RESEARCH ARTICLE

Graphics Processing Unit-Accelerated Code for Computing Second-Order Wiener Kernels and Spike-Triggered Covariance

Omer Mano[1], **Damon A. Clark**[1,2,3]*

1 Department of Molecular, Cellular, and Developmental Biology, Yale University, New Haven, Connecticut, United States of America, **2** Department of Physics, Yale University, New Haven, Connecticut, United States of America, **3** Interdepartmental Neuroscience Program, Yale University, New Haven, Connecticut, United States of America

* damon.clark@yale.edu

Citation: Mano O, Clark DA (2017) Graphics Processing Unit-Accelerated Code for Computing Second-Order Wiener Kernels and Spike-Triggered Covariance. PLoS ONE 12(1): e0169842. doi:10.1371/journal.pone.0169842

Editor: Matthieu Louis, Centre de Regulacio Genomica, SPAIN

Received: October 28, 2016

Accepted: December 22, 2016

Published: January 9, 2017

Data Availability Statement: All data and code used in this paper are available at https://github.com/ClarkLabCode/GPUFilterExtraction.

Funding: This work was supported by the National Institutes of Health, R01EY026555 to DAC; played no role; salary to DAC and OM, National Science Foundation, IOS1558103 to DAC; played no role; salary to DAC, Alfred P. Sloan Foundation, Fellowship to DAC; played no role; salary to DAC, Searle Foundation, Searle Scholar Award to DAC; played no role; salary to DAC, Smith Family Foundation, Scholar Award to DAC; played no role;

Abstract

Sensory neuroscience seeks to understand and predict how sensory neurons respond to stimuli. Nonlinear components of neural responses are frequently characterized by the second-order Wiener kernel and the closely-related spike-triggered covariance (STC). Recent advances in data acquisition have made it increasingly common and computationally intensive to compute second-order Wiener kernels/STC matrices. In order to speed up this sort of analysis, we developed a graphics processing unit (GPU)-accelerated module that computes the second-order Wiener kernel of a system's response to a stimulus. The generated kernel can be easily transformed for use in standard STC analyses. Our code speeds up such analyses by factors of over 100 relative to current methods that utilize central processing units (CPUs). It works on any modern GPU and may be integrated into many data analysis workflows. This module accelerates data analysis so that more time can be spent exploring parameter space and interpreting data.

Introduction

An important goal in neuroscience is to understand how stimuli influence the activity of neurons. One way to characterize this influence is with linear kernels, which describe how neural responses depend linearly on past and present stimuli. These kernels do not tell the entire story, since neural responses are frequently nonlinear. One solution is to combine linear kernels with static nonlinearities to account for neural nonlinearities [1]. An alternative is to directly determine how the response depends on different correlations in the input, an approach known as nonlinear systems identification [2]. In this approach, the nonlinear response can be approximated to low order by the second-order Wiener kernel of the system [2, 3], which is closely related to spike-triggered covariance (STC) methods [4–9]. The shape of the Wiener kernel is informative about which correlations elicit responses, and its eigenvectors span the space of potential linear feature detectors that could be combined nonlinearly to generate the response [5, 10]. These second-order analyses have been used to characterize

salary to DAC, National Institutes of Health, T32GM007499 to OM; played no role; salary to OM, Yale University; played no role other than salary to DAC and OM. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

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

photoreceptors [11, 12], retinal ganglion cells [6, 13–17], visual direction-selective cells in cortex [18–20] and in insects [10, 21, 22], direction-selective behavior in insects [22, 23], somatosensory neurons [24–26], olfactory receptor neurons [27] and subsequent olfactory processing neurons [28], auditory neurons [29, 30], electroretinograms [31, 32], and functional magnetic resonance imaging [33, 34]. These studies used second-order kernels to gain insights into timescales and lengthscales of motion computations, relevant spatiotemporal features for retinal and cortical visual neurons, and distinct ON and OFF pathway inputs.

The computational load for analyzing data in neuroscience is growing heavier, largely because data sets are increasing in size as the number of recorded cells, the sampling rate of cells, and the duration of recordings all increase. Multi-electrode arrays allow extracellular recordings of hundreds of cells [35, 36], while one- and two-photon imaging techniques now also allow hundreds of cells to be recorded [37, 38]. Genetically encoded indicators of activity are becoming faster [39–43] and imaging techniques are acquiring data with increasingly high sampling rates [42]. Covering relevant timescales at faster sampling rates increases the load for analysis.

Calculating second-order filters can require trillions of mathematical operations per filter. To date, second-order kernels have been extracted using standard libraries on central processing units (CPUs) [44]. Graphics processing units (GPUs) can perform some types of computations far more efficiently than CPUs [45]. In our own research, we were computing and assessing the significance of second-order Wiener kernels for thousands of 2-photon calcium imaging traces, each with more than 10,000 samples in time [22]. In order to speed up these computations, we wrote code to extract response-weighted stimulus covariance matrices (Wiener kernels) using GPUs.

