

Appendix D

Supplemental figures

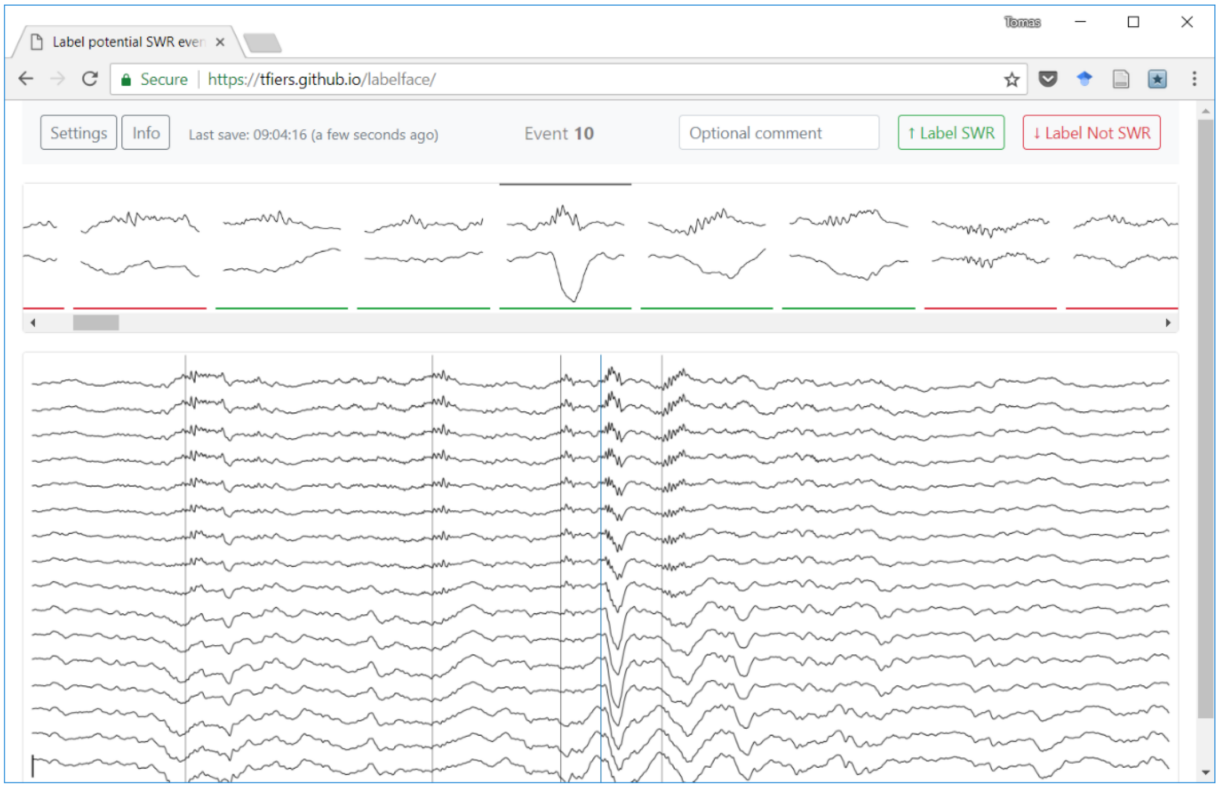
