frontiers in Neuroinformatics

TECHNOLOGY REPORT
published: 01 September 2017
doi: 10.3389/fninf.2017.00049

PRANAS: A New Platform for Retinal Analysis and Simulation

Bruno Cessac[1*†], *Pierre Kornprobst*[1], *Selim Kraria*[1], *Hassan Nasser*[1], *Daniela Pamplona*[1], *Geoffrey Portelli*[1] *and Thierry Viéville*[2]

[1] *Biovision Team, Inria, Université Côte d'Azur, Sophia Antipolis, France,* [2] *Inria, Mnemosyne Project Team, Bordeaux, France*

The retina encodes visual scenes by trains of action potentials that are sent to the brain via the optic nerve. In this paper, we describe a new free access user-end software allowing to better understand this coding. It is called PRANAS (https://pranas.inria.fr), standing for Platform for Retinal ANalysis And Simulation. PRANAS targets neuroscientists and modelers by providing a unique set of retina-related tools. PRANAS integrates a retina simulator allowing large scale simulations while keeping a strong biological plausibility and a toolbox for the analysis of spike train population statistics. The statistical method (entropy maximization under constraints) takes into account both spatial and temporal correlations as constraints, allowing to analyze the effects of memory on statistics. PRANAS also integrates a tool computing and representing in 3D (time-space) receptive fields. All these tools are accessible through a friendly graphical user interface. The most CPU-costly of them have been implemented to run in parallel.

Keywords: retina simulator, spike train statistics, population coding, maximum entropy, Gibbs distributions, large scale spiking activity, spike train generation, multi-electrode array recordings

OPEN ACCESS

Edited by:
Andrew P. Davison, Centre National de la Recherche Scientifique (CNRS), France

Reviewed by:
Thomas Wachtler, Ludwig-Maximilians-Universität München, Germany
Michael Denker, Forschungszentrum Jülich, Germany

***Correspondence:**
Bruno Cessac
bruno.cessac@inria.fr

[†] *The authors are listed in alphabetical order.*

Received: *27 October 2016*
Accepted: *17 July 2017*
Published: *01 September 2017*

Citation:
Cessac B, Kornprobst P, Kraria S, Nasser H, Pamplona D, Portelli G and Viéville T (2017) PRANAS: A New Platform for Retinal Analysis and Simulation. Front. Neuroinform. 11:49. doi: 10.3389/fninf.2017.00049

1. INTRODUCTION

The retina is one of the most developed sensing devices (Gollisch and Meister, 2010; Masland, 2011, 2012). It transforms the incoming light into a set of electrical impulses, called spikes, which are sent asynchronously to higher level structures in the visual cortex through the optic nerve. Although Cajal's neuron doctrine was postulated more than one century ago, how information is encoded and transmitted by neurons is still not entirely understood today. Especially, the role of spatio-temporal correlations in population coding raises up deep theoretical and practical questions that are far from being answered (Rieke et al., 1996; Cessac and Palacios, 2012), particularly for the visual information transmitted from the retina to the visual cortex. To address these questions and make progress in their understanding, one needs to develop joint modeling and experimental studies with efficient software to analyse data. In this paper we present a new Platform for Retinal ANalysis And Simulation called PRANAS (https://pranas.inria.fr). It was designed as a user-friendly tool dedicated to neuroscientist community in a large sense, i.e., not only experienced computational neuroscientists. It has two main goals, analyse retina data, especially spatio-temporal correlations, and simulate the spike response of the retina to a visual flow. This makes this tool a unique platform to better understand how the retina works.

The first goal of PRANAS is to provide methods to analyse retinal recordings at single cell and population levels. With the advent of new techniques, the recording of the simultaneous activity of groups of neurons provides a critical database to unravel the role of specific neural assemblies in spike coding. The acquisition capacity of Multi-Electrode Arrays (MEA) has been exponentially increasing over years (Stevenson and Kording, 2011). Systems like the 256–MEA has become a standard although new high-density MEA are now available such as the APS CMOS 4096–electrodes (Ferrea et al., 2012). Using these systems, one can record from hundreds to thousands of neurons simultaneously in the retina (and more generally *in vivo* or *in vitro*, from cultures of neurons, and from other brain areas). As one gains more intuitions and results on the importance of concerted activity in spike trains, models are developed to extract, from animal recordings, possible canonical principles underlying spike coding. This is a major challenge with many potential outcomes. Beyond the dream that population coding is ruled by a few first principles, several applications could emerge such as the development of artificial systems having similar levels of performance as biological systems. To this end, accurate tools are required to analyse and compare spike trains, experimental or computer generated ones. Progress has been made recently to analyse spike trains (Schneidman et al., 2006; Shlens et al., 2006; Marre et al., 2009; Roudi et al., 2009a,b; Tkačik et al., 2010; Roudi and Hertz, 2011; Tkačik et al., 2013). Especially, the spatio-temporal aspects (memory) and causality have been shown to be relevant for exploring neural activity (Cessac and Palacios, 2012). For instance, Vasquez et al. (2012) and Tang et al. (2008) demonstrated the importance of temporal statistics and Nasser et al. (2013b) and Nasser and Cessac (2014) developed new tools to analyse spatio-temporal activity for large scale spiking networks. These methods shed a new light on spike train analysis. However, they require time and expertise to be implemented efficiently, making them hard to use. The idea of developing a new software came from our motivation to share these recent developments with the neuroscience community in a broad sense. PRANAS integrates all our expertise in terms of spike trains statistical analysis. Note that other methods analyzing correlations in massively parallel data using completely different approaches have also been proposed (Hillar and Effenberger, 2015; Takahashi et al., 2015; Torre et al., 2016; Quaglio et al., 2017).

The second goal of PRANAS is to provide a customizable retina simulator which could evolve in synergy with experimental data analysis. Currently, there is a large and expanding body of literature concerning models of retinal processing. There are three main classes of models. The first class regroups the linear-nonlinear-poisson (LNP) models (Odermatt et al., 2012). LNP models can simulate the spiking activity of ganglion cells (and of cortical cells) in response to synthetic or natural images (Carandini et al., 2005) but they voluntarily ignore the neuronal mechanisms and the details of the inner retinal layers that transform the image into a continuous input to the ganglion cell (or any type of cell) stages. The second class of models has been developed to serve as a front-end for subsequent computer vision task. They provide bio-inspired modules for low level image processing. One interesting example is given by Benoit et al. (2010) and Hérault (2010), where the model includes parvocellular and magnocellular pathways using different non-separable spatio-temporal filter that are optimal for form- or motion-detection. The third class is based on detailed retinal models reproducing its circuitry, in order to predict the individual or collective responses measured at the ganglion cells level (Pelayo et al., 2004; Wohrer and Kornprobst, 2009; Lorach et al., 2012; Martinez-Alvarez et al., 2013). In PRANAS, we are interested in this third class of models because they allow to explore several aspects of retinal image processing such as (i) understanding how to reproduce accurately the statistics of the spiking activity at the population level (Nasser et al., 2013a), (ii) reconciling connectomics and simple computational rules for visual motion detection (Kim et al., 2014), and (iii) investigating how such canonical microcircuits can implement the different retinal processing modules cited in e.g., Gollisch and Meister (2010). More precisely, the PRANAS platform has integrated and extended the VIRTUAL RETINA simulator (Wohrer and Kornprobst, 2009)[1] initially developed in our team to do large scale retina simulations. VIRTUAL RETINA has been used in several theoretical studies (Masquelier, 2012; Mohemmed et al., 2012; Basalyga et al., 2013; Doutsi et al., 2015a,b; Vance et al., 2015).

This paper, aiming at presenting this new platform PRANAS, is organized as follows. In Section 2, we give an overview of PRANAS and compare it with a selection of other tools currently available focusing on spike train analysis methods. In Section 3, we present the main features of the software. Illustrations and step by step procedures are given in several cases allowing readers to reproduce them. In Section 4, we discuss future developments.

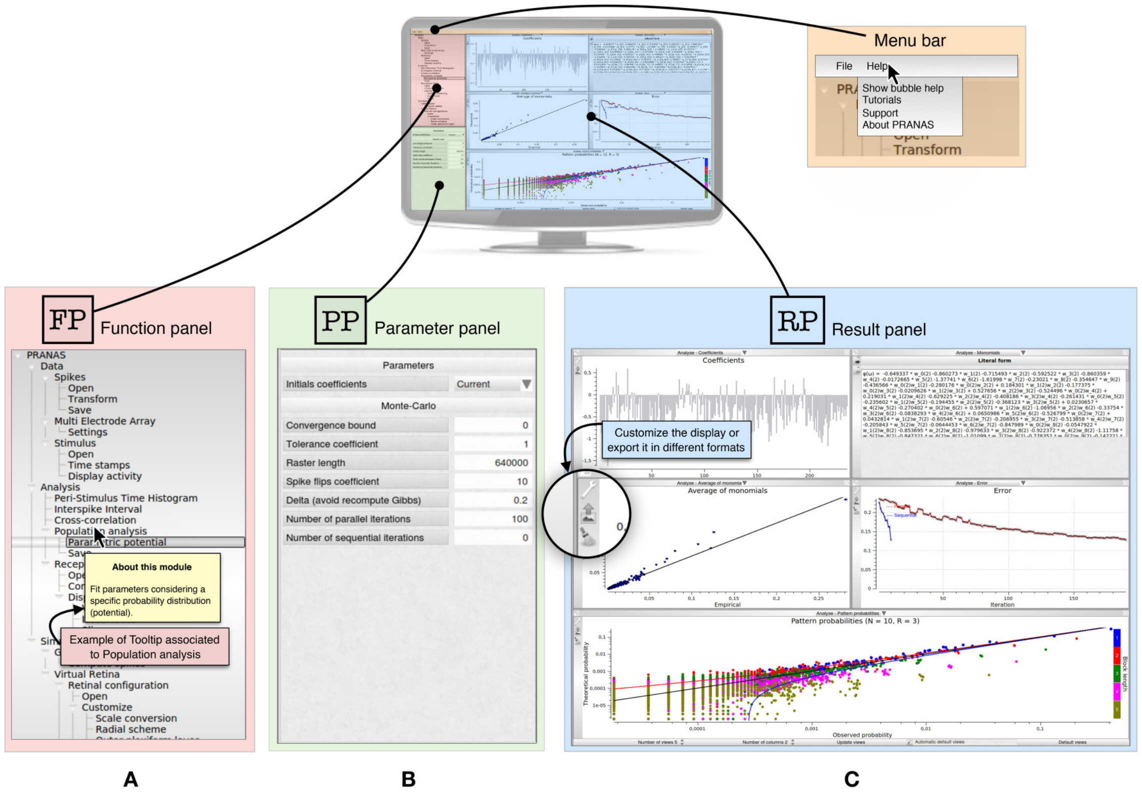
2. GENERAL PRESENTATION

PRANAS targets a broad community of scientists interested in exploring spike coding, in particular at retina level. It provides new tools for analyzing spike trains at the population level and several methods to generate them, either with a prescribed statistics (including spatio-temporal correlations) or by emulating the retinal response to a visual scene. This is done through a user-friendly Graphical User Interface (GUI). PRANAS runs on multiple platforms (Linux, Mac OS, Windows) and it supports parallel architectures. The software is provided as binary code or as source code on request upon acceptance on the terms of the license given on the website (http://pranas.inria.fr). Documentation, tutorials, and samples of spike trains are also available.