Results

Our code uses the computer's GPU to compute second-order Wiener kernels (see Methods for the formula and scaling of the kernel). To test our GPU-accelerated code, we generated a synthetic dataset that creates a response from a second-order Volterra kernel acting on a discrete, Gaussian, uncorrelated input (Fig 1A). We then computed the second-order Wiener kernel for this stimulus-response set (Fig 1B). The Wiener kernel is an unbiased estimate of the Volterra kernel when the system has no higher-order responses and when the response is mean-subtracted. The error in the kernel estimate can be attributed to the finite stimulus presentation. This simulated kernel has very high rank for illustrative purposes, but a real extracted kernel is likely to be much lower rank [5, 6].

To quantify the advantages of the GPU-based algorithm, we compared its performance to that of a CPU-based algorithm on synthetic datasets with different properties. We first varied the duration of the stimulus, expressed in the number of samples. We found that over a wide range of samples, the GPU-algorithm beat the CPU-algorithm by a factor of over 100 (Fig 1C). The speedup increases as the number of samples increases because the GPU module's initialization time, while shorter in absolute terms than the CPU implementation's initialization time, makes up a larger proportion of the total runtime. As the number of samples increases, this initialization time becomes a smaller fraction of the total time.

The number of operations needed to compute the second-order kernel scales with the square of the number lags in the filter. As the number of lags increases, the GPU-algorithm speeds up relative to the CPU-algorithm (Fig 1D). The optimal filter size for the GPU-algorithm on the tested GPU is 64 lags, due to how cores are grouped in this particular GPU (different GPUs may have different optimal lags). Above 64 lags, the GPU-algorithm's relative speed up remains roughly constant. When the user requests fewer than 64 lags, fewer GPU

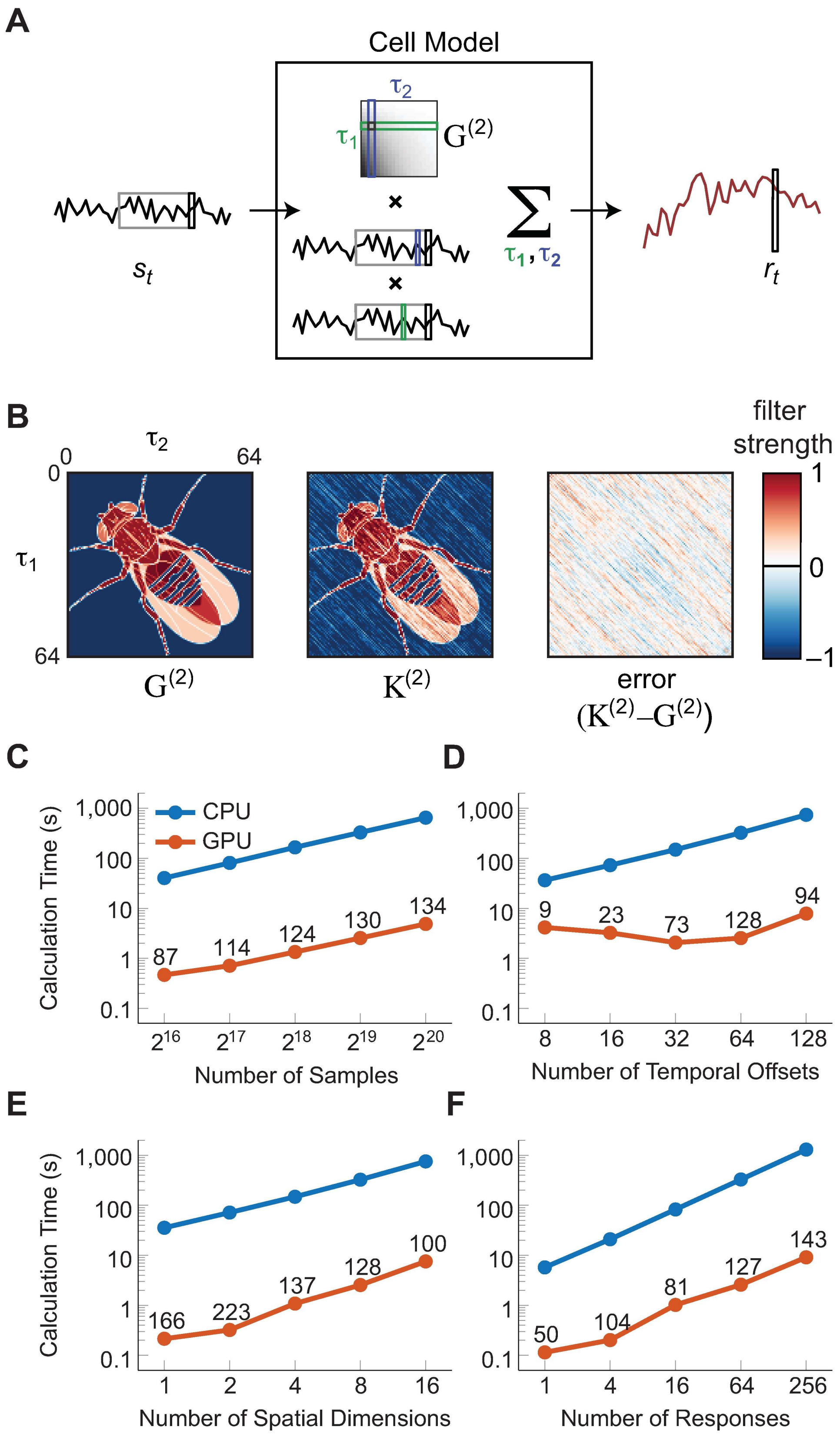
A
Cell Model
τ_2
τ_1
$G^{(2)}$
s_t
$\sum_{\tau_1,\tau_2}$
r_t
B
τ_2
0
64
0
τ_1
64
$G^{(2)}$
$K^{(2)}$
error
$(K^{(2)}-G^{(2)})$
filter strength
1
0
−1
C
CPU
GPU
Calculation Time (s)
1,000
100
10
1
0.1
87
114
124
130
134
2^{16}
2^{17}
2^{18}
2^{19}
2^{20}
Number of Samples
D
9
23
73
128
94
8
16
32
64
128
Number of Temporal Offsets
E
166
223
137
128
100
1
2
4
8
16
Number of Spatial Dimensions
F
50
104
81
127
143
1
4
16
64
256
Number of Responses

