Noise correlations and template matching on neural recordings

by

Miguel I Veloso O'Donell

to obtain the degree of Master of Science

at the Delft University of Technology

January 2018

Student number: 4513274

Supervisor Dr. Felix Franke, ETH Zurich

Supervisor Prof. Dr. ir. Wouter Serdijn, TU Delft

Acknowledgment

First of all I would like to thank my parents and my brother for all their support during my master studies and master thesis. Thanks to them I was able to go abroad for studying and making a new life. All the love, tools and support they have given me has no measure. I will be always in debt with the three of them.

The people that shared with me every day of my first year in Delft, my dear house 7 in Aan't Verlaat, thank you guys for every moment, every dinner, and all the laughs.

Special thanks to Gianmarco for being such a good friend and such a bad influence.

Thanks also to Severine, Felix, and Joaquín for being the best friends and companion not only in The Netherlands, but also in Switzerland.

My thesis period in Switzerland would have been completely boring without Anja. Thank you for giving me the perfect working-spare-time balance. Thanks for being the biggest support during the thesis. Thanks also for every moment in Basel.

Last but not least, I want to thank Felix Franke for letting me work with him on this project. Thank you for your advice, support, and input during the thesis. It was an enriching experience and an honor to work with you.

Table of Contents

Introduction

To study how information is processed in the brain, we need to look at the activity of brain's main cells, the neurons [1]. Neurons process information in groups or ensembles [2], called neural networks, so we need to not only be able to determine the contribution of individual neurons, but also establish correlations among them to understand their connectivity and function [1]. For this we need three components, first to simultaneously record the activity from most of the neurons in the neural network of interest [1] [3] [4]. Second, to stimulate or inhibit specific neurons within the network in order to map its input-output relation [1] [3] [4] [5], in a reverse engineering way. Third, a step that allows us to extract the neural activity from the recordings and process it by mean of signal processing and data analysis techniques [3] [6] [7] [8]. These steps of extracting and processing the neural activity provide the means to later determine the contribution of neurons alone, establish correlations and connectivity, calculate, in some cases, the right stimulation, and study the network's function. Figure 1 shows a diagram of a neural network, the three components, and their interaction.

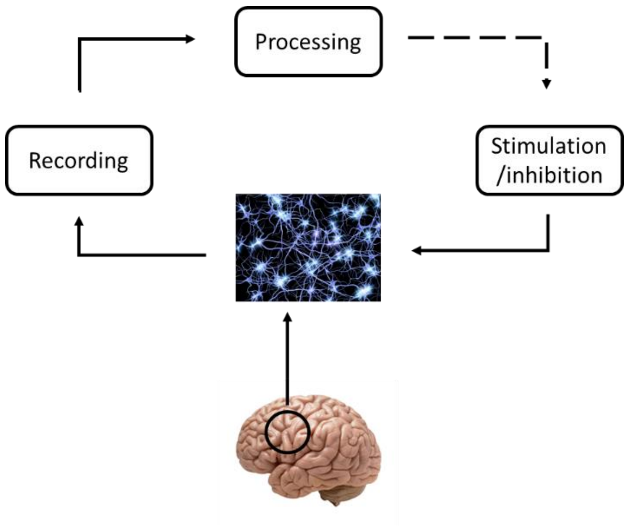


Figure 1. Diagram sketching a neural network from the brain and how the three components: recording, processing and stimulation/inhibition interact with each other in order to study the network. The dashed line between processing and stimulation/inhibition means there is not always a relation between both.

Recording

The neural activity we want to record is the action potential (AP), shown in Figure 2. There have been different techniques that enable the recording of neurons' APs [1] [3] [4] [6] [9]. These techniques range from direct measurements of tens of neurons such as patch-clamping [10] to indirect measurements that can include thousands of neurons, like electroencephalography (EEG) [4].

In this work, however, we are going to focus on the CMOS-based high density microelectrode array (HDMEA) device [1], shown in Figure 3, developed at ETH Zurich. This device falls within the category of multi-microelectrode designs, has an active area of 3.85 x 2.10 mm^2 where brain or retina slices are placed, and possess 26,400 platinum microelectrodes, from which 1024 can be selected for recording. The electrodes' signals from the CMOS HDMEA are amplified, bandpass filtered, and sampled at 20 kHz with a 10 bit analog-to digital converter. The digital signals are sent to a Field Programmable Gate Array (FPGA) and then to a computer for processing. These capabilities allow the implementation of in-vitro extracellular electrophysiology of neural networks containing thousands of neurons.

Extracellular electrophysiology and multelectrode spikes

Extracellular electrophysiology, one of the leading techniques in neuroscience [6], is the measure of the electrical potentials in the extracellular medium of a group of neurons via microelectrodes. The measures contain noise coming from different sources [4] and the extracellular representation of the APs: the spikes, as shown in Figure 2.

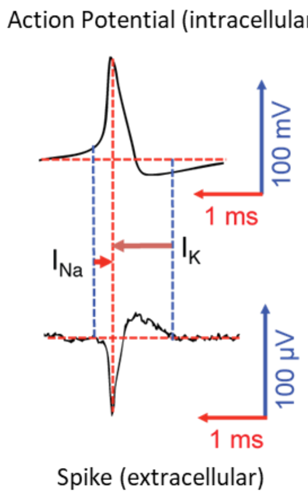


Figure 2. The action potential and its extracellular representation: the spike [11]

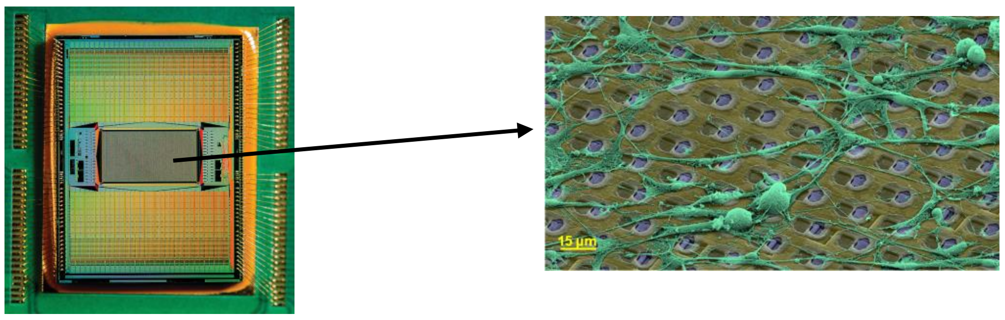


Figure 3. High resolution CMOS HDMEA platform [1] (left), and image of neurons on top of it [11] (right)

Since we are dealing with multiple electrodes to record the neural network under study, whenever a neuron fires, generating an AP, the spike is not only measured in one electrode but in many of them surrounding the neuron. The electrodes closer to the neuron's soma would present a larger spike amplitude, while the further away the electrode is the lower the spike amplitude. We therefore say, we have a multielectrode spike representation of the neurons. Additionally, if we average the spikes of a neuron in every electrode we find its multielectrode spike waveform, called the neuron's footprint, and shown in Figure 4.

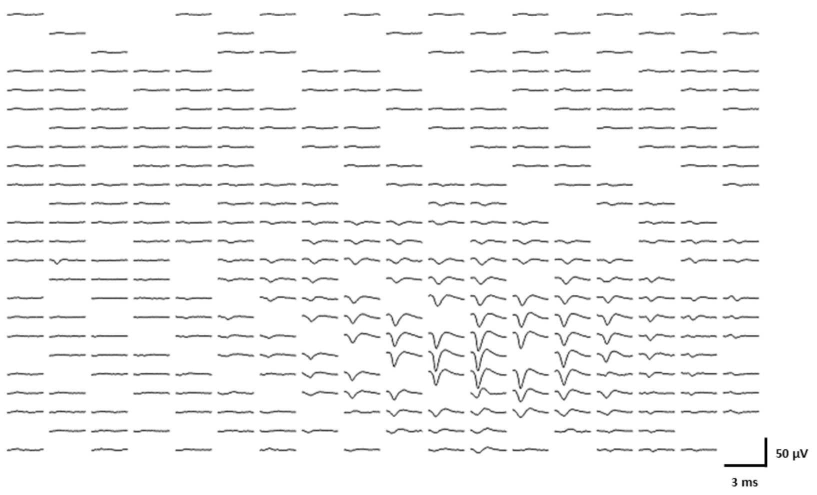


Figure 4. Footprint of a neuron on top of a group of approximately 400 electrodes on the CMOS HDMEA. From the spike amplitude it is possible to infer the position of the neuron. The spikes seem noise free thanks to the averaging used to compute the footprint [11]

Stimulation

For stimulating or inhibiting the neural network there are also different approaches [3] [4] [9]. Chemical, optical, and electrical techniques have been develop in order to manipulate the activity of the neurons in a network. Even though we are not going to talk about any stimulation or inhibition, it is important to mention that the CMOS HDMEA device let us to set up to 32 channels to electrically stimulate neurons in the network under study [1]. This characteristic gives us the possibility to apply the stimulation based on ongoing recordings in a closed-loop fashion, allowing us to establish and study the input-output relations mentioned before.

Processing

Finally, we need a processing step to extract the neurons' spikes from the recorded data. For extracellular electrophysiological recordings the processing consists of two steps: finding the spikes within the recording, and second, figuring out to which neurons those spikes belong [6] [7]. We can summarize the two steps as: detection and classification of spikes. In neuroscience the standard method for spike detection and classification is called spike sorting [3] [6] [7] [8] [12]. With spike sorting we are able to reconstruct the spike train of each neuron, i.e. a time series that tells us when the neuron actually spiked.

Spike sorting

Spike sorting is not an easy task for several reasons [8]. First, the recorded signals always present background noise coming from the electronic circuitry of the recording device, as well as from the influence of attenuated spikes from neurons far away from the electrode of interest [4]. Second, spikes vary in shape, not only among different neurons, but also from the same neuron along time, presenting bursts in some case, i.e. periods of high spiking frequencies. Third, spikes from different neurons can overlap, say, if neurons close to each other spike at the same time, then the recorded spike waveform in the electrodes close to the neurons would be the superposition of the spikes, leading to a larger, smaller or even unrecognizable waveforms [8]. Fourth, the position between the neurons and the electrodes can change over time, changing the shape of the spike waveform recorded on the microelectrodes. All these problems, noise, spike variations, overlaps, and electrode-neuron drift affect the detection and classification task of spike sorting [8]. These problems lead to not only the detection of noise as spikes, known as false positives, or the omission of spikes, named false negatives, but also to classification errors, i.e. misclassification of spikes to their respective neurons. Furthermore, since the spikes are digitalized in order to be processed, this digitalization causes problems when classifying the spikes into the different neurons due to jittering [8]. Figure 5 presents a chunk of a recording in the CMOS HDMEA, where five adjacent electrodes presenting different spike waveforms, background noise, and overlaps can be observed.

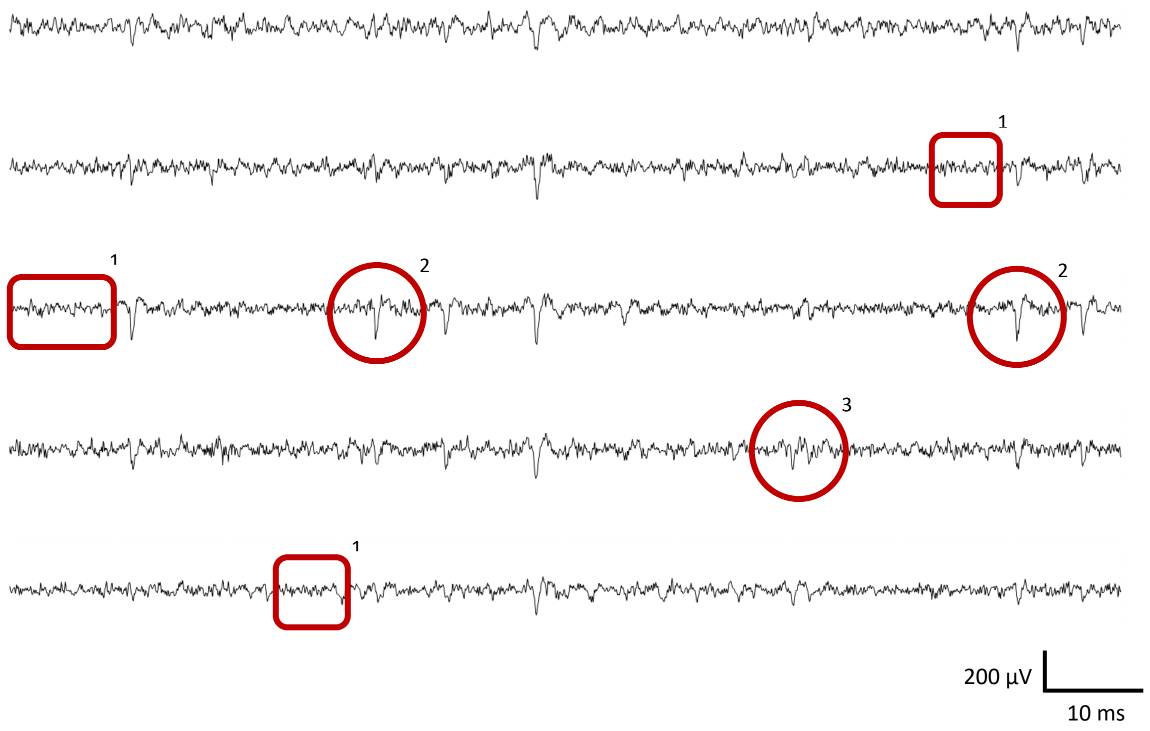


Figure 5. Data recorded with the CMOS HDMEA from a retina slice. The red areas represent: 1: background noise. 2: Two spikes from the same neuron presenting different waveforms. 3: A possible overlap of two spikes

The general procedure of spike sorting consists of a series of steps [7] [8] [12]. First, the recorded data is bandpass filtered. Second, when the filtered data crosses a, predefined, voltage threshold a spike is detected. Third, a number of data points around the detected spike are considered part of the spike waveform and extracted from the recording. From those extracted data points a number of features are extracted. Fourth, the extracted features are used to classify the spikes assigning them to their putative neurons by means of a clustering algorithm. The aforementioned steps are presented in Figure 6.

