SPIKY: A graphical user interface for monitoring spike train synchrony

Nebojsa Bozanic, Thomas Kreuz

Institute for complex systems, CNR, Sesto Fiorentino, Italy

Abstract

Techniques for recording neuronal spiking activity are developing so fast that there is an increasing demand for algorithms which provide the possibility to study the similarity patterns of larger and larger number of spike trains across different spatial scales and with a very high temporal resolution. In recent years, three such time-resolved measures of spike train synchrony have been proposed, ISI-distance, SPIKE-distance, and event synchronization. The Matlab source codes for calculating and visualizing these measures have been made publicly available. However, due to the many different possible representations of the results the use of these codes is not very intuitive and their application requires some basic knowledge of Matlab. Thus it became desirable to provide a more user-friendly and interactive interface. Here we address this need and present the graphical user interface SPIKY which facilitates the application of all three methods to both simulated and real data. SPIKY has been optimized with respect to computation speed and memory demand. It also includes complementary programs aimed at the collective analysis of large numbers of datasets and the estimation of significance levels.

1. Introduction

Spike train distances are measures that when applied to a set of spike trains yield low values for very similar spike trains and high values for very dissimilar spike trains. Essentially there are two major scenarios for using spike train distances to estimate the degree of synchrony between two or more spike trains.

The most common scenario is the simultaneous recording of a neuronal population, typically in some kind of spatial multi-channel setup. If different neurons emit spikes simultaneously, these spikes are 'synchronous' in the classical sense of the word, which is derived from Greek and describes events 'occurring at the same time' (Pikovsky et al., 2001). Synchronization between individual neurons has been proven to be of high prevalence in many different neuronal circuits. Examples include the visual cortex (Tiesinga et al., 2008) and the retina (Shlens et al., 2008) for which numbers of spike coincidences considerably above chance level have been reported. As of now many open questions remain regarding the spatial scale and the nature of interactions (e.g., pairwise or higher order, see (Nirenberg and Victor, 2007)) as well as their functional significance for neuronal coding and information processing (Kumar et al., 2010).

In the other major scenario the neuronal spiking response is recorded in different time intervals. In order to allow a meaningful confrontation there has to be a temporal reference point which is typically set by some kind of trigger (e.g., the onset of an external stimulation). There are two prominent applications for this successive trials scenario. The first one is to test for the reliability of individual neurons by presenting the same stimulus repeatedly (e.g., Mainen and Sejnowski (1995)). In contrast, when different stimuli are used one is mostly interested in the features of the response that provide the optimal discrimination since this might allow drawing conclusions regarding the nature of the neuronal coding (e.g., Victor (2005), for a more general introduction to neural coding cf. Quian Quiroga et al., 2002). These two applications are related since for a good clustering performance one needs not only a pronounced discrimination between stimuli (high inter-stimulus spike train distances) but also a high reliability for the same stimulus (low intra-stimulus spike train distances).

Electrophysiology and other modern recording techniques are developing fast and for both simultaneous population and successive trial recordings they often provide more data than available methods of spike train analysis can handle. There is an increasing demand for algorithms which provide the possibility to study the similarity patterns of many spike trains across different spatial scales. Another very desirable property of such analysis methods would be a high temporal resolution. Analyzing the vary-

---

Email addresses: `strance@gmail.com` (Nebojsa Bozanic), `thomas.kreuz@cnr.it` (Thomas Kreuz).

ing spiking patterns of simultaneously recorded ensembles of neurons in epilepsy patients could lead to a better understanding of the mechanisms of seizure generation, propagation, and termination (Truccolo et al., 2011; Bower et al., 2012) in the same way as the analysis of neuronal responses to successive presentations of time-dependent stimuli could help to understand the importance of synchronous firing in neural coding (Miller and Wilson, 2008). Moreover, in population recordings it would be even more advantageous to be able to monitor spike train synchrony in realtime. This would be a necessary condition for a prospective epileptic seizure prediction algorithm (Mormann et al., 2007), but it could also be very useful for the rapid online decoding needed to control prosthetics (Hochberg et al., 2006; Sanchez, 2008).

In recent years two such time-resolved measures have been proposed. The ISI-distance (Kreuz et al., 2007) and the SPIKE-distance (Kreuz et al., 2013) rely on instantaneous estimates of spike train dissimilarity which makes it possible to track changes in instantaneous clustering, i.e., time-localized patterns of (dis)similarity among multiple spike trains. Both measures are also parameter-free and time-scale independent. Furthermore, the SPIKE-distance also comes in a causal variant (Kreuz et al., 2013) which is defined such that the instantaneous values of dissimilarity are derived from past information only so that time-resolved spike train synchrony can be estimated in real-time. Finally, another time-scale independent and time-resolved method is event synchronization (Quian Quiroga et al., 2002), a sophisticated coincidence detector which quantifies the level of synchrony from the number of quasi-simultaneous appearances of spikes. Originally, it was proposed and used in a bivariate context only but here we will extend it to the multivariate framework.

For all of these measures there are several levels of information reduction (Kreuz, 2012). The starting point is the most detailed representation in which one instantaneous value is obtained for each pair of spike trains. This results in a matrix of size number of sampled time instants squared number of spike trains (i.e. $\#(t_n)N^2$). By selecting a pair of spike trains one obtains a bivariate dissimilarity profile whereas the selection of a time instant yields an instantaneous matrix of pairwise spike train dissimilarities which can be used to divide the spike trains into instantaneous clusters, i.e., groups of spike trains with low intra-group and high inter-group dissimilarity. Another way to reduce the information is averaging. The spatial average over spike train pairs yields a dissimilarity profile for the selected (sub)population, whereas temporal averaging leads to a bivariate distance matrix for the selected interval or the selected trigger points. Finally, application of the remaining average results in one distance value which describes the overall level of synchrony for a group of spike trains over a given time interval.

The Matlab source codes for calculating and visualizing these measures have been made publicly available (`http://wwwold.fi.isc.cnr.it/users/thomas.kreuz/sourcecode.html`) XXXXX and have already been widely used in various contexts (e.g., XXXXX Citations XXXXX). However, due to the many different possible representations of the results the use of these codes is not very intuitive and their application requires some basic knowledge of Matlab. Thus it became desirable to provide a more intuitive and interactive tool for analyzing spike train data. Here we address this need and present the graphical user interface SPIKY. Once given a set of real or simulated spike train data (importable from many different formats), SPIKY calculates the measures of choice and allows to switch between many different visualizations such as measure profiles, pairwise dissimilarity matrices, or hierarchical cluster trees. SPIKY also includes the possibility to generate movies which are very useful in order to track the varying patterns of (dis)similarity. Finally, SPIKY has been optimized with respect to both computation speed (by using MEX-files, i.e. C-based Matlab executables) and memory demand (by taking advantage of the piecewise linear nature of the dissimilarity profiles).

The remainder of this paper is organized as follows. In Section 2 we present the different measures available in SPIKY. These include the ISI-distance and the SPIKE-distance as well as the latter's realtime variant, and, introduced here, its future variant (Subsection 2.1). In Subsection 2.2 we present the improvements realized in this new implementation of the measures. Most prominent are the correction of the edge-effect (spurious decrease to zero) for the SPIKE-distance and the increase of memory efficiency by avoiding sampling. Recently, we added another complimentary measure to SPIKY, event synchronization. In Subsection 2.3 we show how it can be modified such that it can be used within the time-resolved SPIKY-framework. An overview of the different levels of information reduction is given in Section 3. These range from the most detailed representation in which one instantaneous value is obtained for each pair of spike trains to the most condensed representation in which successive temporal and spatial averaging leads to one single distance value which describes the overall level of synchrony for a group of spike trains over a given time interval.