Fig 1. Verified algorithm and degree of speed up. (A) The second order component of a cell's response is modeled as a linear weighting of pairwise products in the stimulus. The linear weighting matrix is the Volterra kernel, $G^{(2)}$. (B) The GPU code correctly extracts the Wiener kernel from the stimulus-response pair. We simulated a model cell that used a Volterra kernel $G^{(2)}$ to respond to a Gaussian input with unit variance. We then used the GPU code to estimate the Weiner kernel $K^{(2)}$. Differences between the actual and estimated kernel go to 0 as the number of stimulus-response samples increases. In this case, the Wiener kernel equals the Volterra kernel because the response does not depend on higher order functions of the stimulus. (C-F) Performance of the GPU-enabled module versus the reference CPU implementation. Unless otherwise stated for each simulation parameter sweep, the dataset included 64 responses of 2^{19} samples, with inputs of 8 spatial dimensions each. The code extracted filters with 64 temporal offsets. All calculation times are given in seconds; code was run on hardware as described in Methods. Numbers above the GPU datapoints indicate the factor speedup of the GPU module relative to the CPU module. (C) Calculation time comparison for a sweep of the number of samples of each response. (D) Calculation time comparison for a sweep of the number of temporal offsets. (E) Calculation time comparison for a sweep of the number of spatial dimensions. (F) Calculation time comparison for a sweep of the number of responses (e.g. recorded cells in an imaging dataset).

doi:10.1371/journal.pone.0169842.g001

cores are allocated to the task, so that the code slows somewhat (in terms of calculations per second) when fewer than 64 lags are requested.

As the number of dimensions of the stimulus increases, the number of elements in the second-order kernel increases quadratically. The GPU-algorithm remains about 100 times faster than the CPU-algorithm as the number of dimensions increases (Fig 1E), but the gains are greater at lower numbers of dimensions. The CPU implementation becomes relatively more efficient as the number of spatial dimensions increases, but does not catch up to the GPU implementation.

New methods of data acquisition can simultaneously acquire many neural responses to a single stimulus. In the GPU-algorithm implementation, the speed up relative to the CPU increases as the number of responses increases (Fig 1F). This is because computing the Wiener kernels of fewer than 64 responses typically does not contain enough work to keep all the cores of a GPU busy.

Perhaps the most common application of the kernels we compute is in spike triggered covariance (STC) analysis, when these kernels are used to determine the space of linear filters that contribute to a cell's activity. In order to demonstrate how our algorithm can be used for this purpose, we simulated a spiking neuron whose firing rate is the sum of two linear-nonlinear models (Fig 2A), reminiscent of ON and OFF inputs to cells that have been similarly analyzed [46]. We presented the model cell with Gaussian random inputs, and simulated the resulting spike train as a Poisson spiking process. The spike-triggered average (STA) does not accurately represent either one of the input filters, and instead mixes the two input filters together (Fig 2B). In order to reconstitute the input filters, we first extracted the response-weighted stimulus covariance using the algorithm presented here. (This is proportional to the Wiener kernel; see Methods.) We then performed computationally trivial matrix operations to convert the resulting kernel into the most commonly used forms of the spike triggered covariance matrix (Fig 2C, see Methods for derivation and conversion instructions). We first computed the raw response-weighted covariance matrix (Fig 2C, *left*). We computed two other manipulations of this matrix popular for STC analysis, in which the STA is either subtracted (Fig 2C, *middle*, [6]) or projected out of the matrix (Fig 2C, *right*, [18]). The Methods section describes these operations in detail.

