IMA Project: Extraction of neural ensembles from the activity of single neurons

Victor Piriou / Supervisor: Thibault Lagache (Biological Image Analysis Unit, Institut Pasteur, Paris)

May 28, 2021

Contents

1 Introduction

The hydra is an animal that has one of the simplest nervous systems in the animal kingdom. It has only a few thousand neurons, which is few compared to the billions of neurons that we humans have.

Therefore, tracking the activity of all the hydra's neurons is possible and could lead to the first decoded nervous system.

Breaking the neural code of this tiny animal would involve linking time series information of the electrical activity of neurons or sets of neurons to each specific action of the animal.

The activity of each neuron is quantified by calcium imaging. Neurons are labeled with a fluorescent indicator. Activity is deduced from changes in fluorescence due to changes in intracellular calcium ion concentration. Indeed, calcium ions enter the cell body membrane in response to the emission of action potentials.

The challenge is to understand how interacting neurons can integrate signals from the environment, calculate the animal's state and trigger appropriate behaviors.

Since the time dedicated to the project is limited, it covers the 3 steps (framed in purple) in Figure 1.

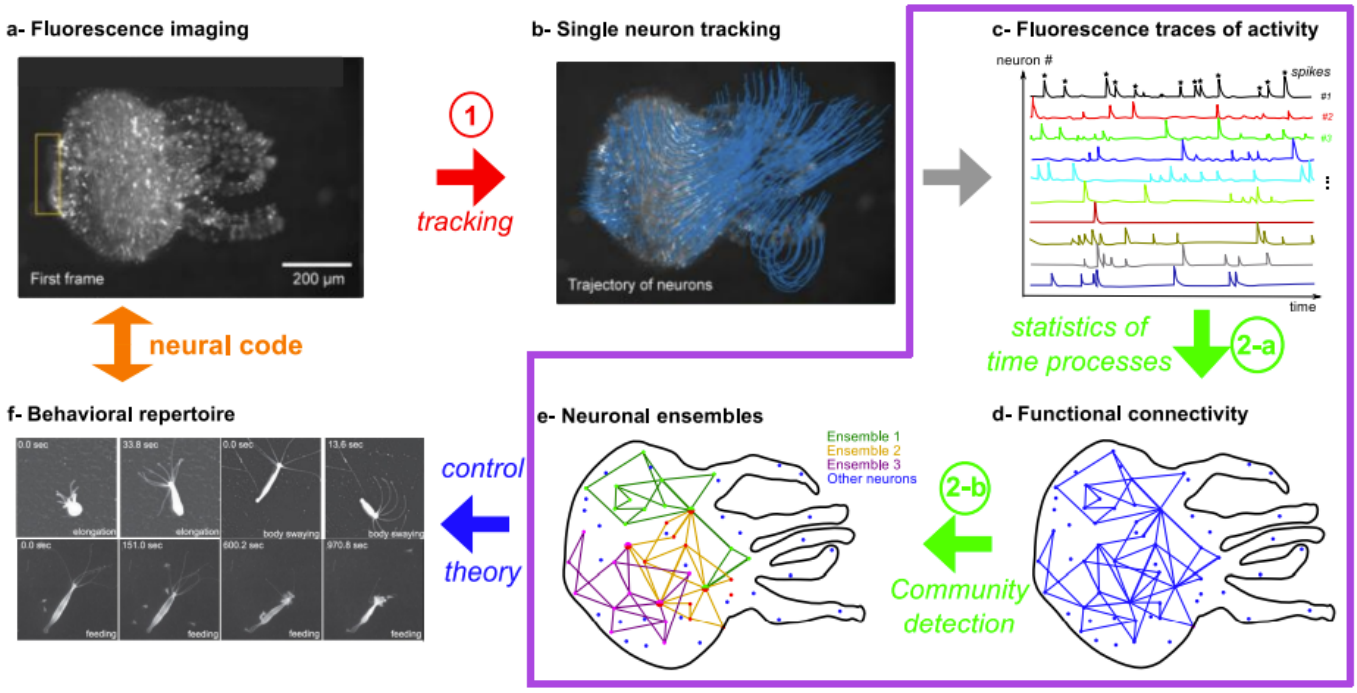


Fig. (1) **Breaking the neural code of Hydra** consists in relating the sequential activity of neurons or ensemble of neurons, observed with fluorescence imaging **(a)** to each specific action of the animal **(f)**, as they document significant relations of causality between the time series of individual neuron's activities. This multi-step approach consists in **(b)** long-term, single particle tracking of $\approx 1000 - 2000$ neurons in freely-behaving and deforming animal (manual tracking over only 200 frames (20 s) is shown here), **(c)** extraction of individual fluorescence traces and spikes (highlighted with black stars for neuron 1), **(d)** statistical inference of neuron's functional connectivity (line thickness indicate connection weights) and, **(e)** clustering into significant neuronal ensembles. Finally, recasting the activity and functional connectivity of individual neurons in an optimal control theory framework will help to understand how the coordinated activity of hundreds to thousands neurons control the animal's state **(f)**.

This work aims at proposing a method to automatically detect nerve impulses. We are only interested in the times at which the impulses take place. We want to derive, from the recording of the calcium fluorescence trace of neuron i, a vector T_i such that $T_i^j = 1$ if there is a spike at the j^{th} frame, $T_i^j = 0$ otherwise. Each action potential is characterized by an almost instantaneous increase of the fluorescence followed by a slow decrease (see figure 1-c). The fluorescence signal being particularly noisy, the detection of the pulses is far from being obvious and is a very active research field.

This detection of the impulses allows us to carry out more reliably the second step which is a statistical analysis aiming at establishing a matrix of the estimates of the similarity between the electrical activities of the neurons, based on the co-occurrence of the neural spikes. To estimate these similarities, we will use distance metrics that are sensitive only to the spike times as well as distance metrics that are also sensitive to interspike intervals.

The third and last step consists in creating an undirected graph whose vertices are the neurons and whose edge weights are the similarities between two neurons. Then, we will perform a binarization of the weights of the edges of the graph: only those whose weight (*i.e.* the similarity between the two neurons) exceeds a certain threshold will be kept. Then, an algorithm allowing to extract the communities of neurons presenting similar activities will be applied.

To validate our results, the program will be tested on a set of time series of neuron activities with a known number of communities. We will compare these distance metrics in terms of their ability to classify neurons into distinct communities.

For each of the tested distance measures, this capacity will be mainly deduced from the number of communities found compared to the one we are supposed to obtain.

2 The data

Real data were available. They correspond to the recording of the following of 954 neurons and are distributed on 4 files. It is on this simulation that has been tested the naive neural spike detection method.

Synthetic data have also been provided. They constitute a simulation (spread over two files) of the tracking of 500 equidistributed neurons in 10 communities and, a simulation (spread over two other files) of the tracking of 500 equidistributed neurons in 5 communities. The noise level of these simulations is moderately high. These simulations were used to validate the results.

For each instant and for each neuron, these files gather statistics (mean, maximum, minimum, median, total, variance) on the distribution of the intensities of the pixels assigned to this neuron.

	A	B	C	D	E
1	Frame number	sub-trackgroup#0	sub-trackgroup#1	sub-trackgroup#2	sub-trackgroup#3
2	0				702.96
3	1				693.36
4	2		581.28		710.24
5	3		575		710.08
6	4		577.08	775.08	703.68
7	5		580.24	749.08	703.96
8	6		580.72	772	700.8
9	7	802.92	579.68	752.88	710.24
10	8	796.32	577.44	770.2	706.76
11	9	805.08	575.84	751.4	715.4
12	10	794.2	578.24	765.2	702.28
	11	793.2	581.56	757.16	714.4

Channel 0 - Mean intensity | Channel 0 - Max intensity | Channel 0 - Min intensity | Channel 0 - Median intensity | Channel 0 - Intensity sum | Channel 0 - Intensity variance

Fig. (2) **Example of spreadsheet provided.** To each column is associated a neuron. To each row is associated an instant. The sampling frequency is 10 Hz. The start and end times of activity differ from one neuron to another.

In the following, we consider that the electrical activity of a neuron is the average of the intensities of each of the pixels assigned to this neuron.

3 First step: neural spike detection

The purpose of the first part is to determine the instants at which there is a spike (see figure 1-c). This makes it possible to distinguish action potential from noise and interference, but it is also an essential step before calculating distances, since the baseline is very variable from one recording to another. For example, the intensity of the global minimum of the recording of a neuron can be higher than the maximum intensity of the recording of another neuron. Consequently, direct signal correlation is not usable because of the baseline. We therefore focus on the co-occurrences of the impulses.

3.1 State of the art overview

Thibault Lagache *et al* [15] compared the accuracy and robustness of 4 methods for detecting peaks:

- OASIS (Online Active Set method to Infer Spikes);
- CDfoopsi (Constrained fast nonnegative deconvolution);
- MLspike;
- A Naive method that consists of smoothing the signal with a wavelet thresholding, before computing the first derivative of the signal and estimates the spikes locations with derivatives greater that one standard deviation of the derivative over the entire calcium trace.

Their performance was evaluated by observing the error rate proposed in [4] by varying 3 parameters:

- The signal-to-noise ratio;
- The firing rate inside burst (a burst is a group of action potentials generated in rapid succession);
- The baseline amplitude.

It has been shown that the CDfoopsi method seems to provide the best compromise between accuracy and robustness to the variation of the 3 parameters mentioned above. Robustness is crucial because, in practice, the captured activities do not all present the same pattern of triggering of action potentials (peaks can be isolated or grouped within bursts) nor the same signal to noise ratio.

This method [14] thus allows us to obtain a vector $\hat{\boldsymbol{s}}$ such that its t^{th} component denotes the number of action potentials that have already been triggered at time t. This vector is obtained by solving the following convex program:

$$\begin{aligned} \min_{\hat{c},\hat{b},\hat{\boldsymbol{s}}} \quad & 1_T^{\mathrm{T}}\hat{\boldsymbol{s}} \\ \text{subject to:} \quad & \hat{\boldsymbol{s}} \geq 0, \hat{\boldsymbol{s}} = G\hat{\boldsymbol{c}}, \left\|\boldsymbol{y} - \hat{\boldsymbol{c}} - \hat{b}\mathbf{1}_T\right\| \leq \hat{\sigma}\sqrt{T} \end{aligned}$$

where:

- 1_T^{T} is a column vector of length T composed of 1;
- $\hat{c}$ is the vector of calcium concentration over time;
- $\hat{b}$ is the vector of the calcium concentration of the baseline over time;
- σ is the variance of the law which models the noise appearing during the observation of the fluorescence.