SPIKY, our graphical user interface (GUI) for monitoring spike train synchrony, is presented in Section 4. In Subsection 4.1 explain how to get access to the codes and the large library of documentation (which includes text files, images, movies, as well as screen recordings with voice-over in which the user is guided step by step through some of the most important features of SPIKY). Subsequently, in Subsection 4.2 we introduce the structure and the workflow of the program. We show how to input spike train data, how to change the layout of the figures and how to export results. We also present two programs included in the SPIKY-package which are complementary to the graphical user interface itself. While the program 'SPIKY_loop' (Subsection 4.3) is meant to be used for the collective analysis of many different datasets (e.g. when the statistics of a certain quantity should be evaluated over all avail-

able datasets), the program 'SPIKY_loop_surro' (Subsection 4.4) was designed to evaluate the statistical significance of the results obtained for the original dataset by comparing them against the results obtained for spike train surrogates generated from that dataset. In Subsection 4.5 we investigate the performance of our new implementation of the time-resolved measures of spike train synchrony and compare it with previously published codes (e.g. Rusu and Florian (2014) XXXXX). Finally, in Section 5 we summarize both the methods and the program, discuss their limitations and present an outlook on future developments.

2. Measures

SPIKY contains implementations of four measures. The first one, included for comparison, is the standard multivariate Peri-Stimulus Time Histogram (PSTH), a measure of overall firing rate. In this Section we give an overview over the other three measures: First we review the ISI-distance and the SPIKE-distance (including the latter's realtime and future variants). We also discuss the improvements in the implementation of the algorithm. Finally, we show how event synchronization can be extended and modified such that we can make use of the multivariate and time-resolved framework of SPIKY.

In the following we show the definitions of the bivariate variants but for all of these measures there exists a straightforward extension to the case $N > 2$ (with N denoting the number of spike trains), the averaged bivariate distance. This average over all pairs of neurons commutes with the average over time, so it is possible to achieve the same kind of time-resolved visualization as in the bivariate case by first calculating the instantaneous average $S^{\mathrm{a}}(t)$ (here for the regular SPIKE-distance) over all pairwise instantaneous values $S^{mn}(t)$,

$$S^{\mathrm{a}}(t) = \frac{1}{N(N-1)/2} \sum_{n=1}^{N-1} \sum_{m=n+1}^{N} S^{mn}(t). \quad (1)$$

Also, all of these measures are time-resolved and rely on instantaneous values which taken together result in a dissimilarity profile (cf. Section 3.1). The overall measure values are defined as the temporal average of these dissimilarity profile, e.g for the regular SPIKE-distance:

$$D_S^{\mathrm{a}} = \frac{1}{T} \int_0^T dt S^{\mathrm{a}}(t). \quad (2)$$

In the following we denote with $\{t_i^{(n)}\} = t_1^{(n)}, ..., t_{M_n}^{(n)}$ the spike times and with M_n the number of spikes for neuron n with $n = 1, ..., N$.

2.1. The ISI- and the SPIKE-distance

Both the bivariate ISI- and the bivariate SPIKE-distance rely on instantaneous values in the sense that in a first step the two sequences of discrete spike times are transformed into dissimilarity profiles, $I(t)$ and $S(t)$. These dissimilarity profiles are based on three piecewise constant quantities which for each neuron $n = 1, 2$ are assigned to every time instant (see Fig. 1A). These are the time of the preceding spike

$$t_{\mathrm{P}}^{(n)}(t) = \max(t_i^{(n)}|t_i^{(n)} \leq t) \quad t_1^{(n)} \leq t \leq t_{M_n}^{(n)}, \quad (3)$$

the time of the following spike

$$t_{\mathrm{F}}^{(n)}(t) = \min(t_i^{(n)}|t_i^{(n)} > t) \quad t_1^{(n)} \leq t \leq t_{M_n}^{(n)}, \quad (4)$$

as well as the interspike interval

$$x_{\mathrm{ISI}}^{(n)}(t) = t_{\mathrm{F}}^{(n)}(t) - t_{\mathrm{P}}^{(n)}(t). \quad (5)$$

The ambiguity regarding the definition of the very first and the very last interspike interval is resolved by placing for each spike train auxiliary leading spikes at time $t = 0$ and auxiliary trailing spikes at time $t = T$ (but see also Section 2.2.1).

2.1.1. The ISI-distance

The ISI-distance, proposed as a bivariate measure in (Kreuz et al., 2007) and extended to the multi-neuron case in (Kreuz et al., 2009), was the first spike train distance directly defined as the temporal average of an instantaneous dissimilarity profile. This profile is calculated as the instantaneous ratio between the interspike intervals $x_{\mathrm{ISI}}^{(1)}$ and $x_{\mathrm{ISI}}^{(2)}$ (see Eq. 5 and Fig. 1) according to:

$$I(t) = \begin{cases} x_{\mathrm{ISI}}^{(1)}(t)/x_{\mathrm{ISI}}^{(2)}(t) - 1 & \text{if } x_{\mathrm{ISI}}^{(1)}(t) \leq x_{\mathrm{ISI}}^{(2)}(t) \\ -(x_{\mathrm{ISI}}^{(2)}(t)/x_{\mathrm{ISI}}^{(1)}(t) - 1) & \text{otherwise.} \end{cases} \quad (6)$$

Since the ISI-values only change at the times of spikes, the dissimilarity profile is piecewise constant (with potential discontinuities at the spikes). The ISI-ratio equals 0 for identical ISI in the two spike trains, and approaches -1 and 1, respectively, in intervals in which the first or the second spike train is much faster than the other. In order to treat both kinds of deviations equally, the temporal averaging is performed on the absolute value of the ISI-ratio:

$$D_I = \frac{1}{T} \int_0^T dt |I(t)|. \quad (7)$$

2.1.2. The SPIKE-distance

In contrast to the ISI-distance, the SPIKE-distance (see Kreuz et al. (2011) for the original proposal and Kreuz et al. (2013); Kreuz (2012) for the definite version presented here) considers the exact timing of the spikes since it is extracted from differences between the spike times of the two spike trains.

The dissimilarity profile is calculated in two steps: First for each spike the distance to the nearest spike in the other spike train is calculated, then for each time instant the relevant spike time differences are selected, weighted, and normalized. Here 'relevant' means local; each time instant is surrounded by four corner spikes: the preceding spike from the first spike train $t_{\mathrm{P}}^{(1)}$, the following spike from the
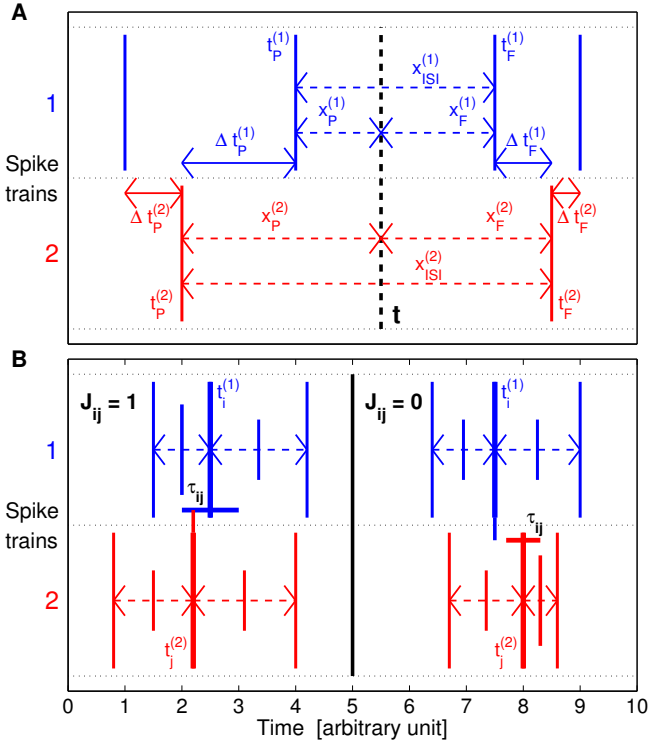
3

Fig. 1. A. ISI- and SPIKE-distance. Illustration of the local quantities needed to define the dissimilarity profiles $I(t)$ and $S(t)$ for an arbitrary time instant t. B. Event synchronization. Illustration of the first variant with adaptive coincidence window. While the middle spikes in the first half are coincident, the middle spikes in the second half are not.