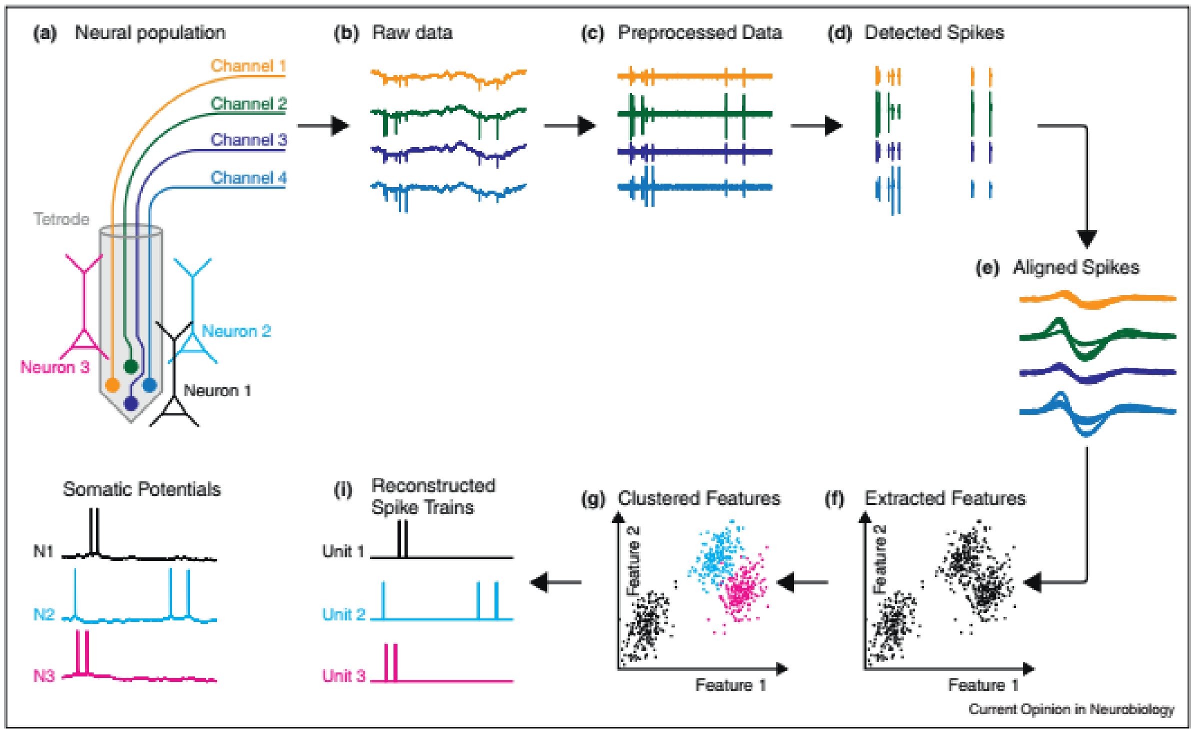


Figure 6. Overview of the spike sorting process [12]. In (a) the data is recorded with the electrodes. The raw data in (b) is bandpass filtered and (c) is obtained. Threshold crossing is done in (d) to detect the spikes, which are then extracted and aligned in (e). The relevant features of the spikes are extracted in (f) and used for clustering in (g). Finally in (i) the spikes of the different clusters are used to classify the neurons and reconstruct the spike trains

The main disadvantages with the classical spike sorting approach are that it is not scalable to process thousands of neurons, and present an unsupervised step, clustering, not suitable for real-time implementations [12] [13] [14].

Since we are dealing with the CMOS HDMEA device to record and stimulate neurons. We seek a processing method that enable not only the recording of thousands of neurons at the same time, but also, closed-loop experiments involving electrical stimulation on real-time.

Template matching

One way to overcome the two aforementioned disadvantages of the classical spike sorting approach is by implementing a template matching based algorithm to detect and classify the spikes [8]. In template matching, a known signal, called the template, is to be found within the noise [13]. In our case, if we would have the averaged multielectrode spike waveform of every neuron, i.e. its footprint, we could use them as templates to match the spikes within the recording, solving both detection and classification.

However, to find the footprint of every neuron we still need to apply spike sorting. After spike sorting, the footprint of each neuron is computed by taking the centroid of each cluster, i.e. its mean. Applying template matching seems redundant because the templates are made after the clustering step of spike sorting, which is basically the one we want to avoid. However, if a spike sorting procedure is applied at the beginning of the recorded data to create the clusters and obtain the templates, then these templates can be used to detect and classify spikes along the rest of the recording. This won't require any supervision step, and will be much faster than the classical spike sorting approach.

Bayes optimal template matching (BOTM)

In this work we are going to focus on the template matching based algorithm for spike detection and classification presented by Franke et. al. [13], the Bayes optimal template matching (BOTM). This algorithm is scalable to thousands of neurons, parallelizable, optimal under certain considerations [13], and finally, faster than the common spike sorting [13] due to the lack of a clustering step. The BOTM equation can be summarized as follows:

$$d_i(t) = X(t)^T C^{-1} \xi_i - c_i \qquad \text{Equation 1}$$

Where d_i is called the discriminant function of neuron i, ξ_i is the template for that neuron, c_i is a constant that depends solely on the neuron, $X(t)^T$ is the transpose of the multichannel recording, and C^{-1} is the inverse of the noise covariance matrix [13] [15]. The BOTM algorithm computes the discriminant function of every neuron along the data and compares them with a threshold for spike detection, this threshold is defined by the noise's discriminant function [13]. Once a spike is detected, the algorithm compares the discriminant functions of the different neurons, assigning the detected spike to the neuron with the largest value.

BOTM and electrode correlations

After introducing the three components we use for studying neural networks: recording, stimulation and processing. And stating we are going to focus on the processing algorithm, and specifically on the BOTM, we can discuss further the motivation of this work.

Simplify the noise covariance matrix

The BOTM algorithm requires us to compute the noise covariance matrix C [13] [15], and its inverse C^{-1}. This matrix is computed by calculating the correlation between electrodes' signals where only noise, and no spikes, is present. The size of the matrix is proportional to the square number of electrodes times the squared length of a spike in samples. Therefore, since we are dealing with 1024 readout channels and a spike is approximately 50 samples long (considering the 20 kHz sampling frequency), the matrix would have around $2{,}5 \times 10^9$ components, making it too large to compute and handle.

The first aim of this work is to simplify the noise covariance matrix to make it easier to compute and handle.

The noise covariance matrix C presents at the same time: spatial noise correlations among electrodes, temporal noise correlations of electrodes with themselves, and spatio-temporal noise correlations among different electrodes. The inverse of the noise covariance matrix, as presented in Equation 1, is working as a whitening transformation [15] [16] [17], whitening the data $X(t)^T$.

Here we consider separating the spatial, temporal and spatio-temporal correlations. We make two smaller matrices as in [16], one containing the spatial noise correlations and the other containing the temporal noise correlations. With this approach we evaluate the BOTM's performance on detection and classification on three scenarios:

- First, considering only spatial correlations among the electrodes and neither temporal nor spatio-temporal, applying what is known as spatial noise whitening.
- Second, considering only temporal and neither spatial nor spatio-temporal correlations, called temporal noise whitening.
- Third, taking both spatial and temporal correlations into account, by applying one whitening after the other, called spatial-and-temporal noise whitening.

With these approaches we can build simpler matrices which are smaller and easier to handle, but also evaluate the contribution of each of these correlations, and whitenings, to the performance of the BOTM algorithm.

Data and signal correlations

The second aim would be to look at the other two correlations we find among electrodes: spike correlations and signal correlations. The first one is the correlation between electrodes just taking the spikes into account, and the second case taking both noise and spikes all together into account to compute the correlation of the signals between electrodes. Even though we were not able to evaluate spike correlations, we show how the signal correlations affect the BOTM performance.

The relation between the three correlations we find among the electrodes, as presented in [18], is:

$$R = H + C \qquad \textit{Equation 2}$$

Where R is the signal covariance matrix and contains the spatial signal correlations among electrodes, temporal signal correlations of electrodes with themselves, and spatio-temporal signal correlations among different electrodes. H is the spike covariance matrix, containing spatial spike correlations among electrodes, temporal spike correlations of electrodes with themselves, and spatio-temporal spike correlations among different electrodes. And C is previously mentioned noise covariance matrix. For the evaluation of R we again divide the correlations into spatial, temporal and spatial-and-temporal in order to be able to handle the matrices.

By looking at R and H we would like to know if there is a way to utilize the correlations among electrodes, as the noise covariance matrix C is used in Equation 1, to generate optimal filters from every template, i.e. filters that suppress every template except for one [18].

Organization of the rest of this work

In the next section, Bayes optimal template matching (BOTM), we introduce the details of the BOTM algorithm and the noise covariance matrix involved on it. Then in the Correlations and whitening section we introduce the importance of taking the correlations, present in the noise covariance matrix, into account, and the concept of data whitening. We also introduce the effects of spatial and temporal whitening. In the next section, Data and performance measures, we discuss about how to build the data to evaluate the BOTM and how to assess its performance. Then we enter the results of this work. We first show the effects of whitening on the BOTM applied to a small group of electrodes in the section Comparing different whitening approaches (Part 1). After this we make a break in the results and show, in the section called The problem of high-frequency components in inserted footprints, how we got erroneous conclusion about the performance of the BOTM algorithm, and how the problem behind this can affect any template matching based algorithm. In the next section called Comparing different whitening approaches (Part 2), we show the performance of the BOTM with different whitening strategies and over a group of 289 electrodes. To wrap up, in the last chapter: Conclusions, outcomes and future work, we discuss about the remarks and conclusions we came up within this work.

Bayes optimal template matching (BOTM)

We are going to focus now on the Bayes optimal template matching algorithm (BOTM) presented in [13]. This algorithm presents the optimal way to perform spike detection and classification by a template matching process in an extracellular electrophysiological recording. The template matching process is a filtering operation, where the filter proposed is derived linearly from the templates, and it is the matched filter that at the same time increases the signal-to-noise ratio (SNR) and leads to an optimal classification in a Bayesian sense. The BOTM solves both detection and classification problem at once. However, it is important to mention that in order to apply the algorithm, as in any template matching based algorithm, the number of neurons and their templates have to be known in advance.

There are two definitions we have to understand before going on to understand the BOTM equations: the templates, and the noise covariance matrix. The templates ξ represent the mean of multivariate normally distributed clusters, and have the following structure: the template for neuron i is $\xi_i = [{\xi_{i,1}}^T, {\xi_{i,2}}^T, \dots, {\xi_{i,M}}^T]^T$ where $\xi_{i,b}$ is the spike waveform for neuron i on electrode b. Since every spike is digitalized by the CMOS HDMEA platform, and every spike waveform in every electrode is L samples long, the template is a vector of length LM, where M is the number of electrodes. The noise covariance matrix among the M electrodes is represented by C. This matrix contains noise correlations among the electrodes. These noise correlations come in the form of noise spatial correlations among channels, noise temporal correlations of individual channels with themselves, i.e. noise autocorrelation of every channel, and spatio-temporal noise correlations between channels, i.e. noise cross-correlations among channels.

Now we can elaborate on the details of the BOTM equations. In [13], after analyzing the spike classification problem as a multi-class classification problem, the following function is obtained:

$$d_i(t) = X(t)^T C^{-1} \xi_i - \frac{1}{2} {\xi_i}^T C^{-1} \xi_i + \ln(p(i)) \qquad \textit{Equation 3}$$

Where the sub-index i stands for neuron i, d_i is called the discriminant function of the corresponding neuron, X^T represents the transpose of the data recorded by the M electrodes, ξ_i the template of neuron i, C^{-1} the inverse of the noise covariance matrix, ${\xi_i}^T C^{-1} \xi_i$ represents the energy of the template, and $p(i)$ is the prior firing probability of the neuron. This is the same function used in linear discriminant analysis (LDA) [19].

Taking the matched filter defined for maximizing SNR in the context of spike sorting [20]:

$$f = \frac{C^{-1}\xi}{\beta} \qquad \textit{Equation 4}$$

Where β is a normalization constant. It becomes clear that Equation 3 is a filtering operation of the data with the matched filter.

In order to detect and classify spikes in the data, the discriminant function d_i for every neuron is calculated. When the discriminant function crosses a certain detection threshold, a spike is detected, then the discriminant function values of all the neurons are compared and the spike is assigned to the neuron with the largest value. It is important to mention that the BOTM equation provides the detection threshold in an analytic way, being equal to the discriminant function of the noise template, obtained by setting the template to zero in Equation 3:

$$d_n(t) = X(t)^T C^{-1} 0 - \frac{1}{2} 0^T C^{-1} 0 + \ln(p(n)) \qquad \text{Equation 5}$$

One the of the important features of the BOTM algorithm is that it is completely parallelizable, as the discriminant functions are all independent from each other and can be computed separately. Moreover, the computational time of the algorithm is reduced to just the filtering operation, since the second and third terms are constants for every neuron [13]. The BOTM equation can be finally rewritten in the following compact form:

$$d_i(t) = X(t)^T C^{-1} \xi_i + c_i \qquad \text{Equation 6}$$

This simplification is possible because the energy and the prior probability of spike of every neuron are both constant and independent among neurons.

Figure 9 shows, in a very simple example of one electrode and two neurons, how the BOTM detects and classifies spikes to their putative neurons.