The constraint $\hat{\boldsymbol{s}} = G\hat{\boldsymbol{c}}$:

$$\text{where } G = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -\gamma_1 & 1 & 0 & \dots & 0 \\ -\gamma_2 & -\gamma_1 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & -\gamma_2 & -\gamma_1 & 1 \end{bmatrix},$$

corresponds to the matrix expression of the calcium concentration equation modeled by this autoregressive process of order p:

$$c(t) = \textstyle\sum_{j=1}^{p} \gamma_j c(t-j) + s(\mathrm{t}).$$

$\left\|\boldsymbol{y} - \hat{\boldsymbol{c}} - \hat{b}\mathbf{1}_T\right\| \leq \hat{\sigma}\sqrt{T}$ allows to constrain the residual energy. Indeed, the formula linking the fluorescence to the calcium concentration is:

$$y(t) = \alpha(c(t) + b) + \varepsilon(t), \quad \varepsilon(t) \sim N\left(0, \sigma^2\right),$$

where α is a positive scalar, b is the baseline and the noise ε is assumed to be i.i.d. and sampled according to a Gausian centered at 0 with a variance σ^2.

whence :

$$\mathbb{E}(y(t) - \alpha(c(t) + b))^2 = \sigma^2,$$

by summing over all the frames, we obtain:

$$\mathbb{E}\left\|\boldsymbol{y} - \alpha\left(\boldsymbol{c} + b\mathbf{1}_T\right)\right\|^2 = \sigma^2 T,$$

finally, according to the Law of large numbers:

$$\left\|\boldsymbol{y} - \alpha\left(\boldsymbol{c} + b\mathbf{1}_T\right)\right\|^2 \approx \sigma^2 T.$$

Minimizing $1_T^{\mathrm{T}}\hat{\boldsymbol{s}}$ allows to promote spike sparsity and avoid overfitting.

3.2 Naive method used

It was recommended by the project supervisor to design a simple pulse detection algorithm, rather than implementing sophisticated methods that required several years of research.

The description of the algorithm, without going into implementation details, is given below:

The algorithm starts by smoothing the signal of the electrical activity of a single neuron, with an Gaussian filter, in order to attenuate the noises due to the acquisition during imaging.

Next, we slide a window of width equal to D, which is the minimum width that a pulse can have on the smoothed signal. To simplify the problem, the overlap of the pulses is limited. Thus, if we detect a pulse per frame n, we will not be able to redetect it before frame $n + D + 1$.

If the intensity corresponding to the center of the window is the maximum value of this window, it is considered that it is perhaps the maximum value of a pulse.

To find out whether this is the case or not, we must iteratively search for the local minimums of the signal on either side of the peak potential: as long as the neighboring intensity is lower, we move away from the peak potential. We know that we have reached the minimum when the neighboring intensity increases.

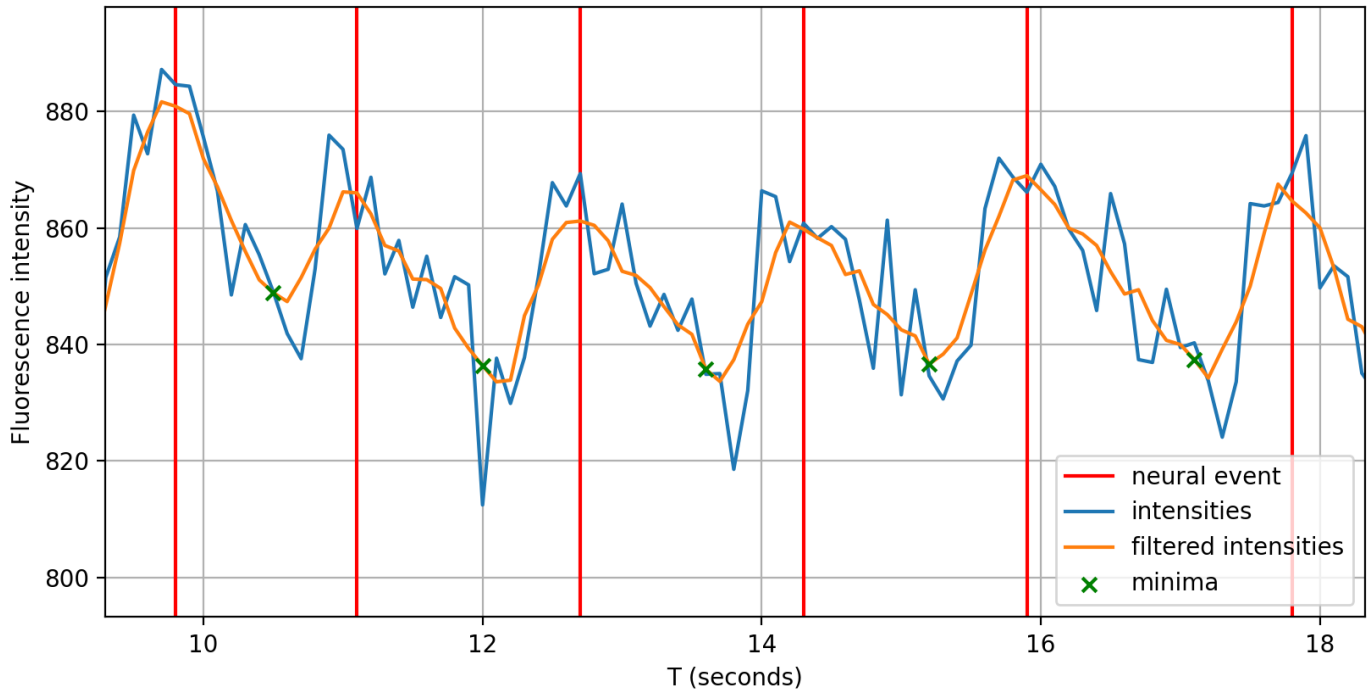


Fig. (3) **Visualization of the pulse detection**. We can notice that filtering allows us to increase our chances of falling in the right minima.

It is to facilitate this search for a minimum that it is necessary to have a previously smoothed signal. Once the minimums on both sides of the peak potential are found, the estimation of the growth and decay rates of the slopes is performed. If the growth (respectively decay) rate exceeds the 98th percentile of the distribution of positive (respectively negative) discrete derivatives, a peak is detected.

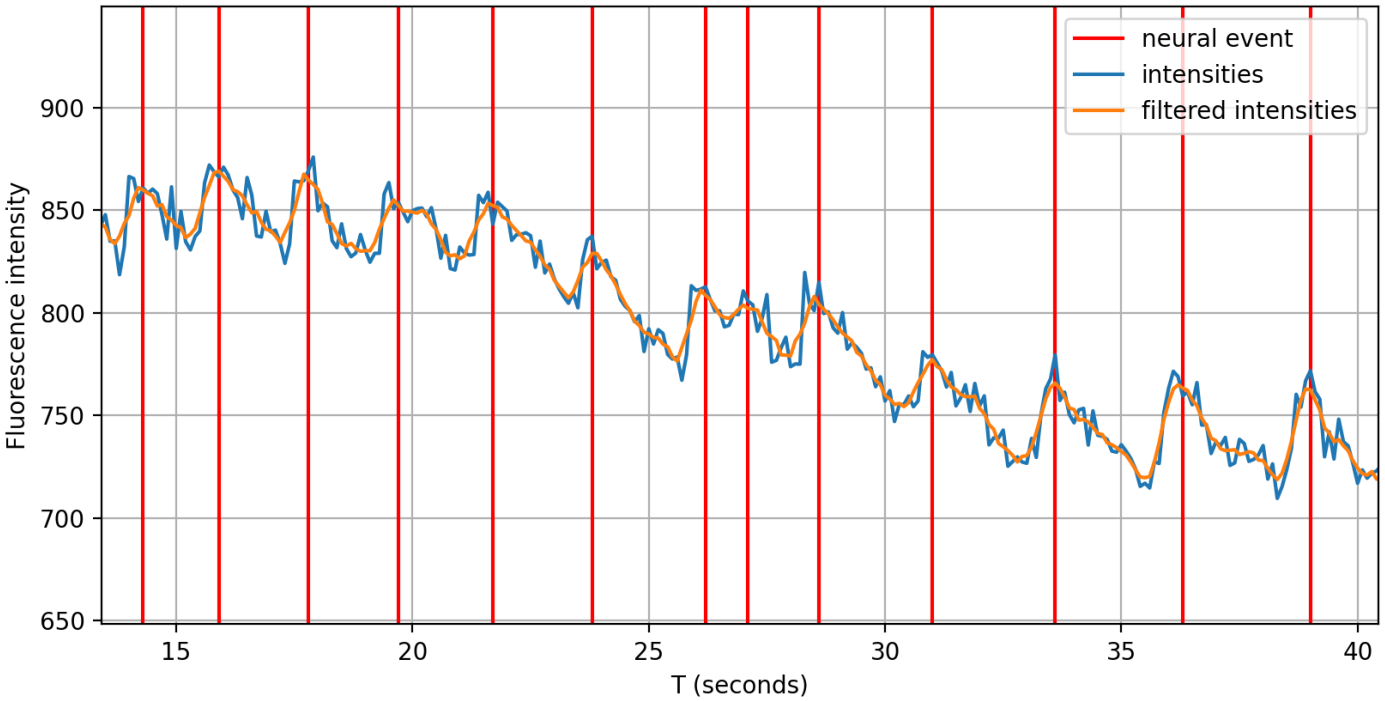


Fig. (4) **Visualization of the pulse detection on a fairly regular portion of the signal.**

3.2.1 Limits

The pre-smoothing of the signal implies that the spikes inside the bursts will be less likely to be detected as the firing rate inside burst increases.

Moreover, the way to estimate the growth and decay rates is not perfect. Indeed, to be able to find the right minimums, the slopes would have to be strictly monotonic. However, when using an averaging filter, there is a trade-off between preserving the shape of the signal (and thus preserving the pulses) and smoothing the signal.