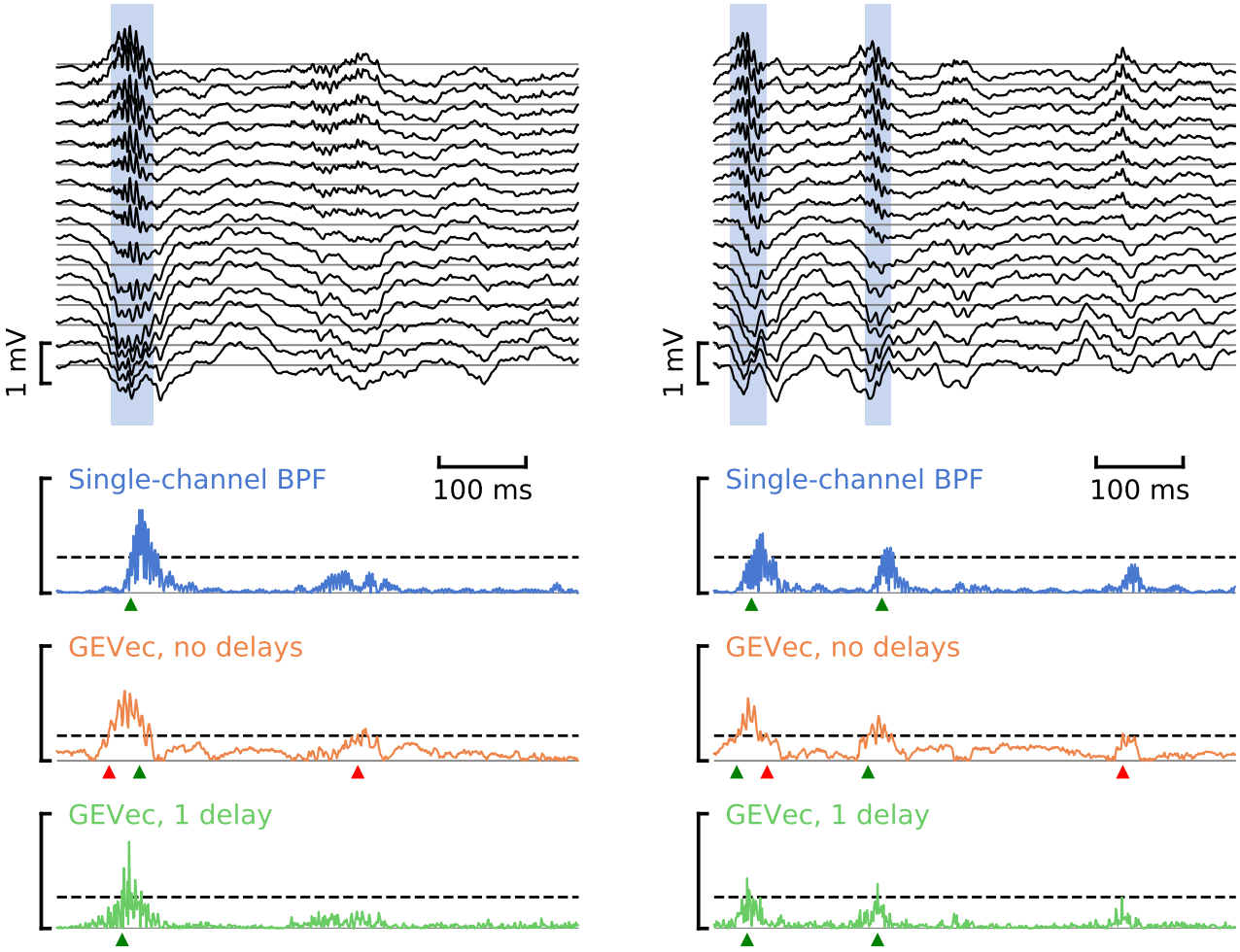