In each of the three cases, we calculated the eigenspectrum of the STC matrix and selected the eigenvectors with significant eigenvalues (Fig 2D and 2E). When the STA is projected out of the STC matrix, only one significant eigenvector remains; the STA itself is used as an additional vector for further analysis. The eigenvectors of these matrices span the space of possible linear filters in the model cell. The actual input vectors used in the model cell can be reconstituted from linear combinations of the significant eigenvectors (Fig 2F).

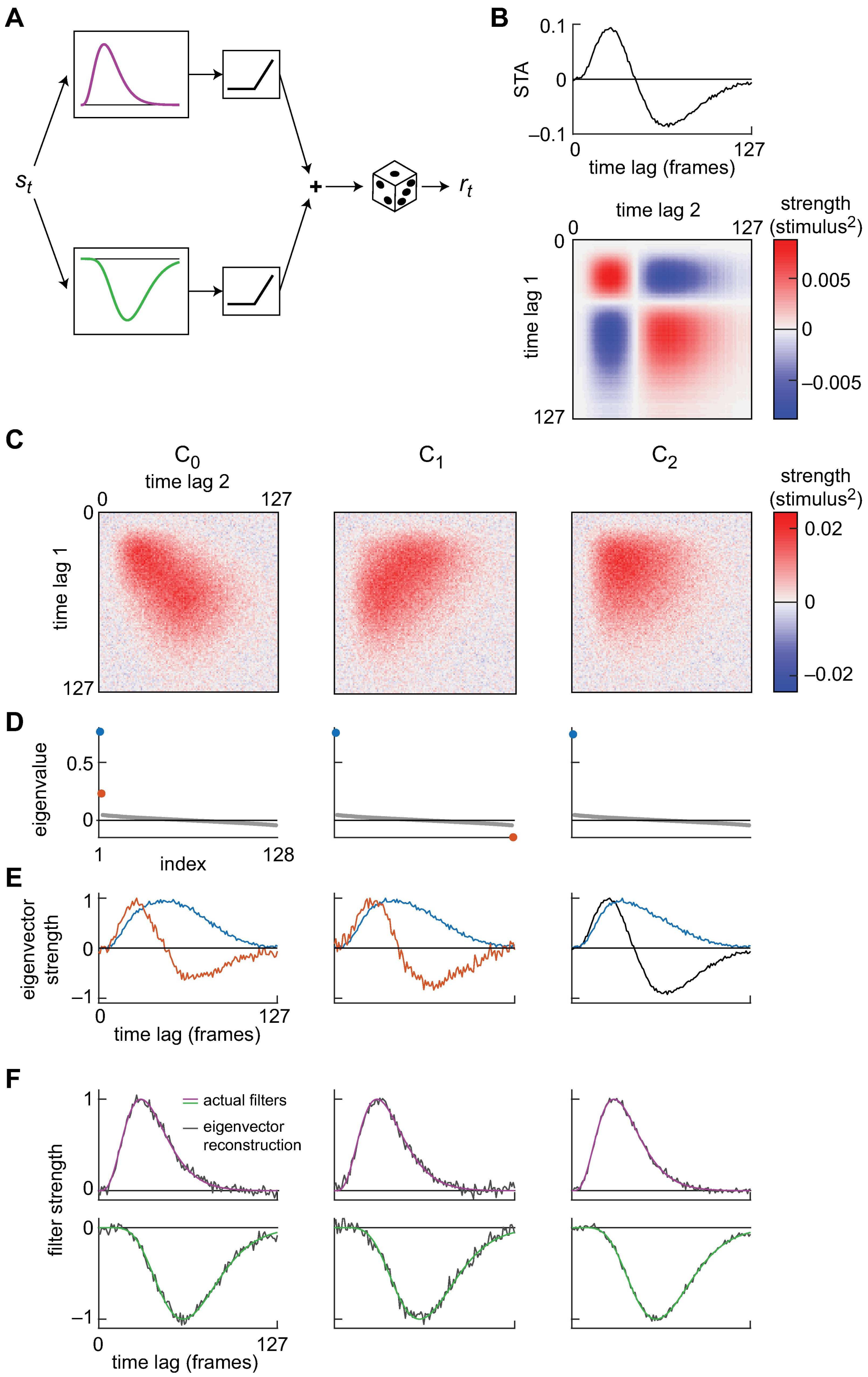

Fig 2. Demonstration of input filter extraction from a model spiking cell. (A) A model spiking neuron. Two linear filters act on the input, are rectified, and then summed. This determines the probability of firing, which is modeled as a Poisson process. Gaussian random inputs were fed into the model and resulting spikes computed