first spike train $t_F^{(1)}$, the preceding spike from the second spike train $t_P^{(2)}$, and, finally, the following spike from the second spike train $t_F^{(2)}$. Each of these corner spikes can be identified with a spike time difference, for example, for the previous spike of the first spike train

$$\Delta t_P^{(1)}(t) = \min_i(|t_P^{(1)}(t) - t_i^{(2)}|) \qquad (8)$$

and analogously for $t_F^{(1)}$, $t_P^{(2)}$, and $t_F^{(2)}$ (see Fig. 1). For each spike train separately a locally weighted average is employed such that the differences for the closer spike dominate; the weighting factors depend on

$$x_P^{(n)}(t) = t - t_P^{(n)}(t) \qquad (9)$$

and

$$x_F^{(n)}(t) = t_F^{(n)}(t) - t, \qquad (10)$$

the intervals to the previous and the following spikes for each neuron $n = 1, 2$. The local weighting for the spike time differences of the first spike train reads

$$S_1(t) = \frac{\Delta t_P^{(1)}(t) x_F^{(1)}(t) + \Delta t_F^{(1)}(t) x_P^{(1)}(t)}{x_{ISI}^{(1)}(t)} \qquad (11)$$

and analogously $S_2(t)$ is obtained for the second spike train. Averaging over the two spike train contributions and normalizing by the mean interspike interval yields

$$S''(t) = \frac{S_1(t) + S_2(t)}{2\langle x_{ISI}^{(n)}(t)\rangle_n}. \qquad (12)$$

This quantity weights the spike time differences for each spike train according to the relative distance of the corner spike from the time instant under investigation. This way relative distances within each spike train are taken care of, while relative distances between spike trains are not. In order to get these ratios straight and to account for differences in firing rate, in a last step the two contributions from the two spike trains are locally weighted by their instantaneous interspike intervals. This leads to the definition of the dissimilarity profile

$$S(t) = \frac{S_1(t) x_{ISI}^{(2)}(t) + S_2(t) x_{ISI}^{(1)}(t)}{2\langle x_{ISI}^{(n)}(t)\rangle_n^2}. \qquad (13)$$

Since the dissimilarity profile $S(t)$ is obtained from a linear interpolation of piecewise constant quantities, it is piecewise linear (with potential discontinuities at the spikes). This dissimilarity profile and the SPIKE-distance D_S as its average are both bounded in the interval $[0, 1]$. The distance value $D_S = 0$ is obtained for identical spike trains only.

2.1.3. Realtime SPIKE-distance

The realtime SPIKE-distance D_{S_r} is a modification of the SPIKE-distance with the key difference that the corresponding time profile $S_r(t)$ can be calculated online because it relies on past information only [1]. From the perspective of an online measure, the information provided by the following spikes, both their position and the length of the interspike interval, is not yet available. Like the regular (improved) SPIKE-distance D_S, this causal variant is also based on local spike time differences but now only two corner spikes are available, and the spikes of comparison are restricted to past spikes, e.g., for the preceding spike of the first spike train

$$\Delta t_P^{(1)}(t) = \min_i(|t_P^{(1)}(t) - t_i^{(2)}|), t_i \leq t. \qquad (14)$$

Since there are no following spikes available, there is no local weighting, and since there is no interspike interval, the normalization is achieved by dividing the average corner spike difference by twice the average time interval to the preceding spikes (Eq. 9). This yields a causal indicator of local spike train dissimilarity:

$$S_r(t) = \frac{\Delta t_P^{(1)}(t) + \Delta t_P^{(2)}(t)}{4\langle x_P^{(n)}(t)\rangle_n}. \qquad (15)$$

2.1.4. Future SPIKE-distance

The future SPIKE-distance D_{S_f} can be used in triggered temporal averaging in order to evaluate the effect of certain spikes or of certain stimuli features on future spiking. It is the inverse measure to the realtime SPIKE-distance but instead of relying on past information only it relies on future information only. Again for each time instant there are just

---

[1] For the ISI-distance no such causal realtime extension is possible, since it relies on the instantaneous ISI-values and thus requires knowledge about the following spikes.

two corner spikes and the potential nearest spikes in the other spike train are future spikes only. Thus the spike time difference for the following spike of the first spike train reads

$$\Delta t_{\mathrm{F}}^{(1)}(t) = \min_i(|t_{\mathrm{F}}^{(1)}(t) - t_i^{(2)}|), t_i > t, \qquad (16)$$

and accordingly for the following spike of the second spike train. In analogy to Eq. 15, an indicator of local spike train dissimilarity is obtained as follows:

$$S_f(t) = \frac{\Delta t_{\mathrm{F}}^{(1)}(t) + \Delta t_{\mathrm{F}}^{(2)}(t)}{4\langle x_{\mathrm{F}}^{(n)}(t)\rangle_n}. \qquad (17)$$

2.2. Improvements in this implementation

2.2.1. Correction of the edge effect for the SPIKE-distance

In previous versions of the SPIKE-distance the ambiguity regarding the definition of the initial (final) distance to the preceding (following) spike as well as the very first and the very last interspike intervals was resolved by adding to each spike train an auxiliary leading spike at time $t = 0$ and an auxiliary trailing spike at time $t = T$. This lead to spurious synchrony at the edges where by construction the dissimilarity profile reached the zero value. Here we follow a suggestion by Conor Houghton (personal communication) and at least partly correct this edge effect. We describe the correction for the beginning of the recording, it is an analogous mirror image at the end of the recording.

We count the auxiliary spikes as normal spikes which can be nearest neighbor to other spikes. But instead of calculating their spike time distance (which is always zero) we use the spike time difference of the first real spike. For the first interspike interval we know that it is at least the distance to the first spike $t_1 - t_0 = t_1$ but it could be longer. So to take the local firing rate (or its inverse) into consideration we set

$$x_{\mathrm{ISI}}(0) = \max(t_1, t_2 - t_1). \qquad (18)$$

where we use the length of the first known interspike interval $t_2 - t_1$ as an upper limit of the inverse firing rate. This way we get at least a crude estimate of how much longer the first interspike interval could be.

2.2.2. Increase of memory efficiency by avoiding sampling

Earlier versions of the codes for calculating event synchronization and the ISI- and the SPIKE-distance relied on sampled dissimilarity profiles. Typically the precision was set to the sampling interval of the neuronal recording. Since the dissimilarity profile has to be calculated and stored for each pair of spike trains, this resulted, for each measure, in a matrix of order 'number of sampled time instants' × 'number of spike train pairs' (i.e., $\#(t_s) \times N(N-1)/2$). For small sampling intervals and a large number of recorded spike trains this lead to memory problems.

In SPIKY we use an optimized and more memory-efficient way of storing the data where we make use of the fact that the dissimilarity profiles $I(t)$ of the ISI-distance
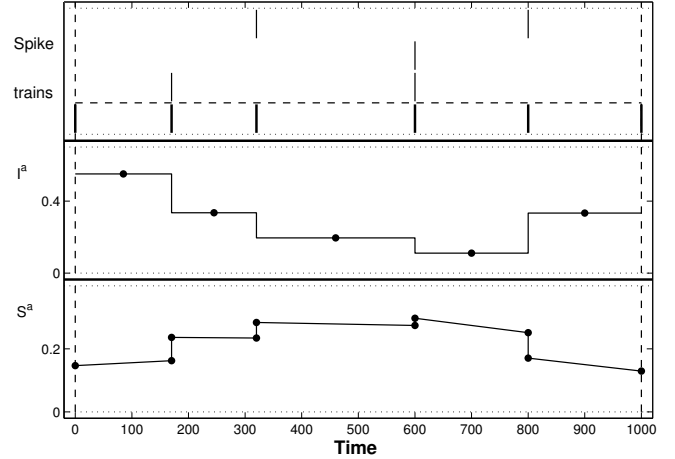

Fig. 2. Illustration of the memory efficient storage of the dissimilarity profile for both the ISI- and the SPIKE-distance.

is piecewise constant and the dissimilarity profile $S(t)$ of the SPIKE-distance is piecewise linear with each interval running from one spike of the pooled spike train to the next one. Thus for each such interval (and for each pair of spike trains) we have to store only one value for the ISI-distance and two values for the SPIKE-distance, one at the beginning and one at the end of the interval (see Figure 2). Typically the storage space required will be much smaller than for dissimilarity profiles sampled with a reasonable precision.