Figure D.1: **User interface for SWR labelling**. Each event in the list at the top is represented by two voltage traces: one from an electrode in the pyramidal cell layer (the top trace), and one from an electrode in the stratum radiatum. The large plot at the bottom gives more comprehensive view of the current event: all 16 channels are plotted (instead of only two), and the plot ranges from 1000 ms before to 1000 ms after the event. In this large plot, the blue vertical line marks the currently active event. The grey vertical lines correspond to other detected events. The scalebar at the beginning of this plot represents 1 mV. Users decide whether the currently active event is an SWR by clicking the buttons in the top-right corner or by using keyboard shortcuts. The source code for this labelling web app is available at `https://github.com/tfiers/labelface`.


1 mV
Single-channel BPF
100 ms
GEVec, no delays
GEVec, 1 delay
a
1 mV
Single-channel BPF
100 ms
GEVec, no delays
GEVec, 1 delay
b

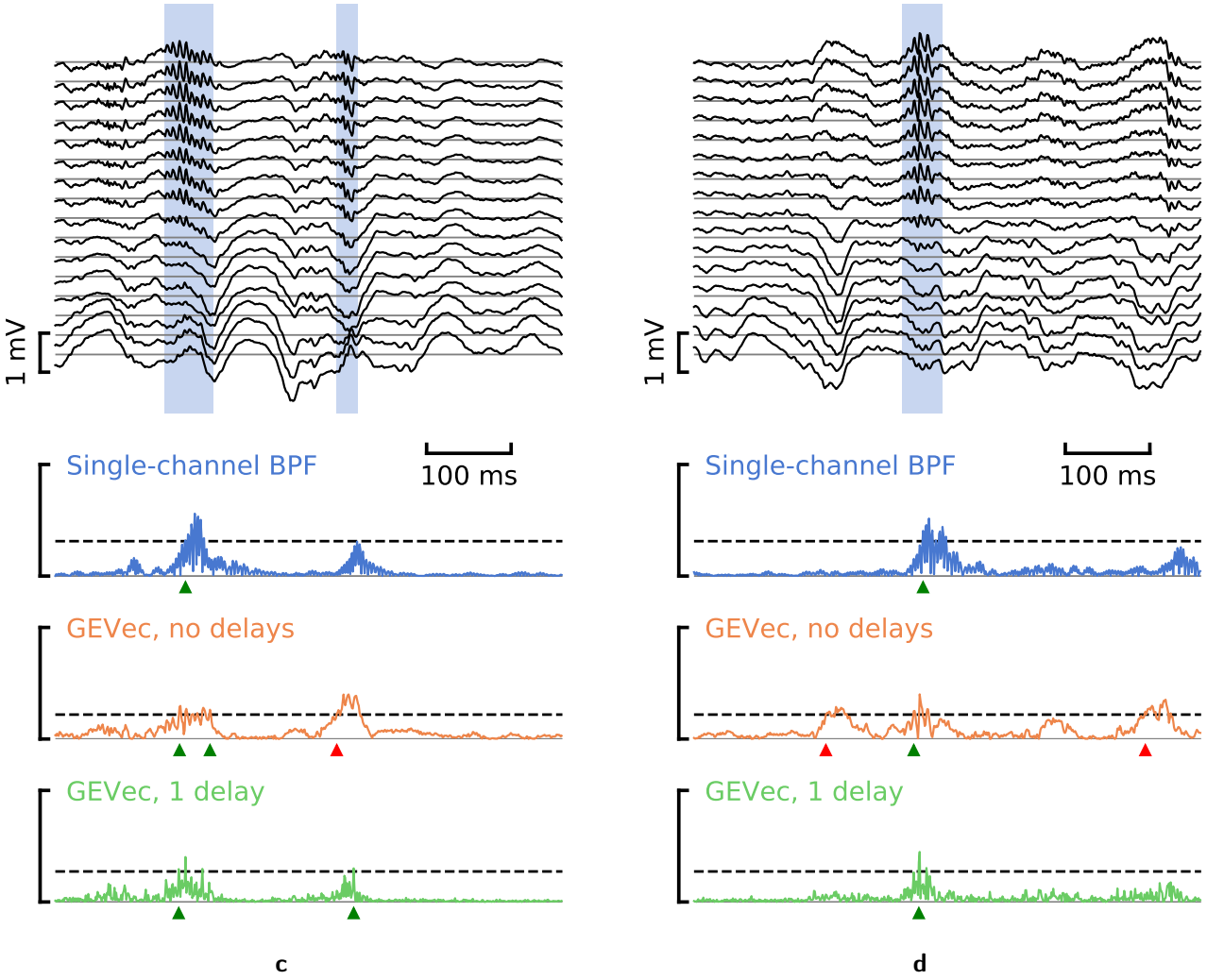


Figure D.2: **Extracts from input data and corresponding linear filter output envelopes**. See fig. 5.2 for legend.

References

[1] C. J. Van Rijsbergen. *Information Retrieval*. 2nd edition. Newton, MA, USA: Butterworth-Heinemann, 1979. URL: http://www.dcs.gla.ac.uk/Keith/Preface.html.