with a mean rate of 0.0045 per time bin. (B) We extracted a spike triggered average from the responses (top). The extracted filter is a mix of the two input filters, and does not accurately represent the underlying linear processing. The outer product of the spike triggered average (bottom) will be used to compute forms of the STC matrix. (C) Three forms of the spike triggered covariance (STC) matrix: raw (C_0), STA subtracted (C_1), and with the STA projected out (C_2). See Methods for details. (D) The eigenspectrum of each STC matrix, with significant eigenvalues highlighted. (E) Eigenvectors of each STC matrix that correspond to significant eigenvalues. In the case of C_2, the STA serves as the second filter. (F) Comparison of the actual input filters (purple and green) with linear combinations of the extracted filters (grey); the eigenvectors span the space of linear input filters.

doi:10.1371/journal.pone.0169842.g002

Discussion

This code speeds up second-order Wiener kernel computations and STC analyses. The speed-up occurs over a wide range of data parameters (Fig 1), and is especially useful for large data-sets that are sampled frequently in time and for experiments in which many neurons are simultaneously measured. The computational speed is increased by using the GPU to quickly perform low-level computations; this contributes to a trend of using GPU-processing to speed up computation-intensive tasks like machine learning and image analysis [47–49].

The second-order Wiener kernel is closely related to spike triggered covariance analysis (STC) [4]. The Wiener kernel allows for continuous and negative inputs, such as voltage, current, and light intensity. To use this code with spikes in computing an STC matrix, the response can simply code the number of spikes in each bin. In some implementations of STC, the spike triggered average (STA) is subtracted from the stimulus before computing covariance [6]. Other implementations of STC have subtracted off the component of the stimulus in the direction of the STA before computing the kernel [18]. To compute the STC in both these cases, this code can be used and its output can be transformed appropriately using the STA (see Methods). Fig 2 shows how the different methods might be applied to a simple example case, in each case first computing a response-weighted stimulus covariance matrix with the GPU code, then transforming it into the more familiar STC matrices that appear in the literature [6, 18].

It can be difficult to assess the statistical significance of Wiener kernels/STC matrices, since noise in the filter tends to be correlated across many elements. One method for significance estimation is to decorrelate the stimulus and response, time shifting them relative to one another, to obtain a filter estimate that would occur if the stimulus and response were unrelated [8]. This preserves the correlations in both the response and stimulus, so it has an appropriate noise distribution for a null hypothesis. To compare a measured filter to the decorrelated null set of filters, one needs to compute thousands or tens of thousands of decorrelated filters. If each filter set takes 100 seconds to compute, as it can with CPU code, this can be prohibitively difficult, taking approximately 10 days to compute a full null set of 10,000 filters. With the GPU acceleration, this can be narrowed to a few hours of computation. Thus, this code makes this method of significance testing far easier for Wiener kernels and STC analysis.

Overall, the increased analysis speed of this tool allows researchers to spend more time devising analyses and interpreting results and less time waiting for computations to finish. The time-savings encourages broader exploration of large neural datasets.

Methods

We wrote code to compute the response-weighted stimulus covariance, C, from a paired stimulus and response:

$$C_{ij} = \frac{1}{T-\tau} \sum_{t=\tau+1}^{T} r_t s_{t-i} s_{t-j}$$

where r_t is the response at time t, $s_{t'}$ is the stimulus at time t', T is the total length of the stimulus-response sampling, and i and j represent lags in the filter ranging from 0 to τ. When the stimulus is multidimensional and can be divided into separate time series, for instance representing multiple pixels on a screen, C is organized block-wise for each stimulus pairing, as described in the code documentation. This weighted average of the stimulus covariance is proportional to the second-order Wiener kernel when $\langle r_t \rangle = 0$. The second-order Wiener kernel, $K^{(2)}$, is

$$K^{(2)} = \frac{1}{2\sigma^4 \Delta t^2} C$$

where σ is the standard deviation of the input stimulus and Δt is the sampling interval [2].

