ReScience

[Re] Spike Synchronization and Rate Modulation Differentially Involved in Motor Cortical Function

Vahid Rostami[1], **Junji Ito**[1], **Michael Denker**[1], **and Sonja Grün**[1,2]

1 Inst. of Neuroscience & Medicine (INM-6) and Inst. for Advanced Simulation (IAS-6), JARA Brain Institute I, Jülich Research Center, Jülich, Germany **2** Theoretical Systems Neurobiology, RWTH Aachen University, Aachen, Germany

v.rostami@fz-juelich.de

Editor
Nicolas P. Rougier

Reviewers
Georgios Detorakis
Fabien Benureau

Received Oct 28, 2016
Accepted Apr 6, 2017
Published May 29, 2017

Competing Interests:
The authors have declared that no competing interests exist.

Article repository
Code repository
Data repository

A reference implementation of

→ Spike synchronization and rate modulation differentially involved in motor cortical function. Alexa Riehle, Sonja Grün, Markus Diesmann, and Ad Aertsen (1997) Science 278:1950-1953. DOI:10.1126/science.278.5345.19 50

Introduction

Understanding how information is processed by networks of neurons in the brain is a major goal in neuroscience. There has been a long-standing debate in the community on the contribution of the firing activity of individual neurons or populations of neurons to information processing: one perspective is that the firing rate of the neurons, i.e. the number of spikes per time unit, represents information or is relevant for the processing. Another perspective is that neurons communicate through precisely timed coordination of spiking, a view that results from the insight that a neuron fires most efficiently if it receives synchronous spike input, i.e., in the range of a few milliseconds [1].

To test whether temporal coordination of spiking activity is indeed relevant for neuronal information processing, advanced data analysis methods are required that perform correlation analysis between simultaneously recorded single unit spike trains. The Unitary Events (UE) analysis method [12, 18, 13, 14] is able to extract significant spike synchrony between neuronal activities that is beyond what is expected by chance (given the rates) and to follow the dynamics of this coordination. The tricky issue with such analyses is that one has to take into account that experimental spike data are typically non-stationary in the sense that the firing rates are not stationary in time and not homogeneous across trials, and that the statistics of the individual spike trains deviate from those of a Poisson process [16, 17]. If such features are ignored there is a considerable danger of occurrence of false positives and thus wrong interpretation of the data. In particular, changes in the firing rates are the most prominent generators of false positives if ignored. The original UE method considers this aspect by performing the analysis in a sliding time window fashion. In later versions of the analysis method other features were corrected for by considering the respective features in the null-hypothesis of the tests [15, 22, 21, 24], either by using extended analytical descriptions of the null-hypotheses or by use of surrogate methods [16, 17, 21].

The UE method had been applied to experimental parallel spike data from, e.g., the motor cortex of awake behaving non-human primates [26, 18, 10, 25, 15, 20, 6, 7] and to data from visual cortices [22, 19]. Generally it was found that UEs, i.e.,

synchronous spike events across neurons that are in excess of the expectation, occur in relation to behavior, e.g., when the animal expects a signal to occur, however the signal does not occur [26]. This finding, in particular, demonstrates the core feature of the method: by performing a time resolved analysis, the method accounts for changes in the firing rates and captures modulations of significant spike synchrony in time. It was also shown that the time of occurrence of UEs may change to a new requested timing in the behavior during a learning process [20].

Several of these studies based on the UE method, in particular [26], have been widely cited and the method has been recognized as one of the standard tools to analyze temporal coordination of neuronal spiking activities [4, 23]. The analysis method has been and is taught in international data analysis courses. However, a publicly available and open source implementation of the UE method had not been available. Only very recently, one of the authors of this study (VR) reimplemented the UE analysis as part of the Electrophysiology Analysis Toolbox[1] (Elephant), a Python library that provides implementations for the analysis of electrophysiological data. This reimplementation of the method was particularly required for, in addition to the general motivation for a reimplementation to confirm the results not depending on small implementation details or mistakes, the following two specific reasons.

First, the custom data object model used to represent the primary data and meta-data in the original analysis code was not documented. Therefore, any data represented in a specific file format had to be converted by implementing a custom data loading routine for this data model. Our new implementation of the UE method is part of the Elephant and Neo[2] libraries and is based on the internal data object model provided by the Neo library [9], a package for representing electrophysiology data in Python. Neo provides support for reading a wide range of neurophysiology proprietary file formats, and supports writing to a subset of these formats, including non-proprietary formats such as HDF5.

Second, the implementation of the UE method in the original publication has experienced several updates after its publication in order to include improvements and extensions of the method to accommodate more features of experimental data. Since no version control was employed in this development process, the original code used in Riehle et al. [26] was lost at some point. Considering that no systematic testing was performed each time the code base of the Unitary Event analysis was updated after the original publication to check whether the code gives the same results as before, it was not clear whether the latest version could exactly reproduce the results of [26].

In this paper we illustrate the successful reproduction of the results shown in [26] using our new Python implementation of the UE method. In particular, we reproduce Figure 2 and Figure 4A of the original paper, which represent the central results of the original study. The remainder of the original paper consisted of more example analyses of individual data sets and a meta statistics across many data sets, all of which were based on the same analysis method and thus do not provide additional insight when reproduced. In the original publication the authors used two different UE implementations: one implemented by one of the coauthors of the present study (SG) in IDL[3], and the other in Matlab (Mathworks, Nattick, MA) implemented later by Markus Diesmann (MD) and SG. The IDL implementation is not available anymore. At a later point we were provided with a Matlab implementation of the extended UE method, however it does not preserve the original implementation used in [26] and was not considered in this study.

For the reproduction of the original results we contacted and communicated with Alexa Riehle (AR), CNRS-AMU, Marseille, and MD, Research Centre Jülich. AR is the first author of the original publication and performed the experiments and data analysis. MD is the third author of the original publication and contributed with

[1] http://neuralensemble.org/elephant/
[2] http://neuralensemble.org/neo/
[3] http://www.harrisgeospatial.com/ProductsandSolutions/GeospatialProducts/IDL.aspx

the Matlab implementation and quality checks of the software implementations. The reproduction of the results of [26] would not have been possible without contacting these authors, since the information in the original publication is not sufficient for reproducing the results. AR provided us with the original data and information, including an old written report by MD, which was crucial to reproduce Figure 4A.

Methods

For reproducing the results of [26] we use our reimplementation of the UE analysis method in Python which is made available in the `unitary_event_analysis` module of the Elephant library (accepted pull-request: [27]). The method is accompanied by unit tests for individual functions (test coverage: 88.54%) and documented as part of the library documentation. The structure and algorithm of our UE implementation is explained in the pseudo-code shown below.