[2] György Buzsáki. "Hippocampal Sharp Wave-ripple: A Cognitive Biomarker for Episodic Memory and Planning". In: *Hippocampus* 25.10 (Sept. 26, 2015), pp. 1073–1188. ISSN: 1050-9631. DOI: 10.1002/hipo.22488. URL: http://onlinelibrary.wiley.com/doi/full/10.1002/hipo.22488 (visited on 04/23/2018).

[3] Tom Roelandts. *How to Create a Configurable Filter Using a Kaiser Window*. Dec. 26, 2016. URL: https://tomroelandts.com/articles/how-to-create-a-configurable-filter-using-a-kaiser-window (visited on 11/25/2018).

[4] Eric Jones, Travis Oliphant, Pearu Peterson, et al. "SciPy: Open Source Scientific Tools for Python". In: (2018). URL: http://www.scipy.org/.

[5] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN: 978-0-13-198842-2.

[6] Richard G. Lyons. *Understanding Digital Signal Processing*. Pearson Education, Nov. 1, 2010. 1166 pp. ISBN: 978-0-13-702852-8.

[7] Zoltán Nádasdy, Hajime Hirase, András Czurkó, Jozsef Csicsvari, and György Buzsáki. "Replay and Time Compression of Recurring Spike Sequences in the Hippocampus". In: *Journal of Neuroscience* 19.21 (Nov. 1, 1999), pp. 9497–9507. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.19-21-09497.1999. pmid: 10531452. URL: http://www.jneurosci.org/content/19/21/9497 (visited on 08/02/2018).

[8] Jozsef Csicsvari, Hajime Hirase, Akira Mamiya, and György Buzsáki. "Ensemble Patterns of Hippocampal CA3-CA1 Neurons during Sharp Wave–Associated Population Events". In: *Neuron* 28.2 (Nov. 1, 2000), pp. 585–594. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(00)00135-5. URL: http://www.sciencedirect.com/science/article/pii/S0896627300001355 (visited on 08/01/2018).

[9] Shayok Dutta, Etienne Ackermann, and Caleb Kemere. "Analysis of an Open Source, Closed-Loop, Realtime System for Hippocampal Sharp-Wave Ripple Disruption". In: *bioRxiv* (Apr. 11, 2018). DOI: 10.1101/298661. URL: https://www.biorxiv.org/content/early/2018/04/11/298661 (visited on 05/30/2018).

[10] Christoph J. Behrens, Leander P. van den Boom, Livia de Hoz, Alon Friedman, and Uwe Heinemann. "Induction of Sharp Wave–Ripple Complexes *in Vitro* and Reorganization of Hippocampal Networks". In: *Nature Neuroscience* 8.11 (Nov. 2005), pp. 1560–1567. ISSN: 1546-1726. DOI: 10.1038/nn1571. URL: https://www.nature.com/articles/nn1571 (visited on 07/31/2018).

[11] Josef H. L. P. Sadowski, Matthew W. Jones, and Jack R. Mellor. "Sharp-Wave Ripples Orchestrate the Induction of Synaptic Plasticity during Reactivation of Place Cell Firing Patterns in the Hippocampus". In: *Cell Reports* 14.8 (Mar. 1, 2016), pp. 1916–1929. ISSN: 2211-1247. DOI: 10.1016/j.celrep.2016.01.061. URL: http://www.sciencedirect.com/science/article/pii/S2211124716300390 (visited on 06/17/2018).

[12] Davide Ciliberti and Fabian Kloosterman. "Falcon: A Highly Flexible Open-Source Software for Closed-Loop Neuroscience". In: *Journal of neural engineering* 14.4 (2017), p. 045004.

[13] Valérie Ego-Stengel and Matthew A Wilson. "Disruption of Ripple-Associated Hippocampal Activity during Rest Impairs Spatial Learning in the Rat". In: *Hippocampus* 20.1 (2009), pp. 1–10. URL: http://onlinelibrary.wiley.com.kuleuven.ezproxy.kuleuven.be/doi/10.1002/hipo.20707/full.