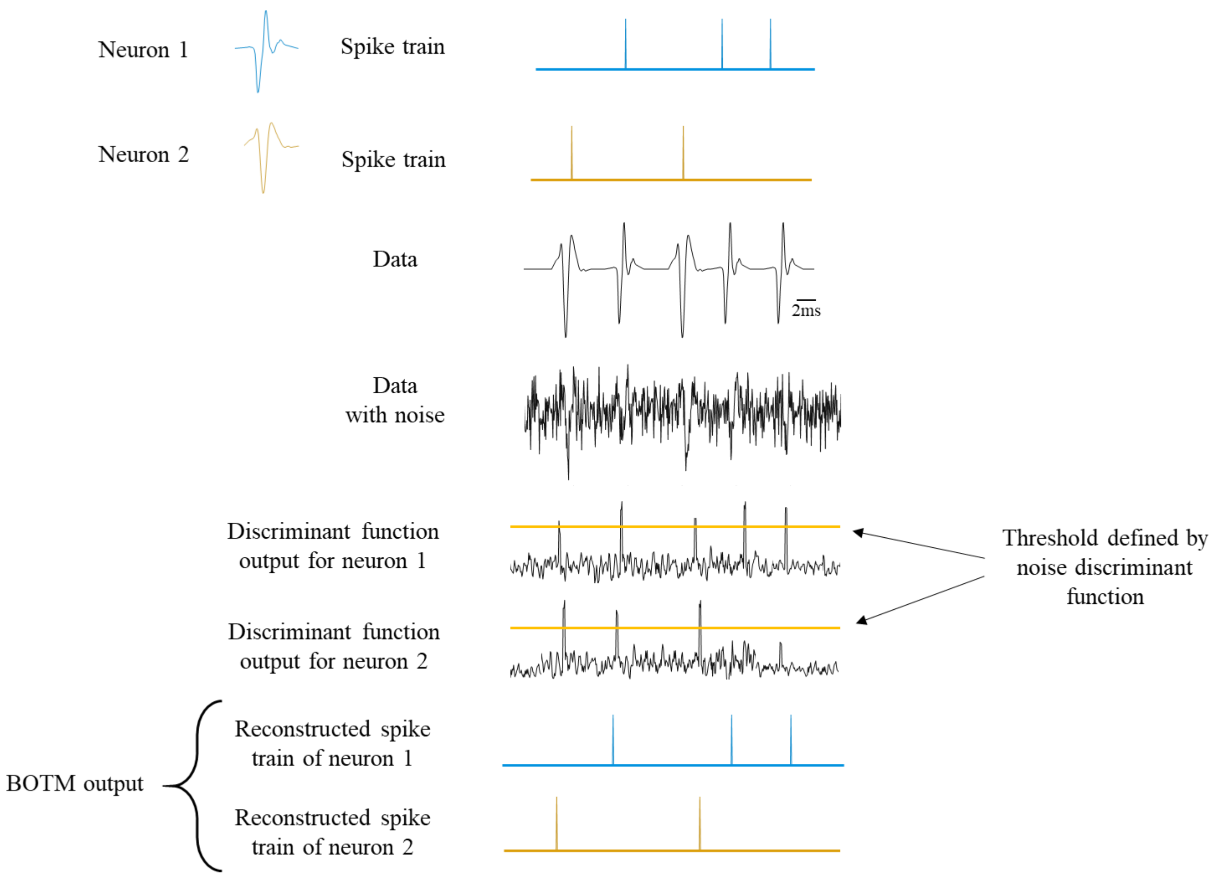

Neuron 1
Spike train
Neuron 2
Spike train
Data
2ms
Data with noise
Discriminant function output for neuron 1
Discriminant function output for neuron 2
Threshold defined by noise discriminant function
BOTM output
Reconstructed spike train of neuron 1
Reconstructed spike train of neuron 2

Figure 7. Data created by two spiking neurons and noise. Here we show how the BOTM algorithm is able to detect the spikes within the noise and assign them to their correct putative neurons

Noise covariance matrix

The noise covariance matrix is extensively used in the spike sorting literature [15] [16] [21] [22]. It has a special block structure:

$$C = \begin{matrix} C_{1,1} & \cdots & C_{M,1} \\ \vdots & \ddots & \vdots \\ C_{1,M} & \cdots & C_{M,M} \end{matrix} \qquad \text{equation 7}$$

Where the $C_{i,k}$ block is the noise cross-correlation matrix between electrodes i and k. Each of these blocks is a $T \times T$ matrix, where T is a time lag equal to the defined spike length, with a Toeplitz structure [23], meaning that its first row contains the noise cross-correlation between electrodes i and k, its second row contains a shifted version of the first, the next row a shifted version of the previous one and so on. It is important to mention that, as the name implies, the noise cross-correlation between electrodes is calculated along chunks of the recording where there is just noise and no spikes detected as shown in Figure 8. The noise covariance matrix C has a symmetric structure with $C_{i,i} = {C_{i,i}}^T$ and $C_{i,k} = {C_{k,i}}^T$. Since every block is $T \times T$ matrix and there is one block for every pair of electrodes, the dimensionality of C is $TM \times TM$, where, has before M is the number of electrodes used.

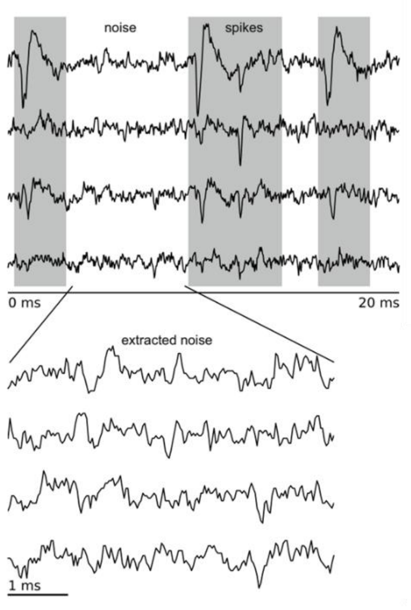


Figure 8. Voltage traces of four electrodes. Noise and spikes can be clearly identify. Noise chunks are extracted to compute noise correlations [24]

This noise covariance matrix presents at the same time: spatial noise correlations among electrodes, temporal noise correlations of electrodes with themselves, and spatio-temporal noise correlations among different electrodes.

Correlations and whitening

To understand the effect of the noise correlations and the noise covariance matrix over the BOTM, we can consider again a simple case were we have a recording with only two spiking neurons and their respective templates. If the data were noise free, the value of the discriminant function for each neuron considered in the BOTM algorithm would be always the same. In this case finding a boundary to separate the BOTM output for every neuron is trivial, as shown in Figure 9 (a), and spike detection and classification would be straight forward. However, due to the noise present in the data, $X(t)$, the discriminant functions have a distribution of values around the ideal noise free value, as shown in Figure 9 (b). Here, finding a threshold for detection and a boundary to separate the two distributions for spike classification is not straight forward. Nonetheless, since the noise present in the data is taken into account when deriving the BOTM, as having a covariance matrix C among electrodes [13], the final BOTM equation gives us the best way to separate the discriminant function values even in the presence of noise.

We can summarize the best way to separate both discriminant function values by:

1. Multiplying data and templates by the inverse of the noise covariance matrix C^{-1}, as the first term in Equation 6
2. Adding the respective constant for each neuron, as the second term in Equation 6
3. Taking the identity line that divides the plane by two halves comparing the value of the two discriminant functions. This is equivalent to saying that whenever a spike is detected, we assign it to the neuron with the largest discriminant function

Understanding the second point mentioned is straight forward: when we add the constant we move the discriminant function value of every neuron away from the noise threshold. The first bullet point however, requires a more detailed explanation.

The inverse of the noise covariance matrix can be expressed as a product of two upper triangular matrices, as follows:

$$C^{-1} = U^T U \qquad \textit{Equation 8}$$

By definition the matrix U is a whitening matrix [17]. Therefore, when multiplying C^{-1} in Equation 6 we are applying a whitening transformation to the data as $X(t)^T U^T$ [15] [16], and reshaping the templates as $U\xi_i$. Data whitening is important for sphering the noise distributions [15], as seen in Figure 9 (c), ensuring the optimal classification of spikes [13]. It is also important to apply the same transformation to the templates to preserve their matching with the spike waveforms in the data.

The whitening transform sets the variance of the noise on every electrode to one and eliminates correlations among every pair of electrodes, i.e. decorrelate the noise among electrodes. With the reshaped distributions the right boundary is the identity line that provides the best separation between every pair of neurons. With this boundary we are able to assign the detected spikes to their corresponding neurons in an optimal way.

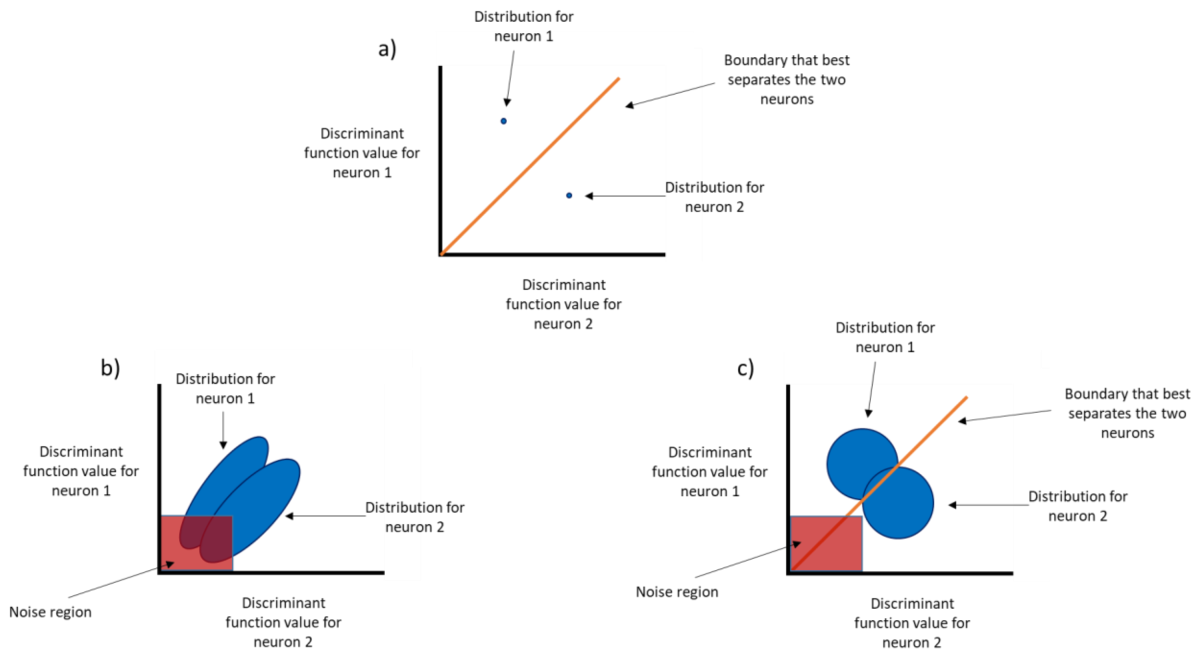

a)
Distribution for neuron 1
Boundary that best separates the two neurons
Discriminant function value for neuron 1
Distribution for neuron 2
Discriminant function value for neuron 2
b)
Distribution for neuron 1
Discriminant function value for neuron 1
Distribution for neuron 2
Noise region
Discriminant function value for neuron 2
c)
Distribution for neuron 1
Boundary that best separates the two neurons
Discriminant function value for neuron 1
Distribution for neuron 2
Noise region
Discriminant function value for neuron 2

Figure 9. (a) Decision boundary and discriminant function values in the ideal case of recording without noise. (b) Distribution of discriminant function values considering noise in the recording. Notice that the discriminant function values become a distribution around the ideal noise free case. The red square close to the origin represents the values that are consider as noise. (c) Decision boundary and distribution of discriminant function values when applying the BOTM equation, it is clear that the whitening transformation is sphering the distributions, and the constant added is moving them away from the noise region

Despite the fact the BOTM equation provides an implicit whitening transformation that enables the decorrelation needed for the classification of spikes, we previously discussed the difficulty in computing and handling the full noise covariance matrix C. So we implement an approach of dividing the full noise covariance matrix in two simpler matrices, applying two whitenings: a spatial whitening, and a temporal whitening. Now we introduce what is the effect of these two whitenings alone.

Spatial noise whitening

Spatial noise whitening is a transformation that sets the noise variance over every electrode to the unity and the cross-correlation between every pair of electrodes to zero, i.e. reshapes the noise distributions into normal distributions. The noise correlations among electrodes are used to compute the spatial noise whitening transformation. Figure 10 shows how the noise correlation between two electrodes can be reshaped by a spatial noise whitening transformation in order to have a normal distributed noise.