The GUI of PRANAS (version 1.0.0) has three main panels, as shown in **Figure 1**:

- Function panel (`FP`) is the main panel containing Input/Output interfaces and functions. It is organized in three sections: `Data`, `Analysis` and `Simulation`. (see Section 3 for more details).
- The parameter panel (`PP`) allows one to set parameters related to the chosen function from the function panel.
- The results panel (`RP`) contains the results. Several results can be displayed simultaneously in different subpanels, and each result can be exported as a figure in a variety of formats (e.g., PDF, PNG, JPG). The user can change the `Number of views` and which result should be displayed in each subpanel. In most subpanels, there are icons on the left-hand side to

[1]VIRTUAL RETINA website: http://virtualretina.inria.fr



Parameters

Initials coefficients	Current
Monte-Carlo	
Convergence bound	0
Tolerance coefficient	1
Raster length	640000
Spike flips coefficient	10
Delta (avoid recompute Gibbs)	0.2
Number of parallel iterations	100
Number of sequential iterations	0

FIGURE 1 | Presentation of PRANAS GUI panels. **(A)** Function panel (`FP`) is the main panel containing Input/Output interfaces and functions. It is organized in three sections: `Data`, `Analysis` and `Simulation`. Note that in the `Help` option from the menu bar users can select `Show bubble help` which will trigger tooltips giving information for each function. **(B)** Depending on the function selected, the parameter panel (`PP`) gives the opportunity to set the corresponding parameters. **(C)** Result panel (`RP`) can show all results the user want. The number of sub-panels and the windows to show can be changed. Note that for each result shown in a sub-panel, the user can customize the display or export it in different formats.

open, save, export or change plot settings (see **Figure 1C**). Another example is shown in **Figure 4A** (`Stimulus view`) where the user can export, show/hide grid and spikes and select neurons graphically.

Finally, another convenient feature of the software is that user can save what he has done in term of analysis in a HDF5 file (Zordan et al., 2015) (`File>Save`) and load it later to continue the work.

PRANAS is implemented in C++. It has its own dedicated libraries. It also uses other libraries for storage (HDF5), analysis (GSL, SFMT), display and GUI (Qt4, Qwt, VTK, CImg). Some parts of the software, especially the statistical estimations, run on multiple processors thanks to OpenMP framework. The software takes all the available processors in the machine automatically, without any interaction with the user. Parallelization allows to boost heavy computations and save processing time and memory.

To our best knowledge there is no other platform integrating together the functionalities proposed by PRANAS, to both analyse and stimulate retinal activities. There are however several platforms performing efficient spike train analysis. **Table 1** provides a qualitative comparison between PRANAS and a selection of such tools for spike train analysis. We note that the implementation of PRANAS as a stand-alone application in C++/Qt4 rather than a library (such as, e.g., the FIND toolbox) makes this tool readily available without the need of an external interpreter, such as Python or the commercially available Matlab suite (MathWorks, Natick, VA). Thus, access to PRANAS functions from a scripted analysis is performed via command line calls (see Section 4).

3. PRANAS MAIN FUNCTIONS

3.1. Data

In section `Data`, user can load spike trains files. Spikes may come either from real cells recordings or from a simulated spiking neural network. They can be imported from different formats such as simple text file (`.txt`), DAT file for Windows (`.dat`), NEXUS file (`.nex`) or HDF5 file format (`.hdf5`, see Zordan et al., 2015) allowing to store not only the spikes but also other information related to experimental protocole (e.g., time stamps, MEA characteristics).

Depending on how the spikes were generated, the user can also load other information such as:

TABLE 1 | Comparison between a selection of existing software for spike train analysis.

	CX	FD	NT	SL	NY	OY	ET	SR	NE	SV	PS
Version	**2.12**	**2.0**	**2.0**	**1.00**	**0.1**	**0.3.5**	**0.3.0**	**0.3-2**	**5.037**	**0.4.2**	**1.0.0**
Language	**Matlab**	**Matlab**	**Matlab**	**Matlab**	**Python**	**Python**	**Python**	**R**	**VB**	**Python**	**C++**
GUI	●	●		●		●			●	●	●
Scripting	●	●	●	●	●	●	●	●	●		
Free binary	●	●	●	●	●	●	●	●	●	●	●
File formats (Nex)		●		●	●	●	●		●	●	●
File formats (HDF5)		●				●	●			●	●
Rate histogram PSTH				●	●		●		●	●	●
Population PSTH	●	●		●	●		●		●	●	●
ISI	●	●		●	●		●	●	●	●	●
Cross-correlograms	●	●		●	●		●	●	●	●	●
Joint spike and stimulus visu.											●
STA					●	●	●		●		●
PCA									●		
Spectral analysis	●				●	●	●		●	●	
MaxtEnt (x/x-t)			GLM								●
Raster generation				●	●						●
...with Poisson/Non Poisson					●	●					●
...with VIRTUAL RETINA											●

Abbreviations for software names (first and last initials) have been chosen for the presentation. Following software is discussed: CX, CHRONUX (Ince et al., 2010); FD, FIND (Meier et al., 2008); NT, NSTAT (Goldberg et al., 2009); SL, SIGTOOL (Lidierth, 2009); NY, NEUROPY (Spacek et al., 2008); OY, OPENELECTROPHY (Garcia and Fourcaud-Trocmé, 2009); ET, ELEPHANT; SR, STAR (Pouzat and Chaffiol, 2009); NE, NEUROEXPLORER; SV, SPYKEVIEWER (Pröpper and Obermayer, 2013); PS, PRANAS. Note that features selected in this table have essentially been chosen according to what PRANAS does. It is not an exhaustive list and information applies to the time of writing. Software we mention can have additional features not commented herein.

- The MEA configuration for animal cell recordings, allowing the spiking activity to be displayed relative to the grid of electrode.
- The sequence of images in case of a visual experiment, allowing the spiking activity to be displayed together with the stimulus but also to emulate a retinal response (see Section 3.3.2).
- The image time stamps, i.e., the precise time at which each image was shown, allowing to perform Spike-Triggered-Average (STA) (Chichilnisky, 2001).

3.2. Analysis

3.2.1. Classical Analysis

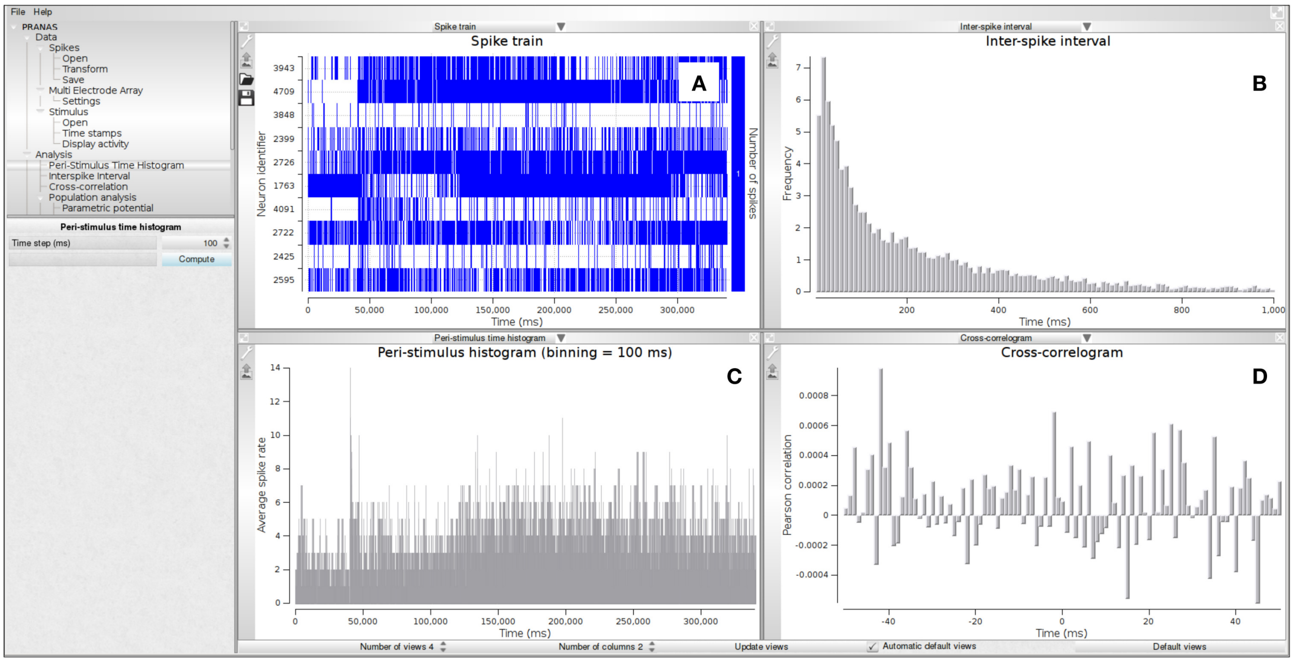
Once the spikes are available (simulated or imported), different analysis can be performed starting with basic visualizations and classical analysis such as peri-stimulus histogram, interspike interval, and cross-correlation (see **Figure 2** and Example 1). The number of neurons used for the analysis can be of several thousands, although the user can select subsets of neurons according to different criteria, e.g., individual activity or receptive field localization (if applicable, i.e., if data allows to compute receptive fields). Even for a large population of neurons (about 4,000 neurons), on any modern computer, the computational time of most of the functions in the classical analysis is neglectable.

Example 1 How to obtain the peri-stimulus time histogram (PSTH) on a selection of neurons having the highest spike rates?

1. `FP` `Data>Spikes>Open`
2. `PP` Browse and select the file of spikes.
3. `RP` (Optional) In the sub-panel showing the list of neurons, click on `Spike rate` to sort neurons with respect to their averaged spike rate, select the set of neurons with higher spike rate and click on `Keep selected`.
4. `FP` `Analysis>Peri-stimulus time histogram`.
5. `PP` Choose the `bin size` (time step) in milliseconds and click on `Compute`.