In the following, we will explain the algorithm in detail. The primary data entering the UE method is a set of parallel, i.e., simultaneously recorded, spike trains, recorded in one or multiple trials. In Elephant, an individual spike train of a particular neuron in a particular trial data is represented as a `Spiketrain` object in the data object model provided by the Neo library, which stores the time points of spike occurrences along with additional information describing the spike train, such as the start and end times. The UE method consumes a nested list of `Spiketrain` objects relating to the spike trains of individual neurons in the individual trials. In order to perform the necessary calculations, the spike data must be converted to an alternative time-binned representation (line 1 in the pseudo code) where the parallel spike trains are stored as binary sequences of ones (marking time bins containing at least 1 spike) and zeros (bins with no spike). The bin size is a parameter provided as input (parameter ***bin_size*** in the pseudo code) to the analysis, and defines the temporal precision in detecting spike synchrony. A spike pattern, which is a representation of spike synchrony in the analysis, is defined as a specific vector of zeros and ones in one time bin of a given trial across all neurons. Since the aim of the UE method is to detect significant spike synchrony, the total number of spike patterns of concern is given by $2^N - N - 1$, where N is the number of neurons, the first term is the number of all possible spike patterns, the second term is the number of spike patterns composed of only one spike, and the third term is the number of the pattern without a spike. In order to limit the analysis to a set of patterns of interest, the input parameter ***pattern_hash_values*** specifies the patterns to consider in the analysis in the form of a hash value which uniquely represents each spike pattern (line 2 in the pseudo code). The hash value is obtained by interpreting the binary spike pattern as a binary number, where the n-th neuron is represented by bit $n - 1$.

The UE analysis is performed in a sliding time window fashion, i.e. the data in each window are analyzed separately. This approach is chosen to account for potential changes of the firing rates in time and to follow the dynamics of the correlation. Here, a sliding time window is defined based on trial time, meaning that a certain window position includes the activity of all neurons in all trials in a certain time interval of the trial. In the algorithm, at each position of the window, the data contained in the window are extracted in order to compute the significance of the specified patterns (line 3 in the pseudo code). For doing that the UE analysis requires the empirical number of occurrences of each pattern as well as its expected number given the firing rates. While the empirical number is directly extracted from the data (line 5 in the pseudo code), the method to compute the expected number can be chosen using the input parameter ***method*** depending on the assumptions regarding the data (line 6-19 in the pseudo code). For cross-trial homogeneous data, selecting *"analytic_TrialAverage"* as ***method*** (line 7 in the pseudo code) computes the expected number by estimating the rate of each neuron from the average spike count across trials [13, 14]. For data with

Algorithm of the Unitary Events method: Black color shows the part of the code used for reproduction of the results of Riehle et al (1997) and the gray color indicates the additional features which are now available in our reimplementation but not used specifically here.

input : `spike_trains`, `bin_size`, `window_size`, `window_step`, `pattern_hash_values`, `method`

output: time series of: suprise `S`, number of empirical coincidences `n_emp`, number of expected coincidences `n_exp`, firing rate of each neuron `rate`

1 Discretize the input spike train data (`spike_trains`) and represent them as binary sequences (‘BinnedSpikeTrain()‘ from `conversion` module in Elephant)
2 **for** *each hash value from* `pattern_hash_values` **do**
3 Define the sliding window positions over the whole length of the data given `window_size` and `window_step` (‘_winpos()‘ and ‘`_bintime()`‘)
4 **for** *each window position* **do**
5 Calculate `n_emp` (‘n_emp_mat()‘ and ‘n_emp_mat_sum_trial()‘)
6 **if** *method is “analytic_TrialAverage” or “analytic_TrialByTrial”* **then**
7 **if** *method is “analytic_TrialAverage”* **then**
8 calculate `n_exp` based on *“analytic_TrialAverage”* (‘_n_exp_mat_analytic()‘ and ‘n_exp_mat_sum_trial()‘)
9 **end**
10 **else if** *method is “analytic_TrialByTrial”* **then**
11 Calculate `n_exp` based on *“analytic_TrialByTrial”* (‘_n_exp_mat_analytic()‘ and ‘n_exp_mat_sum_trial()‘)
12 **end**
13 Generate Poisson distribution with the mean `n_exp`
14 **end**
15 **else if** *method is “surrogate_TrialByTrial”* **then**
16 Generate `n_exp` distribution based on surrogates (‘_n_exp_mat_surrogate()‘ and ‘n_exp_mat_sum_trial()‘)
17 **end**
18 Calculate p-value of `n_emp` given the distribution of `n_exp`
19 **end**
20 Calculate the surprise as a function of time from the p-values at each window position (‘jointJ()‘)
21 **end**

cross-trial variability, the *“analytic_ TrialByTrial”* method (line 10 in the pseudo code) should be used, which accounts for cross-trial changes in the firing rates by computing the expected number of occurrences of the spike pattern based on the product of single-

trial estimates of firing rates [15]. Both methods perform a parametric test, where the number of occurrences of the spike pattern is assumed to be a Poisson-distributed. As an alternative non-parametric option, selecting *"surrogate_TrialByTrial"* as `method` (line 15 in the pseudo code) will numerically compute the distribution of the expected number by implementing the null-hypothesis based on surrogate spike trains [16]. In the following, we will explain these methods in greater detail.

In case of selecting *"analytic_TrialAverage"* - as used in the remainder of this study for the reproduction of [26] - the number of spikes per neuron within the sliding window is summed across trials and divided by the number of trials and bins contained, thus yielding the average probability p_i to have a spike of neuron i in a bin of the time window. The probability to find a particular pattern by chance in a bin is then computed by multiplication of the relevant probabilities, e.g. for a pattern [1,0,1] the probability p of occurrence is given by $p_{[1,0,1]} = p_1 * (1 - p_2) * p_3$. Note that $(1 - p_2)$ is the probability for neuron 2 to contribute no spike to the pattern. The expected number of pattern occurrences, computed as the product of the occurrence probability p of the pattern, e.g. $p_{[1,0,1]}$, and the number of bins (across all trials) covered by the sliding window. The distribution of pattern occurrence numbers is given by a Poisson distribution with the mean equal to the expected number.

Alternatively, for the *"analytic_TrialByTrial"* method, the firing probability of each neuron is calculated in a trial-by-trial manner based on the spike counts per trial. The probability p of finding the pattern by chance is calculated by summing the products of the firing probabilities obtained individually from each trial. As for the trial-averaging method, the expected number of pattern occurrences is given by multiplication with the number of bins of the trial in the sliding window, used as the mean of a Poisson distribution to obtain the distribution of expected pattern occurrence numbers.

As a third alternative, a surrogate method for estimating the expected number can be selected using *"surrogate_TrialByTrial"* for the parameter `method`. In this Monte-Carlo approach, a surrogate version of the spike trains is generated repeatedly, and from each surrogate the number of occurrences of the pattern of interest is counted. The method by which surrogates are generated from the input spike trains is spike time randomization of the spikes per trial and per neuron within the sliding window. The pattern counts obtained from this procedure form a distribution of the expected number of occurrences of the pattern, thus implementing the null-hypothesis under the constraints implied by the surrogate method.