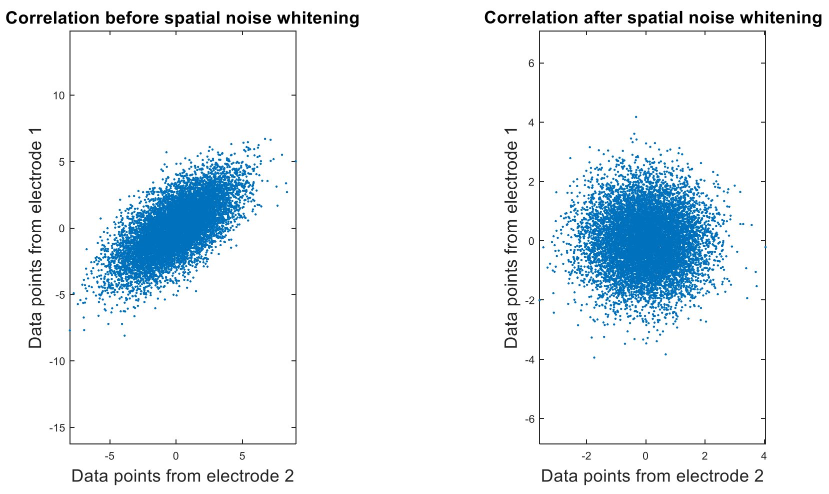


Figure 10. Electrode data before and after spatial noise whitening

Calculating the spatial noise whitening

To calculate the spatial noise whitening transformation we use the noise correlations among electrodes by calculating the cross-correlation matrix between the noisy chunks of every pair of electrodes. The cross-correlation between every pair of electrodes is given by:

$$corr_{j,k} = \sum_{i=1}^{N} \frac{corr(n_{j_i}, n_{k_i})}{length(i)}$$ *Equation 9*

Where n_{j_i} and n_{k_i} represent the i-th chunk of noise, i.e. data trace where no spikes were detected, between electrodes j and k respectively. Then, the correlations among electrodes are arranged in the cross-correlation matrix as follows:

$$corr = \begin{matrix} corr_{1,1} & \cdots & corr_{M,1} \\ \vdots & \ddots & \vdots \\ corr_{1,M} & \cdots & corr_{M,M} \end{matrix}$$ *Equation 10*

If we apply a Cholesky factorization of this cross-correlation matrix we decompose the matrix in an upper triangular matrix and its transpose. Finally, when inverting the upper triangular matrix we obtain the matrix S_n that represents the spatial noise whitening transformation as shown in Equation 12.

$$Chol(corr) = Y^T Y$$ *Equation 11*

$$S_n = Y^{-1}$$ *Equation 12*

Temporal noise whitening

Temporal noise whitening is a transformation that reduce noise correlations in a single electrode over time, i.e. it decorrelates noise on every single electrode. Meaning that the power spectral density of the noise of every electrode will be characterized by white noise, i.e. it will be a flat spectrum. Figure 11 shows the correlation of a piece of data with a shifted version of it on one electrode, and the effect of applying the temporal noise whitening on that electrode. To make it clearer for this purpose, Figure 12 represents the power spectral density (PSD) of the electrode data before whitening, with a specific frequency shape, and after whitening, with a flatter spectrum.

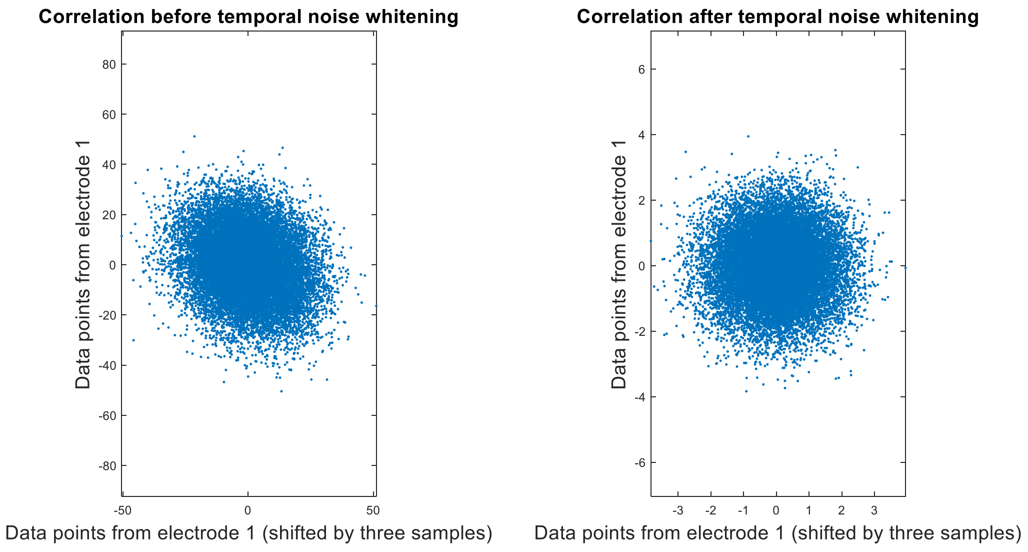


Figure 11. Electrode data before and after temporal noise whitening

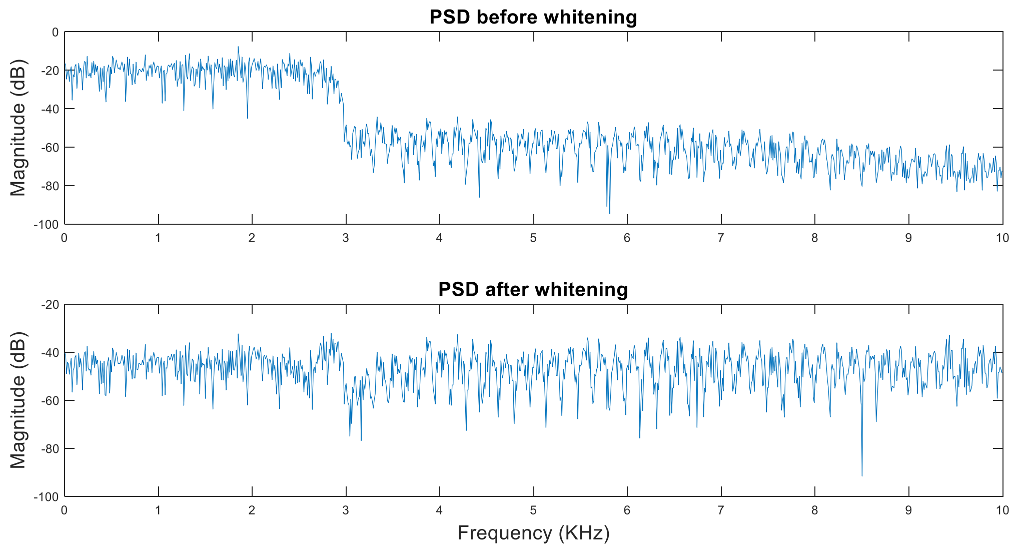


Figure 12. PSD of a single electrode before and after temporal noise whitening

Calculating the temporal noise whitening

To compute the temporal noise whitening transformation we follow an approach similar to [16]. For each electrode we take the autocorrelation of the chunks of data where no spikes were detected, i.e. noise chunks, in the electrode. The autocorrelation is calculated by lagging the noise chunks a value equal to the spike length, as:

$$autocorr_j = \sum_{i=1}^{N} \frac{corr(n_{j_i}, n_{j-\tau_i})}{length(i)} \qquad \text{Equation 13}$$

Where n_{j_i} represent the i-th chunk of noise in electrodes j, and τ represents the lag, which goes from zero to the spike length in sample. Then, since the autocorrelation is a symmetric vector by definition with length equal to twice the maximum lag, we take its half and build a Toeplitz [23] matrix out of it.

$$toeplitz(autocorr_j[0, \dots, \max(lag)]) \qquad \text{Equation 14}$$

The Toeplitz matrix is as follows:

$$\begin{matrix} autocorr_j(0) & \cdots & autocorr_j(\max lag)) \\ \vdots & \ddots & \vdots \\ autocorr_j(\max lag)) & \cdots & autocorr_j(0) \end{matrix} \quad \text{Equation 15}$$

We then find the inverse of the Toeplitz matrix and calculate its square root:

$$inv_{toep_j} = toep_{autocorr_j}{}^{-1} \quad \text{Equation 16}$$

The square root in this context is defined as a matrix M such that:

$$inv_{toep_j} = MM \quad \text{Equation 17}$$

Finally, from the matrix M, we extract the values from the middle column. This middle column is a vector that represents the temporal noise whitening for the specific electrode j, such as:

$$T_{n\,j} = [M_{1,k}, \ldots M_{n,k}] \quad \text{Equation 18}$$

Where n is the total number of rows and k represents the middle column.

Data and performance measures

Data and templates

To evaluate the BOTM algorithm's performance with different whitening approaches we need data recorded by the CMOS HDMEA. Moreover, we need, first, the ground truth, i.e. the information of every spike from every neuron in the recorded data, and second the template of every neuron present in the recording.

There are different ways to obtain the ground truth. The simplest one is to create a fake multichannel noise trace, define a number of neurons and their respective footprints, and then insert the footprints as spikes in the multichannel noise trace. This way we would have the exact spike times and their respective putative neurons. However, this approach presents two problems, first we run into the risk of having footprint waveforms that are not real, and second, the noise correlations of a real recording would be impossible simulate.

A more robust way to create a data set with a known ground truth would be to take a real recording from the CMOS HDMEA and apply a spike sorting algorithm to identify the neurons and their spikes. Then take the multielectrode footprint of some of the neurons we found and use them to create new fake multielectrode footprints. These new footprints representing new neurons are then reinserted in the original recording at known times. With this approach we are able to know exactly when our fake neurons are spiking, also we are keeping the background noise and real signals in the electrodes, which would be impossible to reproduce by creating the full fake data. Every time we want to compare the performance of the BOTM with any whitening approach we just have to look at the fake neurons and how well were they detected and classified.

Performance measures

Once we have detected and classified the spikes with the BOTM algorithm we have to find a measure of how good both tasks were performed. The way to measure the performance of the BOTM is by looking at the false positive spikes (FP), false negative spikes (FN), and classification errors (CLE).

False positive spikes are spikes that are detected by the algorithm but are not actually in the data, as shown in the topmost BOTM output in Figure 13. On the other hand false negative spikes are spikes that are actually happening in the data but the BOTM was not able to detect them, as shown in the middle row of

Figure 13. In this case the BOTM didn't detect two spikes within the data. Finally a classification error is a spike that was correctly detected, but the algorithm miss-classified it, assigning it to the wrong neuron, as shown in the last row of Figure 13.

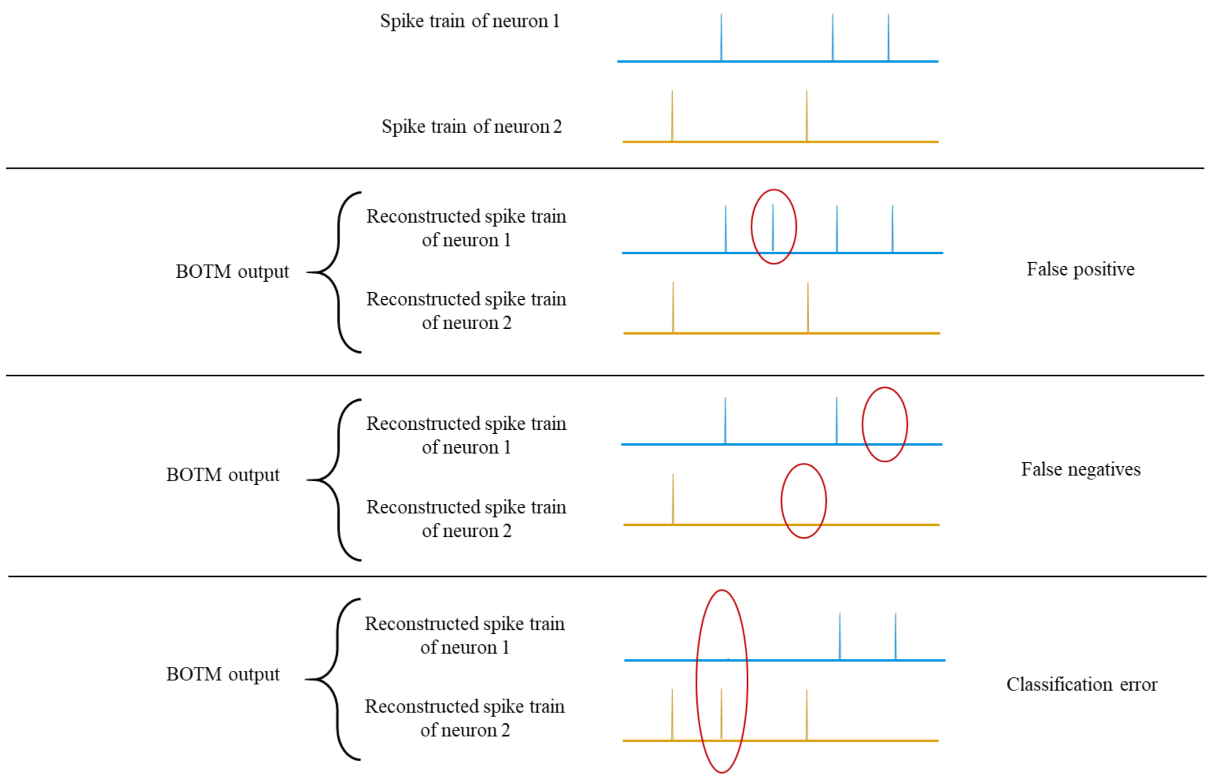

Spike train of neuron 1
Spike train of neuron 2
BOTM output
Reconstructed spike train of neuron 1
Reconstructed spike train of neuron 2
False positive
BOTM output
Reconstructed spike train of neuron 1
Reconstructed spike train of neuron 2
False negatives
BOTM output
Reconstructed spike train of neuron 1
Reconstructed spike train of neuron 2
Classification error

Figure 13. Examples of FP, FN, and CLE

Simplifying the noise covariance matrix and studying signal correlations

A new approach implementing the BOTM algorithm

We already discussed the importance of the noise covariance matrix on the BOTM algorithm: its inverse provides a whitening transformation that is crucial for spike detection and classification. We also saw that computing this noise covariance matrix becomes impossible when large number of electrodes are used for recording. However we can split the noise covariance matrix into two simpler matrices: one containing spatial correlations, which its inverse provides spatial whitening, and one containing temporal correlations, which its inverse provides temporal whitening. Therefore, in the following sections we evaluate the performance of the BOTM algorithm by applying spatial whitening alone, temporal whitening alone, and both spatial and temporal whitening together, one after the other.