3.2.2 Potential improvements

An idea to overcome this problem would be to replace the filtering used by a spline regression, a least squares cost function and a penalty on the integral of the squared second derivative in order to avoid that the generated f curve fits the data too well [6].

$$\sum_{i=1}^{n} \left(y_i - f\left(x_i\right)\right)^2 + \lambda \sum_{i=1}^{n-2} f''(x_i)^2,$$

where :

- n is the number of frames in the record;
- y_i represents the signal strength at frame i;
- $f(x_i)$ represents the value obtained by regression at frame i;
- λ is the regulirization parameter.

However, this would introduce this hyper-parameter λ. I found out that there is a more sophisticated (and probably more efficient) method called Bayesian Adaptative Regression Spline (BARS) used to estimate these rates. This method assumes that the pulses are generated from a Poisson process [7], but this is not the case. A more suitable method would be the BAKS (Bayesian Adaptive Kernel Smoother) method which uses a kernel smoothing technique [1].

The time of the project being limited, I made the choice to be satisfied with the first results obtained for this part 1 and to go to part 2.

4 Second step: calculation of distances between neurons

4.1 State of the art overview

By studying the scientific literature, I have identified mainly two types of distances suitable for similarity estimation between neurons. There are distances that depend on the time scale (Van Rossum distance, Victor-Purpura distance, etc.), and thus on the sampling, and those that do not. The advantage of the latter is that they do not require parameters, unlike the former. Generally, it is preferable to use a method that does not introduce parameters because:

- It is difficult to define what the optimal parameter could be [3].
- There is no guarantee that there is an optimal parameter [3].

Moreover, these time-dependent methods assume that neural information is essentially characterized by the frequency of action potentials [16]. That is, they are based on the assumption that intensity is positively correlated with frequency. However, our spike detection algorithm assumes little overlap between spikes, so there is little correlation between the frequency of our detected spikes and their intensity. Therefore, it is likely that in our case, the best methods are those that do not require parameters.

Distances that do not require parameters include:

- Jaccard index;
- Hamming distance;
- Cosine similarity;
- SPIKE distance;
- ISI distance;
- Event-Synchronisation distance;
- SPIKE-Synchronisation distance.

Mathematically, they are not all distances [17]. For example, the SPIKE distance does not respect the triangular inequality, the Spike Synchronization distance does not respect either the separation property or the transitivity property [12]. Among the above mentioned distances, only the Hamming distance - which associates the number of frames where two spike trains differ - is a distance in the mathematical sense. Nevertheless, this does not mean that the other distances will not lead to a satisfactory classification of neurons into distinct communities.

Finally, ideally, we would also need a method that does not rely on the assumption that the neurons of the same community are active exactly at the same time. Indeed, the propagation of nervous messages from one end of a neuronal ensemble to the other is not necessarily instantaneous. However, the low temporal resolution that we have, greatly approximates the times of the impulses. Consequently, we cannot perceive these temporal shifts.

4.2 Explanations of the experimented distance measurements

All of the distance measurements I have experienced lead to an estimate of the similarity between 0 and 1.

The SPIKE and ISI methods measure dissimilarity. However, similarity can be inferred by subtracting them from 1.

Moreover, these similarities are not all of the same qualitative nature. Indeed, the SPIKE, ISI and SPIKE-Synchronization distances are sensitive to inter-spike intervals. Whereas the Jaccard index and the cosine similarity are essentially sensitive to the times of the pulses. However, with respect to our data, it will generally be unlikely that two pulses occur at exactly the same time. Therefore, we know, in advance, that these two methods alone will not give satisfactory results. In order to overcome this problem, for each of the two signals to be compared, a convolution is first performed between the spike trains and a kernel having the typical shape of a pulse.

This kernel is based on the equation modeling the average evolution of the fluorescence during a pulse [11] :

$$f(t) = A\frac{\exp\left(-\left(\frac{t-t_0}{\tau_{\text{decay}}}\right)^{\beta}\right)}{1+\exp\left(\frac{-t-t_0-\mu}{\tau_{\text{rise}}}\right)}. \tag{1}$$

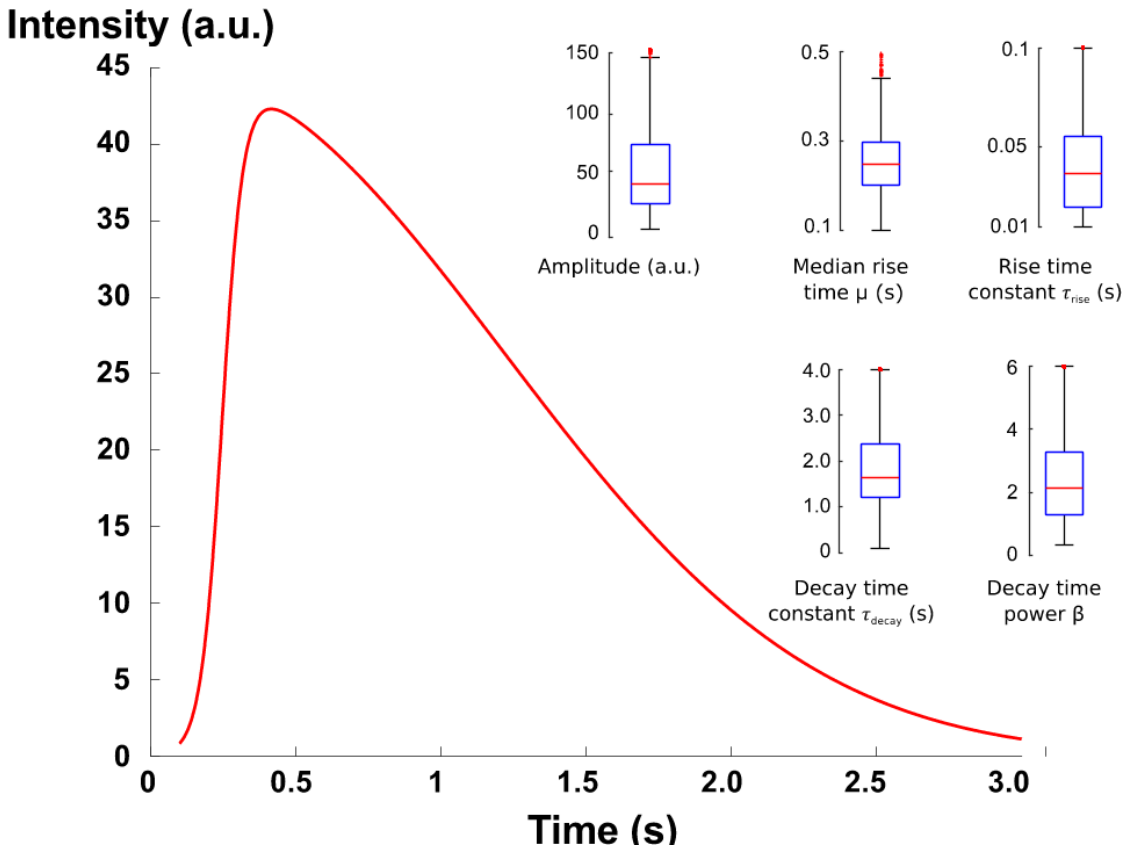


Fig. (5) **Average evolution of the fluorescence of a pulse**, calculated from 3075 pulses from the monitoring of 444 single neurons [11].

4.2.1 Distance metrics sensitive only to spike times

4.2.1.1 Jaccard index

Let be two spike trains T_1 and T_2 of n frames :

\$T$_1 = (T_1^1, T_1^2, ..., T_1^n)$,
$T_2 = (T_2^1, T_2^2, ..., T_2^n)$,

with $T_i^j = 1$ if there is a spike at the j$^{\text{th}}$ frame of T_i, $T_i^j = 0$ otherwise.

If we wanted the Jaccard index from our two spike trains, we would use the formula:

$$\text{Jaccard } (T_1,T_2) = \frac{M_{11}}{M_{01}+M_{10}+M_{11}}.$$

M_{11} represents the total number of frames where T_1 and T_2 both have a value of 1.
M_{01} represents the total number of frames where T_1 is 0 and T_2 is 1.
M_{10} represents the total number of frames where T_1 is 0 and T_2 is 1.
M_{00} represents the total number of frames where T_1 and T_2 both have a value of 0.

Nevertheless, as said before, each spike train is convolved with the kernel based from the equation 1.

Thus, the Jaccard index formula can be adapted to convolved spike trains:

$$\text{Jaccard } (C_1,C_2) = \frac{\sum_{i=1}^{n} \min\left(C_1^i,C_2^i\right)}{\sum_{i=1}^{n}\left(C_1^i+C_2^i-\min\left(C_1^i,C_2^i\right)\right)},$$

where :
$C_1^i = (T_1 * f)[i] = \sum_{m=0}^{\min(n,i)} T_1[m] f[i-m]$,
$C_2^i = (T_2 * f)[i] = \sum_{m=0}^{\min(n,i)} T_2[m] f[i-m]$.

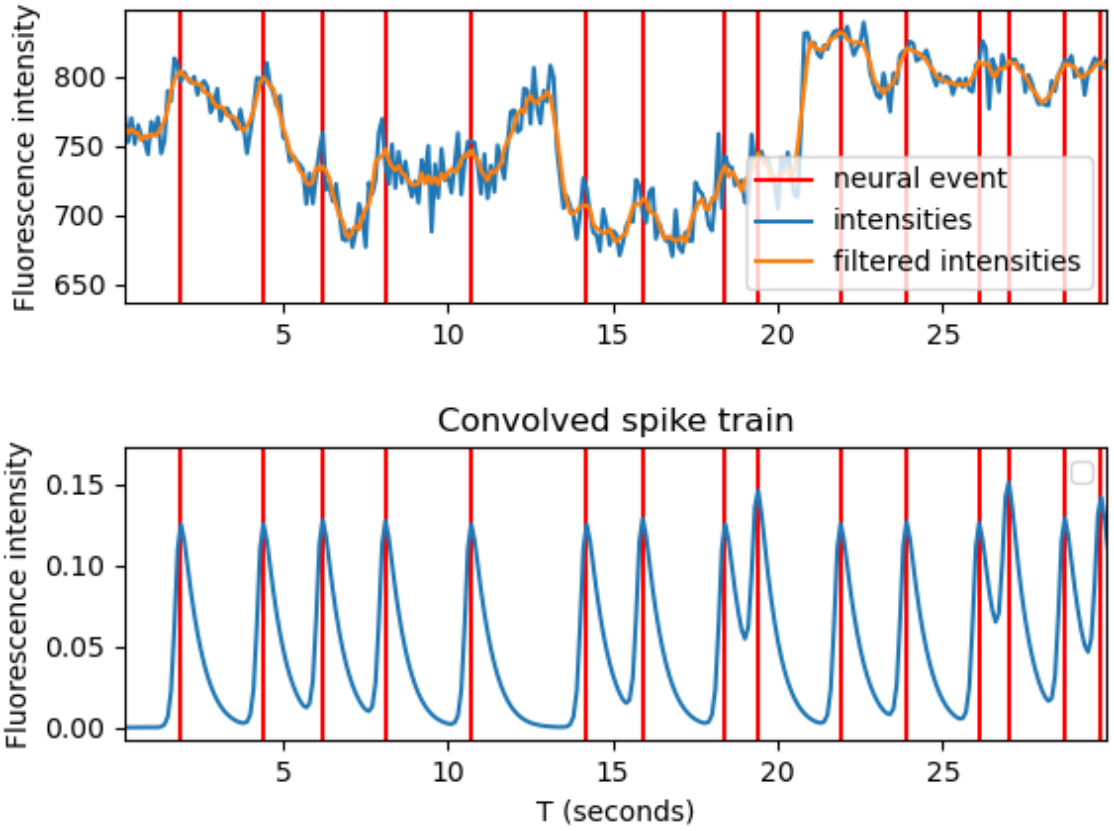


Fig. (6) **Portion of a spike train convolved** with the kernel based on f.