For the dissimilarity profiles of both the ISI- and the SPIKE-distance there are instantaneous jumps at the times of the spikes since this is where the lengths of the interspike intervals and the identity of the previous and the following spikes change abruptly. In contrast to the calculation based on sampling we get the exact result since each spike is both the previous and the next spike and there is no need anymore to 'cut the corners' of the dissimilarity profiles as had to be done for the sampled dissimilarity profiles. The dissimilarity profiles $S_r(t)$ and $S_f(t)$ of the real-time and the future SPIKE-distances are hyperbolic but for these measures the exact result can be obtained by piecewise integration over all intervals of the pooled spike train. While we use these integration schemes to calculate the exact values, for the time-resolved visualization we use linear interpolations within the individual intervals in order to reach the same low computational complexity that we obtain for the regular SPIKE-distance.

2.3. Event synchronization

Event synchronization (Quian Quiroga et al., 2002; cf. Kreuz et al., 2007 and Kreuz, 2011) works as a coincidence detector quantifying the level of synchrony from the number of quasi-simultaneous appearances of spikes. There are two variants. The main variant is parameter- and scale-free, since the time lag τ_{ij} up to which two spikes $t_i^{(1)}$ and $t_j^{(2)}$ are considered to be synchronous is adapted to the local spike rates (see Fig. 1B):

5

$$\tau_{ij} = \min\{t_{i+1}^{(1)} - t_i^{(1)}, t_i^{(1)} - t_{i-1}^{(1)}, t_{j+1}^{(2)} - t_j^{(2)}, t_j^{(2)} - t_{j-1}^{(2)}\}/2. \tag{19}$$

In contrast, the second variant uses a fixed coincidence window τ.

The function $c(1|2)$ counts the appearances of a spike in the first spike train shortly after a spike in the second spike train:

$$c(1|2) = \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} J_{ij}, \tag{20}$$

where

$$J_{ij} = \begin{cases} 1 & \text{if } 0 < t_i^{(1)} - t_j^{(2)} \leq \tau_{ij} \\ 1/2 & \text{if } t_i^{(1)} = t_j^{(2)} \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Here we used the definition of the first (scale-free) variant, for the second variant just replace τ_{ij} with τ. Then with $c(2|1)$ defined accordingly, event synchronization can be written as

$$Q = \frac{c(2|1) + c(1|2)}{\sqrt{M_1 M_2}}. \tag{22}$$

In order to get a time-resolved measure of event synchronization, we calculate the local derivative of the Q-function

$$E = \frac{Q(n) - Q(n - \Delta n)}{\sqrt{\Delta n_1 \Delta n_2}}. \tag{23}$$

where Δn_1 and Δn_2 are the numbers of events in the interval $[n - \Delta n, n]$. Unfortunately, E is not any longer parameter-free. The time resolution is governed by the window size Δn. For the second variant of event synchronization the fixed coincidence window τ constitutes a second parameter.

A suitably normalized distance is obtained by inversion $D_E = E - 1$, with $D_E = 0$ if and only if all spikes within the window coincide (or if there are no spikes at all).

Fig. 3 shows a comparison between the the Peri-Stimulus Time Histogram and event synchronization. It shows that the PSTH, in contrast to event synchronization, is not a measure of spike train synchrony (Kreuz et al., 2011) since it is invariant to shuffling spikes among the spike trains yielding the same value regardless of how spikes are distributed among the different spike trains. XXXXX Rep XXXXX

not yet finished XXXXX

3. Levels of information reduction

The ISI- and the SPIKE-distance combine a variety of properties that make them well suited for applications to real data. In particular, they are conceptually simple, computationally efficient, and easy to visualize in a time-resolved manner. By taking into account only the preceding and the following spike in each spike train, these distances rely on local information only. They are also time-scale-adaptive since the information used is not contained within a window of fixed size but rather within a time frame whose size depends on the local rate of each spike train.
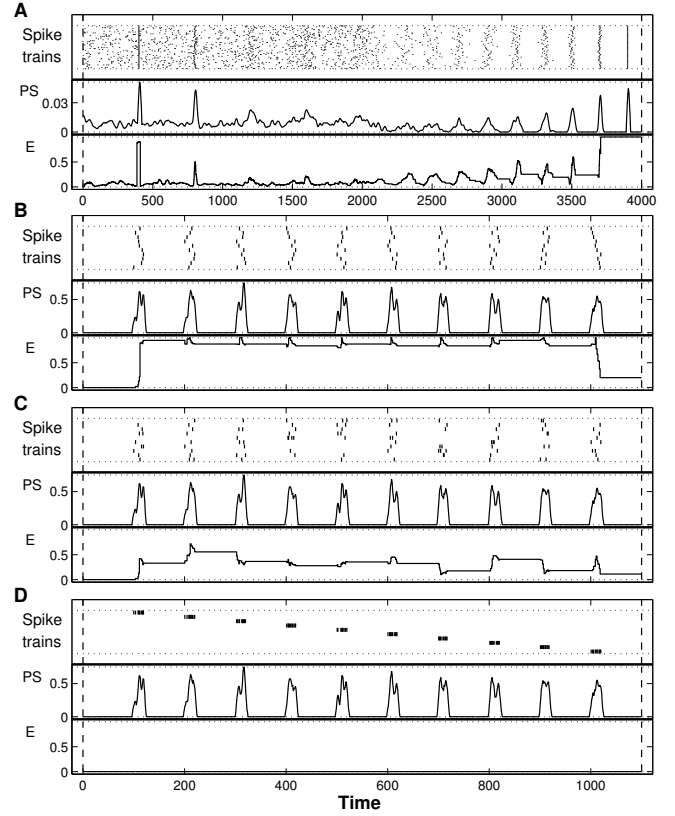

Fig. 3. Comparison of PSTH and event synchronization. (Quian Quiroga et al., 2002) XXXXX

Moreover, the sensitivity to spike timing and the instantaneous reliability achieved by the SPIKE-distance opens up many new possibilities in multi-neuron spike train analysis (Kreuz et al., 2013). These build upon the fact that there are several levels of information reduction all of which we describe in the following. As illustration we use the detailed analysis of an artificially generated spike train dataset (see upper subplot of Fig. 4A for the rasterplot).

3.1. Full matrix and cross sections

The starting point is the most detailed representation in which one instantaneous value is obtained for each pair of spike trains (see Eq. 13). This representation could be viewed as a movie of a symmetric pairwise dissimilarity matrix in which each frame corresponds to one time instant (an example can be found in the supplementary material of (Kreuz et al., 2013)). For a movie of finite length the time axis necessarily has to be sampled but in principle this most detailed representation is continuous and consists of an infinite number of values. However, since all dissimilarity profiles are piecewise linear (with potential discontinuities only at the times of the spikes) there is a lot of redundancy. Using the most compact and memory-efficient representation one can store all pairwise dissimilarity profiles in a matrix of size 'number of interspike intervals in the pooled spike train' × 'number of spike train pairs' (×2 for the SPIKE-distance, see Section 2.2.2).
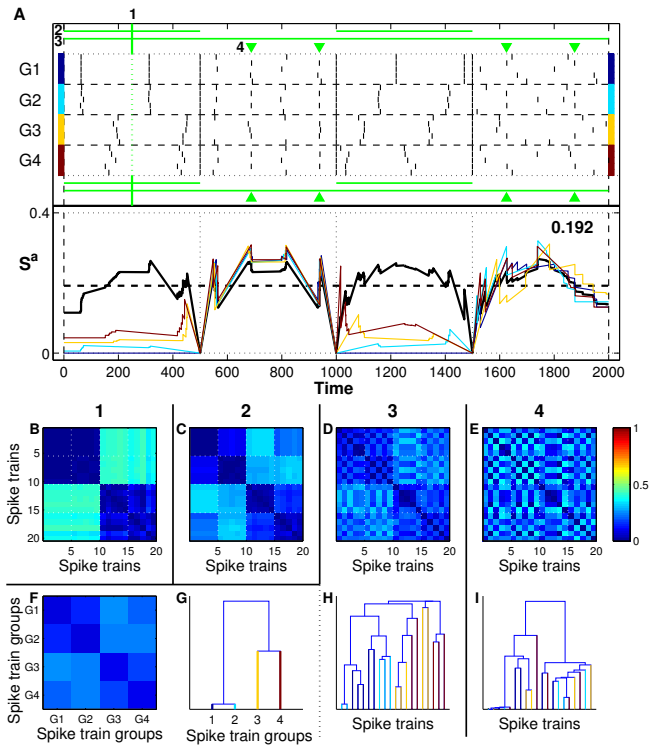
Fig. 4. The different levels of information reduction for the SPIKE-distance. A. Top: Spike rasterplot of 20 artificially generated spike trains divided in 4 spike train groups of 5 spike trains each. The clustering behavior changes every 500 ms. Bottom: Dissimilarity profiles of the SPIKE-distance for the four spike train groups (thin color-coded lines) and for all spike trains (thick black line). The overall dissimilarity is defined as the temporal average of the dissimilarity profile of all spike trains (0.192 in this case) and is marked by a black horizontal line. B-E. Matrices of pairwise instantaneous dissimilarity values for a single time instant, for two selective averages and for a triggered average. F. Matrices of overall pairwise instantaneous dissimilarity values for the 4 spike train groups. G. Dendrogram of spike train group matrix in F. H-I. Dendrogram of spike train matrices in D and E. Note that the triggered averages in contrast to the overall average captures the local similarity between 5 of the spike trains.