The distribution obtained by either of the three methods above is then used for the significance test of the pattern on the basis of the empirical occurrence count. The p-value resulting from the test is then transformed by a logarithmic transformation to the surprise value (line 20 in the pseudo code), which indicates by positive or negative values more or less occurrences of the pattern than expected by chance, respectively. If the p-value is below a fixed prescribed level (e.g. below 5%, which corresponds to a surprise value exceeding 1.27), the occurrences of the spike pattern under investigation in the sliding window are marked as UEs for that pattern. This procedure is performed for each pattern of interest, and in each sliding window.

In the present study, in order to reproduce the original results we used the *"analytic_TrialAverage"* method, which reflects the analysis performed in the original publication [26]. The *"analytic_TrialByTrial"* and *"surrogate_TrialByTrial"* methods are extensions of the original UE method, which were developed after the original publication and introduced in subsequent works [15, 16].

Our reimplementation of the UE method is based on the data object model provided by the Neo library, upon which the Elephant library is based. The Neo library provides loading routines for a variety of data formats, including proprietary and generic data formats. The data sets available for reproducing Figures 2 and 4A of [26] were tab-separated ASCII text files containing two columns of integers (informally often referred to as "GDF-format"): the first column provides event codes (behavioral events or

neuron IDs), and the second column contains the time of the occurrence of these events (time stamps). The units of the time stamps are not contained in the data file. We partly extracted metadata information, in particular the time units and the meaning of the event codes, from a Matlab routine (provided by AR) operating on the GDF data file. However, only after further communication with AR we were able to identify the exact meaning of the content of the data files. Using this information, we wrote a new loading routine that loads the GDF data as Neo data objects.

Our reimplementation uses the `conversion` module of Elephant for converting the spike data (represented as a series of timestamps) into the binary sequence to guarantee a unique, global binning mechanism for all analysis methods provided in Elephant. The bin size to be set for the analysis was extracted from the original publication. However, defining the time point to start the binning of each single trial data required to know the alignment event in each trial and how much time before this event (pre-time) is considered. Since this information was not documented in the original paper, we tried several possibilities until we got an agreement with the original figures as will be shown in the Results.

To check if our Python implementation produces the same results as the implementation(s) used in the original publication, we compare each of our figures in detail with the original figures. For this comparison, as the original results used to generate these figures are not available to us, we first extract the times of the spikes and unitary events from the vector graphic image (PDF) in the original paper. Then, we directly compare these times to the times of spikes and unitary events in our reproduced results by plotting the former against the latter, as well as examining the distribution of the differences between them.

Results

For the reproduction of the original results in [26], we have to focus on reproducing Figure 2A-F and Figure 4A, since for the rest of the figures data were either incomplete in respect to metadata (original Figure 3) or not available (original Figure 4B,C). Figure 2 represents the main result of the study and includes the UE analysis that underlies the subsequent analyses. Figure 4A is an example of the application of the UE method to data with more than 2 neurons. In terms of complexity of the code the implementation of the UE analysis for three or more neurons is considerably more demanding than for only two neurons. With this example we show that our implementation is capable of performing the UE analysis for the generic case of arbitrary number of neurons.

We apply our reimplementation of the UE method to preprocessed versions of the spike train data available to us after communication with AR, which in part were identical to those used in the original analysis. Also, we learned from AR that Figure 2 was generated by the Matlab implementation of the UE method while all remaining figures of the original publication, including Figure 4A, were generated by the older implementation in IDL (see Methods regarding versions of the original code).

Let us start with the reproduction of Figure 2 of the original publication. We first give a brief description of the experiment (see the original publication for details). After the monkey was presented with the preparatory signal (PS) he had to sit still and wait for a response signal (RS) to start his arm movement (i.e. equivalent to a GO signal). The duration of the waiting period was randomly selected on a trial-by-trial basis to be either 600, 900, 1200 or 1500 ms. In Figure 2 of [26] only trials of the longest waiting period (1500 ms) were used for the analysis. In these trials, times marked as expected signals ES1, ES2, and ES3 corresponded to the ends of the three shorter waiting periods, at which the monkey could have gotten the RS signal but did not. As the monkey was trained to recognize and distinguish the four waiting periods, but was not informed of the randomly selected period for a given trial, ES1-ES3 were