Signal correlations and signal whitening

It is important to mention that we also evaluate what happens if we take signal correlations, i.e. not only noise but also spikes, into account. To build the signal covariance matrix we take the full trace of every electrode, and not the noise chunks as before, and again compute both: the spatial covariance matrix and the temporal covariance matrix. After this we calculate their respective spatial whitening and temporal whitening matrices.

Comparing different whitening approaches (Part 1)

BOTM on toy data with fake noise

As a first approach to implement the BOTM with different whitening strategies we studied the very simplest case. We took the multielectrode footprint of four neurons in five electrodes close to each other. The footprint of each neuron can be seen in Figure 14.

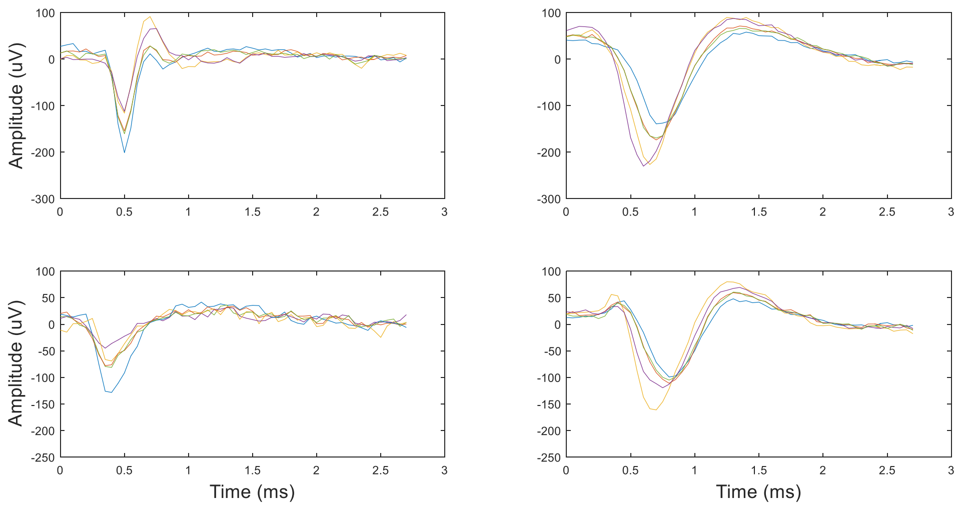


Figure 14. Five-electrode footprint of four neurons. Each color in the plot represents the waveform in each electrode

These footprints were obtained by the classical spike sorting procedure. Then we created a fake five electrodes noise trace of 20 seconds, 400000 samples, and inserted the known footprints by adding them to the background noise. The footprints were randomly inserted as multielectrode spikes in the noise trace, keeping record of the exact spiking time to be used later as a ground truth. In total 2045 multielectrode spikes were inserted. To resemble real spike variability, the amplitude of the spikes was randomly modified within a range of $\pm 10\%$. We also checked for refractory period violations, eliminating spikes from the same neuron that were too close to each other. The noise, multielectrode footprints in noise free data, and final toy data containing noise and spikes is presented in Figure 15.

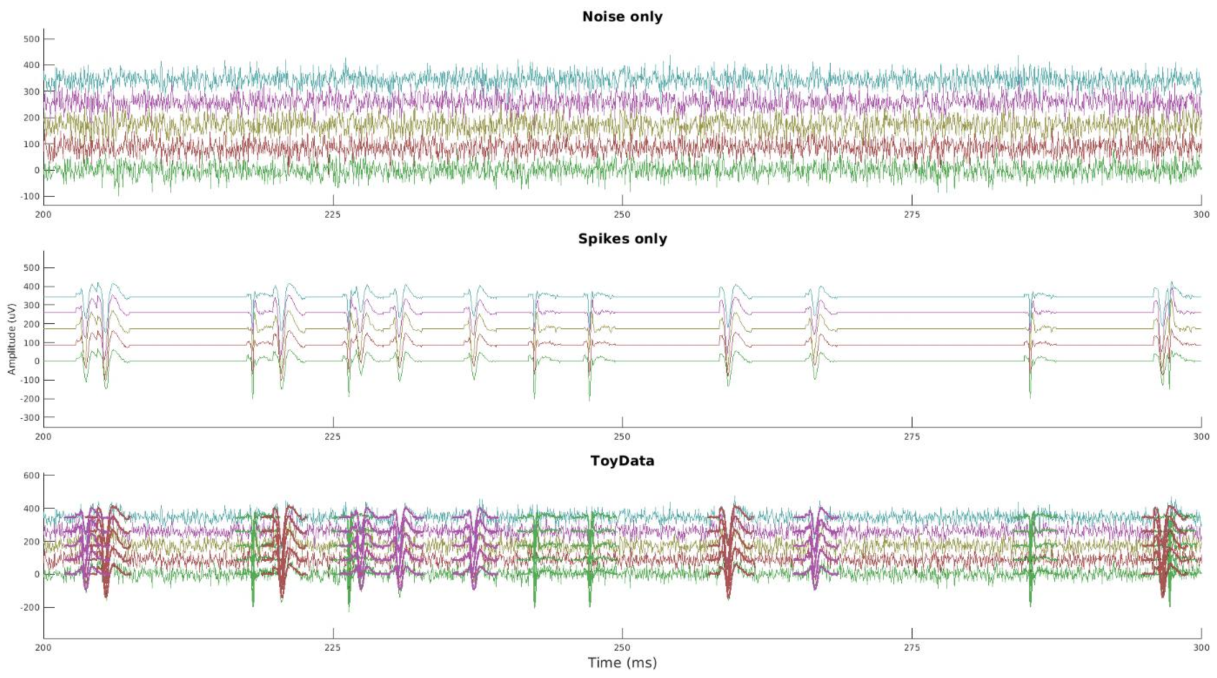


Figure 15. Generated noise, spikes, and full toy data with noise and spikes. The voltage traces are spaced by 100uV and the spikes are highlighted in the toy data to have a better visualization

We applied the BOTM algorithm using the inserted footprints as templates for matching. In Figure 16 we compare how do the noise whitening transformations affect the performance of the BOTM. As expected when the data, and the templates, are not whitened the algorithm gives a larger number of errors. Moreover spatial whitening seems to provide a better performance than temporal whitening. Finally the application of both whitenings provides the least number of errors.

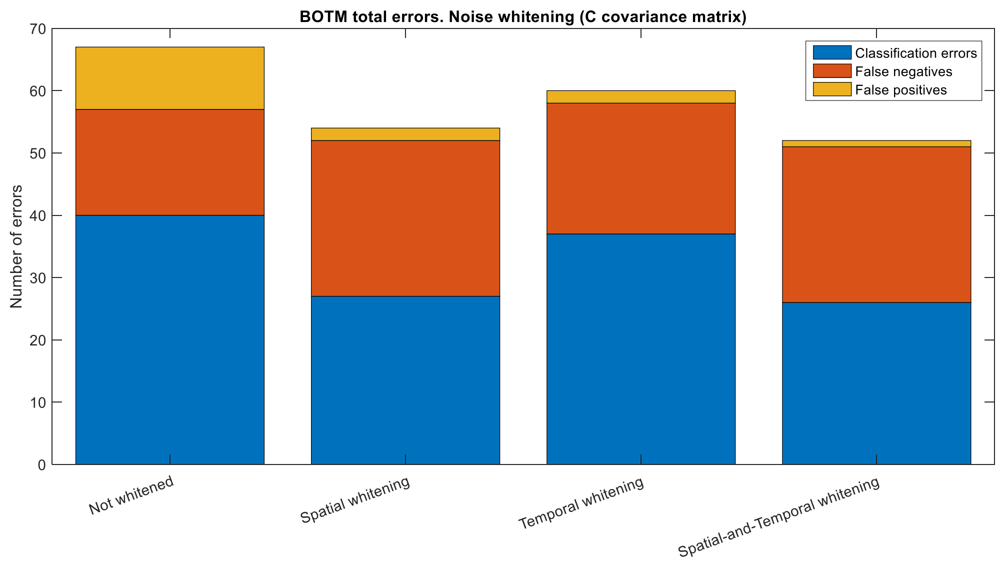


Figure 16. BOTM performance with different noise whitening approaches

In Figure 17 we see the comparison when applying signal whitening. Here again any of the whitenings provide a better performance than the not whitened case. However two things have to be highlighted, the first is that when spatial whitening is involved (spatial and spatial-and-temporal whitening) the algorithm becomes noise robust, eliminating all the FP. The second is that the performance of the spatial-and-temporal whitening is not better than the only spatial whitening. The reason of this is that spatial whitening and temporal whitening alone are not completely independent of each other, therefore, when applying both, one after the other, it can happen that the whitening one is producing is affected by the next whitening, affecting the final performance of the algorithm.

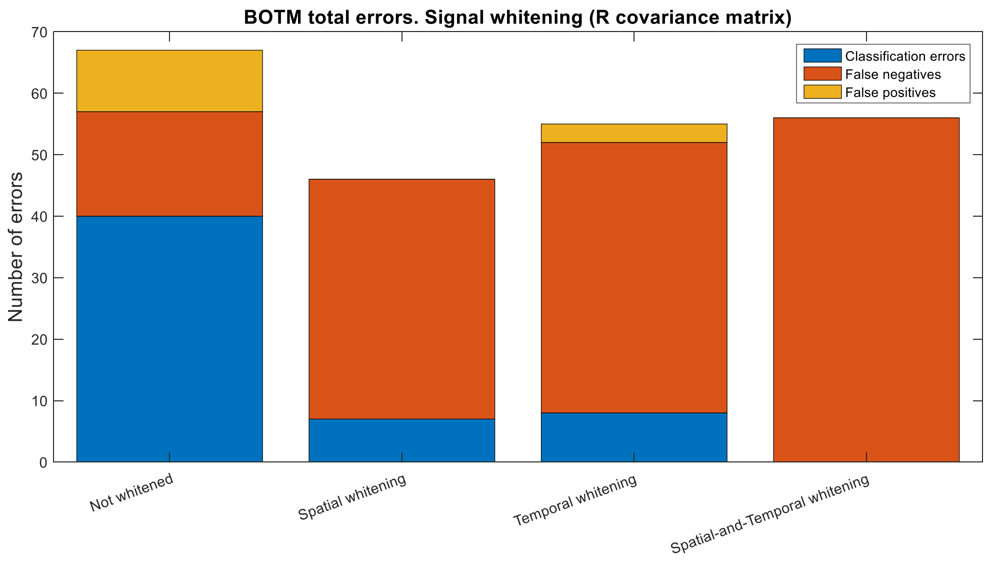


Figure 17. BOTM performance with different signal whitening approaches

Using real multielectrode spikes (derived from the footprints) and inserting them into a generated fake noise is an easy way to have reliable ground truth, and a fast way to run and test the algorithm. But, this does not provide a picture of what would happen in a real recording. In a real recording there would be more spikes than just the ones from the four neurons we considered, these spikes would look like noise in some cases, but would also be confused among the matching of the BOTM, leading to an overall increase on the number of errors. Additionally a real recording contains spatial, temporal, and spatio-temporal correlations among electrodes we are not taking into account when creating the fake noise. The only way to test the performance of the BOTM algorithm with real electrode correlations is by using noise from real recordings.

BOTM on toy data with real noise

In this case we took the same footprints used with the fake noise. However, the noise was taken from a real recording. We extracted a five electrodes trace of 20 seconds, 400000 samples, and added the known footprints to this trace. Figure 18 shows the noise, the added spikes, and the final toy data. Notice in this case the noise also contains spikes. These spikes, which we are considering as background noise, will affect the performance of the BOTM, affecting the number of errors compared to the previous case of fake noise, but this is something we cannot avoid if we want to have real correlations in our data. However, there is a way to overcome this problem as we will see later.

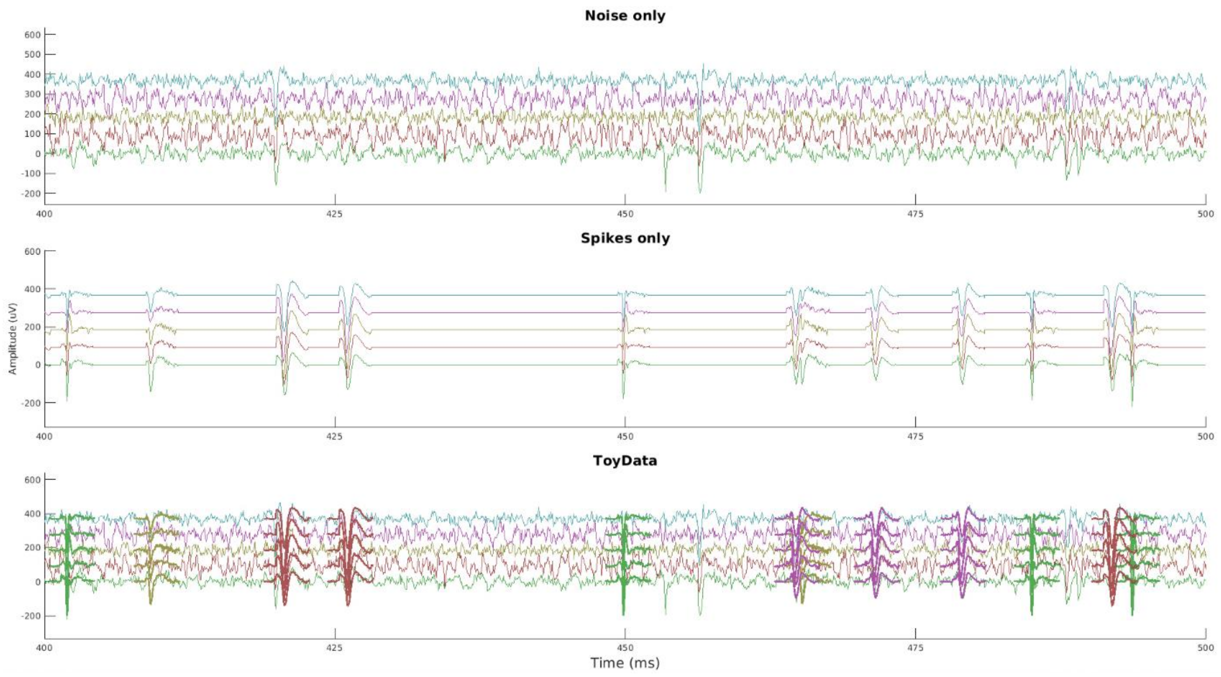


Figure 18. Real noise, spikes, and full toy data with noise and spikes. The voltage traces are spaced by 100uV and the spikes are highlighted in the toy data to have a better visualization

We applied the BOTM algorithm to this new data and the variations of the whitening as before. In this case we still used the same four templates coming from the neurons we previously inserted. Figure 19 shows what happens when applying noise whitening. Figure 20 shows, on the other hand, the performance of the BOTM when applying signal whitening. Notice that in this case the number of errors is presented in a logarithmic scale.