To see how the matrix C, computed by this algorithm, relates to different spike-triggered methods, we must define a few more quantities (for a more detailed analysis, see also [4]). First, in keeping with convention of the spike-triggered literature, we set r_t to be the integer number of spikes in the time window around time t, and we do not set the mean response to 0. Then, the spike triggered average (STA), $\mathbf{a}$, is

$$\mathbf{a} = \frac{1}{n_r} \sum_t r_t \mathbf{s}_t$$

where $\mathbf{s}_t$ is a column vector of the stimulus preceding time t, and $n_r = \sum_t r_t$ is the total number of spikes. Because the response is non-zero in this case, we must also compute the stimulus covariance. That stimulus covariance may be computed by the GPU algorithm by setting all responses to 1, so that $S_{ij} = \frac{1}{T-\tau} \sum_{t=\tau+1}^{T} s_{t-i} s_{t-j}$, or $S = \frac{1}{T-\tau} \sum_{t=\tau+1}^{T} \mathbf{s}_t \mathbf{s}_t^T$. Then the raw spike triggered covariance (STC) matrix, C_0, is just

$$C_0 = \frac{1}{n_r} \sum_t r_t \mathbf{s}_t \mathbf{s}_t^T - S = \frac{T-\tau}{n_r} C - S$$

which is the response-weighted covariance we found above, C, scaled by the mean firing rate, with the basal stimulus covariance subtracted off [10]. Here and throughout, we assume that the number of samples for estimating the covariance is much larger than 1, so that the denominator may be written as n_r rather than n_r–1 without appreciable error.

In some versions of STC, the STA is subtracted before computing the covariance [6]. In such a version, C_1 is computed as

$$C_1 = \frac{1}{n_r} \sum_t r_t (\mathbf{s}_t - \mathbf{a})(\mathbf{s}_t - \mathbf{a})^T - S$$

This version reduces to $C_1 = C_0 - A$, where $A = \mathbf{a}\mathbf{a}^T$ is the outer product of the STA [4]. This means that the matrix computed by this algorithm, C, can be transformed into the form of C_1 by simple matrix operations.

An alternative method computes the STC matrix by first removing the component of each stimulus vector in the direction of the STA [18]. This means subtracting a term $\mathbf{b}_t = \mathbf{a}(\mathbf{a}^T \mathbf{s}_t)/$

$(\mathbf{a}^T\mathbf{a}) = A'\mathbf{s}_t$ from each stimulus vector, and results in an STC matrix C_2 defined as:

$$C_2 = \frac{1}{n_r}\sum_t r_t(\mathbf{s}_t - \mathbf{b}_t)(\mathbf{s}_t - \mathbf{b}_t)^T - \frac{1}{T-\tau}\sum_t(\mathbf{s}_t - \mathbf{b}_t)(\mathbf{s}_t - \mathbf{b}_t)^T$$
$$C_2 = \frac{1}{n_r}\sum_t r_t(I - A')\mathbf{s}_t\mathbf{s}_t^T(I - A')^T - \frac{1}{T-\tau}\sum_t(I - A')\mathbf{s}_t\mathbf{s}_t^T(I - A')^T$$

where I is the identity matrix and $A' = \frac{\mathbf{a}\mathbf{a}^T}{\mathbf{a}^T\mathbf{a}}$ is the outer product of the normalized STA. In the equations above, the stimulus has the projection subtracted before computing covariance, in the case of both the response-weighted covariance (first term) and the unweighted covariance (second term). This equation simplifies to $C_2 = (I–A')C_0(I–A')^T$. Thus, in this case, too, the kernel C_0 can be computed and transformed into the desired form with matrix multiplications.

GPUs achieve very high computational throughput by incorporating thousands of relatively slow "cores" in contrast to CPUs which contain a small number of relatively fast cores. GPUs can attain higher performance than CPUs when each core can perform an independent computation. Because each element C_{ij} can be computed independently, the Wiener kernel computation maps well to this architecture. Our module has also been tuned to remove bottlenecks in GPU computation (especially in the domain of memory accesses). Overall, we are able to achieve speeds which are very close to the considerable peak throughput of modern GPUs.

This code is available as a repository on Github:

https://github.com/ClarkLabCode/GPUFilterExtraction and is available for use under a GPL license. The code is written in C++ and OpenCL and will work for most modern GPUs (they must be compatible with OpenCL). We provide code that integrates it into Matlab, and it can be substituted into STC-packages like iSTAC [44]. Integration with R or Python is possible through each language's C bindings. We have also included the code used to generate Figs 1 and 2.

We tested the GPU-accelerated code against standard existing code for second-order Wiener kernel computation (iSTAC [44]), which takes advantage of optimized linear algebra libraries to accelerate computations (Fig 1C–1F). For comparisons, the CPU and GPU code was run on the same machine with an Intel Core i7 6700K (4.0 Ghz) processor, an AMD Fury X GPU, and 32 GB of DDR4 RAM. The factors of speed up will be different using different CPUs and GPUs, but these are each reasonable high-end pieces of hardware for comparison.