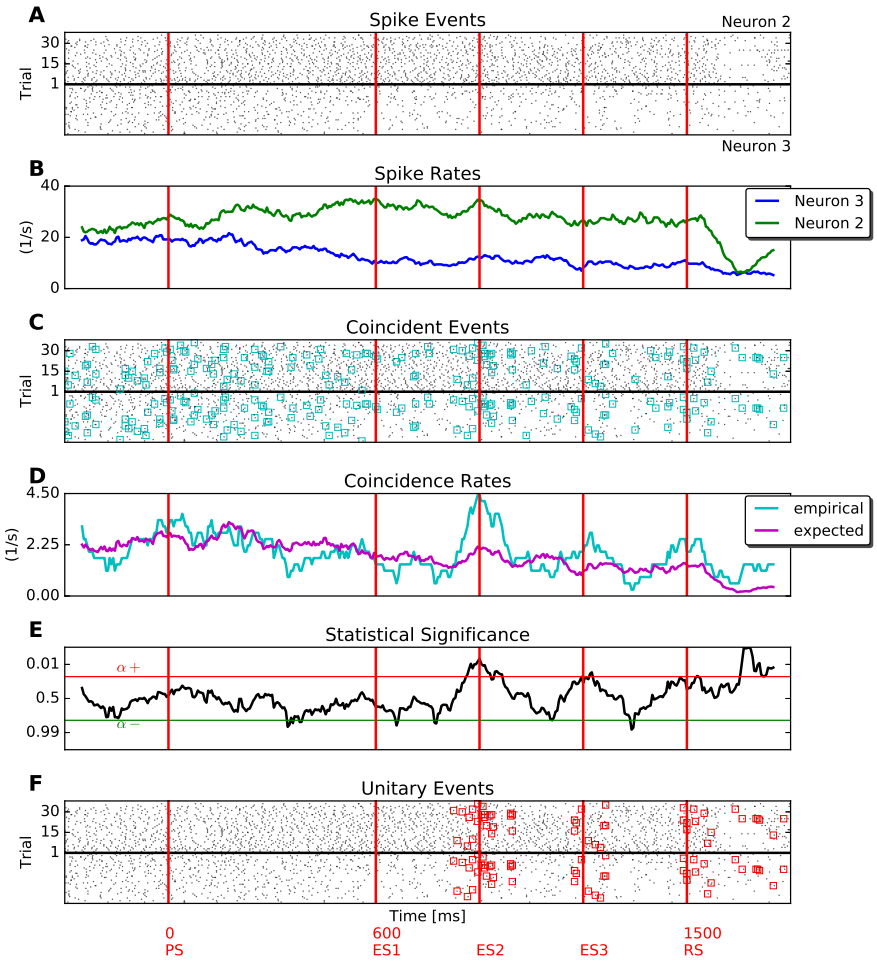

Figure 1: Initial attempt to reproduce Figure 2 of the original publication with trial alignment to PS. **A)** Raster plot of two neurons (neuron 2: top of panel; neuron 3: bottom of panel) in 32 trials (sorted identically for both neurons). **B)** Average firing rate of each neuron calculated across trials in a sliding window of length 100 ms in steps of 5 ms. **C)** Same raster plot as in panel A with spike coincidences (i.e., pattern [1,1]) between the two neurons marked by cyan squares. **D)** Empirical (cyan) and expected (magenta) number of coincidences calculated in a time-resolved manner (parameters of sliding window identical to panel B). **E)** Time course of the surprise measure, calculated in same sliding windows as in panel B. Surprise values that correspond to positive and negative significance levels $\alpha+ = 0.05$ and $\alpha- = 0.95$ are shown with by horizontal red and green lines, respectively. **F)** Same raster plot as in panel A with significant coincidences, i.e. UEs, marked by red squares.

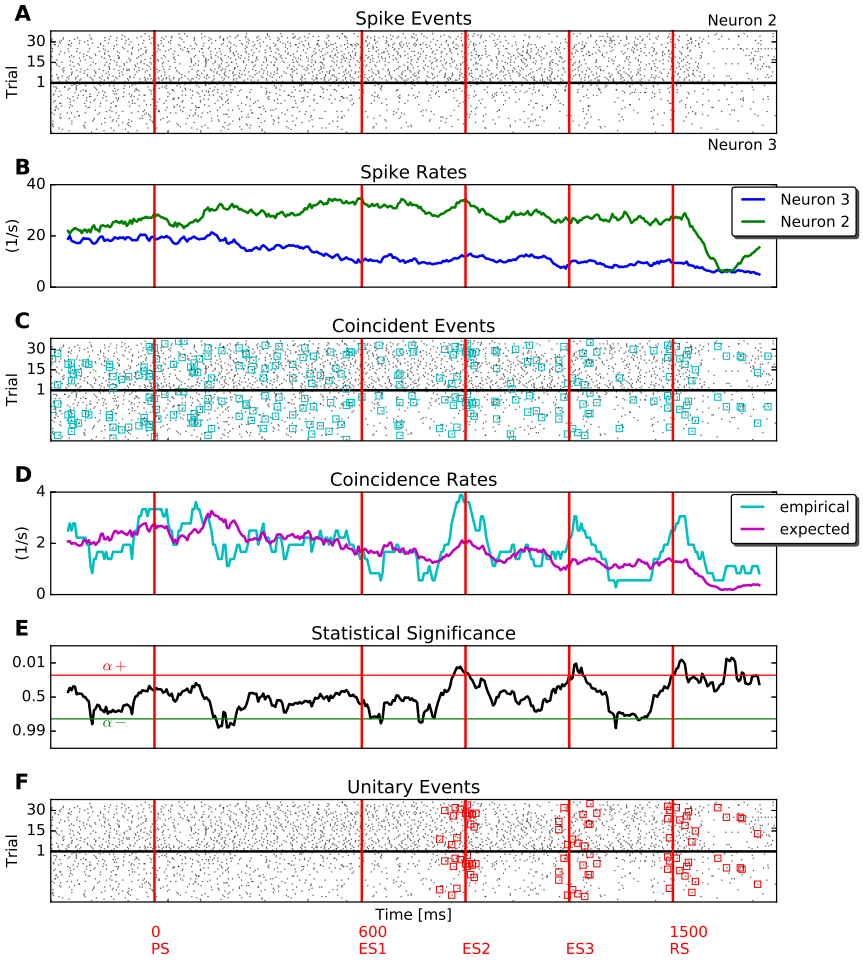

Figure 2: Reproduction of Figure 2 of the original publication with trial alignment to RS. The same conventions as in Figure 1 apply to the respective panels.

time points at which the monkey expected that a signal could occur.

Since the data file for Figure 2 contained the data as a continuous recording of one recording session ("winny131.gdf"; 2 neurons, and behavioral events), we extract the trials by cutting the data in a time window around specific trigger events that belong to trials of the longest waiting period, such that the complete trial is contained in the cut-out. In a subsequent step, the spike times in the individual trials are aligned to the trigger event, such that spike times in each trial are given as relative to the trigger.

The original publication does not provide information which event was used as the trigger. In this experiment, 2 events that occur in every trial could serve as trigger events, the preparatory signal PS (event code 114) and the response signal RS (event code 124). We noticed that the time interval between PS and RS for the longest trials was not identical across the respective trials and varied by ± 1 ms. Given the UE method is applied on a time scale of 5 ms, the analysis results therefore are expected to depend on whether trials are aligned to PS or RS. Thus, we decide to generate the results for both alignments.

Figures1 and 2 show the results of performing the UE analysis for PS- and RS-aligned data, respectively. Here, the analysis parameters are set to the identical values as reported in the original publication (bin size: 5 ms, analysis time window size: 100 ms, time step of the sliding window: 5 ms, significance level $\alpha = 0.05$). The comparison of the two figures to the original figure shows agreement in the raster displays (panel A) and the time-resolved, trial-averaged firing rate estimates (B). However, although the graphs of the number of coincidences per sliding window (panel D) and the surprise measure (panel E) are similar in their overall general behavior, they differ in the details. Thus, indeed the choice of the alignment influences the analysis result. In order to test if one of the two alignments is in agreement with the original publication, we perform a detailed visual comparison of our two figures and the original one on the basis of the spikes marked as coincident (panels C) and as part of a UE (panel D). We notice that when aligning to PS (Figure1), the marked spikes do not agree in all details with the original figure. However, in Figure2, with trials aligned to RS, we find no disagreements with the original figure. After this visual comparison we check if our results are exactly identical to the original ones. Therefore, we extract the positions of the data points representing spikes and UEs in the original figure by reading the plotting commands in the PDF file of the original paper, and compare them to our reproduced results. In Figure3 A the reproduced spike times in Figure2 A are plotted against the extracted original spike times. The plotted data points lie on the diagonal line, indicating that the reproduced spike times correspond to the original ones. Figure3 B shows the same plot for the UEs, indicating the identity of the original and reproduced UE timings as well. To further confirm the identity, we plot the distributions of the differences between the original and the reproduced timings for the spikes (Figure3 C; black) and the UEs (Figure3 D). The differences are at most +/- 0.3 ms, which are considerably narrower than the +/- 1 ms differences caused by the misaligned data shown in Figure1 A (see for comparison the gray plot in Figure3 C, which shows the differences between the original spike times and the misaligned spike times). The remaining minor differences of the spike and UE times of the correctly aligned data and the original data are only due to slight errors in the extraction of the spike times from the original figure, which is inevitable because of a limited precision of the plotting commands in the PDF file. Thus, we confirm that spike timings in the correctly aligned data are identical to those in the original data, and our implementation of the UE analysis applied to these data reproduces exactly the same results as shown in the original paper.