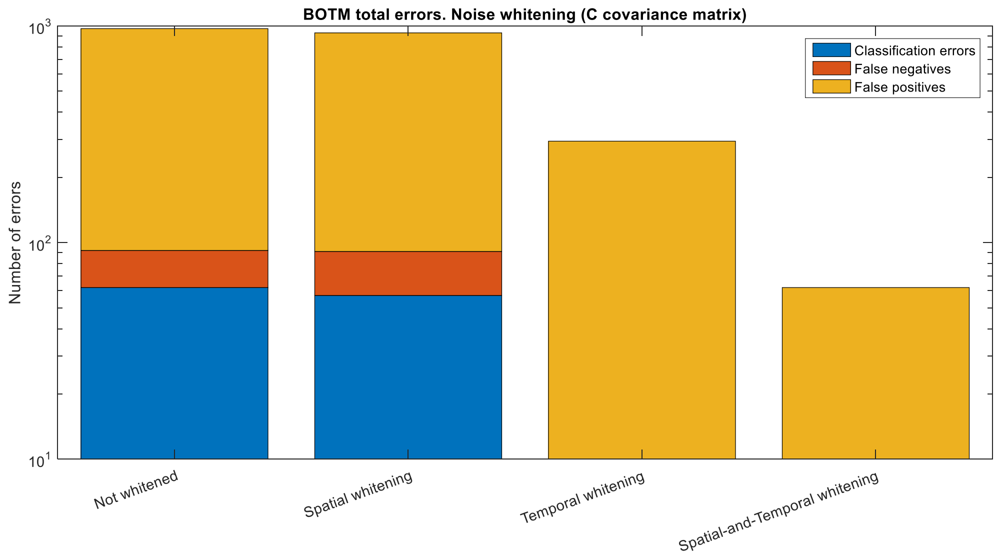


Figure 19. BOTM performance with different noise whitening approaches

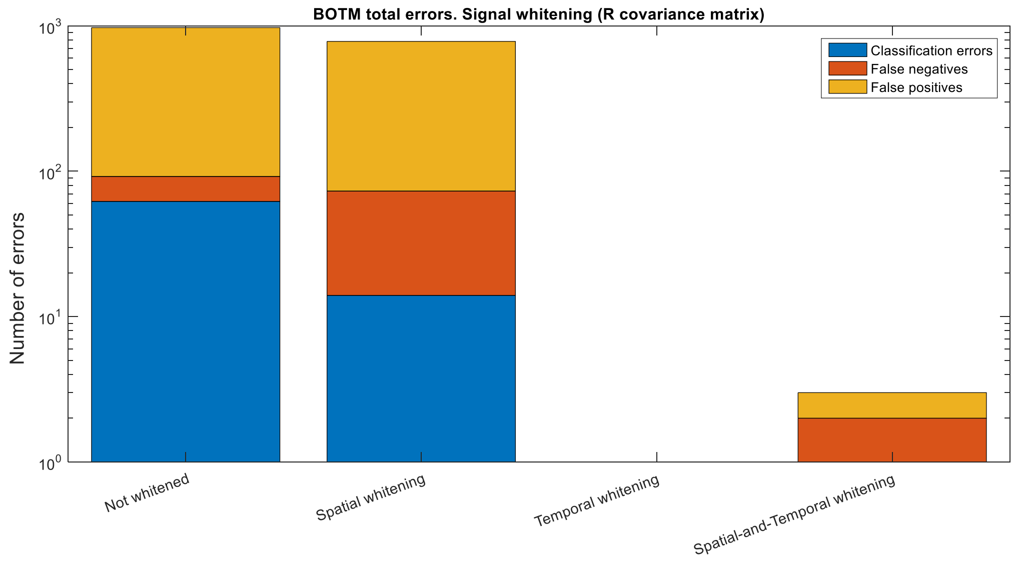


Figure 20. BOTM performance with different signal whitening approaches

The number of errors for the not whitened case increased by more than 1400% with respect to the fake noise case, mainly because of the FP. This drastic change is due to the spikes in the noise that are confused by the BOTM algorithm, we will tackle this problem later. Furthermore, and more interesting is the fact that even though the spatial whitening provide some improvement in the number of errors, the temporal, and the spatial-and-temporal whitenings drastically reduced the number of total errors. Specifically, the temporal signal whitening approach produces a perfect matching leading to zero errors, and the spatial-and-temporal signal whitening an almost perfect performance with only 2 errors.

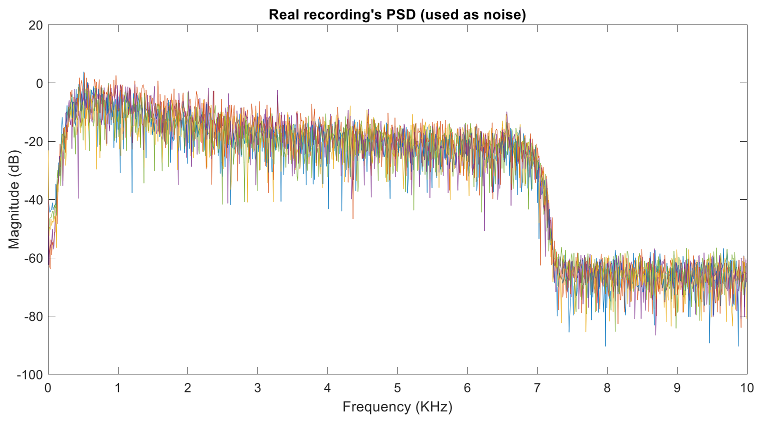
After looking at this results, we can think the temporal signal whitening is transforming the data and templates in such a way that we can perfectly detect and classify any spike with the BOTM. However we now from [13] that this is not the case. We then decided to investigate on what was going on: why the temporal signal whitening was providing such a good BOTM performance when using noise from a real recording?

The problem of high-frequency components in inserted footprints

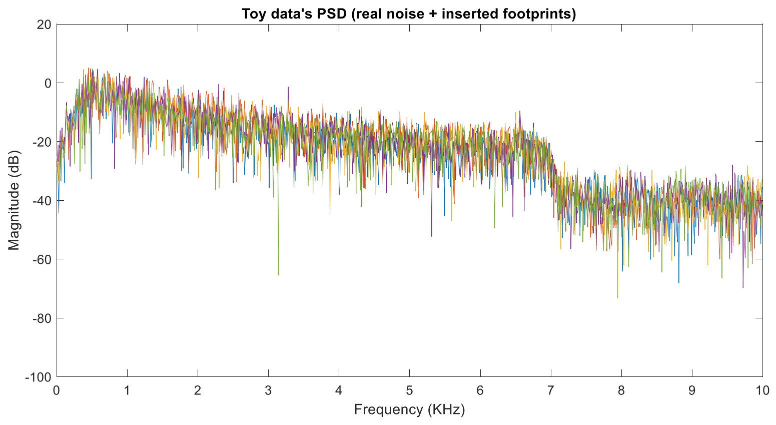
We were able to find out that the inserted footprints have high-frequency components not present in the real noise, which are enhanced after temporal whitening, making them easier to match.

To understand this we first have to take a look at the high-frequencies of the footprints, then, discuss why they are there but not in the real noise, later, why is the temporal whitening enhancing those frequencies, and finally understand why this is an actual problem, and how to solve it.

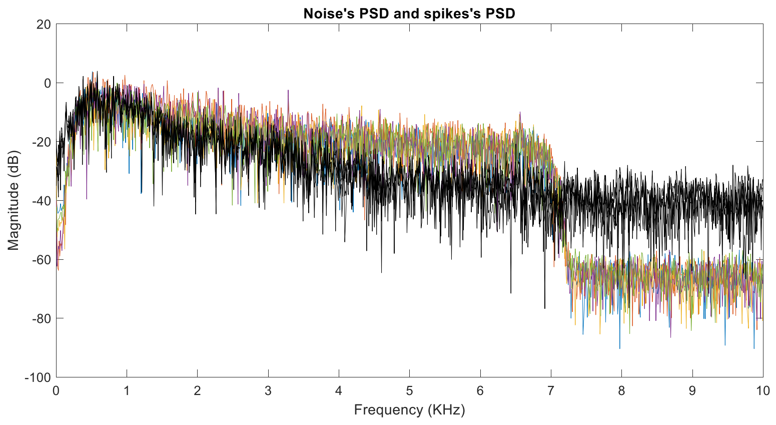
As mentioned before, a real recording was used as noise, and then on top of that noise we added the fake footprints. In Figure 21 we see the PSD of the real recording used as noise (a), the toy data generated with the noise and the inserted footprints (b), and the noise and footprints separated (c). In Figure 1Figure 21 (a) we can observe that the real recording has the shape of the bandpass filter used as the first digital signal processing step of the classical spike sorting algorithm. However, when we add the footprints that are going to define the spikes of our ground truth for the BOTM algorithm, we obtain the PSD shown in (b). Even though the footprints are extracted from the neurons previously classified in the spike sorting algorithm, when we add them back into the data, they change the shape of PSD, adding high-frequencies. By comparing (a) and (b) in Figure 21, we clearly see that there is a change, mainly, in the high-frequency components of the data. This change must come from the added spikes, since was the only thing that we added. To make this clearer, we have replotted in (c) the PSD of the noise but now with the PSD of the signal containing only spikes and no noise on top. When looking at this figure it becomes obvious that the new PSD is the sum of the real recording's PSD and the PSD of the signal containing only spikes.



(a)



(b)



(c)

Figure 21. (a) Power spectral density (PSD) of the recording used as noise. (b) PSD of the toy data containing noise as well as inserted footprints. (c) PSD of the recording used as noise, and the noise free signal composed solely by added spikes (black)

To understand why the footprints are adding such high-frequencies components we must understand where these footprints are coming from. In principle the footprints should not have any difference in PSD compared to the real recording since they are obtained from the same band pass filtered data. But, there are two sources for these high frequencies which we will call: the offset problem, and the alignment problem.

Before introducing both problems we have to remember how we build the footprint for each neuron. Following the classical spike sorting approach: first we detect the spikes, then we extract some samples around the spike peak in order to get its waveform, and finally we align the waveforms coming from the same spikes in the same electrodes and average them (get the centroid of their distribution) in order to get the footprint.

The offset problem

The offset problem arises when we add the footprints with the noise. Since we are just adding both signals the transition between the background noise and the added footprint is not smooth, introducing the aforementioned high-frequencies.

The alignment problem

The alignment problem comes into play when we are aligning the spike waveforms extracted from the recording to build the footprints. Because the spikes are not locked to our sampling frequency, and since we are dealing with sampled spikes (i.e digitalized), when trying to align different spike waveforms, the alignment can differ if the starting sample of the waveforms is not the same. This is known as sampling jittering and is presented in [15]. Then, when we sum up waveforms that are not perfectly aligned, but present a time shift, we create new spike waveforms that contain high-frequency fluctuations.

Temporal whitening and its effect on the high-frequency components

As explained before in the Correlations and whitening section, the temporal whitening is basically flattening the PSD of the recording (see Figure 12). Now, if we look at the PSD of Figure 21 (a) again, we would think the temporal whitening should be just a function that leads to a flat spectrum. And that is exactly what is happening, as is shown in Figure 22. The problem is that because the temporal whitening filter is suppressing the mid-frequencies and enhancing the high-frequencies, and in these high-frequencies there are only spike components and no noise, the temporal whitening is actually suppressing the noise while enhancing the inserted spikes.

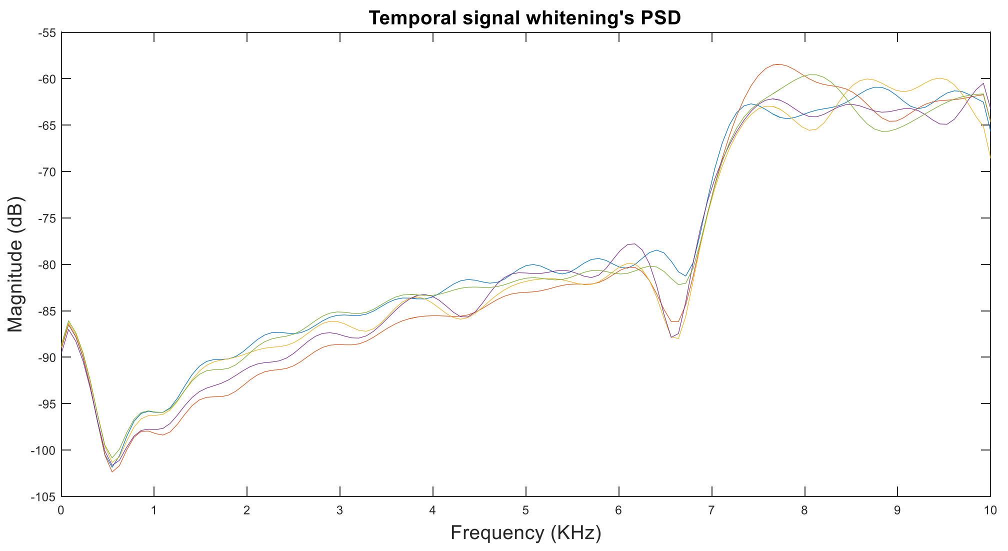


Figure 22. PSD of the temporal signal whitening transformation

The actual problem

Now that we understand that the temporal whitening is enhancing high-frequencies, where spike but not noise components are present, we can discuss whether this is an actual problem. We can ask ourselves, with this perfect matching that leads to zero errors, are we not solving the detection and classification problem? The actual problem is that these high-frequencies are there just because we are inserting fake footprints (spikes) to be able to have a ground truth. In a real case scenario, there will not be frequencies higher than the cutoff frequency of the bandpass filter used. Therefore the enhancement of spikes over noise due to the temporal whitening will never happen. The only way to actually measure the performance of the BOTM under different whitening circumstances is by removing those undesirable high-frequency components.