4.2.1.2 Cosine similarity

Let be two time series C_1 and C_2 (of the same length) corresponding to the convolved spike trains.

The Cosine similarity between these two series is obtained by dividing their scalar product by the product of their norms:

$$\text{Cosinus } (C_1, C_2) = \frac{C_1 \cdot C_2}{\|C_1\| \|C_2\|}.$$

4.2.2 Distance metrics sensitive to the intervals between peaks

4.2.2.1 ISI distance

The ISI distance [9] was introduced by Thomas Kreuz *et al* in 2007. It is based on the instantaneous value $I(t)$ of the interval between the nearest spike preceding t and the nearest spike following t.

The global distance is calculated via an integration:

$$D_I = \frac{1}{T} \int_0^T I(t) \mathrm{d}t,$$

where T is the total duration of the record.

his dissimilarity profile $I(t)$ is based on the ratio between the intervals, of each sequence of peaks, comprising the time t.

Let $\{t_i^{(1)}\}$ be the spike times of the train 1. The intervals between the spikes are defined by :

$$\nu_i^{(1)} = t_{i+1}^{(1)} - t_i^{(1)},$$

and similarly for $\nu_i^{(2)}$, the second spike train. In order to obtain a time-dependent profile, these sequences of inter-spike distances are transformed into piecewise constant functions:

$$\nu^{(1),(2)}(t) = \nu_i^{(1),(2)} \text{ where } t_i^{(1),(2)} \leq t < t_{i+1}^{(1),(2)}.$$

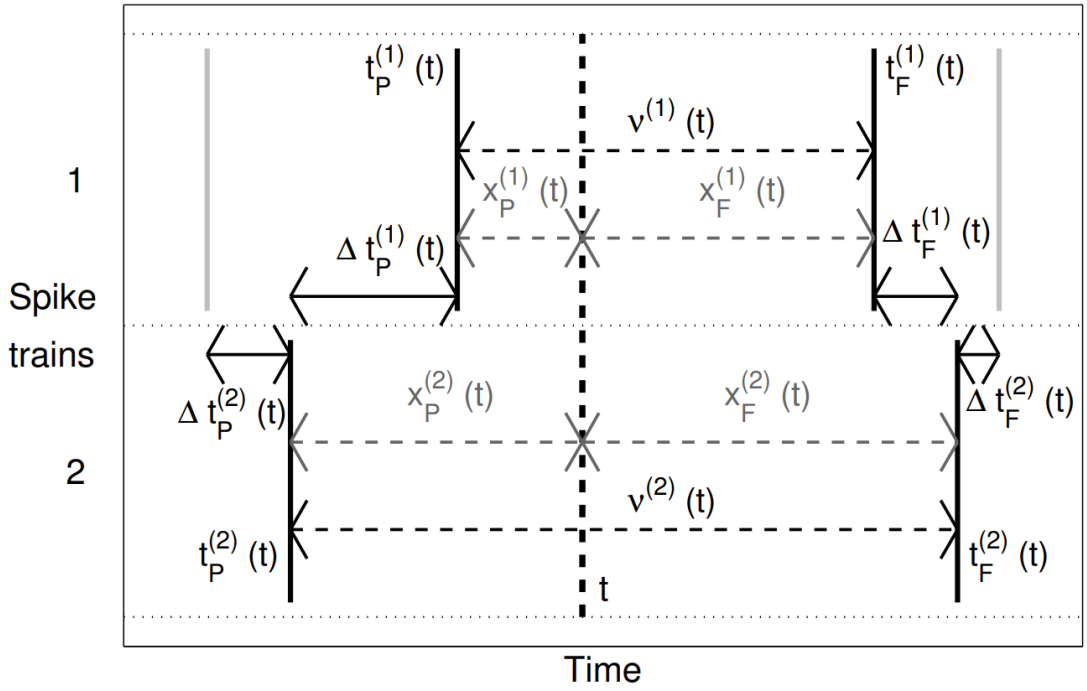


Fig. (7) **Definitions, for a given time t, of the intervals required for the calculation of the ISI and SPIKE profiles.**

The ISI profile is then given by the absolute value of the difference between the intervals, normalized:

$$I(t) = \frac{\left|\nu^{(1)}(t) - \nu^{(2)}(t)\right|}{\max\left\{\nu^{(1)}(t), \nu^{(2)}(t)\right\}}, \quad t \in [0, T).$$

However, there is a potential ambiguity regarding the first and last interval if there are no pulses at the beginning and end of the sequence. To correct this, a peak is added at the beginning and end of the sequence.

4.2.2.2 SPIKE distance

The SPIKE distance [10] (Thomas Kreuz et *al*, 2013) is also based on the instantaneous value $S(t)$ on the interval between the nearest spike preceding t and the nearest spike succeeding t.

The global distance is calculated via an integration.

$$D_I = \frac{1}{T} \int_0^T S(t) \mathrm{d}t,$$

where T is the total duration of the time series.

The calculation of $S(t)$ is based on the previous spikes $t_P^{(1),(2)}(t)$ and the following spikes $t_F^{(1),(2)}(t)$ of each spike train (cf. Fig. 7) The interval associated with time t is defined by:

$$\nu^{(1),(2)}(t) = t_F^{(1),(2)}(t) - t_P^{(1),(2)}(t). \tag{2}$$

For each of these 4 spikes, the distance, from the nearest spike at time t to the other spike, is calculated, e.g.:

$$\Delta t_P^{(1)}(t) = \min_i \left\{ \left| t_P^{(1)} - t_i^{(2)} \right| \right\},$$

and similarly for $\Delta t_P^{(2)}$ and $\Delta t_F^{(1),(2)}$. These distances are then weighted by the distances separating time t from the 4 spikes:

$$x_P^{(1),(2)}(t) = t - t_P^{(1),(2)} \quad \text{et} \quad x_F^{(1),(2)}(t) = t_F^{(1),(2)} - t,$$

avec $x_P^{(1),(2)}(t) + x_F^{(1),(2)}(t) = \nu^{(1),(2)}(t)$, (cf. Fig. 7). The weighted distance for the first spike train is therefore:

$$S_1(t) = \frac{\Delta t_P^{(1)}(t) x_P^{(1)}(t) + \Delta t_F^{(1)}(t) x_F^{(1)}}{\nu^{(1)}(t)},$$

and similarly $S_2(t)$ is defined for the second spike train. Finally, these local distances are weighted by the local intervals and, by normalizing, we obtain the definition of the SPIKE profile:

$$S(t) = \frac{S_1(t)\nu^{(2)}(t) + S_2(t)\nu^{(1)}(t)}{\frac{1}{2}\left(\nu^{(1)}(t) + \nu^{(2)}(t)\right)^2}, \quad t \in [0, T).$$

As with the ISI distance, the ambiguity one has about the beginning and end of the spike train is removed by adding a spike at the beginning and end of the spike train.

4.2.2.3 SPIKE-Synchronisation distance

The SPIKE-Synchronization distance [8] (Thomas Kreuz *et al*, 2015) can be seen as a detector of simultaneous pulses.

For the SPIKE-Synchronization profile, a coincidence indicator $C_i^{(1),(2)}$ is defined for each of the two peak sequences $S^{(1),(2)}$. This indicator is defined by:

$$C_i^{(1)} = \begin{cases} 1 & \text{if } \min\limits_j \left(\left| t_i^{(1)} - t_j^{(2)} \right| \right) < \tau_{ij}^{(1,2)}, \\ 0 & \text{otherwise .} \end{cases} \tag{3}$$

This means that a coincidence is found for spike i (of the first spike train) if the distance to the first spike j (of the second spike train) is smaller than the coincidence window $\tau_{ij}^{(1,2)}$. The time window is defined adaptively:

$$\tau_{ij}^{(1,2)} = \tfrac{1}{2} \min \left\{ \nu_i^{(1)}, \nu_{i-1}^{(1)}, \nu_j^{(2)}, \nu_{j-1}^{(2)} \right\},$$

with $\nu^{(1),(2)}$ the inter-spike interval given in equation 2.

The coincidence indicator of the spike train $C_i^{(2)}$ is calculated in the same way as in equation 3 but, with the permuted indices $(1) \leftrightarrow (2)$. The SPIKE-Synchronization profile is then obtained by merging the coincidence indices of the two spike trains $\{C_k = \left\{ C_i^{(1)} \cup \left\{ C_i^{(2)} \right\} \right.$ as well as the spike times: $\left\{ t_k' \right\} = \left\{ t_i^{(1)} \right\} \cup \left\{ t_i^{(2)} \right\}$, which results in a discrete function defining a SPIKE-Synchronization value for each spike time of the two spike trains: $t_k' \mapsto C_k$.

Unlike the ISI and SPIKE profiles described above, the SPIKE-Synchronization distance profile is only defined for spike times.