As a next step we aim at reproducing Figure 4A of [26]. This figure contains the result of the analysis of three neurons recorded simultaneously, in contrast to Figure 2 where only two neurons are considered. We analyze the original data for this figure provided by AR with the parameter values given in [26] and compare our result to the original figure. Figure 4A of the original publication contains the raster displays

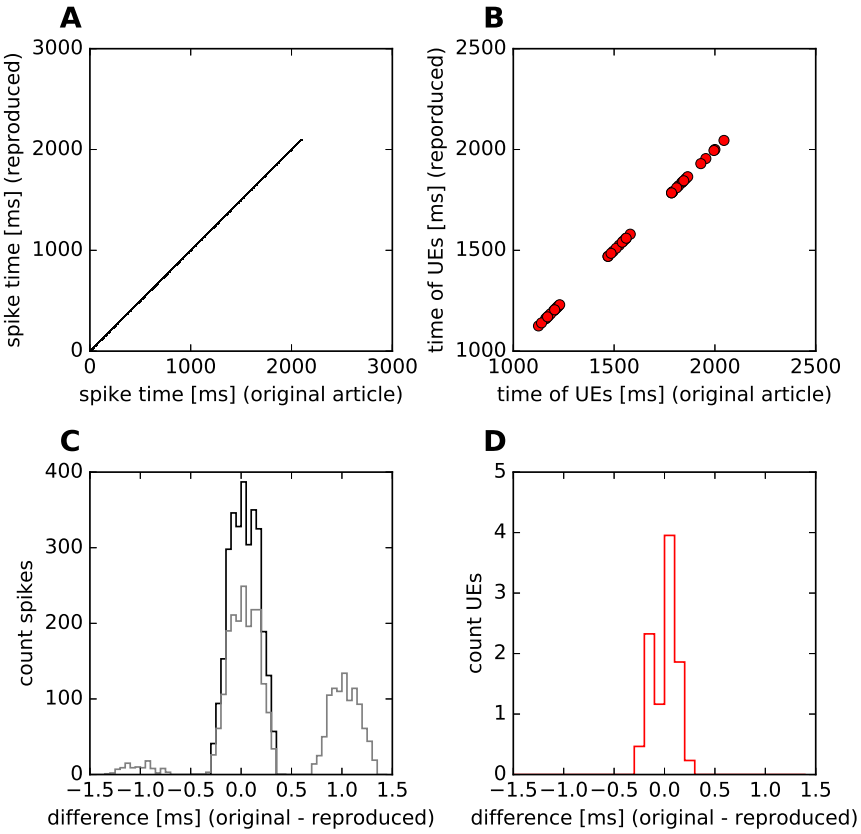

Figure 3: **A)** The scatter plot of the reproduced spike times in Figure2 A plotted against the extracted original spike times. **B)** Same plot as in A but for the time of occurrences of UEs. **C)** The distributions of the differences between the original and the reproduced timings for spikes in Figure 2 and Figure1 are shown in black and gray, respectively. **D)** Same plot as in C but for UEs in Figure 2.

of the data in the top panel, the raster displays with the marked coincident spikes (blue marks) in the middle panel, and the raster displays with the marked spikes that are part of a UE (red marks) in the bottom panel. We find that the UE result is different, as the UEs occur at different times and between different neurons compared to the original publication. Thus we check whether the spike times of the individual spikes are identical between the original and our results. Figure4 shows a segment of the raster plot of the original figure and the corresponding segment of our reproduced figure. We compare the positions of the single spikes and find that there are small discrepancies between the two raster plots in some of the spike times. Figure4 shows examples of clusters of spikes marked in red that should be identical in both raster plots but contain a few individual spikes that are slightly shifted in our figure compared to the original figure by a very small amount.

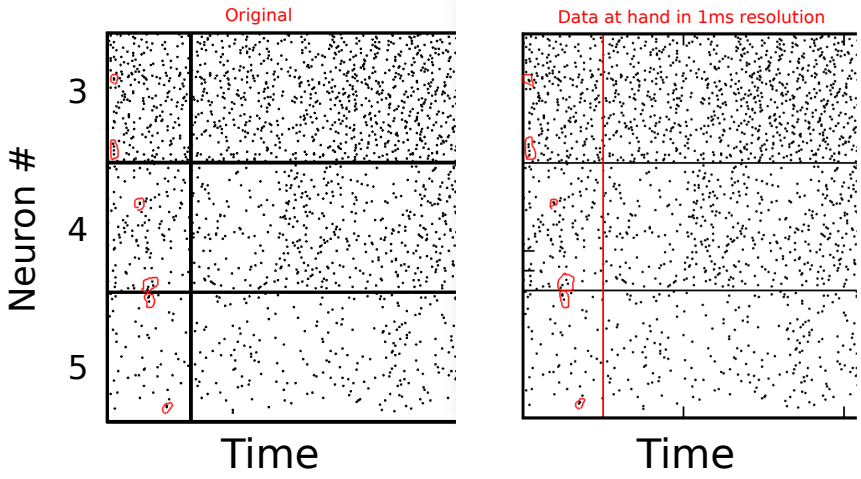

Figure 4: Close-ups of the original raster display in Figure 4A of the original publication (left) and the first data file available at hand for reproduction (right) reveal slight differences in the positions of some spikes. Data on the right are aligned (similar to the data on the left) to ES1 (event code 15 in the GDF data file). The time before the alignment event is chosen as 700 ms, and a bin size of 5 ms is used. The red marks indicate spike clusters with identified differences between the left and the right panel, where at least one spike is shifted in the right panel compared to the left one.

This leads us to the suspicion that the data are binned in a fashion that is not consistent with the data shown in the original publication. Personal communication with AR revealed that while Figure 2 had been generated by the Matlab implementation of the UE analysis, Figure 4 had been generated by the IDL implementation (see Introduction). A report by MD written before the time of the original publication summarized a comparison of the IDL and the Matlab implementations, and concluded that both were correct implementations of the method, but differed in their results due to a slightly different implementation of the down-sampling and binning of the raw data (recorded at 10 kHz). In the workflow for the IDL implementation, as illustrated in Figure5 (the leftmost branch of the diagram), the raw data were first down-sampled to a temporal resolution of 0.5 ms (by a program `2gdf`) and then further rounded to 1 ms resolution integer values inside the IDL implementation. The data available to us had a resolution of 1 ms, which must have been a result of another down-sampling procedure than the one for the IDL implementation. This explains the difference in the raster displays, and this difference is likely also the cause that we were initially not able to reproduce the original UE result.