Acknowledgments

We acknowledge helpful discussions with and feedback from Margarida Agrochao, Juyue Chen, Matthew Creamer, and Emilio Salazar-Gatzimas.

Author Contributions

Conceptualization: OM DAC.

Data curation: OM.

Formal analysis: OM DAC.

Funding acquisition: DAC.

Investigation: OM.

Methodology: OM DAC.

Software: OM.

Validation: OM DAC.

Visualization: OM DAC.

Writing – original draft: OM DAC.

Writing – review & editing: OM DAC.

References

1. Dayan P, Abbott LF. Theoretical neuroscience: Cambridge, MA: MIT Press; 2001.
2. Marmarelis VZ. Nonlinear Dynamic Modeling of Physiological Systems. Piscataway, NJ: IEEE Press; 2004.
3. Wiener N. Nonlinear problems in random theory. Nonlinear Problems in Random Theory, by Wiener Norbert, pp 142 ISBN 0-262-73012-X Cambridge, Massachusetts, USA: The MIT Press, August 1966 (Paper). 1966;1.
4. Sandler RA, Marmarelis VZ. Understanding spike-triggered covariance using Wiener theory for receptive field identification. J Vis. 2015; 15(9):16–. doi: 10.1167/15.9.16 PMID: 26230978
5. Simoncelli EP, Paninski L, Pillow J, Schwartz O. Characterization of neural responses with stochastic stimuli. The cognitive neurosciences. 2004; 3:327–38.
6. Aljadeff J, Lansdell BJ, Fairhall AL, Kleinfeld D. Analysis of neuronal spike trains, deconstructed. Neuron. 2016; 91(2):221–59. doi: 10.1016/j.neuron.2016.05.039 PMID: 27477016
7. Samengo I, Gollisch T. Spike-triggered covariance: geometric proof, symmetry properties, and extension beyond Gaussian stimuli. J Comput Neurosci. 2013; 34(1):137–61. doi: 10.1007/s10827-012-0411-y PMID: 22798148
8. Schwartz O, Pillow JW, Rust NC, Simoncelli EP. Spike-triggered neural characterization. J Vis. 2006; 6 (4):13.
9. Van Steveninck RDR, Bialek W. Real-time performance of a movement-sensitive neuron in the blowfly visual system: coding and information transfer in short spike sequences. Proceedings of the Royal Society of London B: Biological Sciences. 1988; 234(1277):379–414.
10. Bialek W, van Steveninck RR. Features and dimensions: Motion estimation in fly vision. arXiv preprint q-bio/0505003. 2005.
11. French A, Korenberg M, Järvilehto M, Kouvalainen E, Juusola M, Weckström M. The dynamic nonlinear behavior of fly photoreceptors evoked by a wide range of light intensities. Biophys J. 1993; 65(2):832. doi: 10.1016/S0006-3495(93)81116-0 PMID: 8218908
12. Pece A, French A, Korenberg M, Kuster J. Nonlinear mechanisms for gain adaptation in locust photoreceptors. Biophys J. 1990; 57(4):733. doi: 10.1016/S0006-3495(90)82594-7 PMID: 2344461
13. Liu JK, Gollisch T. Spike-Triggered Covariance Analysis Reveals Phenomenological Diversity of Contrast Adaptation in the Retina. PLoS Comput Biol. 2015; 11(7):e1004425. doi: 10.1371/journal.pcbi.1004425 PMID: 26230927
14. Schwartz O, Chichilnisky E, Simoncelli EP, editors. Characterizing neural gain control using spike-triggered covariance. Advances in neural information processing systems; 2002.
15. Pillow JW, Paninski L, Uzzell VJ, Simoncelli EP, Chichilnisky E. Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. J Neurosci. 2005; 25(47):11003–13. doi: 10.1523/JNEUROSCI.3305-05.2005 PMID: 16306413
16. Cantrell DR, Cang J, Troy JB, Liu X. Non-centered spike-triggered covariance analysis reveals neurotrophin-3 as a developmental regulator of receptive field properties of ON-OFF retinal ganglion cells. PLoS Comput Biol. 2010; 6(10):e1000967. doi: 10.1371/journal.pcbi.1000967 PMID: 20975932
17. Victor JD, Shapley RM. Receptive field mechanisms of cat X and Y retinal ganglion cells. J Gen Physiol. 1979; 74(2):275–98. PMID: 490143
18. Rust NC, Schwartz O, Movshon JA, Simoncelli EP. Spatiotemporal elements of macaque v1 receptive fields. Neuron. 2005; 46(6):945–56. doi: 10.1016/j.neuron.2005.05.021 PMID: 15953422
19. Lochmann T, Blanche TJ, Butts DA. Construction of direction selectivity through local energy computations in primary visual cortex. PloS one. 2013; 8(3):e58666. doi: 10.1371/journal.pone.0058666 PMID: 23554913
20. Baker CL. Linear filtering and nonlinear interactions in direction-selective visual cortex neurons: a noise correlation analysis. Vis Neurosci. 2001; 18(03):465–85.