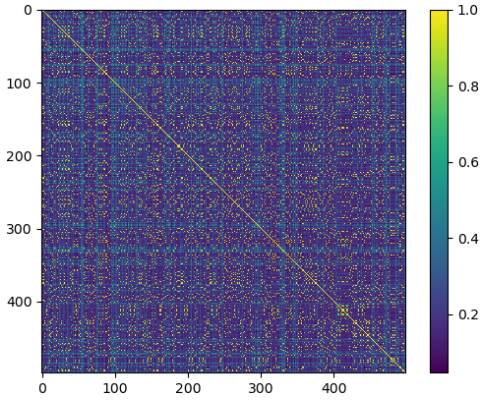
Finally, the SPIKE-Synchronization similarity is obtained by:

$$\text{SYNC} = \frac{1}{M} \sum_{k=1}^{M} C_k = \frac{C}{M},$$

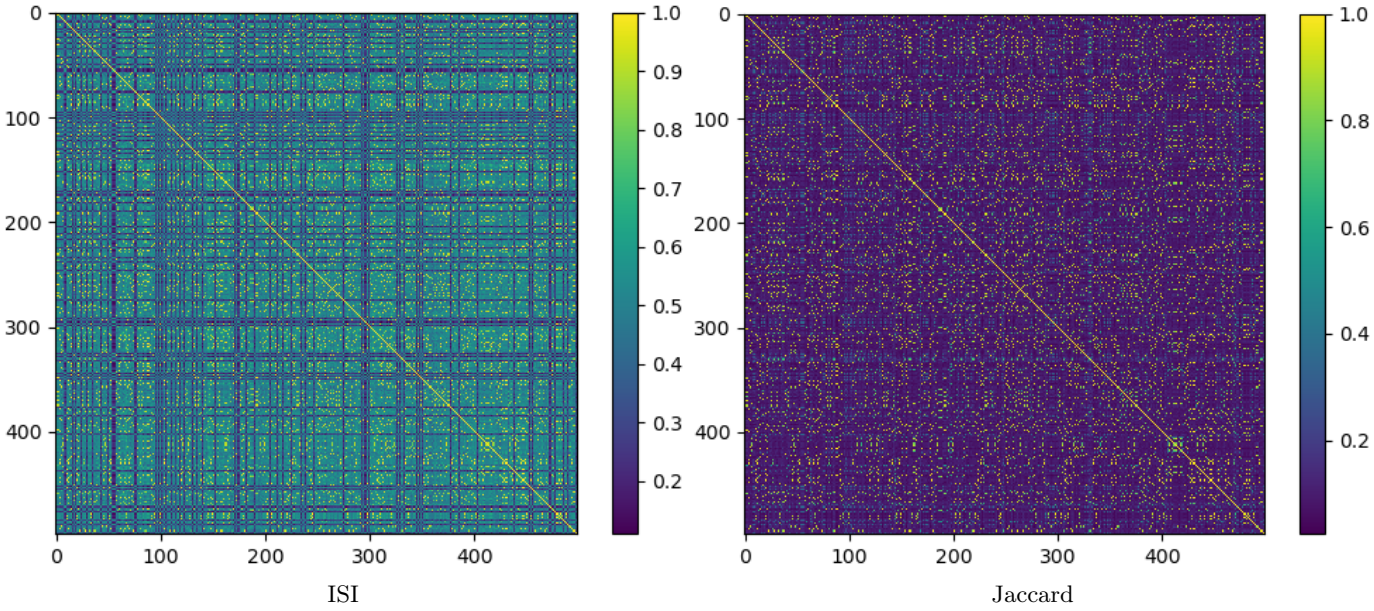
with M the total number of spikes in the merged spike trains t_k^{prime} and C the total number of simultaneous spikes.

4.3 Visualization of the obtained similarity matrices

The SPIKE, ISI, and SPIKE Synchronization distance matrices were coded using the PySpike library [13].

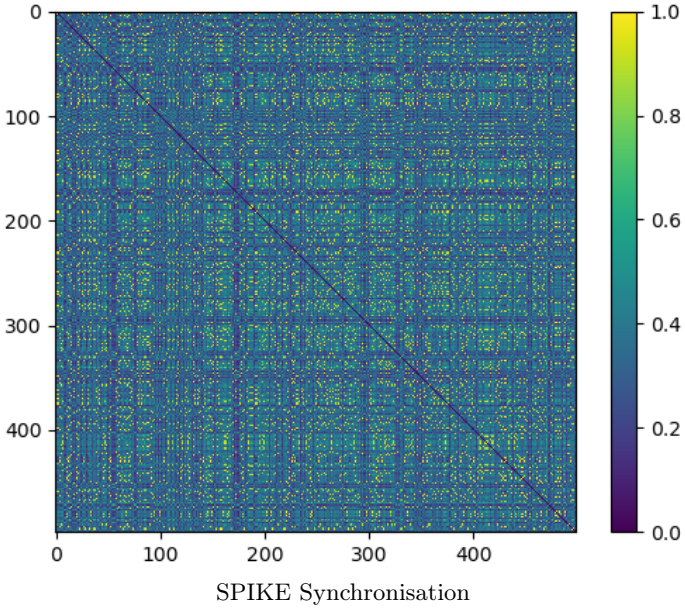


Cosinus

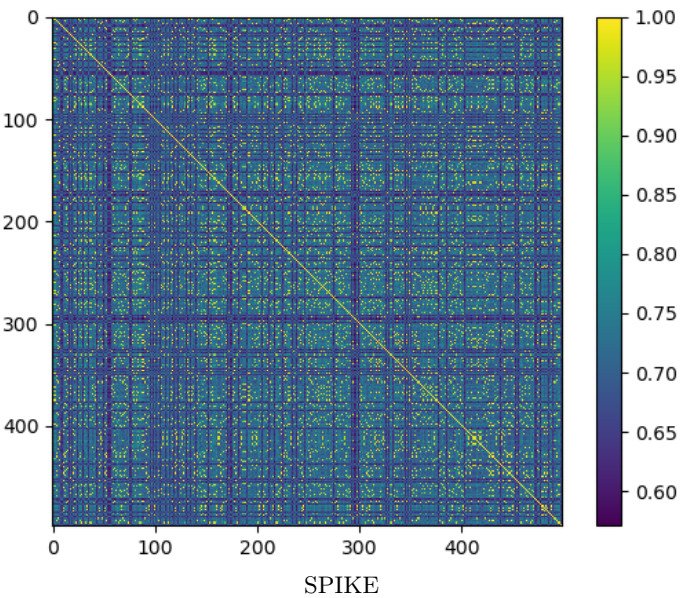


ISI

Jaccard



SPIKE Synchronisation



SPIKE

Fig. (8) **Distances matrices between neurons** obtained from the tracking simulation of 500 neurons.

It can be seen that the distribution of similarities varies depending on the measure used. The cosine similarity and the Jaccard index seem less sensitive than the ISI, SPIKE and SPIKE-Synchronization distances since the average similarity of their matrix is lower.

5 Third step: extraction of the neuronal ensembles

Once the similarity measures are obtained, a binarization of these similarities must be performed by thresholding in order to establish an undirected graph linking neurons with similar activity. This threshold cannot be fixed because it depends at least on the distance used.

5.1 Computation of the thresholds used to binarized the similarity matrix

5.1.1 Thresholds to be used to test the program on real data

When the data is real, we don't know how many communities we are supposed to find.

5.1.1.1 Probabilistic thresholds

I implemented a Monte-Carlo method to automatically determine these thresholds.

The threshold we want to find is the one that will allow us, for example, to ensure with a 95% confidence level that the similarity obtained between a spike train 1 and a spike train 2 is higher than the one we would have obtained between two other random spike trains having respectively the same number of spike as the spike trains 1 and 2.

The algorithm used is the following:

For each pair of neurons (i,j), with $i \neq j$, we create two random spike trains S_1 and S_2 having respectively the same size and the same number of spikes as S_i and S_j, the spike train of neuron i and i. Then, we compute the similarity between S_1 and S_2. We repeate this process a large number of times, n times. We store the n similarities obtained in a list that we sort in ascending order. The threshold obtained is the value of this list having the index $\lceil 0.95 \times n \rceil$.

Thus, for each pair of neurons (i,j) a new threshold is calculated since the size of the spike trains as well as the number of spikes differ according to the spike trains.

Therefore, for each distance metric, a square (threshold) matrix must be calculated whose order is equal to the number of neurons.

5.1.1.2 Single threshold depending on the distance measurement

These probabilistic threshold matrices being much too long to compute for the available hardware, I finally chose as threshold the 95^{th} percentile of the similarities distribution so that the numbers of communities obtained with the different distance measures are at least comparable between them.

I still coded the functions that allow to store the probabilistic threshold matrices in files and to load them from these files.

5.1.2 Threshold used to measure performance on synthetic data

When the data are synthetic, we know the number of communities we are supposed to find but also the number of neurons showing no activity (thus belonging to no community).

We can therefore choose for each distance measure a unique threshold t^* such that:

$$t^* = \arg\min_t g(t),$$

where :

- $g(t) = \big|C - f(t)\big|$;
- C is the number of inactive neurons (C is 100 in the 500 neuron tracking simulation);
- $f(t)$ is the number of isolated vertices (*i.e.* the number of « communities » found containing a single neuron).

This minimum is easily found since the function g is convex.

5.2 Method used: Louvain algorithm

The Louvain method [2] is a hierarchical community extraction algorithm. It was proposed by Vincent Blondel *et al* from the University of Louvain in 2008.

5.2.1 Principle of the algorithm

The value to optimize is the modularity, defined between -1 and 1, measuring the quality of a partitioning of the graph nodes into communities.

For a weighted graph, the modularity Q is defined as :

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta \left(c_i, c_j \right),$$

where :

- A_{ij} represents the weight of the edge between nodes i and j;
- k_i and k_j are the sum of the weights of the edges linked to nodes i and j respectively;
- $2m$ is the sum of all the weights of the edges of the graph;
- c_i et c_j are the node communities;
- δ is a delta function.

The principle is that a good partitioning of the graph implies a large number of intra-community edges and a small number of inter-community edges.

The modularity is positive for a graph and a partition for which there is no edge connecting 2 nodes of different classes but at least one edge connecting 2 nodes of the same class.

The modularity is negative for a graph and a partition for which there is no edge connecting 2 nodes of the same class but, at least one edge connecting 2 nodes of different classes.

The modularity tends to 0 when we randomly partition a graph in which the edges are randomly distributed.

We can consider that a graph has a significant community structure when a partition obtains a modularity score higher than 0.3.

The algorithm follows two phases **A** and **B** which are repeated iteratively.