In our reproduction of Figure 2 of the original paper we use preprocessed data available in 1 ms resolution, that likely experienced the `alexa2gdf` program for conversion as shown in Figure5 (the rightmost branch), before data are loaded into our reimple-

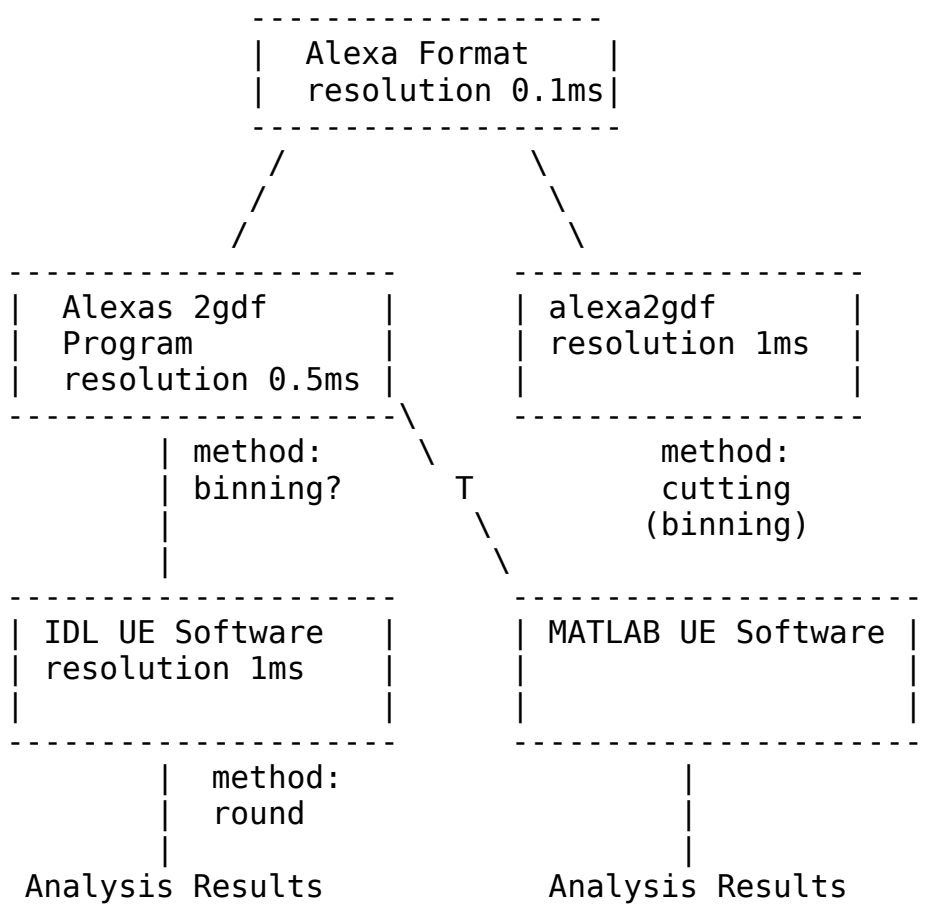

Figure 5: Illustration of the data preprocessing workflow, translated from the 1997 report of MD (in German) on the comparison of the first UE implementation in IDL (left branch) and the second implementation in Matlab [8] (right branch). Data entering both analysis branches have a resolution of 0.1 ms (top box). In the IDL branch spike times t in the original data are first transformed to a resolution of $h = 0.5$ ms (by the `2gdf` program, middle left) using the method of binning $\lfloor t/h \rfloor$. Then the data are read into the "IDL UE Software" (lower left box) and therein converted to $h = 1$ ms resolution by the method of "round half up" $\lfloor t/h + 1./2 \rfloor$ prior to analysis. Alternatively, one can load the 0.5 ms resolution data into the "MATLAB UE Software" (lower right box). Here the bin width is a parameter of the analysis and thus data can be converted to a 1 ms resolution but results are different from the IDL branch. Results are only identical if a prior transformation "T" (diagonal in center) performs a round half up and no further binning is done in the MATLAB program. At a later point in time the `alexa2gdf` converter function (written in MATLAB) became available such that data in the original 0.1 ms resolution could directly be converted to the 1 ms resolution by binning. The full report ("Report_by_MarkusDiesmann.txt") is included in the data folder of the repository for this paper.

mentation of the UE analysis. However, according to the aforementioned report by MD, we only have a chance to reproduce Figure 4A of [26] if we have the original data or a version of them with a time resolution lower than 1 ms available. The original raw data with 0.1 ms resolution are presumably only available on a storage medium and format that at present we are not able to read and interpret. However, after we contacted AR she found the data ("jenny201_345_preprocessed.gdf") of Figure 4A of [26] with a time resolution of 0.5 ms, which likely experienced the `2gdf` program for conversion (middle left box in Figure5).

We loaded this data at 0.5 ms resolution into Python and converted the data from the 0.5 ms to the required 1 ms resolution by the mathematical operation $\lfloor x + \frac{1}{2} \rfloor$, called "rounding half up". In numerical software packages, including IDL, this operation is typically implemented by a function named round(). However, the round() implementation of NumPy (version 1.11.0) performs an even rounding, i.e., values exactly halfway between two integers are rounded to the nearest even integer. Indeed, the latter implementation of rounding did not reproduce the result of the original publication. Thus, we used the expression `floor(x+0.5)` to perform rounding as it is implemented by IDL. The procedure completely reproduces panel A of Figure 4 in the original publication (see Figure6).