From this matrix, it is possible to extract any desired information. By selecting a pair of spike trains, one obtains the bivariate dissimilarity profile $S(t)$ for this pair of spike trains. Selecting a time instant t_s (and using linear interpolation for time instants in between spikes) yields an instantaneous matrix of pairwise spike train dissimilarities $S_{mn}(t_s)$ (see Fig. 4B). This matrix can be used to divide the spike trains into instantaneous clusters, that is, groups of spike trains with low intra-group and high inter-group dissimilarity.

3.2. Spatial and temporal averaging

Another way to reduce the information of the dissimilarity matrix is averaging. There are two possibilities that commute: the spatial average over spike train pairs and the temporal average. Since the spatial average over spike train pairs can be done locally it yields a dissimilarity profile for the whole population. Examples for averages over four dif-

ferent spike train groups as well as over all spike trains are shown in the lower subplot of Fig. 4A. Temporal averaging over certain intervals on the other hand leads to a bivariate distance matrix (see Fig. 4C and D for examples of non-continuous and continuous intervals). In real data, these temporal intervals could be chosen to correspond to different external conditions such as normal vs. pathological, asleep vs. awake, target vs. non-target stimulus, or presence/absence of a certain channel blocker.

A combination of temporal and spatial averaging can be seen in Fig. 4F. This dissimilarity matrix is obtained from the overall temporal average shown in Fig. 4D by (spatially) averaging over the 16 submatrices and thus depicts the pairwise spike train group dissimilarity (4×4 instead of 20×20). Fig. 4G shows the respective dendrogram. In applications to real data, these groups could be different neuronal populations or responses to different stimuli, depending on whether the spike trains were recorded simultaneously or successively. Finally, successive application of the spatial average over all spike train pairs and the temporal average over the whole interval results in just one single distance value that describes the overall level of synchrony for the whole dataset. In Fig. 4A this value is stated in the upper right of the lower subplot.

3.3. Triggered averaging

The fact that there are no limits to the temporal resolution allows further analyses such as internally or externally triggered temporal averaging. Here, the matrices are averaged over certain trigger time instants only. The idea is to check whether this triggered temporal average is significantly different from the global average since this would indicate that something peculiar is happening at these trigger instants. The trigger times can either be obtained from external influences (such as the occurrence of certain features in a stimulus) or from internal conditions (such as the spike times of a certain spike train). External triggering is a standard tool to address questions of neural coding, for example, it can be used to evaluate the influence of localized stimulus features on the reliability of neurons under repeated stimulation. In multi-neuron data, internal triggering might help to uncover the connectivity in neural networks or to detect converging or diverging patterns of firing propagation. An example is shown in Fig. 4E. Here, neurons 2, 9, 14, and 19 follow the 7-th neuron during the 2nd and the 4th 500-ms subinterval. This can be revealed by triggering on the spike times of neuron 7 during these subintervals. The difference between the triggered dissimilarity matrix of 4E and the dissimilarity matrix of the full interval in Fig. 4E can be seen best by comparing the respective dendrograms in Fig. 4H and I.

4. SPIKY

SPIKY is a graphical user interface for monitoring synchrony between artificially simulated or experimentally recorded neuronal spike trains. It contains implementations of the ISI- and the SPIKE-distance (including simulations of its realtime and future variants) as well as event synchronization which allow interactive access to all the different representations described in Section 3. All source codes are written in Matlab (MathWorks Inc, Natick, MA, USA) with the most time-consuming loops coded in MEX-files [2]. SPIKY is not stand-alone but requires Matlab to run.

4.1. *Access to SPIKY and how to get started*

SPIKY is distributed under a BSD licence (Copyright (c) 2014, Thomas Kreuz, Nebojsa Bozanic. All rights reserved.). A zip-package containing all the necessary files can be accessed for free on the download page (`http://www.fi.isc.cnr.it/users/thomas.kreuz/Source-Code/SPIKY.html`). This package also contains a folder with lots of documentation (such as a FAQ-file and an introduction to all individual elements and all individual files of SPIKY). Further information and many demonstrations (both images and movies) can be found on the download page and on the SPIKY Facebook-page (`https://www.facebook.com/SPIKYgui`). Both of these pages are used to announce updates and distribute the latest information about new features. They also provide the user with an opportunity to provide feedback and ask questions.

The Facebook-page includes various screen recordings with voice-over in which the user is guided step by step through some of the most important features of SPIKY. All of these movies can also be viewed on the SPIKY Youtube-channel (`https://www.youtube.com/channel/UCgSzOYQ5lWdVFO_Z1FNNOBw`).

After having downloaded SPIKY from the download page the user has to first extract the zip-package which leaves all files in one folder named 'SPIKY'. If the system has a suitable MEX-compiler installed, the MEX-files can be compiled from within this folder by running the m-file 'SPIKY_compile_MEX'. The last step is to run the m-file 'SPIKY'.

When SPIKY is running, the user can find quick information about the individual elements of the graphical user interface by activating the 'Hints'-checkbox in the 'Options'-Menu. Hovering with the mouse cursor above the elements of interest will then show short hints. An overview of all the information contained in the hints can be found in the documentation file 'SPIKY-Elements.doc'. Further-

---

[2] In our case these are subroutines written in C. However, as some users may not have access to a suitable C compiler, SPIKY contains the (slower) pure Matlab code as well.
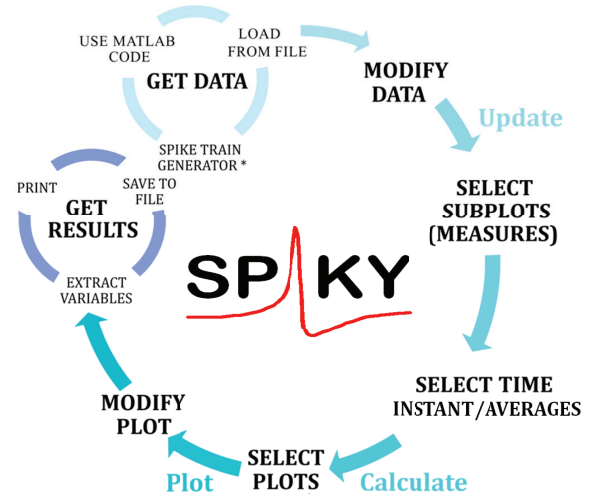

Fig. 5. Flowchart describing the workflow of SPIKY from the input of spike train data to the output of results. A typical SPIKY-session begins at the top with 'Get data', then goes clockwise and ends on the left with 'Get results'. In the center of the circle the SPIKY-logo is depicted.

more, at each step the suggested element for the next user action is highlighted by a bold font.