At the beginning, each node is associated with a different partition. The number of nodes is equal to the number of partitions.

A. Detection of small communities by local optimization of modularity on each node.

1. For each neighbor j of node i, we check if the overall modularity increases by removing i from its own community and moving it into the community of each neighbor j of i. The node i is then moved into the partition of j for which the modularity gain is the greatest. In the case where no modularity gain is possible, i keeps its initial partition.

This gain ΔQ is defined by :

$$\Delta Q = \left[\frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right]$$

Here, Σ_{in} is the sum of the weights of the edges within the destination community of i, Σ_{tot} is the sum of the weights of all edges related to nodes of the community destination of i, k_i is the weighted degree of i, $k_{i,in}$ is the sum of the weights of the edges between and the nodes of the community destination of i, and m is the sum of the weights of all edges in the graph.

2. Step 1 is repeated for all nodes sequentially until no more modularity gain is possible.

B. Grouping of nodes from the same community into a single node.

1. The weights of the edges between two new communities are determined by summing the weights of the edges located between these two communities.

Steps A and B are repeated until the modularity converges.

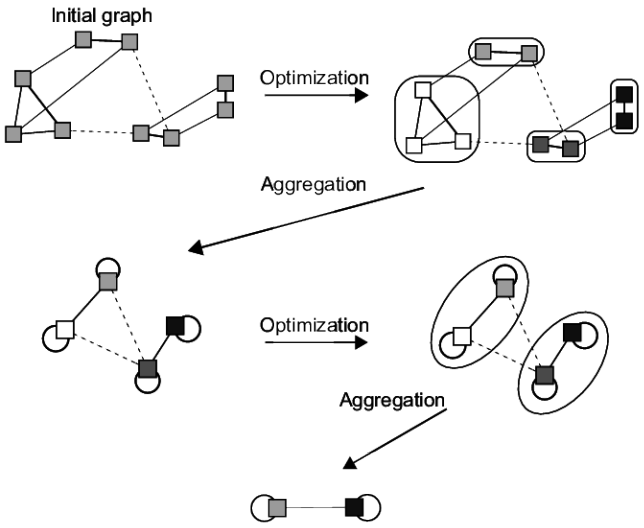


Fig. (9) **Visualization of the phases of the Louvain method** over two iterations.

5.2.2 Avantages

This algorithm has several advantages. First of all, it is easy to implement. It does not require any parameters, nor any preconceived notions about the nature of the graph. Its time complexity is $\mathcal{O}(nlogn)$, which implies that its execution is extremely fast. Most of the computations are performed during the first iterations and decrease exponentially iteration after iteration.

It allows you to choose the resolution, *i.e.* the size of the communities you want to detect.

Another advantage of the Louvain algorithm is its hierarchical nature. Indeed, if one is confronted with communities of different sizes within the same graph, some communities - even well defined - may not be distinguished in the optimal modularity partition. It's a common criticism of modularity which is called the resolution limit. Nevertheless, the Louvain method circumvents the resolution problem thanks to its hierarchical nature. Indeed, the first phase of the algorithm involves moving single nodes from one community to another. Therefore, the probability that two distinct communities are merged by moving the nodes one by one is very low and, therefore, small communities are preserved.

5.2.3 Limits

It is a greedy method, so there is no guarantee that the maximum modularity is achieved. However, the algorithm seems to offer excellent accuracy.

It has also been shown that the order of the nodes does not seem to have a significant influence on the modularity obtained. However, this can impact the computation time. Indeed, if the graph had a much larger number of neurons, we would have to find the best way to order the nodes.

This algorithm associates each neuron to a single community. However, we know that in reality a neuron can be involved in several neural ensembles.

6 Experiments and validation

6.1 Method used to visualize the graph: Fruchterman and Reingold algorithm

The graph is visualized with Fruchterman and Reingold's algorithm (1991) [5] whose goal is to position the nodes so that the edges cross each other as little as possible and are all, more or less, the same length.

The nodes can be seen as particles of the same charge and the edges as springs. At each iteration, the sum of the forces (according to Hooke's and Coulomb's laws) applied to each of the nodes is calculated. The nodes are then moved. The process is repeated until there are no more moves to be made in order to respect the laws of physics. This essentially allows an aesthetic visualization of our results.

6.2 Visualization and analysis of the obtained communities

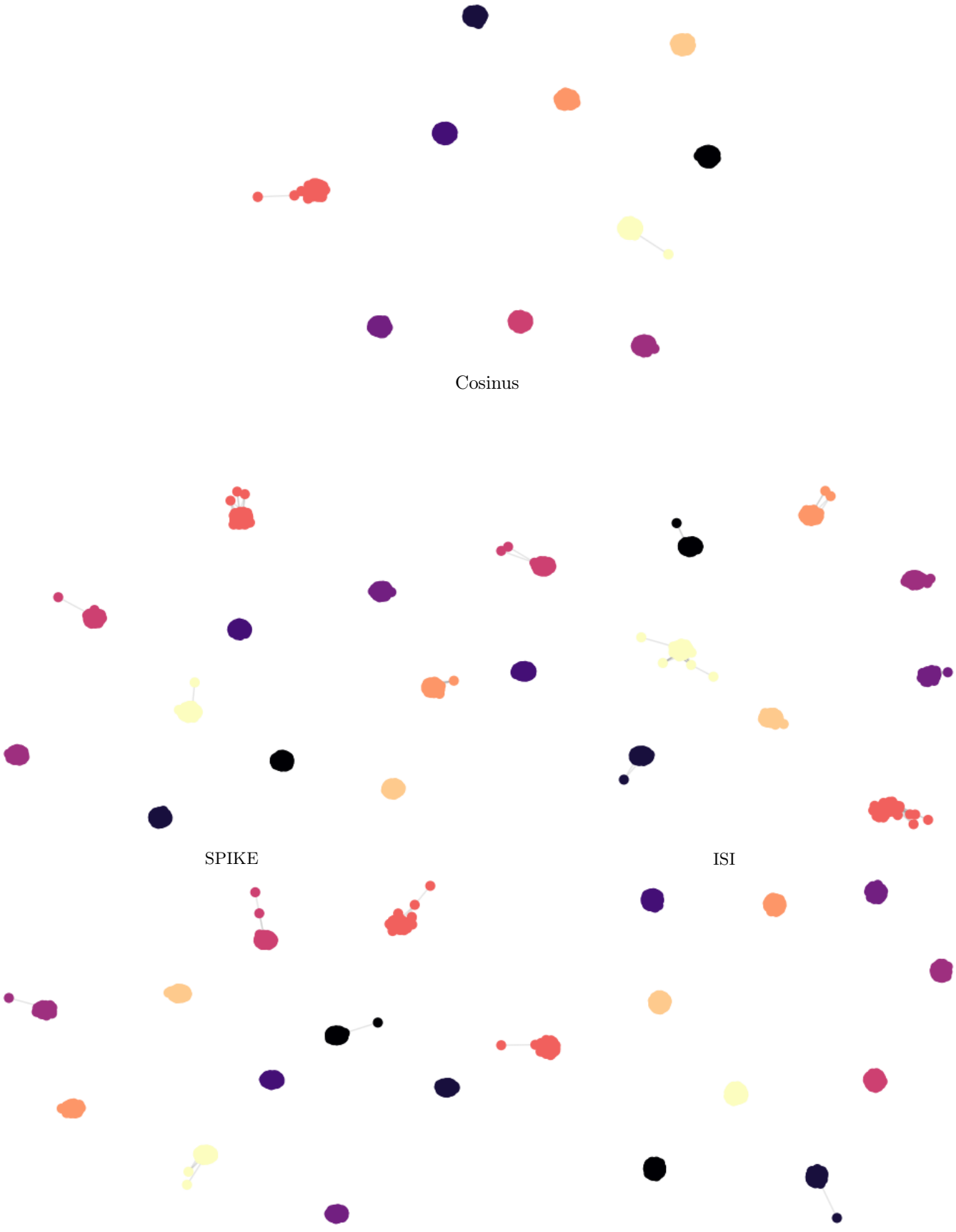


Fig. (10) **Visualization of the classifications obtained for the neurons of the graphs built from the different distance metrics.** The vertices of the neurons of the same class have the same color. The isolated vertices are not represented. For each of the 5 distance measures, the right number of communities (10) has been found. The obtained classes are rather balanced.

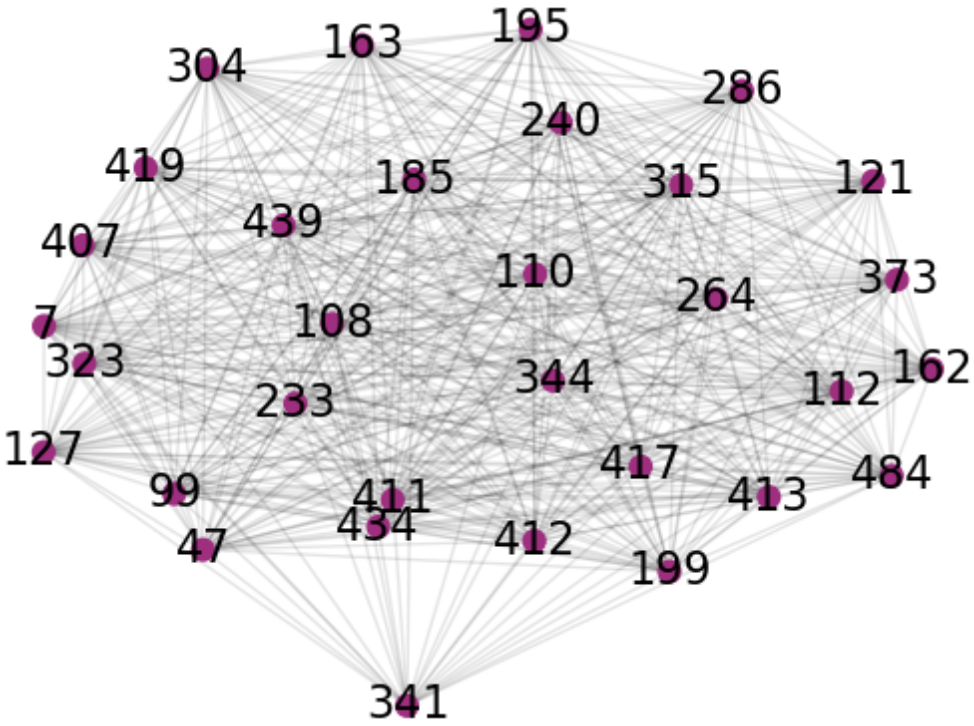


Fig. (11) **Zoom on a community obtained from the SPIKE similarity graph.** On each vertex is displayed the identifier of the corresponding neuron.