One question that may arise is how this problem was handle in the original BOTM publication [13]. Here the effect of the temporal whitening is less obvious since the full noise covariance matrix is used and there is no individual evaluation of the spatial and temporal components. However, the very low magnitude in the high-frequency components of the data is represented by singular values in the noise covariance matrix that are very close to zero, making it ill-conditioned. The ill-conditioned noise covariance matrix is handled by diagonal loading, solving at the same time the inversion of the matrix, and the high-frequency components in the inserted footprints.

In our case, we decided to rebuild the footprints, making sure every spike had a smooth start and end when added to the data, to prevent the offset problem. And, bandpass filtering every footprints after the alignment process, with the same filter used for the data, to eliminate the high-frequencies coming from the alignment problem.

Comparing different whitening approaches (Part 2)

Here we compare the different whitening approaches with the new data set built with the 10 footprints without high-frequency components. Once again we took a 20 seconds, i.e. 400000 data samples, real recording and used it as noise, however to get closer to a real CMOS HDMEA application we used 289 electrodes. To this 289 electrodes 20 seconds recording, we added the new footprints. Figure 23 shows the PSD of the new noise and inserted footprints, i.e. spikes. It is interesting to compare this figure to Figure 21 (c) and look at how the spikes have no high-frequency components anymore.

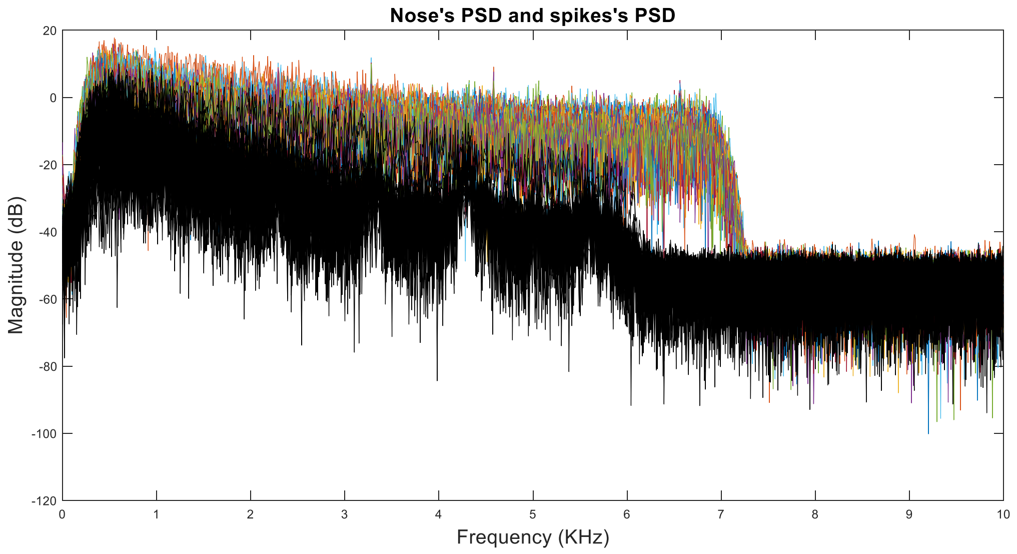


Figure 23. Power spectral density (PSD) of the new recording used as noise, and the signal composed solely by the new spikes on top (black)

In a first attempt to measure the performance of the BOTM on our new data we applied the algorithm without any whitening transformation. In this case we also considered only the inserted neurons' templates and not the templates coming from the neurons in the background noise for the matching. However the more templates we add to the matching the better the performance of the BOTM should be. Therefore, as a second case we added all the neurons in the original recording that were found previously by the spike sorting algorithm.

BOTM without whitening

When using only the templates from the inserted neurons in the BOTM algorithm the total number of errors is 3562, see Figure 24, left column. Table 1 gives a more detailed error outlook of every inserted neuron. However when we use templates from all the neurons found in the recording with a previous spike sorting approach, 620 templates in this case, the number of errors decrease drastically to 748, Figure

24, right column, Table 2. It is important to realize that in this case the number of errors is in a logarithmic scale to have a better visualization of the results.

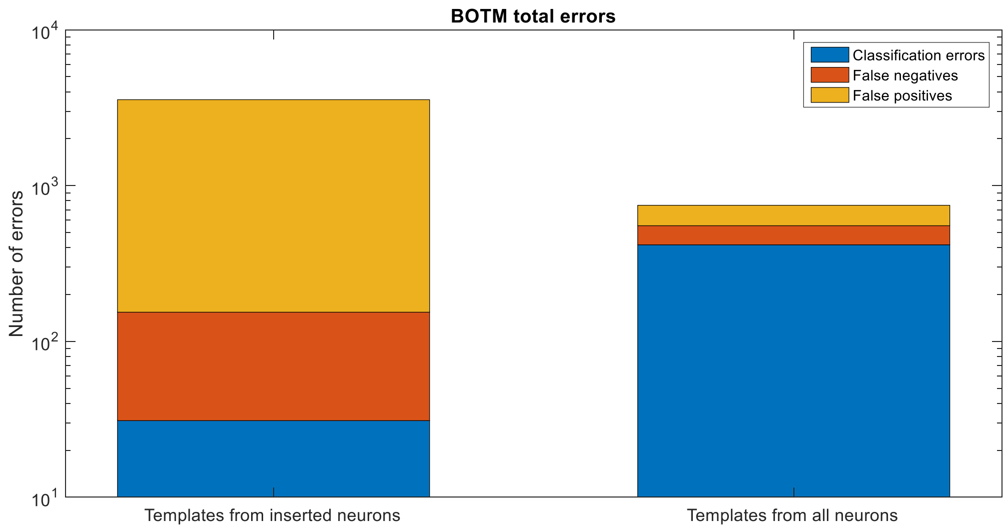


Figure 24. BOTM considering only templates from inserted neurons (left column), and considering templates from all the neurons found in the original recording

Neuron	False Positives (FP)	False Negative (FN)	Classification Errors (CLE)	Total Errors
1	17	0	1	18
2	51	0	0	51
3	54	2	4	60
4	370	102	16	488
5	2385	11	3	2399
6	312	5	4	321
7	117	2	1	120
8	0	0	0	0
9	58	0	0	58
10	44	1	2	47
Total	3408	123	31	3562

Table 1. BOTM output for the 10 inserted neuron footprints when only templates from inserted neurons are used

There are some important observation we have to make here. The first one is that the number of errors decreases when we add more templates. The reduction of errors comes largely due to the reduction in the number of FP, in the first case we have 3408 FP, while in the second 195 FP. The number of FP is large in the first place because any spike detected from the original recording, i.e. not inserted spikes, will be assigned to one of the inserted neurons, leading to a FP, since we are only using their templates in the matching algorithm. But when we use the templates from all the neurons, the algorithm is able to assign the spikes from the original recording to their putative neurons, reducing the number of FP.

Neuron	False Positives (FP)	False Negative (FN)	Classification Errors (CLE)	Total Errors
1	0	19	51	70
2	0	6	11	17
3	2	19	56	77
4	0	18	63	81
5	193	11	32	236
6	0	13	45	58
7	0	18	61	79
8	0	2	15	17
9	0	14	40	54
10	0	16	43	59
Total	195	136	417	748

Table 2. BOTM output for the 10 inserted neuron footprints when templates from all the neurons found in the original recording are used

Another important point to discuss is the number of CLE, which goes from 31 in the case where only the templates from the inserted neurons are used, to 417 in the case where the templates from all neurons are used. This increase in the number of CLE is due to the high number of templates, since we have 620 templates when we considering all the neurons, the algorithm is more likely to confuse spikes, assigning them to the wrong putative neuron. Finally the number of FN seems unchanged in the plots, being 123 for the case of the templates from inserted neurons, and 136 for the case of the temples from all neurons. However if we look at where the errors are coming from, we see the nature of FN is completely different in both cases. For the first case, just one neuron accounts for the most of the FN, while for the second case, most of the neurons present between 11 and 19 FN. To understand what is happening we need to explain what the BOTM does in detail when a spike is detected. Every time a spike is detected, the algorithm sets a “detection window” where any other spike detected is neglected, this to be sure the spike initially detected is not counted many times among the rest of the electrodes. The problem arises when two or more spikes are happening at the same time, but due to the detection window just one is detected. This is exactly what is happening in the second case with the 620 templates, since the BOTM is detecting spikes everywhere in the CMOS HDMEA device every time a spike is detected, the rest of the spikes happening within the detection window are being neglected increasing the number of FN. In the first case where we only have the templates from the inserted neurons, we are not able to detect every spike and therefore there are not many detection windows neglecting the rest of the spikes.

BOTM with noise whitening

After discussing the effect of using the templates from all the neurons, previously detect and classified in a spike sorting approach, in the BOTM algorithm, we finally want to study the effect of the noise whitening transforms. We used the previous case of the BOTM algorithm without whitening, and with the templates from all the neurons, as our base case. Then we looked at the performance of the algorithm when applying a spatial noise whitening, a temporal noise whitening, and a spatial-and-temporal noise whitening. Figure 25 shows the plot comparing the different whitening strategies. Table 3, Table 4, and Table 5 show a more detailed picture of the BOTM's errors.

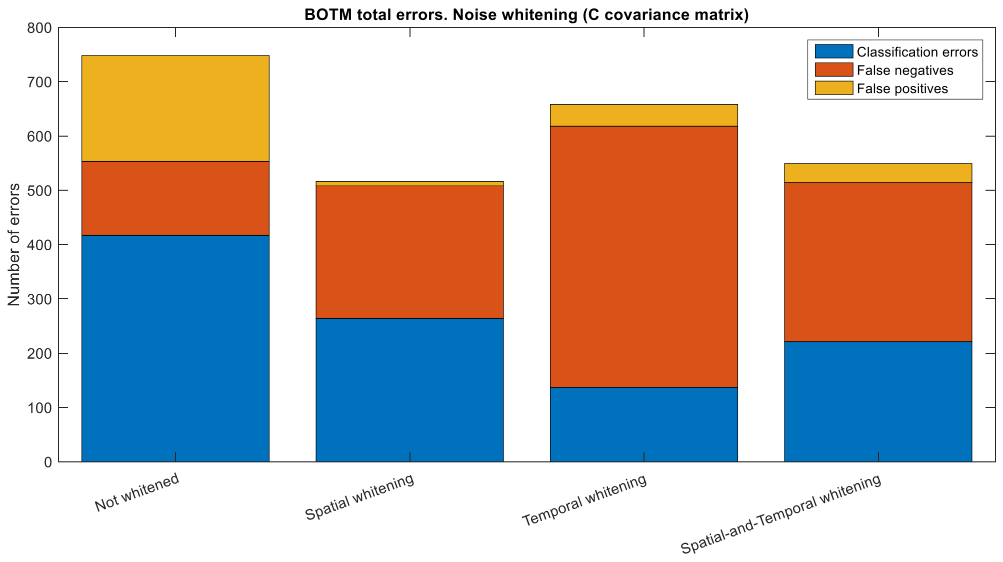


Figure 25. BOTM with different noise whitening approaches

By comparing the bar plots in Figure 25 we can see how all the whitenings are actually reducing the number of errors compared to the base case. What is also interesting is that the number of FN increases when applying any whitening, again compared to the base case. This, is as explained before, due to the detection window the BOTM sets when detecting a spike. Finally we see how the spatial-and-temporal noise whitening is worse than the spatial noise whitening. As stated previously this is because both spatial and temporal whitening transformations are not completely independent from each other, therefore when applying one after the other the data is not perfectly spatio-temporally whitened.

Neuron	False Positives (FP)	False Negative (FN)	Classification Errors (CLE)	Total Errors
1	0	2	34	36
2	0	2	1	3
3	0	43	36	79
4	0	19	36	55
5	3	20	30	53
6	1	19	20	40
7	0	32	38	70
8	4	51	37	92
9	0	31	15	46
10	0	25	17	42
Total	8	244	264	516

Table 3. BOTM performance when applying spatial noise whitening

Looking at the details of the spatial noise whitening we see how it makes the matching more noise robust, taking the number of FP from 195 to just 8. Here we also discuss about three neurons that present a large total error change after spatial noise whitening. Neuron 2 got only three errors after spatial whitening, when we looked at the footprint of this neuron we saw it has large amplitude spikes waveform in a small number of electrodes. After spatial noise whitening the footprint still exhibited the large amplitude waveforms but in a reduced number of electrodes, making the neuron more focalized and harder to confuse with others, reducing CLE. Another interesting neuron is number 5. This neuron had the problem that it is not noise robust: 193 FP in the not whitened case, however thanks to the spatial noise whitening the number of FP is reduced just to 3. Finally neuron 8, this neuron presented the highest spike amplitude and ideally should be detected without problem, but again the number of FN is increasing due to the “detection widow” problem.