To get the user started quickly, SPIKY provides a few (artificial) example datasets from previous publications. The most useful example is the entry 'Clustering' in the 'Selection: Data'-listbox. This dataset has already been used in several figures as well as in the supplementary movie of Kreuz et al. (2013). A good introduction to SPIKY would be to follow this example through till the end advancing from panel to panel by pressing the highlighted button. In a second step one could reset and run the same example again while changing some parameters in order to see the consequences. Note that it is not necessary to set all the parameters each time when SPIKY is started. Rather it is possible to use the file 'SPIKY_f_user_interface' to set and modify the spike train data as well as the parameters (again refer to the dataset 'Clustering' for an example).

4.2. *Structure and workflow of SPIKY*

Overall, SPIKY has a rather linear workflow, however, it is much more interactive than previous implementations of the measures and there are many potential shortcuts and loops along the way. As you can see in the SPIKY-flowchart in Fig. 5, the general flow is clearly directed from the input of spike train data to the output of results. So the first step the user has to do is to give SPIKY spike train data (i.e. sequences of spike times) to work with. There are three possibilities to do so: one can make use of predefined examples, load data from a file, or employ the spike train generator (see Section 4.2.1 for more details).
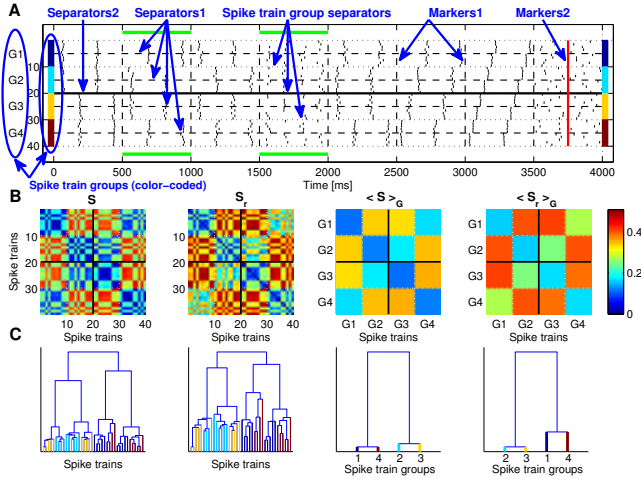
Fig. 6. Annotated screenshot from a movie. A. Artificially generated spike trains. B. Dissimilarity matrices obtained by averaging over two separate time intervals for both the regular and the real-time SPIKE-distance as well as their averages over subgroups of spike trains (denoted by $< \cdot >_G$). C. Corresponding dendrograms.

Once the full dataset is there, modification is still possible. One can restrict the analysis to a specific subset, e.g., select a smaller time window and/or a subset of spike trains. It is also possible to impose some external structure on the rasterplot (spike trains vs time). SPIKY allows the definition of two types of spike train separators (e.g. neurons from the left vs. neurons from the right hemisphere) and two types of time markers (e.g. specific events such as seizure onset and offset in epilepsy, trigger onset during stimulation etc.). It also enables the user to define spike train groups. Depending on the setup these could be spike trains recorded in different brain regions or upon presentation of different kinds of stimuli. Fig. 6 shows an example of a raster plot with annotations marking all these different elements.

After updating all of these data parameters the next step is to select the measures to be calculated. In addition to the measures described in the Appendix 2 we also include the Peri-Stimulus Time Histogram as a time-resolved standard approach widely used in spike train analysis. However, note that the PSTH is a complementary approach which describes overall firing rate and (to some precision) timing. It is not a measure of spike train synchrony (Kreuz et al., 2011) since it is invariant to shuffling spikes among the spike trains yielding the same value regardless of how spikes are distributed among the different spike trains (see Fig. 3). At the same time the user can select successive frames for a temporal analysis of spike train patterns. These can be individual time instants for cross sections, temporal intervals for selective averages and sequences of time instants for triggered averages (see Section 3).

Now the actual calculation of the measures takes place. For reasonably sized datasets this should take at most a few seconds. Larger datasets are divided in smaller subintervals and the calculation will be performed in a loop which might take longer (cf. Section 5.2). It is at this point that SPIKY

becomes truly interactive. Now the user can switch between different representations of the results (such as dissimilarity profiles, dissimilarity matrices and dendrograms). Regarding matrices and dendrograms one can decide whether one wants to compare different representations or look at them in sequence.

The presentation can be restricted to smaller time windows and/or subsets of spike train, and temporal and spatial averaging (for example moving average and average over spike train groups) can be performed. It is possible to add further figure elements such as spike number histograms, overall averages, or dissimilarity profiles for individual spike train groups. At this stage the user can also retrospectively change the appearance of all the individual elements of the figure (see Section 4.2.2 for more details). Finally, SPIKY allows to extract both data and results to the Matlab workspace for further analysis, and it is also possible to save individual figures as postscript-file or a sequence of images as an 'avi'-movie (for more details see Section 4.2.3).

4.2.1. Input

There are three different possibilities to input spike train data into SPIKY.

The first option is to select one of the predefined examples which are generated using Matlab-code. Initially these are the examples used in (Kreuz et al., 2013) but one can also define new examples.

The second option is to load spike train data from a file. Two different file formats are allowed, '.mat' and '.txt' (ASCII) files. For the mat-files SPIKY currently allows three different kinds of input formats (further formats can be added on demand).

– cell arrays (ca) with just the spike times. This is the preferred format used by SPIKY since it is most memory efficient. The two other formats will internally be converted into this format.
– regular matrices with each row being a spike train and zero padding (zp) in case the spike numbers are different.
– matrices representing time bins where each zero/one (01) indicates the absence/presence of a spike

If case of a mat-file SPIKY looks for a variable called 'spikes', if it cannot find it you have the chance to select the variable name (or field name) which contains the spikes via an input mask which provides a hierarchical structure tree of the variable structure. In the text format spike times should be written as a matrix with each row being one spike train. The SPIKY-package contains one example file for all four formats ('testdata_ca.mat', 'testdata_zp.mat', 'testdata_01.mat' and 'testdata.txt').

The third option is to create new spike train data via the spike train generator. After setting some defining variables (number of spike trains, start and end time, sampling interval) you can build your spike trains by using predefined spike train patterns (such as periodic, splay, uniform or Poisson) and/or by manually adding, shifting and deleting

individual spikes or groups of spikes.

4.2.2. Figure-Layout

SPIKY was designed in a way that allows to directly generate figures suitable for publication. To this aim the user is given control over the appearance of every individual element (e.g. fonts, lines etc.) in each type of figure. There are two ways to determine essential properties such as color, font size or line width. Most conveniently, one can use the file 'SPIKY_f_user_interface' to define the standard values for all the parameters that describe the principal layout of the figure. But it is also possible to change elements in the active figure while the program is already running. To do so the user has to simply click the right mouse button on the element to be changed. A context menu will appear which lets the user either edit either the properties of individual elements or of all elements of a certain type. This also includes the string property of any font (title, x- and y-labels etc.).

If a figure contains more than one subplot (besides the combined subplot containing the spike rasterplot and dissimilarity profiles these are typically subplots with dissimilarity matrices and dendrograms), it is also possible to change their position and size. To do so just move the cursor to the respective axis (either just left or just below the subplot) and click the right mouse button. Now one can edit all position variables by hand or change the x-position, the y-position, the width and the height individually. In case there are several dissimilarity matrices / dendrograms one can do this either for an individual matrix / dendrogram or for all of them at the same time.