3.2.2. Population Analysis

With the current evolution of Multi-Electrode Array (MEA) recordings devices, it is possible to record simultaneously thousands of neurons (Ferrea et al., 2012). This opens up the possibility of analyzing the collective neuronal population activity, namely its spatio-temporal correlations. Characterizing spatio-temporal statistics is a fundamental step toward extracting general population coding principles in neuronal networks. For example, it has been observed in several vertebrate species that, even if the pairwise correlations of retinal ganglion cell are weak, they are necessary to explain the statistics of spikes (Ginsburg et al., 1984; Schneidman et al., 2006;



Time step (ms) 100

FIGURE 2 | Example of classical analysis for a spike train shown in sub-panel **(A)**. The following estimations are shown: **(B)** Inter-spike interval (ISI), **(C)** Peri-stimulus time histogram (PSTH) and **(D)** cross-correlogram. Data from electrode array recording of a mouse retina in response to a moving pattern.

Shlens et al., 2006; Tkačik et al., 2010; Greschner et al., 2011).

Analyzing spatio-temporal correlations is, however, a notoriously hard problem from a statistical perspective. The point is not only to measure pairwise or higher order correlations but also to address adequately the question of their significance in neuronal coding taking into account constraints such as limited statistical sample, the reliability of statistical tests, the hypotheses underlying mathematical models and risks of overfitting (Roudi et al., 2009a,b). One possible way of solving this problem is to use Gibbs distributions. Initially introduced in statistical physics by Boltzmann and Gibbs (Jaynes, 1957) but currently used in a broader context, this general class of probability distributions, constitutes a canonical paradigm to explain the spike statistics reproducing as close as possible empirical correlations, without adding additional unnecessary assumptions (Schneidman et al., 2006; Shlens et al., 2006; Marre et al., 2009; Tkačik et al., 2010, 2013; Ganmor et al., 2011a,b). This method has been used to study pairwise correlations (see Vasquez et al., 2012; Nasser et al., 2013b; Nasser and Cessac, 2014) as well as more general spatio-temporal correlations (Cessac et al., 2009; Vasquez et al., 2010; Cofré and Cessac, 2013; Nasser et al., 2013b; Cofre and Cessac, 2014; Herzog et al., 2014; Ravello et al., 2015; Herzog et al., in preparation).

PRANAS provides a toolbox to compute a Gibbs distribution from the neuronal data i.e., to estimate a probability distribution, reproducing as close as possible empirical space- and time-statistics of a raster, without adding additional unnecessary assumptions (maximizing entropy)[2]. In the current version, the population analysis toolbox of PRANAS focuses on stationary distributions, i.e., the statistics are invariant under time-translation. We give first a brief explanation of Gibbs distribution before explaining how to use PRANAS (see Vasquez et al., 2010; Nasser et al., 2013b; Nasser and Cessac, 2014 for more details).

Below, rasters are denoted by ω. We assume that time has been discretized with some time bin. The state of neuron i at time t is denoted by $\omega_i(t) \in \{0, 1\}$. We note by $\omega(t)$ the vector $\omega_i(t)$ (it tells us the spiking state of neurons at time t). Therefore, a raster ω is mathematically a matrix with N lines (number of neurons) and T columns (time bins in the raster). Obviously, it is not stored this way in memory (we do not store 0 s). A Gibbs distribution is a probability measure μ defined by a function E (also called "energy"). The probability of observing a raster ω of time length T, $\mu[\omega]$ is proportional to $e^{E(\omega)}$ where $E(\omega) = \sum_{t=1}^{T} \phi(\omega(t))$. In the following ϕ is called "potential." A paradigmatic example of Gibbs distribution

[2]Note that this toolbox first motivated the creation of the GUI to manipulate these complex notions. In former publications, software was named ENAS standing for Event Neural Assembly Simulations. After fusing of ENAS and the retina simulator VIRTUAL RETINA (see Section 3.3.2) into the same platform, we decided to choose a better suited name (PRANAS) to describe the functional scope of this unique platform allowing to (i) analyse spike trains coming from simulations or MEA recordings, at single cells or population levels, (ii) simulate retinal spike trains from a bio-plausible model, (iii) use additional dedicated tools for retinal recordings.

appears in the Ising[3] model. Here $\phi(\omega(t)) = \sum_i b_i\omega_i(t) + \sum_{i,j} J_{ij}\omega_i(t)\omega_j(t)$ where the sums hold on neurons indices. The terms b_i, J_{ij} are parameters tuning the probability whereas the terms $\omega_i(t), \omega_i(t)\omega_j(t)$ depend on the spike configuration and are called "interactions" (self-interaction for the term $\omega_i(t)$, and pairwise interactions $\omega_i(t)\omega_j(t)$). Here parameters b_i, J_{ij} are independent on time (stationarity assumption) so it is sufficient to define the potential ϕ at time $t = 0$.

In the Ising model, there is no coupling of spikes at different times so that the probability of observing a raster of length T factorizes into probabilities of spike states $\omega(t)$: consecutive time events are therefore independent. We say the statistical model has "no memory" in contrast to the Markovian model presented now. A natural generalization of the Ising form is indeed to define the potential ϕ as:

$$\phi(\omega) = \sum_{l=0}^{L} h_l m_l(\omega) \tag{1}$$

The terms h_l are parameters tuning the probability. They correspond to b_i, J_{ij} in Ising model but they tune more general spike interactions. Again, these parameters are independent of time (stationarity) so that we can define the potential ϕ starting from an initial time 0. The main difference with Ising model is that now interactions involve spikes at different times $t_1, \ldots, t_n$. These interactions correspond to the terms $m_l(\omega)$, with the general form $\omega_{i_1}(t_1)\ldots\omega_{i_n}(t_n)$, i.e., it involves spike events occurring at different times. As an immediate consequence, consecutive time events are not independent anymore. One can show the Gibbs distribution is, in this case, the invariant probability of a Markov chain. Thus, statistics involves memory and has non-vanishing time correlations. For historical reasons, related to the development of our research work, we call from now the m_ls "monomials" instead of "interactions." The monomials correspond thus to the conjunction of events in the raster, varying the space and time. For instance, the condition "neuron zero" is firing at time one, neuron two is firing at time zero and neuron three is firing at time two" corresponds to $m(\omega) = \omega_0(1)\omega_2(0)\omega_3(2)$. More generally, the times t_k defining a monomial are chosen in the interval $[0, D]$ where D is a positive integer characterizing the memory depth of the Markov chain associated with the Gibbs distribution. The model range is given by $R = D + 1$. The potential range is directly connected to the complexity of the algorithm analyzing data, as the higher the range the higher the computational time and memory load. Gibbs distributions satisfy a variational principle: maximizing the statistical entropy under the constraints that the average value of each monomial, $\mu\,[\,m_l\,]$ has a fixed value. Gibbs distributions are thus also called Maximum Entropy (MaxEnt) distributions. In our case, this value is equal to average empirical value $\pi\,[\,m_l\,]$ computed from an experimental raster. In other words, here is what our algorithm does: given an initial form of the potential ϕ, fixed by the user (see below), one seeks the parameters h_l maximizing the statistical entropy under the constraints $\mu\,[\,m_l\,] = \pi\,[\,m_l\,]$. In general, it is not possible to have the strict equality $\mu\,[\,m_l\,] = \pi\,[\,m_l\,]$ so one tries to approach it at best. Equivalently, we minimize the Kullback-Leibler divergence between μ, the "model" and π the empirical measure ("data"). In the ideal case of a raster infinite in time, this is a convex problem, and therefore there is a unique solution. For the realistic case, the input raster is finite, and several solutions can solve the problem (Cessac and Palacios, 2012). The algorithm starts from an initial guess of the h_ls and computes the corresponding averages $\mu(m_l)$ using a Monte Carlo method. From this, it computes a variation of the h_ls giving, if possible, a lower Kullback-Leibler divergence (see Nasser and Cessac, 2014 for details). We proceed this way until no improvement in the Kullback-Leibler divergence is observed any more. At this point, modifying some parameters (see next paragraph) can nevertheless still improve the minimization.

Let us now describe how to compute Gibbs distributions with PRANAS. There are many (an exponential number) a priori possible potentials. In PRANAS we propose four predefined potentials although the user can also define his own one. The options are the following. Here we characterize the potential by the list of monomials $\mathcal{M}$ which compose it.

- `Bernoulli model`: It takes into account individual neurons activity where all neurons are independent. The potential reads $\phi(\omega) = \sum_{l=1}^{N} h_l\omega_l(0)$ where N is the number of neurons. Hence, monomials are of type "neuron i is firing at time t" (Example: $\mathcal{M} = \{\omega_1(0), \omega_2(0), \omega_3(0)\}$ for 3 neurons). This model has range 1 .
- `Ising model`: This paradigmatic model from statistical physics has been used in neuroscience in several papers such as Schneidman et al. (2006), Shlens et al. (2006), Tkačik et al. (2006), and Tkačik et al. (2009). The corresponding potential has been introduced above. The monomials are of type "neuron i is firing at time t" and "neuron i and neuron j are simultaneously firing at time t." Thus events are instantaneous, and this model does not involve memory and causality (consecutive times are independent under μ, the Gibbs distribution). This model has range 1 (Example for 3 neurons: $\mathcal{M} = \{\omega_1(0), \omega_2(0), \omega_3(0), \omega_1(0)\omega_2(0), \omega_1(0)\omega_3(0), \omega_2(0)\omega_3(0)\}$).
- `Pairwise + Triplets model`: This model has been introduced by Ganmor et al. (2011a,b). In addition to Ising terms, there are triplets of interactions (e.g., $\omega_1(0)\omega_2(0)\omega_3(0)$). This model too has range 1.
- `Pairwise model`: This is an extension of Ising model where monomials are of type "neuron i is firing at time t" (single events) and "neuron i is firing at time t and neuron j is firing at time $t + k$," $0 \le k \le D$ (pairwise events). This model integrates memory and causality via time dependent pairwise interactions. (Example with range 2 and 2 neurons: $\mathcal{M} = \{\omega_1(0), \omega_2(0), \omega_1(0)\omega_2(0), \omega_1(0)\omega_1(1), \omega_2(0)\omega_2(1), \omega_1(0)\omega_2(1), \omega_1(1)\omega_2(0)\}$).

[3]The model initially proposed by Ising and Lenz in 1920 had only nearest neighbors interactions and constant couplings. The potential form we use corresponds in fact to a spin-glass but we use here the terminology found in computational neuroscience papers.

- `User`: Here the user defines his own potential (Example $\mathcal{M} = \{\omega_1(0), \omega_2(0), \omega_3(0)\omega_1(0)\omega_2(1)\omega_3(2), \dots\}$).

Example 2 How to calculate the Gibbs distribution of a spike train?