[14] Shantanu P. Jadhav, Caleb Kemere, P. Walter German, and Loren M. Frank. "Awake Hippocampal Sharp-Wave Ripples Support Spatial Memory". In: *Science* 336.6087 (June 15, 2012), pp. 1454–1458. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1217230. pmid: 22555434. URL: http://science.sciencemag.org/content/336/6087/1454 (visited on 04/19/2018).

[15] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Fourth edition. The Johns Hopkins University Press, 2013.

[16] John O'Keefe. "Hippocampal Neurophysiology in the Behaving Animal". In: *The Hippocampus Book*. Oxford University Press, 2007.

[17] Gabrielle Girardeau, Karim Benchenane, Sidney I Wiener, György Buzsáki, and Michaël B Zugaro. "Selective Suppression of Hippocampal Ripples Impairs Spatial Memory". In: *Nature Neuroscience* 12.10 (Oct. 2009), pp. 1222–1223. ISSN: 1097-6256, 1546-1726. DOI: 10.1038/nn.2384. URL: http://www.nature.com/articles/nn.2384 (visited on 04/16/2018).

[18] Laura Lee Colgin. "Rhythms of the Hippocampal Network". In: *Nature Reviews Neuroscience* 17.4 (2016), p. 239.

[19] Omid Talakoub, Andrea Gomez Palacio Schjetnan, Taufik A. Valiante, Milos R. Popovic, and Kari L. Hoffman. "Closed-Loop Interruption of Hippocampal Ripples through Fornix Stimulation in the Non-Human Primate". In: *Brain Stimulation: Basic, Translational, and Clinical Research in Neuromodulation* 9.6 (Nov. 1, 2016), pp. 911–918. ISSN: 1935-861X, 1876-4754. DOI: 10.1016/j.brs.2016.07.010. pmid: 27576185. URL: http://www.brainstimjrnl.com/article/S1935-861X(16)30202-9/abstract (visited on 04/16/2018).

[20] Howard Eichenbaum. "Prefrontal–Hippocampal Interactions in Episodic Memory". In: *Nature Reviews Neuroscience* 18.9 (2017), p. 547.

[21] H Freyja Ólafsdóttir, Daniel Bush, and Caswell Barry. "The Role of Hippocampal Replay in Memory and Planning". In: *Current Biology* 28.1 (2018), R37–R50.

[22] Jozsef Csicsvari, Hajime Hirase, András Czurkó, Akira Mamiya, and György Buzsáki. "Oscillatory Coupling of Hippocampal Pyramidal Cells and Interneurons in the Behaving Rat". In: *Journal of Neuroscience* 19.1 (Jan. 1, 1999), pp. 274–287. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.19-01-00274.1999. pmid: 9870957. URL: http://www.jneurosci.org/content/19/1/274 (visited on 11/13/2018).

[23] Jozsef Csicsvari, Hajime Hirase, András Czurkó, Akira Mamiya, and György Buzsáki. "Fast Network Oscillations in the Hippocampal CA1 Region of the Behaving Rat". In: *The Journal of Neuroscience* 19.16 (Aug. 15, 1999), RC20–RC20. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.19-16-j0001.1999. URL: http://www.jneurosci.org/lookup/doi/10.1523/JNEUROSCI.19-16-j0001.1999 (visited on 11/13/2018).

[24] Sen Cheng and Loren M. Frank. "New Experiences Enhance Coordinated Neural Activity in the Hippocampus". In: *Neuron* 57.2 (Jan. 24, 2008), pp. 303–313. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2007.11.035. pmid: 18215626. URL: https://www.cell.com/neuron/abstract/S0896-6273(07)01030-6 (visited on 11/13/2018).

[25] Caleb Kemere, Margaret F. Carr, Mattias P. Karlsson, and Loren M. Frank. "Rapid and Continuous Modulation of Hippocampal Network State during Exploration of New Places". In: *PLOS ONE* 8.9 (2-Sep-2013), e73114. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0073114. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0073114 (visited on 11/13/2018).

[26] Lloyd N. Trefethen and David III Bau. *Numerical Linear Algebra*. Siam, 1997.