The number of intra-community edges is high. This is a particularly expected result since the low temporal resolution makes the neurons of the same community seem to activate at exactly the same time even if this is not necessarily the case. If the temporal resolution was better, we would perceive the temporal shifts between the signals of the neurons of the same community. Thus, some of the similarity estimates would be reduced - as would the number of edges - at least those estimated by distance measurements that are only sensitive to pulse times.

In order to have an idea of the quality of the classification obtained, let us observe the time series 110 and 185 which are supposed to be similar since they belong to the same community (see Fig. 11).

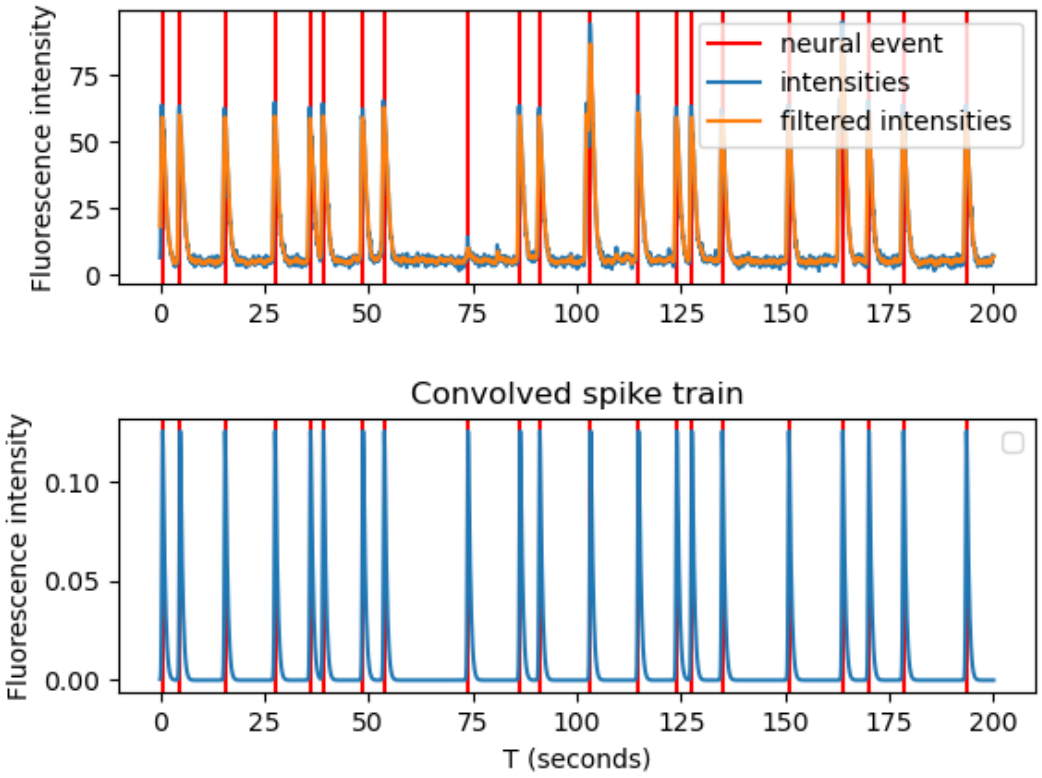


Fig. (12) **Activity of neuron 110.**

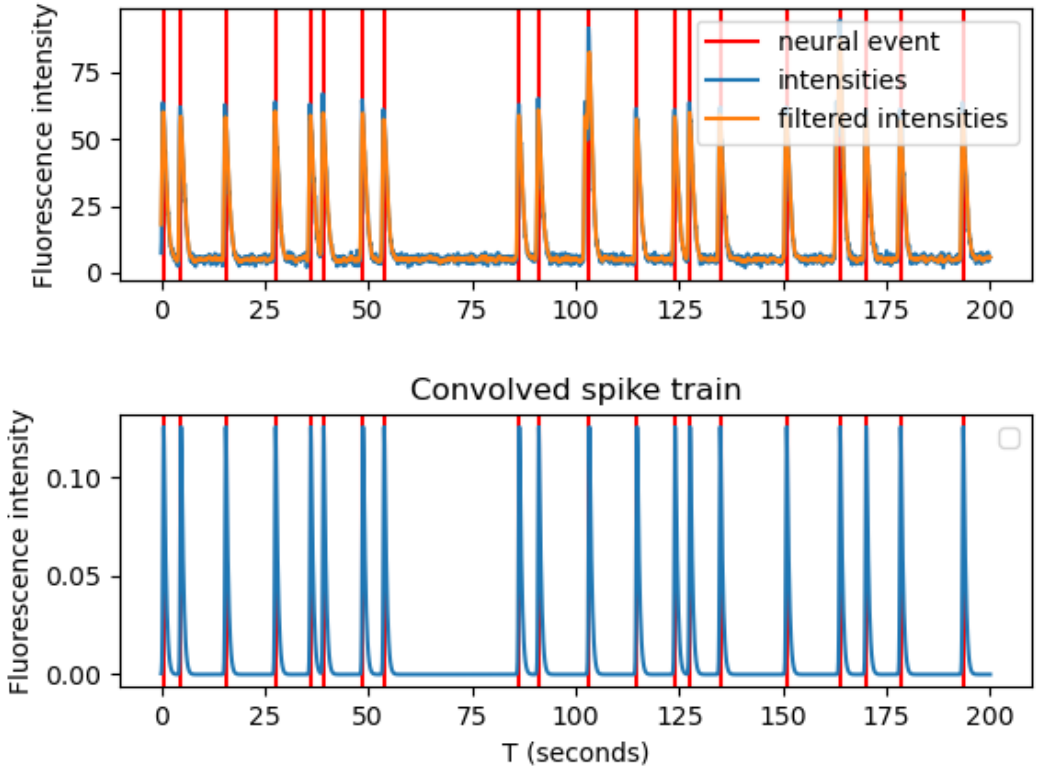


Fig. (13) **Activity of neuron 185.** It can be seen that neurons 110 and 185 show similar activity. The program was also tested on the simulation of the tracking of 500 neurons equidistributed in 5 communities. The 5 distance measurements led to a classification into 5 communities.

7 Evaluation of the robustness to noise of the different methods

The 5 distance metrics tested all led to a classification into 10 communities. In order to evaluate the robustness of these metrics, the signal-to-noise ratio must be decreased. We could redo the simulation of the 500 neurons tracking by specifying a higher noise level. Nevertheless, the naive method used for spike detection is very robust to noise when tested on synthetic data (as opposed to noise in real data). It is preferable to directly modify the spike sequences by removing spikes (to simulate false negatives) and adding spikes (to simulate false positives).

The mean error (« empirical risk ») is defined by :

$$R_{emp}(f) = \frac{1}{l} \sum_{i=1}^{l} \left| 10 - f\left(x_i\right) \right| .$$

where :

- 10 is the number of communities specified during the simulation of the 500 neurons tracking;
- l is the number of times the same proportion of spikes is removed;
- $f\left(x_i\right)$ is the number of communities obtained at the i^{th} iteration.

We note :

- FP, the number of false positives (the number of added spikes);
- FN, the number of false negatives (the number of added spikes);
- TP, the number of true positives (the number of spikes that were already present);
- TN, the number of true negatives (the number of frames where there were no spikes).

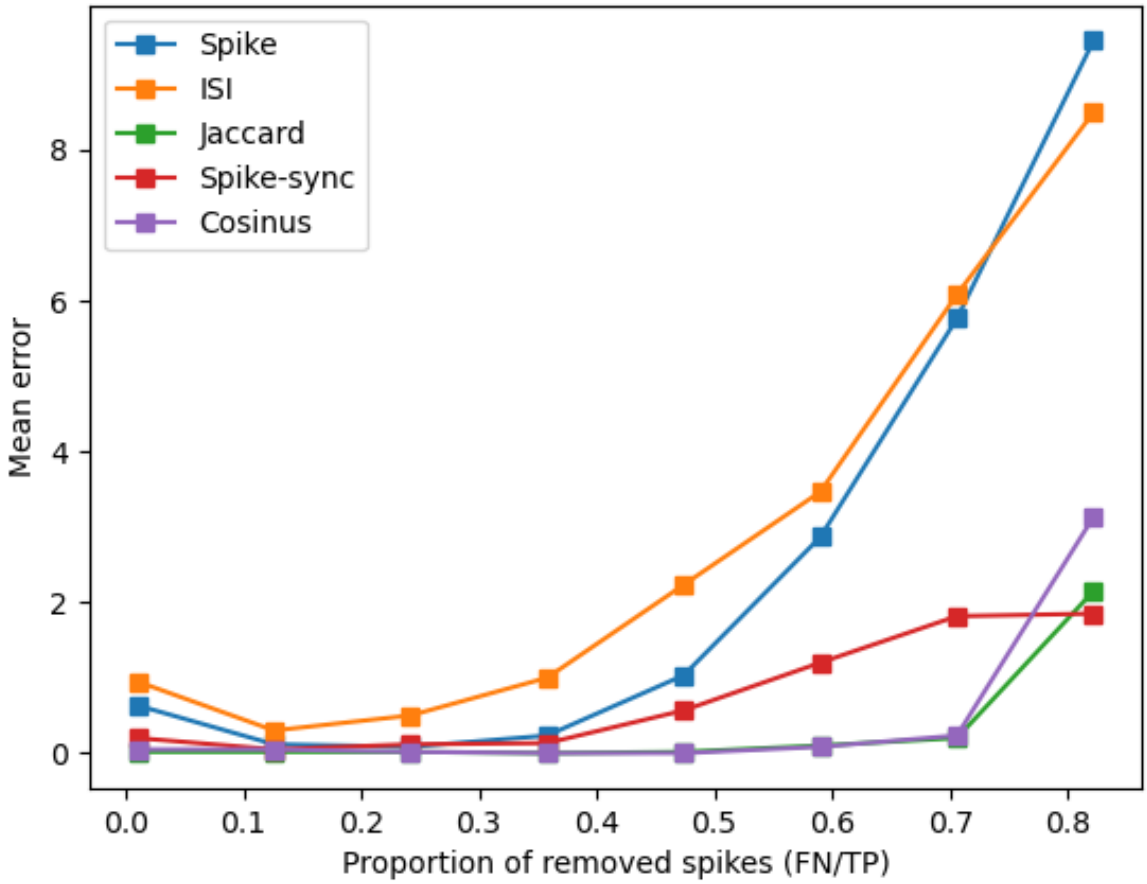


Fig. (14) **Mean error as a function of the proportion of randomly removed spikes.** Each point was obtained with $l = 60$. The uncertainty at each point is proportional to the binomial coefficient $\binom{TP}{FN}$.