1. $\boxed{\text{FP}}$ `Data>Spikes>Open`
2. $\boxed{\text{PP}}$ Browse and select the file of spikes.
3. $\boxed{\text{RP}}$ (Optional) In the sub-panel showing the list of neurons, select a subset of neurons and click on `Keep selected`.
4. $\boxed{\text{FP}}$ `Analysis>Population analysis`.
5. $\boxed{\text{PP}}$ Define the `Potential`, either from a file or by setting the type (e.g., `pairwise`) and the range[4] (e.g., $R = 3$). Set the `Maximal pattern length`. For example, if you select 2, the program will seek *in the data* all spike events occurring within 2 successive time steps, and compare the probability of these events, predicted by the model, to the empirical probabilities. In `Simulation`, choose what you want to compute. Possible choices are:
   (a) `Potential`: Fits the coefficients of the potential.
   (b) `Kullback-Leibler divergence`: An estimation of Kullback-Leibler divergence derived from the entropy estimation in Strong et al. (1998) and the classical relations between K-L divergence, μ average of ϕ and entropy (Cessac and Palacios, 2012).
   (c) `Pattern probabilities`: The confidence plot.
6. $\boxed{\text{FP}}$ `Analysis>Population analysis>Parametric potential`
7. $\boxed{\text{PP}}$ Choose `Initial Coefficients`: possible choices are `Current` (in the case when you have made a run before and you want to keep h_ls) or `0`. For the first run you can choose both as the initial parameters are anyway set to 0 (this corresponds to start from a Bernoulli models where spikes are independent with probability of occurrence $\frac{1}{2}$). Set parameters of the Monte Carlo method to speed up the process (see text for the role of each parameter).
8. $\boxed{\text{FP}}$ `Population analysis`
9. $\boxed{\text{RP}}$ `Compute` to do the estimation. The red bar at the bottom indicates that computation is in progress.

Note that in PRANAS the user can fine-tune the probability estimation and speed up the computation in the first steps of the Monte Carlo method by acting on the following parameters (in $\boxed{\text{FP}}$ `Analysis>Population analysis>Parametric potential`):

- `Convergence bound`: The lower bound of error you require. Simulations stops when the code reaches this value.
- `Tolerance Coefficient`: Allows to filter monomials to compute h_ls. For example, if the event m_l appears only, say 3 times in the raster you may consider that it is not significant and must be eliminated from the potential (1). In this case you set `Tolerance Coefficient` to 3.
- `Raster length`: The length of the Monte Carlo raster used to compute the average of monomials for the current values of h_ls. This number must increase as you get closer to the solution (typically when you do not get any improvement in the error).
- `Spike flips coefficients`: This defines the number of flips per neuron and per iteration in Monte Carlo methods (Nasser and Cessac, 2014).
- `Delta`: If the distance between predicted and empirical distribution is smaller than `Delta` the Monte Carlo raster is not recomputed. Instead, the average value of monomials is computed via linear response (Nasser and Cessac, 2014). The number must be decreased as you get closer to the solution (typically when you do not get any improvement in the error).
- `Number of parallel iterations`: We use two kinds of updating, based upon Dudík et al. (2004): parallel or sequential. Parallel update sets all h_ls in one step. `Number of parallel iterations` tells how many parallel updating of h_ls are done before the program stops (and e.g., plots the confidence plot, when selected).
- `Number of sequential iterations`: Sequential update. It computes only one h_l at each step. It is better to use it at the end of the computation, to fine-tune coefficients.

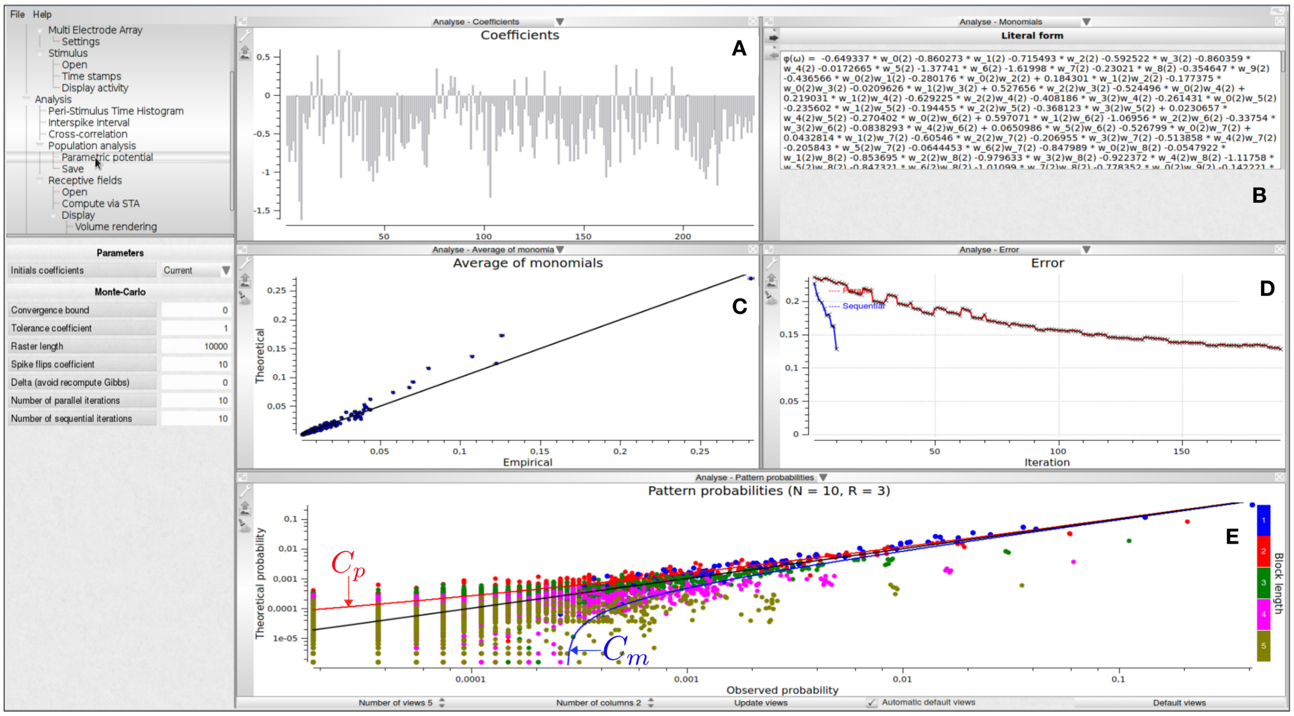
Figure 3 shows different visualizations of the results. In **Figure 3A**, we show a graphical view of the potential's coefficients. In **Figure 3B**, we show the literal form of the potential $\phi(\omega)$. In this example, it is

$$\begin{aligned}\phi(\omega) = &- 0.860273\, w_1(2) - 0.715493\, w_2(2) + \dots \\ &- 0.280176\, w_0(2)w_2(2) \\ &+ \dots - 0.120281\, w_0(2)w_3(2)w_5(3) + \dots,\end{aligned} \quad (2)$$

where, e.g., $w_0(2)w_2(2)$ corresponds to the event neuron zero and neuron two are firing at time two whereas -0.280176 is the corresponding coefficient h_l of this term in the potential. In **Figure 3C**, we show a plot of monomials average value, with empirical average on the abscissa and theoretical average (predicted by μ) on the ordinate. In **Figure 3D**, we show the evolution of the Hellinger[5] distance between the model predictions and empirical data, vs. the number of iterations. Hellinger distance is natural here as it relies directly on the estimation of the average value of monomials. It provides a quantitative measure of the goodness of fit. Finally, in **Figure 3E** we show a *confidence plot*. This figure consists of plotting, in log scale, on the abscissa the empirical probability of observed spike blocks and on the ordinate the expected probability of those blocks for the model μ. If the matching were perfect one would have all points aligning on the diagonal. This perfect matching is only possible, however, if the input spike train were infinite.

[4]The product $N \times R$ somewhat defines the complexity of the model. The larger $N \times R$ the longer the simulation and the memory load. We have been able to run up to $N = 100, R = 2$ in reasonable times. Note however that the statistical estimation of a so-huge number of neurons raises the classical problems (*not inherent to our method*) of statistical estimation on a sample, which, e.g., for the retina rarely exceeds a few millions of spikes.

[5]The Hellinger distance between the empirical probability π and the Gibbs probability μ is $d(\pi, \mu) = \frac{1}{\sqrt{2}}\sqrt{\sum_{l=1}^{L}\left(\sqrt{\pi(m_l)} - \sqrt{\mu(m_l)}\right)^2}$.



Parameters

Initials coefficients	Current

Monte-Carlo

Convergence bound	0
Tolerance coefficient	1
Raster length	10000
Spike flips coefficient	10
Delta (avoid recompute Gibbs)	0
Number of parallel iterations	10
Number of sequential iterations	10

```
φ(ω) =  -0.649337 * w_0(2) -0.860273 * w_1(2) -0.715493 * w_2(2) -0.592522 * w_3(2) -0.860359 *
w_4(2) -0.0172665 * w_5(2) -1.37741 * w_6(2) -1.61998 * w_7(2) -0.23021 * w_8(2) -0.354647 * w_9(2)
-0.436566 * w_0(2)w_1(2) -0.280176 * w_0(2)w_2(2) + 0.184301 * w_1(2)w_2(2) -0.177375 *
w_0(2)w_3(2) -0.0209626 * w_1(2)w_3(2) + 0.527656 * w_2(2)w_3(2) -0.524496 * w_0(2)w_4(2) +
0.219031 * w_1(2)w_4(2) -0.629225 * w_2(2)w_4(2) -0.408186 * w_3(2)w_4(2) -0.261431 * w_0(2)w_5(2)
-0.235602 * w_1(2)w_5(2) -0.194455 * w_2(2)w_5(2) -0.368123 * w_3(2)w_5(2) + 0.0230657 *
w_4(2)w_5(2) -0.270402 * w_0(2)w_6(2) + 0.597071 * w_1(2)w_6(2) -1.06956 * w_2(2)w_6(2) -0.33754 *
w_3(2)w_6(2) -0.0838293 * w_4(2)w_6(2) + 0.0650986 * w_5(2)w_6(2) -0.526799 * w_0(2)w_7(2) +
0.0432814 * w_1(2)w_7(2) -0.60546 * w_2(2)w_7(2) -0.206955 * w_3(2)w_7(2) -0.513858 * w_4(2)w_7(2)
-0.205843 * w_5(2)w_7(2) -0.0644453 * w_6(2)w_7(2) -0.847989 * w_0(2)w_8(2) -0.0547922 *
w_1(2)w_8(2) -0.853695 * w_2(2)w_8(2) -0.979633 * w_3(2)w_8(2) -0.922372 * w_4(2)w_8(2) -1.11758 *
w_5(2)w_8(2) -0.847321 * w_6(2)w_8(2) -1.01099 * w_7(2)w_8(2) -0.778352 * w_0(2)w_9(2) -0.142221 *
```

FIGURE 3 | Population analysis view with $N = 10, R = 2$. **(A)** Graphic view of the potential's coefficients values; the abscissa corresponds to an index associated to a prescribed order on monomials: the N first coefficients corresponds to single events (hence are related to firing rate); by clicking on the histogram bar one makes appear the coefficient value and the corresponding monomial. Ordinate give the value of the coefficient attached to the corresponding monomial. **(B)** Literal form of the potential. **(C)** Theoretical average value of monomials vs. their empirical average value. To which extents the model predicts the empirical data is drawn with this representation. This allows checking that constraints on empirical averages are respected by the model. **(D)** Decay of estimation errors vs. the number of iterations allowing us to graphically check the convergence of the estimation algorithm (red curve). In the figure several successive runs have been performed. The blue curve shows the evolution of error after each of these trials. **(E)** Confidence plot. C_p, C_m corresponds, respectively, to the upper and lower 3σ bound associated with Central Limit Theorem (see text).