4.2.3. Output

From within SPIKY it is possible to extract the spike trains and the results of the analyses (measure profiles, matrices, dendrograms) to the Matlab workspace for further processing. When one clicks the right mouse button on the element whose data one wishes to extract results will be stored in variables such as 'SPIKY_spikes', 'SPIKY_profile_X_1', 'SPIKY_profile_Y_1', 'SPIKY_profile_name_1' as well as 'SPIKY_matrix_1' and 'SPIKY_matrix_name_1'. In addition, the results obtained during an analysis will automatically be stored in the output structure 'SPIKY_results' which will have one field for each measure selected. Depending on the parameter selection within SPIKY, for each measure the structure can contains the following subfields which largely correspond to the different representations identified in Section 3:

– SPIKY_results.<Measure>.name: Name of selected measures (helps to identify the order within all other variables)
– SPIKY_results.<Measure>.distance: Level of dissimilarity over all spike trains and the whole interval. This is just one value, obtained by averaging over both spike trains and time

– SPIKY_results.<Measure>.matrix: Pairwise distance matrices, obtained by averaging over time
– SPIKY_results.<Measure>.x: Time-values of overall dissimilarity profile
– SPIKY_results.<Measure>.y: Overall dissimilarity profile obtained by averaging over spike train pairs

Note that the dissimilarity profiles are not equidistantly sampled. Rather they are stored as memory-efficiently as possible which means just one value for each interval of the pooled spike train for the ISI- and two values for the SPIKE-distance. Since this format can be more difficult to process, the functions 'SPIKY_f_selective_averaging', 'SPIKY_f_triggered_averaging', and 'SPIKY_f_average_pi' are provided in order to compute the selective average over time intervals, the triggered over time instants, or the average over many dissimilarity profiles, respectively. Furthermore, for the ISI-distance the function 'SPIKY_f_pico.m' can be used to obtain the average value as well as the x- and y-vectors for plotting.

Besides the standard way to work with Matlab-figures SPIKY also offers the opportunity to save each figure as a postscript-file. Finally, it is possible to save a sequence of images as an 'avi'-movie.

4.3. GUI vs. loop

SPIKY was mainly designed to facilitate the detailed analysis of one dataset. It enables the user to switch between different representations (see Section 3) and to zoom in on both spatial and temporal features of interest. However, SPIKY is not very suitable for the collective analysis of many different datasets when e.g. the statistics of a certain quantity such as an average over certain time intervals should be evaluated over all available datasets in some kind of loop. For these purposes the SPIKY-package contains a program called 'SPIKY_loop' which is complementary to SPIKY. It is not a graphical user interface but it should be simple enough (and plenty of examples are provided) to allow everyone to run the same kind of analysis for many different datasets and to evaluate and compare their 'SPIKY_results'. 'SPIKY_loop' uses the full functionality of SPIKY such as access to time instants, selective and triggered averages as well as averages over spike train groups.

So combining these two programs it is possible to first use SPIKY for a rather exploratory but detailed analysis of a limited number of individual datasets and then use SPIKY_loop and its output structure 'SPIKY_loop_results' to verify whether any effect discovered on the example dataset is consistently present within all of the datasets.

4.4. Spike train surrogates and significance

An important question that has not yet been asked is the one of statistical significance. Given a certain value of the SPIKE-distance how can one judge whether it reflects

a significant decrease or increase in spike train synchrony and does not just lie within the range of values obtained for random fluctuations. One way to address this question is the use of spike train surrogates (Kass et al., 2005; Gruen, 2009; Louis et al., 2010). The idea is to compare the results obtained for the original dataset versus the results obtained for spike train surrogates generated from that dataset. If the value obtained for the original lies outside the range of values for the surrogates this value can be assumed to be significant to a level defined by the number of surrogates used (e.g. $\alpha = 0.05$ for 19 surrogates or $\alpha = 0.001$ for 999 surrogates).

The SPIKY-package contains a program 'SPIKY_loop_surro' which was designed to look at significance. So far it includes four different types of spike train surrogates. They differ in the properties that are preserved and maintain either the individual spike numbers (obtained by shuffling the spikes), the individual interspike interval distribution (obtained by shuffling the interspike intervals), the pooled spike train (obtained by shuffling spikes among the spike trains) or the distribution using a general method for resampling from a given distribution using *the inverse transformation method* which is based on having a uniform (0, 1) random variable U, where for any distribution function F if there is a defined random variable X such that $X = F^{-1}(U)$ then the random variable X has a distribution function F. [$F-'(u)$ is defined to equal that value x for which $F(x) = u$.]

4.5. *Comparison with other implementations*

Python-Implementation of SPIKE-distance courtesy of Jeremy Fix:
`http://jeremy.fix.free.fr/Softwares/spike.html`
Python-Implementation of the pairwise ISI-distance courtesy of Michael Chary:
`https://pypi.python.org/pypi/ISIpy/1.0.1`
Răzvan Florian:
`https://github.com/modulus-metric/spike-train-metrics`
(Rusu and Florian, 2014)
HRLAnalysis^{TM} (Thibeault et al., 2014)
While all of these previous implementations are restricted to the overall dissimilarity profile and its temporal average (the overall dissimilarity), SPIKY also allows the user to interactively access all of the many different levels of information reduction introduced in Section 3.
XXXXX
Matlab provides a powerful toolbox for processing vectorized data because of the high level parallelization, while its GUI has engaging visualization capabilities. Some possible problems in using Matlab for processing spike train data could relate to the fact that interpreted language are slow. Addressing this issue, we converted the most time consuming functions to C. Under Matlab (2007b), we observed a significant speed up of functions written in C with respect to the ones written in Matlab.
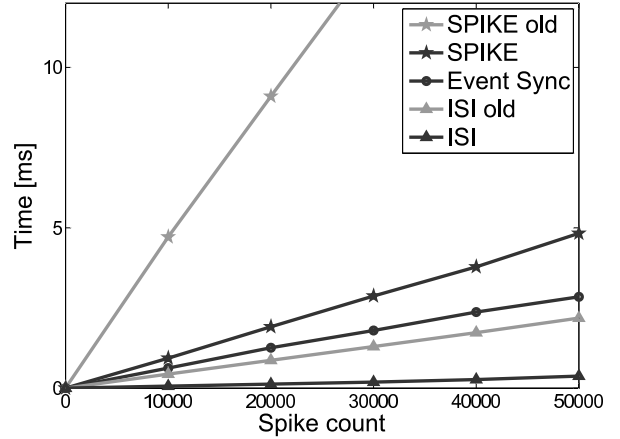C functions do not only lead to an increase in perfor-

Fig. 7. Performance comparison: Time needed for computing various distances, as a function of the number of spikes in the spike trains. For the ISI- and the SPIKE-distance we show the curves for both the old (sampling-based) and the new algorithm.

mance, they can also easily be incorporated into other programming platforms (such as Python).

For the sake of an unbiased comparison all algorithms have been implemented in C++, Visual C++ 12.0, and the complete source codes are available on http://wwwold.fi.isc.cnr.it/users/thomas.kreuz/ XXXXX to upload XXXXX. The following results have been achieved on an Intel Core i7-4700MQ CPU @ 2.4 GHz.

In order to show how different implementation of the same method can have different complexity classes, previous implementations for ISI-distance, SPIKE-distance and modulus metric are taken and compared versus novel ones (Figure 7).

The setup for the comparison was taken from Rusu and Florian (2014). Algorithms were tested using simulated spike trains. As spike train input we used random processes with different numbers of spikes. Figure 7 shows average time needed to compute distances as a function of number of spikes. The results were averaged over 10,000 trials. The simulations confirmed that the new implementations of both ISI- and SPIKE-distance have linear dependence of the number of spikes (see Section 2.2.2), however, the main improvement is not in the performance, but in the fact that the algorithms now compute the exact distance values without any dependence on the integration timestep.

5. Discussion

So far the SPIKE-distance has been applied in the following papers:
(Papoutsi et al., 2013; Di Poppa and Gutkin, 2013; Rusu and Florian, 2014; Sacré and Sepulchre, 2014)
XXXXX Update at the end XXXXX

5.1. *Summary*

Apart from the standard Peri-Stimulus Time Histogram SPIKY contains three parameter-free and time-resolved measures. These measures are complementary to each other since each one addresses a different specific aspect of spike train similarity. While the ISI-distance quantifies local dissimilarities based on covariances of the neurons firing rate profiles, both the SPIKE-distance and event synchronization capture the relative timing of spikes. However, whereas the SPIKE-distance weights and normalizes the differences between nearest neighbor spikes, event synchronization acts as a binary coincidence detector, i.e. there is a cutoff at the (adaptive or fixed) time lag relative to which two neighboring spikes are either considered coincident or not and all detailed information both within or outside this coincidence window is discarded.

freedom

5.2. *Limitations*

The calculation of the SPIKE-distance consists of three steps: First for each spike the distance to the nearest spike in all the other spike trains is calculated. Successively, for each time instant and each pair of spike trains, the distances of the four corner spikes are first locally weighted and then normalized. These latter steps involve matrices of the order 'number of interspike intervals in the pooled spike train' × 'number of spike train pairs', which for very long datasets with many spike trains can lead to memory problems. The solution to this problem is to make the calculation sequential, i.e., to cut the recording interval into smaller segments, and to perform the averaging over all pairs of spike trains for each segment separately. In the end the dissimilarity profiles for the different segments (already averaged over pairs of spike trains) are concatenated, and its temporal average yields the distance value for the whole recording interval. During this sequential calculation SPIKY uses a waitbar (which displays the % completed) to continuously inform the user about the progress.