The very long computation time of these simulations forced to base these tests (unreliable) only on the example of the 500 neurons tracking. Nevertheless, these tests suggest that the Jaccard index and the cosine similarity are the most robust to false negatives and, that the SPIKE, ISI, SPIKE-Synchronization distances are the most robust to false positives, for this data set.

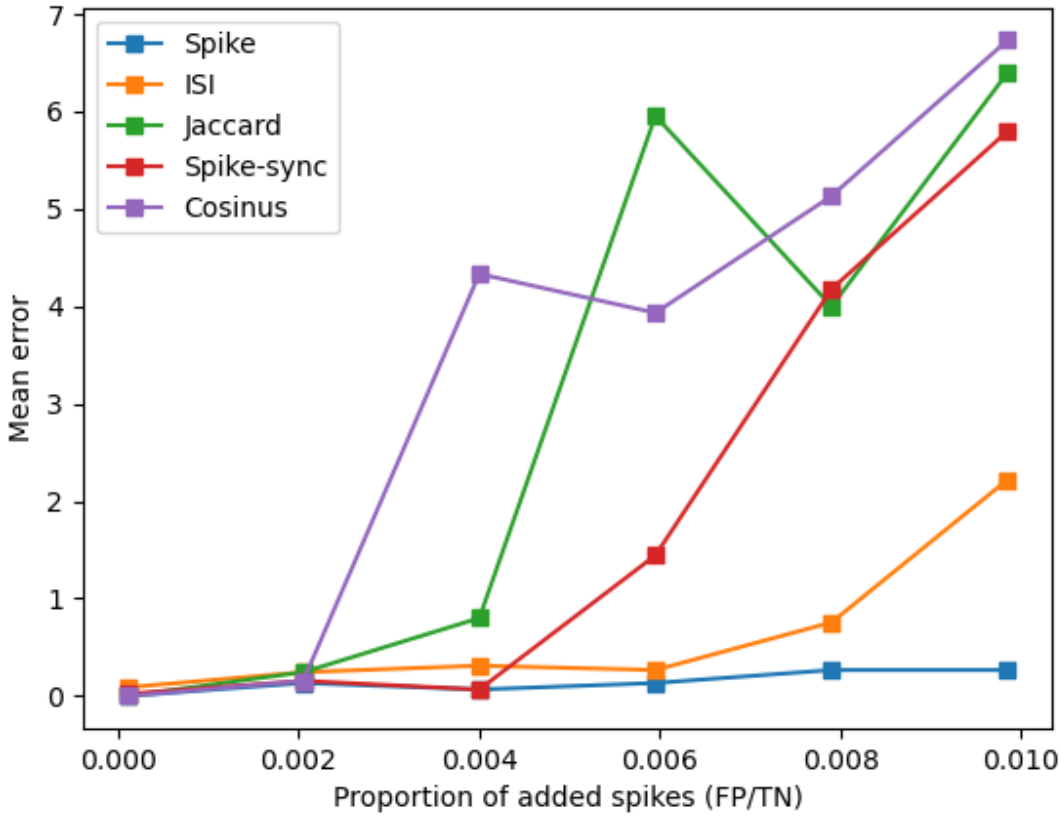


Fig. (15) **Mean error as a function of the proportion of randomly added spikes.** This test is very unreliable because TN is 978 545 and each point was obtained with only $l = 45$.

8 Conclusion and perspectives

The objective of this work was to propose an approach to extract neural ensembles from the activity of single neurons. For this purpose, a simple method for detecting action potentials has been tested. Then, a non-exhaustive review and a synthesis of similarity measures were performed. This second step made it possible to apply a community extraction algorithm on moderately noisy synthetic data. For each of the five distance measures used, the right number of communities were obtained. Finally, the robustness to noise of the distance measures was evaluated. Rather than increasing the noise directly on the signals, simulations of peak detection errors were performed.

In the continuity of this work, it would be a question of creating a simulator of neural ensemble activity in order to test methods offering the possibility of assigning each neuron to several communities.

By modeling the computational properties of neural ensembles, the methodological developments of this field participate, among others, in the search for mathematical principles that would explain the robustness and computational efficiency of biological neurons. Future discoveries will surely bring their contributions in many fields such as Artificial Intelligence.

References

[1] Nur Ahmadi, Timothy G. Constandinou, and Christos-Savvas Bouganis. "Estimation of neuronal firing rate using Bayesian Adaptive Kernel Smoother (BAKS)". In: *PLoS ONE* 13.11 (Nov. 21, 2018). ISSN: 1932-6203. DOI: 10.1371/journal.pone.0206794. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6248928/ (visited on 03/23/2021).

[2] Vincent D. Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.10 (Oct. 9, 2008), P10008. ISSN: 1742-5468. DOI: 10.1088/1742-5468/2008/10/P10008. arXiv: 0803.0476. URL: http://arxiv.org/abs/0803.0476 (visited on 04/26/2021).

[3] Daniel Chicharro, Thomas Kreuz, and Ralph G. Andrzejak. "What can spike train distances tell us about the neural code?" In: *Journal of Neuroscience Methods* 199.1 (July 15, 2011), pp. 146–165. ISSN: 1872-678X. DOI: 10.1016/j.jneumeth.2011.05.002.

[4] Thomas Deneux et al. "Accurate spike estimation from noisy calcium signals for ultrafast three-dimensional imaging of large neuronal populations in vivo". In: *Nature Communications* 7.1 (July 19, 2016). Number: 1 Publisher: Nature Publishing Group, p. 12190. ISSN: 2041-1723. DOI: 10.1038/ncomms12190. URL: https://www.nature.com/articles/ncomms12190 (visited on 04/26/2021).

[5] Thomas M. J. Fruchterman and Edward M. Reingold. "Graph drawing by force-directed placement". In: *Software: Practice and Experience* 21.11 (1991). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.4380211102, pp. 1129–1164. ISSN: 1097-024X. DOI: https://doi.org/10.1002/spe.4380211102. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380211102 (visited on 04/26/2021).

[6] Gareth James et al. "Moving Beyond Linearity". In: *An Introduction to Statistical Learning: with Applications in R*. Ed. by Gareth James et al. Springer Texts in Statistics. New York, NY: Springer, 2013, pp. 265–301. ISBN: 978-1-4614-7138-7. DOI: 10.1007/978-1-4614-7138-7_7. URL: https://doi.org/10.1007/978-1-4614-7138-7_7 (visited on 03/23/2021).

[7] RE Kass. "Adaptive spline smoothing of neural data". In: *Neural Signal Processing: Quantitative Analysis of Neural Activity. Society for Neuroscience* (2008), pp. 35–42.

[8] Thomas Kreuz, Mario Mulansky, and Nebojsa Bozanic. "SPIKY: a graphical user interface for monitoring spike train synchrony". In: *Journal of Neurophysiology* 113.9 (May 2015), pp. 3432–3445. ISSN: 0022-3077, 1522-1598. DOI: 10.1152/jn.00848.2014. URL: https://www.physiology.org/doi/10.1152/jn.00848.2014 (visited on 04/26/2021).

[9] Thomas Kreuz et al. "Measuring spike train synchrony". In: *Journal of Neuroscience Methods* 165.1 (Sept. 15, 2007), pp. 151–161. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2007.05.031.

[10] Thomas Kreuz et al. "Monitoring spike train synchrony". In: *Journal of Neurophysiology* 109.5 (Mar. 2013), pp. 1457–1472. ISSN: 1522-1598. DOI: 10.1152/jn.00873.2012.

[11] Thibault Lagache et al. "Robust single neuron tracking of calcium imaging in behaving Hydra". In: *bioRxiv* (July 6, 2020). Publisher: Cold Spring Harbor Laboratory Section: New Results, p. 2020.06.22.165696. DOI: 10.1101/2020.06.22.165696. URL: https://www.biorxiv.org/content/10.1101/2020.06.22.165696v2 (visited on 03/23/2021).

[12] M. Mulansky et al. "A guide to time-resolved and parameter-free measures of spike train synchrony". In: *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. 2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP). June 2015, pp. 1–8. DOI: 10.1109/EBCCSP.2015.7300693.

[13] Mario Mulansky and Thomas Kreuz. "PySpike—A Python library for analyzing spike train synchrony". In: *SoftwareX* 5 (Jan. 1, 2016), pp. 183–189. ISSN: 2352-7110. DOI: 10.1016/j.softx.2016.07.006. URL: https://www.sciencedirect.com/science/article/pii/S2352711016300255 (visited on 03/23/2021).

[14] Eftychios A. Pnevmatikakis et al. "Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data". eng. In: *Neuron* 89.2 (Jan. 2016), pp. 285–299. ISSN: 1097-4199. DOI: 10.1016/j.neuron.2015.11.037.

[15] Jean-Christophe Olivo-Marin Thibault Lagache Samuel Kubler Suvadip Mukherjee. "A robust and versatile framework to compare spike detection methods in calcium imaging of neuronal activity". preprint. 2021.

[16] Eero Satuvuori and Thomas Kreuz. "Which spike train distance is most suitable for distinguishing rate and temporal coding?" In: *Journal of Neuroscience Methods* 299 (Apr. 1, 2018), pp. 22–33. ISSN: 0165-0270. DOI: 10.1016/j.jneumeth.2018.02.009. URL: https://www.sciencedirect.com/science/article/pii/S0165027018300372 (visited on 03/23/2021).

[17] Jonathan D. Victor. "Spike Train Distance". In: *Encyclopedia of Computational Neuroscience*. Ed. by Dieter Jaeger and Ranu Jung. New York, NY: Springer, 2014, pp. 1–8. ISBN: 978-1-4614-7320-6. DOI: 10.1007/978-1-4614-7320-6_409-1. URL: https://doi.org/10.1007/978-1-4614-7320-6_409-1 (visited on 03/23/2021).