For finite spike trains fluctuations about the exact probability are observed, ruled by Central Limit Theorem. Namely, fluctuations are Gaussian with a variance σ_l, depending on the monomial l and proportional to $\frac{1}{\sqrt{T}}$, T being the raster length (number of time bins). Around the diagonal are drawn error bounds corresponding to $3\sigma_l$, where σ is an estimated empirically. These bounds define a confidence region: if the Gibbs distribution is a perfect estimation of the input raster then points in the confidence plot are distributed inside the confidence region with a probability of 99.7% of the expected averages. The confidence plot provides a qualitative measure of the goodness of the fit.

Reducing the computation time has been a primary concern for us during the development phase. In fact, the most time-consuming routine of PRANAS is for inferring the Gibbs Potential. To face this problem, we introduced a new algorithm based on Monte Carlo sampling (Nasser et al., 2013b). For a large number of neurons, the Monte Carlo-based algorithm offers better computation time than the algorithm proposed in Vasquez et al. (2010). With a cluster of 64 cores of 2.4 GHz speed, PRANAS needs the following amount of time to compute a target distribution (1 iteration): 5 min for 20 neurons with a pairwise model $R = 2$, 10 min for 40 neurons with an Ising model. To infer the Gibbs Distribution, one needs 100–200 iterations on the h_l estimation. For more details on the scalability of this method see Nasser et al. (2013b).

3.2.3. Receptive Fields Estimation

Beyond tools to analyse spike trains at the single cell and population levels, PRANAS provides specific instruments in the case when spike trains come from evoked activity of neurons from the retina. The user can upload MEA characteristics, the sequences of images of the stimulus and the time stamps. Given this information, PRANAS provides a method to estimate the neuron's receptive fields. The method available in version 1.0.0 is Spike Triggered Average (STA) (Chichilnisky, 2001). The STA can be computed for a single neuron, a subset of neurons or the entire population. Example 3 describes the general procedure.

Example 3 How to estimate receptive fields with STA method?[6]

1. $\boxed{\texttt{FP}}$ `Data>Spikes>Open`

[6]To test this example, one can use sample data available on PRANAS website: recording called mouse-P39_17_06_14 (We are indebted to Evelyne Sernagor and Gerrit Hilgen, Newcastle University, who provided us these data and authorized us to use them as a basis for examples.)

2. `PP` Browse and select the file of spikes. Note that the user can select a subset of neurons as in Example 1.
3. `FP` `Data>Multi-Electrode Array>Settings`
4. `PP` Define the MEA configuration, so that, if neurons have a position relative to the array, the spike activity can be displayed at the correct location.
5. `RP` `Stimulus view` shows the activity of individual neurons w.r.t. MEA array. The user may choose a region of interest to select a subset of neurons. Note that if you have a TEXT file for spikes, no position is available, and we chose to arrange neurons following their order, by rows and columns.
6. `FP` `Data>Stimulus>Open`
7. `PP` Browse to choose the directory containing the sequence of white noise images. Note that all images should be in this directory and ordered by name. All standard formats are accepted (e.g., PNG, JPG, TIFF).
8. `FP` `Data>Stimulus>Time stamps`: Time stamps of the exact offset of each image during the experiment.
9. `PP` Browse and select the file. Time stamps can be imported from a `.txt` (row *n* contains the starting time of image *n*) or HDF5 (Zordan et al., 2015).
10. `FP` `Analysis>Receptive fields>Compute via STA`
11. `RP` `Output directory`: Choose the directory where the receptive fields will be saved, as a sequence of PNG files and in HDF5 format. Choose the number of images to be averaged before each spike with `Time depth (number of slices)`. Then click on `Compute`.
12. `FP` `Analysis>Receptive fields>Display` allows four types of visualization.
13. `RP` The user can display the receptive field of interest by selecting the neuron in `Receptive Fields` subpanel and `Update views`.

A receptive field is stored in a regular grid in a three-dimensional space. Each voxel in that grid, denoted by $RF(x, y, z)$, stores the value of the receptive field at each spatial position $(x \times s_d, y \times s_d)$ and temporal depth $z \times t_d$, where s_d and t_d denote, respectively, the spatial and temporal resolutions. The temporal resolution is estimated from the time stamps file as the average difference between two successive time stamps. The number of time slices of the receptive field, n_f, is a user-defined parameter. The first spatial slice $(x, y, z = 0)$ corresponds to the average spike triggered stimuli at time $-(n_f - 1)td$ before the spike, the second slice $(x, y, z = 1)$ corresponds to the average spike triggered stimuli at time $-(n_f - 2) \times t_d$ before the spikes and so on. In PRANAS we propose several displays of that volume (2D and 3D), as illustrated in **Figure 4**.

The time spent on the estimation of receptive fields depends on the parameters. For example, on a computer equipped with a Intel Core i7@2.8 GHz and 32 GB of memory it takes about 2 min to estimate the receptive fields of 4,000 neurons with 64 × 64 × 12 voxels each.

3.3. Simulation of Spike Trains

3.3.1. Simulation of Spike Trains from Statistics

PRANAS gives the possibility to generate spike trains from Gibbs distributions. This can be useful if, for example, one wants to generate spike trains with prescribed spatio-temporal correlations so as to test a statistical method of analysis. There are two different ways:

1. `FP` `Simulation>ComputeSpikes>GibbsRaster>current`: if distribution comes from a population analysis (see Section 3.2.2).
2. `FP` `Simulation>ComputeSpikes>GibbsRaster>File`: if distribution has been defined by the user and stored in a file containing the Gibbs potential form.

A Gibbs probability distribution (defined by Gibbs potential of the form 1) is naturally associated with a Markov chain, with memory $D = R - 1$, whose transition probabilities can be computed from the potential ϕ (Cessac and Palacios, 2012; Vasquez et al., 2012; Cofre and Cessac, 2014). The invariant distribution μ of this chain is the Gibbs distribution associated to ϕ. Therefore, given a potential of the form (1) it is easy to generate a sample raster distributed according to μ using Monte Carlo method (Nasser et al., 2013b). PRANAS affords this functionality allowing to generate rasters with spatio-temporal correlations tuned by the parameters h_l in ϕ.

3.3.2. Simulation of Spike Trains from a Retina Simulator

Another strength of PRANAS is to provide a way to generate spike trains that mimic retina's outputs. To do so, PRANAS has integrated and extended the VIRTUAL RETINA simulator (Wohrer and Kornprobst, 2009) formerly developed in our team[7].

In a nutshell, VIRTUAL RETINA is a software to perform large-scale simulations of biologically-plausible retinas. Given a retina configuration which can be fully customized and an input video as visual stimulus, VIRTUAL RETINA simulates the spiking output for different cell types. VIRTUAL RETINA has been shown to reproduce a broad range of experimental data at single cell level, from salamander, cat and primate retinas, and has been used in several theoretical studies (Masmoudi et al., 2010; Masquelier, 2012; Mohemmed et al., 2012; Basalyga et al., 2013; Doutsi et al., 2015a,b; Vance et al., 2015). The underlying model includes a non-separable spatio-temporal linear model of filtering in the Outer Plexiform Layer, a shunting feedback at the level of bipolar cells, and a spike generation process using a network generalization of the noisy leaky integrate-and-fire neurons to model Ganglion Cells (GCells). Note that VIRTUAL RETINA was designed to be used via a command line (there was no GUI available).

By integrating VIRTUAL RETINA software into PRANAS platform, we allowed this simulation software to be used through a GUI. We also extended VIRTUAL RETINA with the aim to

[7]VIRTUAL RETINA website: http://virtualretina.inria.fr. VIRTUAL RETINA is under Inria CeCILL-C license, IDDN.FR.001.210034.000.S.P.2007.000.31235.

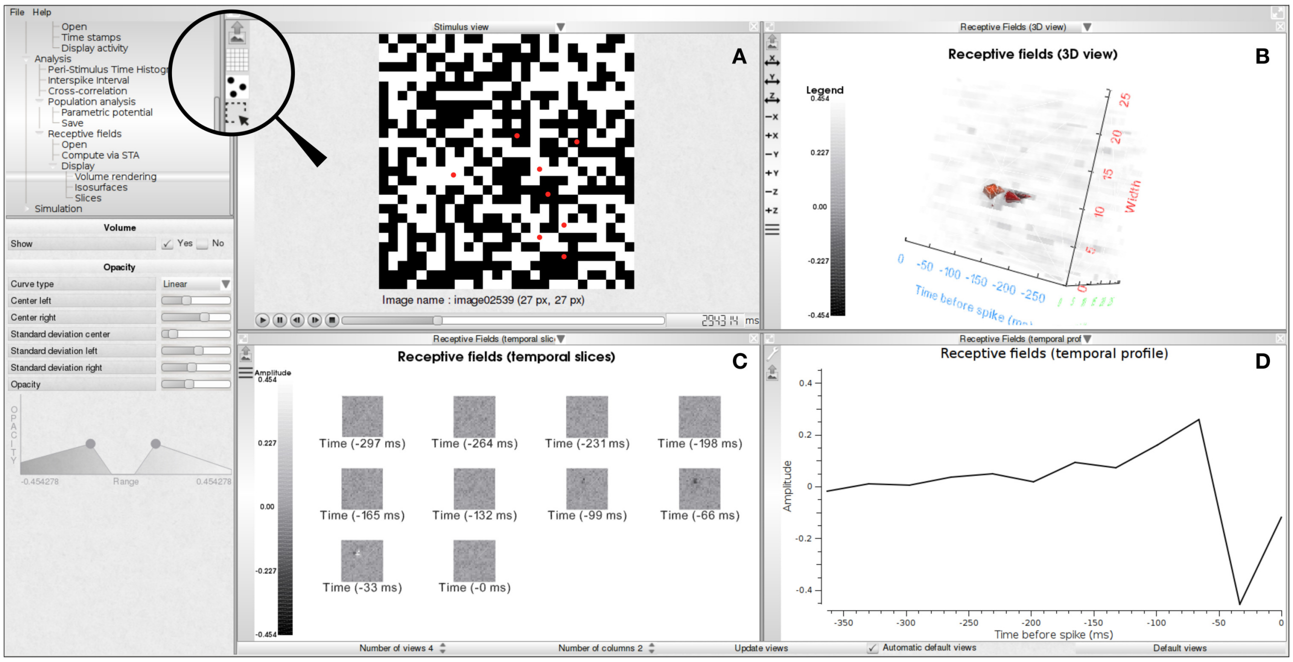


FIGURE 4 | Receptive field estimation using the STA approach. **(A)** White noise image sequence with spiking activity superimposed as red dots. **(B)** 3D volume visualization of the receptive field of one neuron. **(C)** Temporal slices of the receptive field ($RF(\cdot, \cdot, z)$). Slices are in increasing time order from top to bottom and left to right. Time period between two slices is 33 ms. **(D)** Temporal profile passing through the peak activity location of the spatio-temporal volume ($RF(x_M, y_M, \cdot)$ where $(x_M, y_M, z_M) = \text{argmax}_{x,y,z} RF(x, y, z)$).