One should be aware that all of these measures are based on some explicit notion of localness and simultaneousness and thus they all can be fooled by phase lags, no matter whether these are caused by internal delay loops in one spike train or by a common driver that forces the two time series with different delays. Therefore, any such phase lags should be removed by suitably shifting the time series in a preprocessing step.

5.3. *Outlook*

This manuscript is concerned with a new program primarily designed to analyze neuronal spike trains. But in principle it is also applicable to any other kind of discrete data which comes in the form of sequences of time stamps (such as times of bouncing basketballs and coded parent and child acts during children's tantrums just to mention two examples which we already dealt with). Furthermore, in Kreuz et al. (2013) we already applied the SPIKE-distance not only to discrete data but also to an example of continuous data (modified EEG). We plan to include this possibility in SPIKY; in principle the only preprocessing step missing is the transformation of the continuous data to discrete data via some kind of spike / event detection.

One of the measures included in SPIKY is the realtime SPIKE-distance. The present algorithm calculates the realtime values by making use of past information only but it does so in a speed-optimized and parallelized manner which would not be compatible with an actual realtime-implementation. However, such a realtime-implementation should actually be simpler and less problematic even for large numbers of neurons since the only information that would have to be stored at each time instant are the time stamps of the latest spikes of each spike train and their nearest neighbors in the other spike trains.

We chose to write SPIKY in Matlab because of its ease of use, popularity in the neuroscience community, and the well-developed MEX-interface for integrating C functions for performance enhancements. However, in the future we plan to export at least some of its functionality into an open-source environment which could for example be based on Python code within a C++ core.

Another potential direction would be the extension of the methods used here from the quantification of synchrony within one population of two or more neurons to the quantification of synchrony between two (or more) neuronal populations. Similar extensions have been done for two other spike train distances (Aronov, 2003; Houghton and Sen, 2008) but both of these methods are not time-resolved and depend on not one but two parameters. So one particular challenge for our methods could be to try to maintain the high temporal resolution of the original methods and to make the new methods parameter-free.

Note that there are many open questions regarding neuronal synchrony. Among these questions are its role in dynamical diseases like epilepsy and its relevance for neural coding. While a thorough discussion of these issues is beyond the scope of this paper, we argue that SPIKY should be a very useful tool for investigation. If epileptic seizures and/or time-dependent stimulations lead to changes in spike train synchrony or spike train clustering, SPIKY should be able to detect them.

References

Aronov D, Reich DS, Mechler F, Victor JD. Neural coding of spatial phase in V1 of the macaque monkey. J Neurophysiol, 2003;89:3304-27.

Bower MR, Stead M, Meyer FB, Marsh WR, Worrell GA. Spatiotemporal neuronal correlates of seizure generation in focal epilepsy. Epilepsia, 2012;53:807-16.

DiPoppa M, Gutkin BS. Correlations in background activity control persistent state stability and allow execution of working memory tasks. Front Comp Neurosci, 2013;7:139.

Grün S. Data-Driven Significance Estimation for Precise Spike Correlation. J Neurophysiol, 2009;101:1126-40.

Hochberg LR, Serruya MD, Friehs GM, Mukand JA, Saleh M, Caplan AH, Branner A, Chen D, Penn RD, Donoghue JP. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. Nature, 2006;442:164-71.

Houghton C, Sen K. A new multineuron spike train metric. Neural Comput, 2008;20:1495-511.

Kass RS, Ventura V, Brown EN. Statistical issues in the Analysis of Neuronal Data. J Neurophysiol, 2005;94:8-25.

Kreuz T, Haas JS, Morelli A, Abarbanel HDI, Politi A. Measuring spike train synchrony. J Neurosci Methods, 2007;165:151-61.

Kreuz T, Chicharro D, Andrzejak RG, Haas JS, Abarbanel HDI. Measuring multiple spike train synchrony. J Neurosci Methods, 2009;183:287-99.

Kreuz T, Chicharro D, Greschner M, Andrzejak RG. Time-resolved and time-scale adaptive measures of spike train synchrony. J Neurosci Methods, 2011;195:92-106.

Kreuz T. Measures of spike train synchrony. Scholarpedia, 2011;6:11934.

Kreuz T. SPIKE-distance. Scholarpedia, 2012;7:30652.

Kreuz T, Chicharro D, Houghton C, Andrzejak RG, Mormann F. Monitoring spike train synchrony. J Neurophysiol, 2013;109:1457.

Kumar A, Rotter S, Aertsen A. Spiking activity propagation in neuronal networks: reconciling different perspectives on neural coding. Nature Rev Neurosci, 2010;11:615-27.

Louis S, Gerstein GL, Grün S, Diesmann M. Surrogate spike train generation through dithering in operational time. Front Comp Neurosci, 2010;4:127.

Mainen Z, Sejnowski TJ. Reliability of spike timing in neocortical neurons. Science 1995;268:1503-6.

Miller EK, Wilson MA. All My Circuits: Using Multiple Electrodes to Understand Functioning Neural Networks. Neuron, 2008;60:483-8.

Mormann F, Andrzejak RG, Elger CE, Lehnertz K. Seizure prediction: the long and winding road. Brain, 2007;130:314-33.

Nirenberg S, Victor JD. Analyzing the activity of large populations of neurons: how tractable is the problem? Curr Opin Neurobiol 2007;17:397-400.

Papoutsi A, Sidiropoulou K, Cutsuridis V, Poirazi P. Induction and modulation of persistent activity in a layer v pfc microcircuit model. Front Neural Circuits 2013;7:161.

Pikovsky AS, Rosenblum MG, Kurths J. Synchronization. A universal concept in nonlinear sciences. Cambridge Univ. Press, Cambridge, UK 2001.

Quian Quiroga R, Kreuz T, Grassberger P. Event synchronization: A simple and fast method to measure synchronicity and time delay patterns. Phys Rev E 2002;66:041904.

Quian Quiroga R, Panzeri S (Eds.). Principles of neural coding. CRC Taylor and Francis, Boca Raton, FL, USA 2013.

Ross SM. Introduction to Probability Models. Sixth edition. Academic Press, New York, USA 1997.

Rusu CV, RV Florian. A new class of metrics for spike trains. Neural Computation, 2014;26:306-48.

Sacré P, Sepulchre R. Sensitivity analysis of oscillator models in the space of phase-response curves: Oscillators as open systems. Control Systems, IEEE 2014;34:5074.

Sanchez JC, Principe JC, Nishida T, Bashirullah R, Harris JG, Fortes JAB. Technology and Signal Processing for Brain-Machine Interfaces. IEEE Signal Processing, 2008;25:29-40.

Shlens J, Rieke F, Chichilnisky EJ. Synchronized firing in the retina. Curr Opin Neurobiol, 2008;18:396-402.

Thibeault CM, O'Brien MJ, Srinivasa N. Analyzing large-scale spiking neural data with HRLAnalysis. Front. Neuroinform., 2014;8:17.

Tiesinga PHE, Fellous JM, Sejnowski TJ. Regulation of spike timing in visual cortical circuits. Nature Reviews Neuroscience 2008;9:97-107.

Truccolo W, Donoghue JP, Hochberg LR, Eskandar EN, Madsen JR, Anderson WS, Brown EN, Halgren E, Cash SS. Single-neuron dynamics in human focal epilepsy. Nature Neurosci, 2011;14:635-41.

Victor JD. Spike train metrics. Current Opinion in Neurobiology 2005;15:585-592.