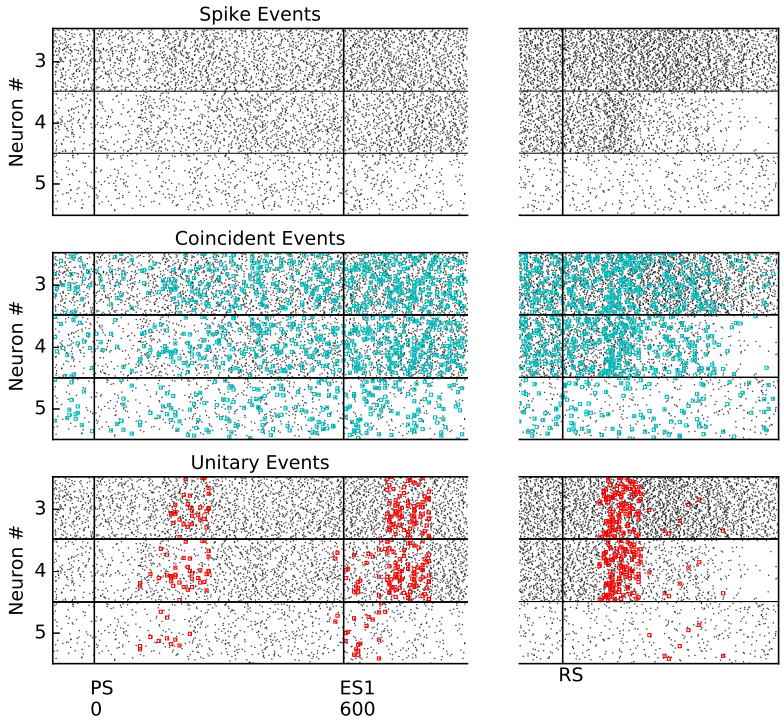

Figure 6: Reproduction of Figure 4A of the original publication. The left part of the figure shows the UE analysis result for the data aligned to ES1 (event code 15 in the GDF data file) with a time before the event (pre-time) set to 699 ms. The right part of the figure shows the analysis of the same data aligned to RS and with a pre-time of 99 ms. The left and right parts of the figure show 96 and 128 trials, respectively.

Conclusion

We are able to reproduce the original results of [26] by applying a new reimplementation of the Unitary Events analysis method in Python to the original data. The method involves a number of numerical computations and is very sensitive, as we show here by the comparison of Figures 1 and 2, which differed in the events that the trials were aligned to. This difference in the alignment would not affect the results if the time difference between the two events (PS and RS) were identical across the trials. But since the latter was not the case due to hardware features of the recording setup (as we learned from the first author of the original publication), the binning of the data started at a slightly different time points in different trials. This likely led to a loss or an addition of a spike in a bin and thus to a small difference of the number of spike synchrony events (see also the discussion on the issues of exclusive binning in [18]). In spite of this sensitivity of the method, we succeeded in generating results that are exactly identical to the original ones, as confirmed by the direct comparison of the positions of spikes and UEs in the original and the reproduced figures. This is a strong indication that our new implementation of the analysis faithfully implemented the UE method.

The event to which the data were aligned and the cut time which then defined the start of the (exclusive) binning was not documented in the original publication. Also the original scripts for the analysis are not available anymore, which could have revealed this information, even without having the original UE software code at hand. Thus, due to the lack of documentation we are only able to reproduce the results of [26] by communicating with some of the authors of the original publication.

The reproduction of Figure 4A of the original publication is a further and important test whether our reimplementation is also correct for N>2 neurons. This is relevant since the implementation requires a more generic, complex algorithm for the analysis than the one that can be used for only N=2. In the case of two neurons, there is only one pattern type which has to be analyzed (i.e. [1,1]). However, in the case of e.g. 3 neurons there are already 4 different spike patterns to analyze ([1,1,0], [0,1,1], [1,0,1], and [1,1,1]), and even much more for more neurons ($2^N - N - 1$). The statistics of each of the patterns is performed separately and, therefore, the bookkeeping needs to be carefully done.

The reproduction of Figure 4A is more complicated than reproducing Figure 2 due to additional reasons. First of all, the data were not available to us. After requesting them from the first author of the original publication we received data and were not able to reproduce the result - in terms of the UE results, but also the data seemed slightly different. After further consultation with the original authors we learned that the original Figure 4A was not generated by the Matlab implementation used for Figure 2 but by another implementation in IDL. Both are not available to us. However, we were told that the third author, MD, of the original publication performed a thorough comparison of the two implementations at the time and the final report on that investigation was made available to us. This enables us to define the correct workflow that reproduces the original result, given we have the data in the correct resolution at hand.

The reimplemented UE analysis software contains extensions for improving the statistics that were developed after the original publication. On the one hand, it contains the option to adjust the statistics to take into account cross-trial inhomogeneity by calculating the number of expected spike synchrony events based on the firing rates in a trial-by-trial fashion (option: *"analytic_TrialByTrial"*) as suggested in [15], in contrast to using trial averages of the firing rates. On the other hand, our reimplementation offers the possibility to calculate the significance of the empirical number of spike synchrony events based on a Monte Carlo approach (option: *"surrogate_TrialByTrial"*). Instead of computing the significance using a parametric distri-

bution based on the estimate of the firing rates, the null-hypothesis of independence is implemented by surrogate data [16, 17, 21]. By repeated intentional manipulation of the original data, potential spike synchrony is deleted. Each of these surrogate data created by this procedure are then searched - as the original data - for spike synchrony, and these numbers create the distribution underlying the significance test of the method. Obviously this version is considerably more computationally expensive than the parametric approach used here for the reproduction of [26] and we are currently working on an HPC implementation to make use of parallelization. The Python implementation of the UE method is publicly available in the open source software package Elephant at http://neuralensemble.org/elephant/.

If the authors of the original paper would not have been accessible, we would not have been able to reproduce the results. Nevertheless, here our final validation of the reproduction is based on the values extracted from the vector graphic image (PDF) in the original paper. In an optimal scenario, we would be able to exactly validate the results based on a numerical comparison. To do so, all of the following pieces of information would have had to be available at hand:

1. the original primary data
2. metadata describing the primary data in detail
3. the original statistics software package (e.g. Unitary Events)
4. the loading routine for the data
5. all specific code required to produce each figure of the original publication
6. detailed documentation of all code
7. the original software environment (with programs available in the original versions used), including, e.g. the interpreter/compiler (here: Matlab) and operating system
8. unique identifiers of the data records that unambiguously identify data from within the analysis code

In the analysis presented in this work, not even the original primary data (1), recorded more than 20 years ago, but only a slightly preprocessed version is available. However, even today many of the pieces of information listed above are often not made available by scientists. In part, this is due to the enormous complexity of the task to record all information in fine detail leading from the experiment to an analysis result. Moreover, there is still a lack of software tools to support researchers in the process of acquiring, storing, and organizing this information. Currently, there are emerging approaches suggested for metadata annotation (2) of electrophysiological data, such as the odML[4] framework (see, e.g. [11, 28]) for storing hierarchical collections of metadata or the NIX[5] data format [2] for linking data and metadata. In our concrete example, the information about the hardware limitations in storing the event times, would have been essential information contained in the metadata. Using modern tools for version control, points (3)-(6) can be easily addressed. There are emerging approaches to keep software environment, i.e., the original Matlab version and the operating system (7), e.g., by freezing the environment in a virtual machine. Point (8) is still challenging, because it requires the data to be addressed in an unambiguous manner from within the analysis scripts. Including data within the code repositories is typically prohibitive due to the size of the data. A solution would be to deposit data in public or private databases that allow data to be identified using a unique identifier

[4] https://github.com/G-Node/python-odml
[5] https://github.com/G-Node/nix/wiki

in combination with a tool to generate a detailed provenance track of the analysis process, but the implementation of tools and services for the workflows used in data analysis of electrophysiological data is still an ongoing endeavor [3, 5]. In summary, there are still components missing such that researchers are put into a position to build complex data acquisition and analysis workflows that enable optimal reproducibility in neuroscience.