reproduce statistically coherent responses at a population level. Indeed, outputs of the initial VIRTUAL RETINA version were successfully compared to experimental data at single cell level but could not reproduce collective statistics at a population level. This is because GCells in the early version were not connected. They were modeled by independent leaky-integrate and fire neurons receiving their input from bipolar cells, but with no lateral connectivity (due to amacrine cells—ACells—in the retina). As a consequence correlations in GCells spikes were only due to statistics of the stimulus and overlapping receptive fields.

In the new version of the retina simulator embedded in PRANAS, GCells are connected laterally so as to explore the effects of connectivity on retinal responses to stimuli. More precisely, the IPL is now composed of discrete time leaky-integrate and fire neurons introduced in Soula et al. (2006) and studied mathematically in Cessac (2008, 2011). We have chosen this model because of its simplicity, fast response, and the fact that its collective dynamics are well known. In a nutshell, the model comes from the time discretization of the leaky-integrate and fire model whose sub-threshold dynamics reads:

$$C\frac{dV}{dt} = -g_L(V - V_L) + I(t) + \sigma_B B(t), \quad (3)$$

for $V < \theta$, where θ is the firing threshold. Here C is the membrane capacity, g_L the leak conductance, V_L is the leak reversal potential, $B(t)$ is a Gaussian noise (mean zero and variance 1) and σ_B controls the amplitude of this noise. Finally, $I(t)$ is an external current. In our case, this is the bipolar current coming from the OPL (see Wohrer and Kornprobst, 2009 for details).

Let $\tau_L = \frac{C}{g_L}$ be the leak characteristic time. We consider now a time discretization with a time step dt (here it is fixed, $dt = 1$ ms) and we set $\gamma = \left(1 - \frac{dt}{\tau_L}\right)$. Note that dt has to be quite smaller than τ_L to have a reasonable description of the biophysics. Therefore $\gamma \in [0, 1]$. Then, the full dynamics (below and above threshold) of the membrane potential V_i of neuron i is given by:

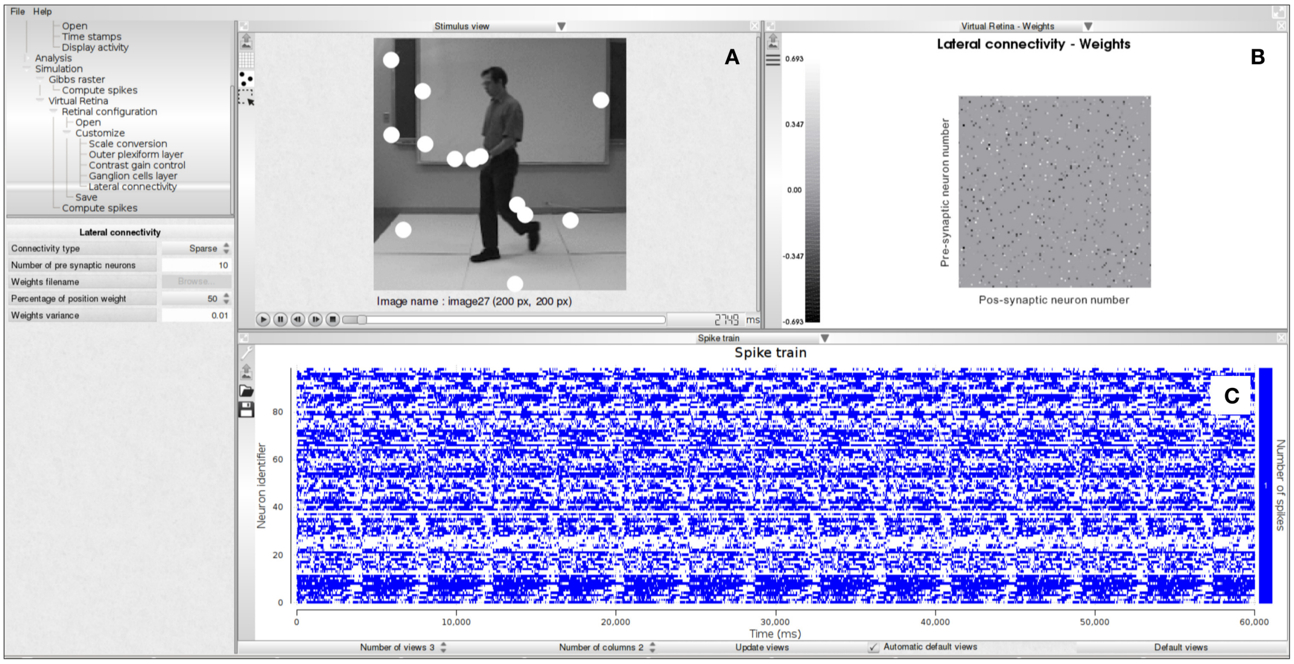
$$V_i(t+1) = \gamma V_i(t)(1 - Z_i) + \frac{I_i(t)}{C} + \frac{\sigma_B}{C}B(t),$$

where Z is the binary spike label:

$$Z_i = \begin{cases} 1, & \text{if } V_i \geq \theta; \\ 0, & \text{otherwise.} \end{cases}$$

Thus Z_i is 1 whenever neuron i spikes. Here $1 - Z_i$ models the reset. In this equation, neurons are not coupled. We add then a connectivity through synaptic weights W_{ij} where neuron j (pre synaptic) acts on neuron i (post synaptic) upon spiking. We have now:

$$V_i(t+1) = \gamma V_i(t)(1 - Z_i) + \frac{I_i(t)}{C} + \frac{1}{C}\sum_{j=1}^{N} W_{ij}Z_j + \frac{\sigma_B}{C}B(t). \quad (4)$$



Lateral connectivity

Connectivity type	Sparse
Number of pre synaptic neurons	10
Weights filename	Browse...
Percentage of position weight	50
Weights variance	0.01

FIGURE 5 | Spike generation from our retina simulator. **(A)** Input image sequence with spiking activity super-imposed (white blobs). **(B)** Connectivity matrix chosen for the simulation, here random. **(C)** Simulated spike train.

For a complete description see Cessac (2008, 2011). Clearly, this modeling of lateral connectivity, although usual in the field of neural networks models, is rough when compared to the real lateral connectivity in the IPL involving amacrine cells having a complex dynamics. Elaborating more realistic coupling is under current investigations.

The simulator allows to tune the parameters leak ($\gamma \in [0, 1]$); neuronal noise ($\sigma_B > 0$); membrane capacity (C); threshold (θ) and the connectivity matrix (W). The connectivity between neurons is set through the connectivity matrix. One can simulate independent neurons by choosing `none` or connected neurons using `sparse` or `dense` schemes. One can also upload a predefined connectivity matrix offering the possibility to explore its impact on spike statistics. For example, we provide on the website an example of ACells-like connectivity pattern.

Example 4 shows how to generate spikes using PRANAS and **Figure 5** illustrates an example of simulation. Note that user can find online retina configuration files and videos sequences to test the simulator.

Example 4 Running the retina simulator[8].

1. FP `Data>Stimulus>Open`
2. PP Browse to choose the directory containing the sequence.
3. RP `Stimulus sequences` subpanel: The sequence of images is loaded with a default display duration of 100 ms per image (`Display time` column). This value can be manually set or defined from time stamps file (see Example 3).
4. FP `Simulation>Virtual Retina>Retinal configuration>Open`[9]
5. PP Browse and select the XML files containing the parameters of the retina.
6. FP `Simulation>Virtual Retina>Retinal configuration>Customize`: Once loaded, the user can still change parameters of the retina.
7. FP `Simulation>Virtual Retina>Compute spike`.
8. PP `Compute` to get the spikes.

The time spent in the generation of spikes with the retina simulator depends on the parameters. For example it takes about 6 min to generate the spikes corresponding to **Figure 5** on a laptop equipped with a Intel Core i7-6820HK CPU@2.70 GHz. This was obtained with the image sequence `martin-walk` (200 × 200 pixels, 41 frames) and the `cat_X_cell_DLIF.xml` retina configuration file (100 neurons, sparse connectivity with 1,000 connections out of 10,000 possible connections).

[8]To run this example one can use data available on PRANAS website, for example the `martin-walk` image sequence and the `cat_X_cell_DLIF.xml` retina configuration file which corresponds to the result shown in **Figure 5**.

[9]Note that steps 4–6 can be skipped and one can directly generate spikes in step 7 using a proposed default configuration.

4. CONCLUSION

In this paper we have introduced PRANAS, a new free[10] platform for retinal analysis and simulation. PRANAS provides a retina simulator allowing to transform any video into spike trains, together with sophisticated methods to analyse spike trains, coming from simulations but also experiments. As explained, PRANAS (version 1.0.0) includes a range of original methods from the state-of-the-art (Wohrer and Kornprobst, 2009; Cessac and Palacios, 2012) which are now available to the neuroscience community. More precisely, PRANAS intends to be useful for the neuroscience community in a large sense, i.e., not only experienced computational neuroscientists. To do so, we decided to make all methods implemented therein accessible through a friendly GUI. No scripting is needed to run analysis and simulation methods. This represents a major advantage to promote interdisciplinary neuroscience and we believe that PRANAS helps in this direction.