Neuron	False Positives (FP)	False Negative (FN)	Classification Errors (CLE)	Total Errors
1	0	42	15	57
2	0	25	4	29
3	0	52	19	71
4	0	78	19	97
5	39	68	26	133
6	0	48	13	61
7	1	58	14	73
8	0	18	3	21
9	0	50	10	60
10	0	42	14	56
Total	40	481	137	658

Table 4. BOTM performance when applying temporal noise whitening

For the case of temporal noise whitening, we again see an improvement over FP errors and the overall error value compared to the not whitened case.

In the last case of spatial-and-temporal whitening again neuron 5 sees a large improvement as for the case of only spatial whitening, and neuron 8 becomes again the neuron with the least number of errors. Furthermore, neuron 2 presents again a large number of errors due to the increase in FP.

Neuron	False Positives (FP)	False Negative (FN)	Classification Errors (CLE)	Total Errors
1	0	44	12	56
2	33	57	28	118
3	0	41	31	72
4	0	47	22	69
5	2	20	23	45
6	0	20	14	34
7	0	0	57	57
8	0	3	2	5
9	0	33	14	47
10	0	28	18	46
Total	35	293	221	549

Table 5. BOTM performance when applying spatial-and-temporal noise whitening

BOTM with signal whitening

We finally applied our last variation over whitening approaches: signal whitening. Here the procedure is the same as for noise whitening, however the structure used to apply the whitening transformations was the signal covariance matrix R.

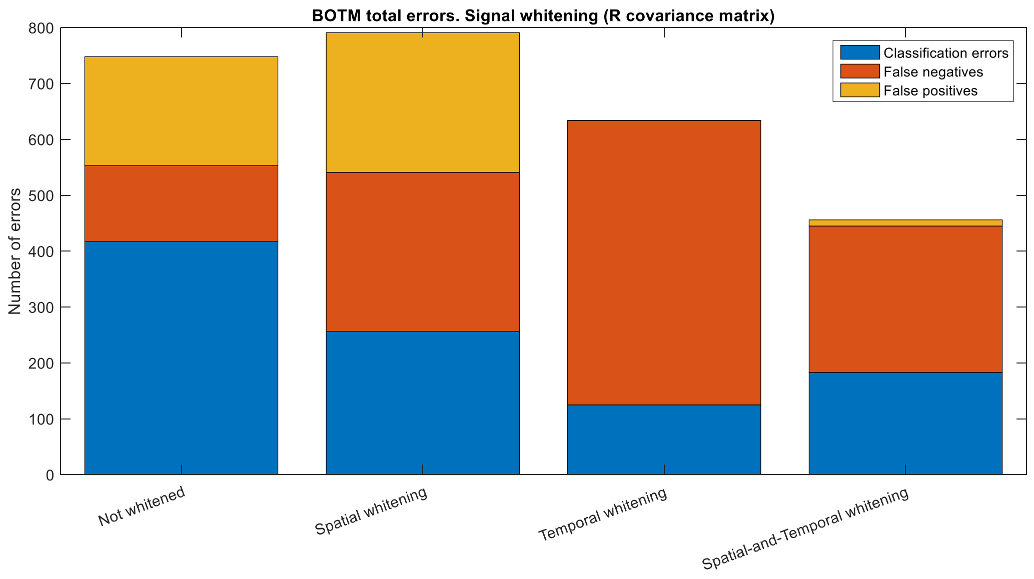


Figure 26. BOTM with different signal whitening approaches

Here we can appreciate the spatial signal whitening is actually not providing any improvement over the base case, we will discuss this later. But as expected the temporal and spatial-and-temporal signal whitening are better over the base case. As in the noise whitening cases the number of FN is increasing due to the BOTM "detection window" issue.

Neuron	**False Positives (FP)**	**False Negative (FN)**	**Classification Errors (CLE)**	**Total Errors**
1	0	21	22	43
2	242	55	31	328
3	0	38	34	72
4	5	67	46	118
5	2	20	21	43
6	1	13	19	33
7	0	40	29	69
8	0	2	0	2
9	0	7	35	42
10	0	22	19	41
Total	250	285	256	791

Table 6. BOTM performance when applying spatial signal whitening

In the particular case of spatial signal whitening it is important to point out that approximately one third of the total errors are coming from the FP of neuron 2. This neuron, when whitened, seems to resemble one of the neurons in the background noise, leading to the increase number of FP.

Neuron	False Positives (FP)	False Negative (FN)	Classification Errors (CLE)	Total Errors
1	0	55	13	68
2	0	23	2	25
3	0	60	17	77
4	0	73	18	91
5	0	72	18	90
6	0	54	9	63
7	0	54	22	76
8	0	23	7	30
9	0	51	7	58
10	0	44	12	56
Total	0	509	125	634

Table 7. BOTM performance when applying temporal signal whitening

In this case is the temporal whitening is providing the most noise robust detection, leading to zero FP. Another interesting outcome to mention is that this whitening reduces the number of classification errors in one third of the base case.

Finally the spatial-and-temporal whitening present a low number of FP, but also of FN, resulting in a reduction in the total number of errors.

Neuron	False Positives (FP)	False Negative (FN)	Classification Errors (CLE)	Total Errors
1	0	41	19	60
2	0	1	2	3
3	0	48	28	76
4	0	46	25	71
5	9	24	14	47
6	2	31	7	40
7	0	0	63	63
8	0	4	2	6
9	0	35	14	49
10	0	32	9	41
Total	11	262	183	456

Table 8. BOTM performance when applying spatial-and-temporal signal whitening

Conclusions, outcomes and future work

Before going into the conclusions let us restate the core ideas we were address during the development of this work. We wanted to implement the BOTM, a template matching based algorithm, for spike detection and classification, on a recording with a large number of electrodes. The BOTM algorithm requires the inverse of the noise covariance matrix among electrodes to apply a spatio-temporal data whitening. However, this matrix becomes too large for large number of electrodes, making it impossible to compute and handle. To solve this we presented an approach of separating this matrix into two smaller and simpler matrices: one providing a spatial noise whitening and another providing a temporal noise whitening. With this separation we were able to evaluate the BOTM on a large number of electrodes and moreover provide insights into the effect of each whitening. We were also interested on evaluating the performance of the BOTM by considering other correlations among electrodes: signal and spike correlations. Even though we were not able to look at the spike correlations, we evaluated the performance of the BOTM when signal correlations are taken into account.

The first observation is that the covariance matrix (both noise and signal) can be split into two simpler matrices: the spatial covariance matrix that leads to spatial whitening, and the temporal covariance matrix leading to temporal whitening. This is something handful for the purpose of the data whitening, however, as we saw in the results, these matrices (and their whitenings) are not completely independent. Therefore applying one after the other can influence the whitening as a whole.

One of the most important outcomes we got during this work was that the toy data to tests the BOTM performance has to have real noise in it, and that we have to be really careful about the footprints we are inserting as ground truth. This is important not only for the BOTM but for any template matching-based algorithm. We have to make sure the frequency components of the footprints lie in the range of the frequency components of the noise we are using, and that there is no offset when adding the footprints into the data, so no high-frequency steps will be added.

Thanks to the insights we got about the temporal whitening and its effect on flattening the PSD of the data, we proposed an idea to be further developed in the future. Considering the temporal whitening is reverting the effect of the bandpass filter, introducing large values in the high-frequency components and leading to instabilities. One approach to avoid these instabilities is to apply the temporal whitening before bandpass filtering the data. This could be done by taking the raw data, calculating a temporal whitening filter for each channel, applying the temporal whitening and flattening the data's spectrum, and then applying the bandpass filter data for spike sorting.

Another important result, as shown in Figure 24, is the fact that the more templates we use for the BOTM the least the number of errors. This is clear since the algorithm has more templates to match, reducing the overall number of errors. The only tradeoff with having large number of templates is that the number of FN increases due to the "detection window" mentioned before that blinds the algorithm whenever a spike is detected.

Comparing Figure 25 and Figure 26 it is not clear whether noise or signal whitening leads the better results. Spatial noise whitening seems to be better, than spatial signal whitening, however for temporal and spatial-and-temporal whitening the results were similar. More performance tests of the BOTM over different data sets would be useful to determine which is better. Another problem is determining the exact effect each whitening is exerting, since 289 electrodes and 620 footprints are not easy to visualize and interpret. It would be interesting to also look at the spike covariance matrix, i.e. the matrix formed by the correlation of chucks where only spikes happened (opposed to the noise covariance matrix). With this matrix we could apply a PCA, reducing the vector space where the footprints are described, and reducing the dimension where the BOTM operates.

Finally, what is also interesting about the different whitening approaches is that the reduction in the number of errors to the base case (not whitened) is very low, from 748 to 456 in the best case, compared to the reduction is already providing the use of all the 620 templates in the recording, from 3562 to 748 errors in total. Therefore we can infer that thanks to the high resolution the CMOS HDMEA is providing, there could be no need to use the noise covariance matrix at all since the reduction in the number of errors is not critical.

References

[1] Jan Müller et al., "High-resolution CMOS MEA platform to study neurons at subcellular, cellular, and network levels," *Lab on a Cchip,* p. 2767–2780, 2015.

[2] E. Marieb and K. Hoehn, Human Anatomy and Physiology, Glenview, IL: Pearson, 2013.

[3] G. Buzsáki, "Large-scale recording of neuronal ensembles," *Nature neuroscience,* vol. 7, no. 5, pp. 446-451, 2004.

[4] M. E. J. Obien, K. Deligkaris, T. Bullmann, D. J. Bakkum and U. Frey, "Revealing neuronal function through microelectrode array recordings," *Frontiers in Neuroscience,* p. 423, 2015.

[5] F. Franke, D. Jäckel, J. Dragas, J. Müller, M. Radivojevic, D. Bakkum and A. Hierlemann, "High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity," *Frontiers in neural circuits,* vol. 6, no. 105, 2012.

[6] K. D. Harris, R. Q. Quiroga, J. Freeman and S. L. Smith, "Improving data quality in neuronal population recordings," *Nature Neuroscience,* pp. 1165-1174, 2016.

[7] H. G. Rey, C. Pedreira and R. Q. Quiroga, "Past, present and future of spike sorting techniques," *Brain Research Bulletin,* pp. 106-117, 2015.

[8] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Computational Neural Systems,* pp. 53-78, 1998.

[9] R. Chen, A. Canales and P. Anikeeva, "Neural recording and modulation technologies," *Nature reviews Materials,* vol. 2, no. 16093, 2017.

[10] U. Marx and V. Sandig, Drug Testing in vitro: Breakthroughs and Trends in Cell Culture Technology, Berlin: Wiley-VCH Verlag GmbH & Co. KGaA, 2007.

[11] E. Z. "Research: Electrophysiology & Neuroscience," September 2017. [Online]. Available: https://www.bsse.ethz.ch/bel/research/electrophysiology-and-neuroscience.html.

[12] G. T. Einevoll, F. Franke, E. Hagen, C. Pouzat and K. D. Harris, "Towards reliable spike-train recordings from thousands of neurons with multielectrodes," *Current Opinion in Neurobiology,* pp. 11-17, 2012.

[13] F. Franke, R. Q. Quiroga, A. Hierlemann and K. Obermayer, "Bayes optimal template matching for spike sorting – combining fisher discriminant analysis with optimal filtering," *Journal of Computational Neuroscience,* p. 439–459, 2015.

[14] J. Dragas, D. Jäckel, A. Hierlemann and F. Franke, "Complexity Optimisation and High-Throughput Low-Latency Hardware Implementation of a Multi-Electrode Spike-Sorting Algorithm," *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* vol. 23, no. 2, pp. 149-158, 2015.

[15] C. Pouzat, O. Mazor and G. Laurent, "Using noise signature to optimize spike-sorting and to assess neuronal classification quality," *Journal of Neuroscience Methods,* vol. 122, no. 1, pp. 43-57, 2002.

[16] J. W. Pillow, J. Shlens, E. J. Chichilnisky and E. P. Simoncelli, "A Model-Based Spike Sorting Algorithm for Removing Correlation Artifacts in Multi-Neuron Recordings," *PLoS One,* vol. 8, no. 5, 2013.

[17] A. Kessy, A. Lewin and K. Strimmer, "Optimal Whitening and Decorrelation," *The American Statistician,* 2017.

[18] R. Vollgraf, "Unsupervised Learning Methods for Statistical Signal Processing," Berlin, 2006.

[19] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics,* pp. 179-188, 1936.

[20] F. Franke, Real-Time Analysis of Extracellular Multielectrode Recordings, Berlin: Technische Universität Berlin, 2011.

[21] F. Franke, M. Natora, C. Boucsein, M. H. Munk and K. Obermayer, "An online spike detection and spike classification algorithm capable of instantaneous resolution of overlapping spikes," *Journal of computational neuroscience,* vol. 29, no. 1-2, pp. 127-148, 2010.

[22] O. Marre, D. Amodei, N. Deshmukh, K. Sadeghi, F. Soo, T. E. Holy and M. J. Berry, "Mapping a Complete Neural Population in the Retina," *The Journal of Neuroscience,* vol. 32, no. 43, pp. 14859-14873, 2012.

[23] A. Böttcher and S. M. Grudsky, Toeplitz Matrices, Asymptotic Linear Algebra, and Functional Analysis, vol. 13, Springer, 2000, pp. 404-424.

[24] F. Franke, R. Pröpper, H. Alle, P. Meier, J. R. P. Geiger, K. Obermayer and M. H. J. Munk, "Spike sorting of synchronous spikes from local neuron ensembles," *Journal Neurophysiology,* vol. 114, pp. 2535-2549, 2015.