Author Contribution

VR: Implementing the UE analysis in Python, Reproduction of the figures, Discussion of the results, Drafting of the manuscript, Revision of the draft. **JI**: Reproduction of the figures (with VR), Discussion of the results (with all other authors), Drafting of the manuscript (with VR), Revision of the draft (with all other authors). **MDe**: Drafting of the manuscript (with VR), Revision of the draft (with all other authors), Assisted in integrating the UE analysis in the Elephant library. **SG**: Conceived the original idea of reproduction, mediated the communications with original authors, provided suggestions for correcting errors in intermediate reproduction results, discussed the results, revised the manuscript (with all other authors).

Acknowledgements

This project received funding from EU Grant 720270 (HBP), Deutsche Forschungsgemeinschaft Grant DE 2175/2-1 and GR 1753/4-2 of the Priority Program (SPP 1665), the German-Japanese Computational Neuroscience Project (German Federal Ministry for Education and Research, BMBF Grant 01GQ1114), from the Helmholtz Portfolio Theme "Supercomputing and Modeling for the Human Brain", and from the Osaka Univ for the project 'Neural mechanism of active vision studied by combining large-scale sampling of neural activity and advanced computational analysis'.

We thank Alexa Riehle and Markus Diesmann for fruitful discussions.

References

[1] Moshe Abeles. *Local Cortical Circuits: An Electrophysiological Study*. Studies of Brain Function. Berlin, Heidelberg, New York: Springer-Verlag, 1982.

[2] Stoewer Adrian et al. "File format and library for neuroscience data and metadata". In: *Frontiers in Neuroinformatics* 8 (2014).

[3] Rosa Badia et al. *INCF Program on Standards for data sharing: new perspectives on workflows and data management for the analysis of electrophysiological data*. Techn. Report. 2015.

[4] Emery N. Brown, Robert E. Kaas, and Partha P. Mitra. "Multiple neural spike train data analysis: state-of-the-art and future challenges". In: *Nature Neuroscience* 7 (2004), pp. 456–461.

[5] Michael Denker and Sonja Grün. "Brain Inspired Computing". In: ed. by Katrin Amunts et al. Springer Series Lecture Notes in Computer Science, in press. Chap. Designing Workflows for the Reproducible Analysis of Electrophysiological Data.

[6] Michael Denker et al. "Estimating the contribution of assembly activity to cortical dynamics from spike and population measures". In: *Journal of Computational Neuroscience* 29 (2010), pp. 599–613.

[7] Michael Denker et al. "The Local Field Potential Reflects Surplus Spike Synchrony". In: *Cerebral Cortex* 21 (2011), pp. 2681–2695.

[8] Markus Diesmann. "Report on comparing the IDL and the Matlab implementations of the UE analysis". In: *personal communication* (2016).

[9] Samuel Garcia et al. "Neo: an object model for handling electrophysiology data in multiple formats". In: *Frontiers in Neuroinformatics* 8 (2014), p. 10.

[10] Franck Grammont and Alexa Riehle. "Precise spike synchronization in monkey motor cortex involved in preparation for movement". In: *Experimental Brain Research* 128 (1999), pp. 118–122.

[11] Jan Grewe, Thomas Wachtler, and Jan Benda. "A Bottom-up Approach to Data Annotation in Neurophysiology". In: *Frontiers in Neuroinformatics* 5 (2011).

[12] S. Grün. *Unitary Joint-Events in Multiple-Neuron Spiking Activity: Detection, Significance, and Interpretation*. Reihe Physik, Band 60. ISBN 3-8171-1506-7. Verlag Harri Deutsch, 1996.

[13] S. Grün, M. Diesmann, and A. Aertsen. "'Unitary Events' in Multiple Single-Neuron Spiking Activity. I. Detection and Significance". In: *Neural Computation* 14 (2002), pp. 43–80.

[14] S. Grün, M. Diesmann, and A. Aertsen. "'Unitary Events' in Multiple Single-Neuron Spiking Activity. II. Non-Stationary Data". In: *Neural Computation* 14 (2002), pp. 81–119.

[15] S. Grün, A. Riehle, and M. Diesmann. "Effect of cross-trial nonstationarity on joint-spike events". In: *Biological Cybernetics* 88 (2003), pp. 335–351.

[16] Sonja Grün. "Data-driven significance estimation of precise spike correlation." In: *Journal of Neurophysiology* 101 (2009), pp. 1126–1140.

[17] Sonja Grün, Markus Diesmann, and Ad Aertsen. "Analysis of Parallel Spike Trains". In: ed. by Sonja Grün and Stefan Rotter. Springer, 2010. Chap. Unitary Event Analysis.

[18] S. Grün et al. "Detecting unitary events without discretization of time". In: *Journal of Neuroscience Methods* 94 (1999), pp. 67–79.

[19] J Ito et al. "Saccade-related modulations of neuronal excitability support synchrony of visually elicited spikes". In: *Cereb Cortex* 21 (2011), pp. 2482–2497.

[20] B. E. Kilavik et al. "Long-term Modifications in Motor Cortical Dynamics induced by intensive practice". In: *Journal of Neuroscience* 29 (2009), pp. 12653–12663.

[21] S. Louis et al. "Surrogate spike train generation through dithering in operational time". In: *Front. Comput. Neurosci.* 4 (2010), p. 127.

[22] Pedro Maldonado et al. "Synchronization of Neuronal Responses in Primary Visual Cortex of Monkeys Viewing Natural Images". In: *Journal of Neurophysiology* 100 (2008), pp. 1523–1532.

[23] H. Nakahara and S. Amari. "Information-geometric measure for neural spikes". In: *Neural Computation* 14 (2002), pp. 2269–2316.

[24] Gordon Pipa, Sonja Grün, and Carl Van Vreeswijk. "Impact of Spike Train Autostructure on Probability Distribution of Joint Spike Events". In: *Neural Computation* 1163 (2013), pp. 1123–1163.

[25] Alexa Riehle et al. "Dynamical changes and temporal precision of synchronized spiking activity in monkey motor cortex during movement preparation". In: *Journal of Physiology Paris* 94 (2000), pp. 569–582.

[26] Alexa Riehle et al. "Spike Synchronization and Rate Modulation Differentially Involved in Motor Cortical Function". In: *Science* 278 (1997), pp. 1950–1953.

[27] Vahid Rostami. "Pull Request of the Unitary Events method in Elephant". In: *Elephant Github* (2016). URL: https://github.com/NeuralEnsemble/elephant/pull/64.

[28] Lyuba Zehl et al. "Handling Metadata in a Neurophysiology Laboratory". In: *Frontiers in Neuroinformatics* 10 (2016), p. 26.