However, we are certainly aware that this advantage could also be seen as a limitation of the software. Indeed, inability to script analyses could be a limitation that can make systematic analysis of large datasets tedious and error prone. It makes also difficult to link the functionalities of this toolbox to more extensive data analysis workflow. For thoses reasons, we also offer some functionalities that run on the command line. A non-exhaustive list of tools that can be used from the command line is `SimulateRetina` (simulate retinal response), `Correlations` (compute the average cross-correlations of a given raster), `SpikeTriggerAverage` (compute the average spike triggered stimuli)[11]. We expect to have the platform fully available with scripting in a forthcoming version.

In this first release of PRANAS, we have focused on the following aspects:

- The user experience: We optimized the ergonomics of interactions between the user and the interface. No scripting is needed to analyse or generate the spike trains.
- The richness of toolboxes: We developed a variety of methods ranging from classical tools to population analysis, including a method for receptive fields estimation. In addition, we provide two simulators, one inspired in the retina, the other from rasters statistics analysis.
- The computational performance: We parallelized a selection of functions to make possible to handle faster large populations of neurons[12].

By putting together functions related to retina simulation and analysis, PRANAS intends to be an original platform that will encourage joint modeling and experimental studies. The synergy between these two areas of functionality will be in particular useful to define better retina simulators by confronting simulated output w.r.t. real cell recordings. A typical use case is to start from a real cell recording of a retina, to analyse its receptive field structure, and use that information to define the XML retina configuration file so to define a virtual retina having similar characteristics as the real one. Another use case that we target is to start again from a real cell recording of a retina, analyse spike train statistics and compare them with the ones given by the simulator. Such a comparison is very informative and will suggest improvements of the retina simulator to increase its biological plausibility.

Here, we presented its main features, and we refer the interested reader to its website[13] for downloads and more information (see e.g., tutorials to get started). We hope that PRANAS becomes a useful tool for neuroscientists to analyse spike trains and we expect to improve it thanks to the users' feedback. Our goal is to progressively enrich PRANAS with the latest research results, to facilitate the transfer of new methods to the community. Future work will focus on improving several existing functions regarding efficiency in time and memory consumptions. We will include methods for population analysis in the non-stationary case as well as other statistical analysis models. We are working on better methods for receptive fields estimation (Drogoul et al., 2016). We are also developing extensions of the retina simulator model so that it could serve as a satisfactory model at a population level. Finally, as a general objective, we also plan to improve the computational power of PRANAS in order to handle even larger networks.

AUTHOR CONTRIBUTIONS

BC supervised the general development of PRANAS and has especially worked on the population analysis toolbox. PK contributed to the GUI design and is one of the developers of VIRTUAL RETINA. SK contributed to the software development, packaging, and GUI design. HN contributed to the population analysis toolbox. DP contributed to the receptive field and classical analysis toolboxes and to the integration and extension of VIRTUAL RETINA in PRANAS. GP contributed to the GUI design and the classical analysis toolbox. TV contributed to the software development. The authors are listed in alphabetical order.

FUNDING

This work was supported by KEOPS ANR-CONICYT, the EC IP project FP7-ICT-2011-9 no. 600847 (RENVISION) and FP7-IDEAS-ERC no. 227747 (NERVI), the French ministry of Research and University of Nice (EDSTIC), INRIA ADT and by the ECOS-Sud program (grant C13E06).

[10]PRANAS is under Inria CeCILL-C license, IDDN.FR.OO1.190004.000.S.P.2014.000.31235. Binaries can be freely downloaded.

[11]These commands can be found in the directory `pranas/bin`. Help of each tool is given with the command line: `toolName -h`.

[12]Parallelization concerns every loop in our C++ code where iterations can be computed independently. To do so, we used OpenMP so that PRANAS parallel processes can run on a personal computer equipped with multi-core processors. More precisely, four functions were parallelized: (i) the loading of rasters for all file formats, (ii) the statistical analysis functions to generate potentials, compute divergence and estimate patterns probabilities, (iii) the estimation of receptive fields and (iv) the retina simulator.

[13]PRANAS website: http://pranas.inria.fr

ACKNOWLEDGMENTS

We warmly acknowledge Michael J. Berry, Maria-Jose Escobar, Matthias Hennig, Ruben Herzog, Gerrit Hilgen, Evgenia Kartsaki, Gaia Lombardi, Olivier Marre, Thierry Mora, Adrian Palacios, Cesar Ravello, Horacio Rostro, Vivien Robinet, Selma Souihel, Evelyne Sernagor, Gasper Tkačik, Juan-Carlos Vasquez for their help in designing PRANAS. The authors also thank the reviewers for their constructive criticisms and valuable comments which helped us improve the manuscript.

REFERENCES

Basalyga, G., Montemurro, M. A., and Wennekers, T. (2013). Information coding in a laminar computational model of cat primary visual cortex. *J. Comput. Neurosci.* 34, 273–283. doi: 10.1007/s10827-012-0420-x

Benoit, A., Caplier, A., Durette, B., and Herault, J. (2010). Using human visual system modeling for bio-inspired low level image processing. *Comput. Vis. Image Understand.* 114, 758–773. doi: 10.1016/j.cviu.2010.01.011

Carandini, M., Demb, J. B., Mante, V., Tollhurst, D. J., Dan, Y., Olshausen, B. A., et al. (2005). Do we know what the early visual system does? *J. Neurosci.* 25, 10577–10597. doi: 10.1523/JNEUROSCI.3726-05.2005

Cessac, B. (2008). A discrete time neural network model with spiking neurons. Rigorous results on the spontaneous dynamics. *J. Math. Biol.* 56, 311–345. doi: 10.1007/s00285-007-0117-3

Cessac, B. (2011). A discrete time neural network model with spiking neurons II. dynamics with noise. *J. Math. Biol.* 62, 863–900. doi: 10.1007/s00285-010-0358-4

Cessac, B., and Palacios, A. (2012). "Spike train statistics from empirical facts to theory: the case of the retina," in *Modeling in Computational Biology and Biomedicine: A Multidisciplinary Endeavor*, Lectures Notes in Mathematical and Computational Biology (LNMCB), eds F. Cazals and P. Kornprobst (Berlin; Heidelberg: Springer-Verlag), 261–302.

Cessac, B., Rostro-Gonzalez, H., Vasquez, J.-C., and Viéville, T. (2009). How gibbs distribution may naturally arise from synaptic adaptation mechanisms: a model based argumentation. *J. Stat. Phys.* 136, 565–602. doi: 10.1007/s10955-009-9786-1

Chichilnisky, E. J. (2001). A simple white noise analysis of neuronal light responses. *Network Comput. Neural Syst.* 12, 199–213. doi: 10.1080/713663221

Cofré, R., and Cessac, B. (2013). Dynamics and spike trains statistics in conductance-based integrate-and-fire neural networks with chemical and electric synapses. *Chaos, Solit. Fract*. 50, 13–31. doi: 10.1016/j.chaos.2012.12.006

Cofre, R., and Cessac, B. (2014). Exact computation of the maximum-entropy potential of spiking neural-network models. *Phys. Rev. E* 89:052117. doi: 10.1103/PhysRevE.89.052117

Doutsi, E., Fillatre, L., Antonini, M., and Gaulmin, J. (2015a). "Retinal-inspired filtering for dynamic image coding," in *IEEE International Conference on Image Processing (ICIP)* (Quebec City, QC), 3505–3509.

Doutsi, E., Fillatre, L., Antonini, M., and Gaulmin, J. (2015b). "Video analysis and synthesis based on a retinal-inspired frame," in *23rd European Signal Processing Conference (EUSIPCO)* (Nice: IEEE), 2226–2230. doi: 10.1109/EUSIPCO.2015.7362780

Drogoul, A., Aubert, G., Cessac, B., and Kornprobst, P. (2016). "A new nonconvex variational approach for sensory neurons receptive field estimation," in *6th International Workshop on New Computational Methods for Inverse Problems (NCMIP)* (Cachan). doi: 10.1088/1742-6596/756/1/012006

Dudík, M., Phillips, S. J., and Schapire, R. E. (2004). *Performance Guarantees for Regularized Maximum Entropy Density Estimation*. Berlin; Heidelberg: Springer. doi: 10.1007/978-3-540-27819-1_33

Ferrea, E., Maccione, A., Medrihan, L., Nieus, T., Ghezzi, D., Baldelli, P., et al. (2012). Large-scale, high-resolution electrophysiological imaging of field potentials in brain slices with microelectronic multielectrode arrays. *Front. Neural Circuits* 6:80. doi: 10.3389/fncir.2012.00080

Ganmor, E., Segev, R., and Schneidman, E. (2011a). The architecture of functional interaction networks in the retina. *J. Neurosci.* 31, 3044–3054. doi: 10.1523/JNEUROSCI.3682-10.2011

Ganmor, E., Segev, R., and Schneidman, E. (2011b). Sparse low-order interaction network underlies a highly correlated and learnable neural population code. *Proc. Natl. Acad. Sci. U.S.A.* 108, 9679–9684. doi: 10.1073/pnas.1019641108

Garcia, S., and Fourcaud-Trocmé, N. (2009). Openelectrophy: an electrophysiological data-and analysis- sharing framework. *Front. Neuroinform.* 3:14. doi: 10.3389/neuro.11.014.2009

Ginsburg, K. S., Johnsen, J. A., and Michael W., L. (1984). Common noise in the firing of neighbouring ganglion cells in goldfish retina. *J. Physiol.* 351, 433–450. doi: 10.1113/jphysiol.1984.sp015254

Goldberg, D. H., Victor, J. D., Gardner, E. P., and Gardner, D. (2009). Spike train analysis toolkit: enabling wider application of information-theoretic techniques to neurophysiology. *Neuroinformatics* 7, 165–178. doi: 10.1007/s12021-009-9049-y

Gollisch, T., and Meister, M. (2010). Eye smarter than scientists believed: neural computations in circuits of the retina. *Neuron* 65, 150–164. doi: 10.1016/j.neuron.2009.12.009

Greschner, M., Shlens, J., Bakolitsa, C., Field, G. D., Gauthier, J. L., Jepson, L. H., et al. (2011). Correlated firing among major ganglion cell types in primate retina. *J. Physiol.* 589(Pt 1), 75–86. doi: 10.1113/jphysiol.2010.193888

Hérault, J. (2010). "Vision: Images, Signals and Neural Networks: Models of Neural Processing in Visual Perception," *Progress in Neural Processing,* Vol. 19 (River Edge, NJ: World Scientific Publishing Co., Inc.).

Herzog, R., Araya, J., Pizarro, M., Cessac, B., Ravello, C., Escobar, M.-J., et al. (2014). "From habitat to retina: neural population coding using natural movies," in *Bernstein Conference* (Heidelberg).

Hillar, C., and Effenberger, F. (2015). Robust discovery of temporal structure in multi-neuron recordings using hopfield networks. *Proc. Comput. Sci.* 53, 365–374. doi: 10.1016/j.procs.2015.07.313

Ince, R. A., Mazzoni, A., Petersen, R. S., and Panzeri, S. (2010). Open source tools for the information theoretic analysis of neural data. *Front. Neurosci.* 4:62. doi: 10.3389/neuro.01.011.2010

Jaynes, E. (1957). Information theory and statistical mechanics. *Phys. Rev.* 106:620. doi: 10.1103/PhysRev.106.620

Kim, J. S., Greene, M. J., Zlateski, A., Lee, K., Richardson, M., Turaga, S. C., et al. (2014). Space-time wiring specificity supports direction selectivity in the retina. *Nature* 509, 331–336. doi: 10.1038/nature13240

Lidierth, M. (2009). sigtool: a matlab-based environment for sharing laboratory-developed software to analyze biological signals. *J. Neurosci. Methods* 178, 188–196. doi: 10.1016/j.jneumeth.2008.11.004

Lorach, H., Benosman, R., Marre, O., Ieng, S.-H., Sahel, J. A., and Picaud, S. (2012). Artificial retina: the multichannel processing of the mammalian retina achieved with a neuromorphic asynchronous light acquisition device. *J. Neural Eng.* 9:066004. doi: 10.1088/1741-2560/9/6/066004

Marre, O., El Boustani, S., Frégnac, Y., and Destexhe, A. (2009). Prediction of spatiotemporal patterns of neural activity from pairwise correlations. *Phys. Rev. Lett.* 102:138101. doi: 10.1103/PhysRevLett.102.138101

Martinez-Alvarez, A., Olmedo-Payá, A., Cuenca-Asensi, S., Ferrandez, J. M., and Fernandez, E. (2013). RetinaStudio: a bioinspired framework to encode visual information. *Neurocomputing* 114, 45–53. doi: 10.1016/j.neucom.2012.07.035

Masland, R. H. (2011). Cell populations of the retina: the proctor lecture. *Invest. Ophthalmol. Visual Sci.* 52, 4581–4591. doi: 10.1167/iovs.10-7083

Masland, R. H. (2012). The neuronal organization of the retina. *Neuron* 76, 266–280. doi: 10.1016/j.neuron.2012.10.002

Masmoudi, K., Antonini, M., and Kornprobst, P. (2010). "Another look at the retina as an image scalar quantizer," in *Proceedings of the International Symposium on Circuits and Systems (ISCAS)* (Paris), 3076–3079.

Masquelier, T. (2012). Relative spike time coding and stdp-based orientation selectivity in the early visual system in natural continuous and saccadic vision: a computational model. *J. Comput. Neurosci.* 32, 425–441. doi: 10.1007/s10827-011-0361-9

Meier, R., Egert, U., Aertsen, A., and Nawrot, M. P. (2008). FIND: a unified framework for neural data analysis. *Neural Netw.* 21, 1085–1093. doi: 10.1016/j.neunet.2008.06.019

Mohemmed, A., Lu, G., and Kasabov, N. (2012). "Evaluating SPAN incremental learning for handwritten digit recognition," in *International Conference on Neural Information Processing* (Doha; Qatar: Springer Berlin Heidelberg), 670–677. doi: 10.1007/978-3-642-34487-9_81

Nasser, H., and Cessac, B. (2014). Parameters estimation for spatio-temporal maximum entropy distributions: application to neural spike trains. *Entropy* 16, 2244–2277. doi: 10.3390/e16042244

Nasser, H., Kraria, S., and Cessac, B. (2013a). "Enas: a new software for neural population analysis in large scale spiking networks," in *Twenty Second Annual Computational Neuroscience Meeting* (Paris: Organization for Computational Neurosciences), 57.

Nasser, H., Marre, O., and Cessac, B. (2013b). Spatio-temporal spike train analysis for large scale networks using the maximum entropy principle and montecarlo method. *J. Stat. Mech. Theory Exp.* 2013:P03006. doi: 10.1088/1742-5468/2013/03/P03006

Odermatt, B., Nikolaev, A., and Lagnado, L. (2012). Encoding of luminance and contrast by linear and nonlinear synapses in the retina. *Neuron* 73, 758–773. doi: 10.1016/j.neuron.2011.12.023

Pelayo, F. J., Romero, S., Morillas, C. A., Martinez, A., Ros, E., and Fernandez, E. (2004). Translating image sequences into spike patterns for cortical neuro-stimulation. *Neurocomputing* 58–60, 885–892. doi: 10.1016/j.neucom.2004.01.142

Pouzat, C., and Chaffiol, A. (2009). Automatic spike train analysis and report generation. An implementation with r, r2html and star. *J. Neurosci. Methods* 181, 119–144. doi: 10.1016/j.jneumeth.2009.01.037

Pröpper, R., and Obermayer, K. (2013). Spyke viewer: a flexible and extensible platform for electrophysiological data analysis. *Front. Neuroinform.* 7:26. doi: 10.3389/fninf.2013.00026

Quaglio, P., Yegenoglu, A., Torre, E., Endres, D. M., and Gruen, S. (2017). Detection and evaluation of spatio-temporal spike patterns in massively parallel spike train data with spade. *Front. Comput. Neurosci.* 11:41. doi: 10.3389/fncom.2017.00041

Ravello, C., Herzog, R., Cessac, B., Escobar, M.-J., and Palacios, A. (2015). "Spectral dimension reduction on parametric models for spike train statistics" in *12e Colloque de la Société des Neurosciences* (Montpellier).

Rieke, F., Warland, D., de Ruyter van Steveninck, R., and Bialek, W. (1996). *Spikes, Exploring the Neural Code*. Cambridge, MA: MIT Press.

Roudi, Y., and Hertz, J. (2011). Mean field theory for non-equilibrium network reconstruction. *Phys. Rev. Lett.* 106:048702. doi: 10.1103/PhysRevLett.106.048702

Roudi, Y., Nirenberg, S., and Latham, P. (2009a). Pairwise maximum entropy models for studying large biological systems: when they can work and when they can't. *PLoS Comput. Biol.* 5:e1000380. doi: 10.1371/journal.pcbi.1000380

Roudi, Y., Tyrcha, J., and Hertz, J. A. (2009b). Ising model for neural data: model quality and approximate methods for extracting functional connectivity. *Phys. Rev. E* 79:051915. doi: 10.1103/PhysRevE.79.051915

Schneidman, E., Berry, M., Segev, R., and Bialek, W. (2006). Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature* 440, 1007–1012. doi: 10.1038/nature04701

Shlens, J., Field, G., Gauthier, J., Grivich, M., Petrusca, D., Sher, A., et al. (2006). The structure of multi-neuron firing patterns in primate retina. *J. Neurosci.* 26:8254. doi: 10.1523/JNEUROSCI.1282-06.2006

Soula, H., Beslon, G., and Mazet, O. (2006). Spontaneous dynamics of asymmetric random recurrent spiking neural networks. *Neural Comput.* 18, 60–79. doi: 10.1162/089976606774841567

Spacek, M., Blanche, T., and Swindale, N. (2008). Python for large-scale electrophysiology. *Front. Neuroinform.* 2:9. doi: 10.3389/neuro.11.009.2008

Stevenson, I. H., and Kording, K. P. (2011). How advances in neural recording affect data analysis. *Nat. Neurosci.* 14, 139–142. doi: 10.1038/nn.2731

Strong, S., Koberle, R., de Ruyter van Steveninck, R., and Bialek, W. (1998). Entropy and information in neural spike trains. *Phys. Rev. Lett.* 80, 197–200. doi: 10.1103/PhysRevLett.80.197

Takahashi, K., Kim, S., Coleman, T., Brown, K., Suminski, A., Best, M., et al. (2015). Large-scale spatiotemporal spike patterning consistent with wave propagation in motor cortex. *Nat. Commun.* 6:7169. doi: 10.1038/ncomms8169

Tang, A., Jackson, D., Hobbs, J., Chen, W., Smith, J. L., Patel, H., et al. (2008). A maximum entropy model applied to spatial and temporal correlations from cortical networks *In vitro*. *J. Neurosci.* 28, 505–518. doi: 10.1523/JNEUROSCI.3359-07.2008

Tkačik, G., Marre, O., Mora, T., Amodei, D. II., M. B., and Bialek, W. (2013). The simplest maximum entropy model for collective behavior in a neural network. *J. Stat. Mech.* 2013:P03011. doi: 10.1088/1742-5468/2013/03/P03011

Tkačik, G., Prentice, J. S., Balasubramanian, V., and Schneidman, E. (2010). Optimal population coding by noisy spiking neurons. *Proc. Natl. Acad. Sci. U.S.A.* 107, 14419–14424. doi: 10.1073/pnas.1004906107

Tkačik, G., Schneidman, E., Berry II, M., and Bialek, W. (2006). Ising models for networks of real neurons. arXiv q-bio/0611072.

Tkačik, G., Schneidman, E., Berry M. J. II., and Bialek, W. (2009). Spin glass models for a network of real neurons. arXiv preprint arXiv:0912.5409.

Torre, E., Quaglio, P., Denker, M., Brochier, T., Riehle, A., and Grun, S. (2016). Synchronous spike patterns in macaque motor cortex during an instructed-delay reach-to-grasp task. *J. Neurosci.* 36, 8329–8340. doi: 10.1523/JNEUROSCI.4375-15.2016

Vance, P., Coleman, S. A., Kerr, D., Das, G., and McGinnity, T. (2015). "Modelling of a retinal ganglion cell with simple spiking models," in *IEEE International Joint Conference on Neural Networks* (Killarney), 1–8.

Vasquez, J.-C., Cessac, B., and Viéville, T. (2010). "Entropy-based parametric estimation of spike train statistics," in *Statistical Mechanics of Learning and Inference* (Mariehamn: Stockholm-Mariehanm).

Vasquez, J. C., Marre, O., Palacios, A. G., Berry, M. J., and Cessac, B. (2012). Gibbs distribution analysis of temporal correlation structure on multicell spike trains from retina ganglion cells. *J. Physiol. Paris* 106, 120–127. doi: 10.1016/j.jphysparis.2011.11.001

Wohrer, A., and Kornprobst, P. (2009). Virtual retina: a biological retina model and simulator, with contrast gain control. *J. Comput. Neurosci.* 26, 219–249. doi: 10.1007/s10827-008-0108-4

Zordan, S., Zanotto, M., Niels, T., Di Marco, S., Amin, H., Maccione, A., et al. (2015). "A scalable high performance client/server framework to manage and analyze high dimensional datasets recorded by 4096 CMOS-MEAs," in *7th International IEEE/EMBS Conference on Neural Engineering (NER)* (Montpellier), 968–971.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.