21. McCann G. Nonlinear identification theory models for successive stages of visual nervous systems of flies. J Neurophysiol. 1974; 37(5):869. PMID: 4414838

22. Salazar-Gatzimas E, Chen J, Creamer MS, Mano O, Mandel HB, Matulis CA, et al. Direct Measurement of Correlation Responses in Drosophila Elementary Motion Detectors Reveals Fast Timescale Tuning. Neuron. 2016; 92(1):227–39. doi: 10.1016/j.neuron.2016.09.017 PMID: 27710784

23. Clark DA, Bursztyn L, Horowitz MA, Schnitzer MJ, Clandinin TR. Defining the computational structure of the motion detector in Drosophila. Neuron. 2011; 70(6):1165–77. doi: 10.1016/j.neuron.2011.05.023 PMID: 21689602

24. Jing X, Simpson DM, Allen R, Newland PL. Understanding neuronal systems in movement control using Wiener/Volterra kernels: A dominant feature analysis. J Neurosci Methods. 2012; 203(1):220–32. doi: 10.1016/j.jneumeth.2011.09.014 PMID: 21963576

25. Maravall M, Petersen RS, Fairhall AL, Arabzadeh E, Diamond ME. Shifts in coding properties and maintenance of information transmission during adaptation in barrel cortex. PLoS Biol. 2007; 5(2):e19. doi: 10.1371/journal.pbio.0050019 PMID: 17253902

26. Fox JL, Fairhall AL, Daniel TL. Encoding properties of haltere neurons enable motion feature detection in a biological gyroscope. Proc Natl Acad Sci USA. 2010; 107(8):3840–5. doi: 10.1073/pnas.0912548107 PMID: 20133721

27. Kim AJ, Lazar AA, Slutskiy YB. System identification of Drosophila olfactory sensory neurons. J Comput Neurosci. 2011; 30(1):143–61. doi: 10.1007/s10827-010-0265-0 PMID: 20730480

28. Rivera DC, Bitzer S, Kiebel SJ. Modelling Odor Decoding in the Antennal Lobe by Combining Sequential Firing Rate Models with Bayesian Inference. PLoS Comput Biol. 2015; 11(10):e1004528. doi: 10.1371/journal.pcbi.1004528 PMID: 26451888

29. Sharpee TO, Nagel KI, Doupe AJ. Two-dimensional adaptation in the auditory forebrain. J Neurophysiol. 2011; 106(4):1841–61. doi: 10.1152/jn.00905.2010 PMID: 21753019

30. Slee SJ, Higgs MH, Fairhall AL, Spain WJ. Two-dimensional time coding in the auditory brainstem. J Neurosci. 2005; 25(43):9978–88. doi: 10.1523/JNEUROSCI.2666-05.2005 PMID: 16251446

31. Larkin R, Klein S, Ogden T, Fender D. Nonlinear kernels of the human ERG. Biol Cybern. 1979; 35 (3):145–60. PMID: 518936

32. Sutter EE. Imaging visual function with the multifocal m-sequence technique. Vision Res. 2001; 41 (10):1241–55.

33. Kellman P, van Gelderen P, de Zwart JA, Duyn JH. Method for functional MRI mapping of nonlinear response. NeuroImage. 2003; 19(1):190–9. PMID: 12781738

34. Friston KJ, Josephs O, Rees G, Turner R. Nonlinear event-related responses in fMRI. Magn Reson Med. 1998; 39(1):41–52. PMID: 9438436

35. Maccione A, Hennig MH, Gandolfo M, Muthmann O, Coppenhagen J, Eglen SJ, et al. Following the ontogeny of retinal waves: pan-retinal recordings of population dynamics in the neonatal mouse. J Physiol. 2014; 592(7):1545–63. doi: 10.1113/jphysiol.2013.262840 PMID: 24366261

36. Muthmann J-O, Amin H, Sernagor E, Maccione A, Panas D, Berdondini L, et al. Spike detection for large neural populations using high density multielectrode arrays. Frontiers in neuroinformatics. 2015; 9.

37. Peron S, Chen T-W, Svoboda K. Comprehensive imaging of cortical networks. Curr Opin Neurobiol. 2015; 32:115–23. doi: 10.1016/j.conb.2015.03.016 PMID: 25880117

38. Ghosh KK, Burns LD, Cocker ED, Nimmerjahn A, Ziv Y, El Gamal A, et al. Miniaturized integration of a fluorescence microscope. Nat Methods. 2011; 8(10):871–8. doi: 10.1038/nmeth.1694 PMID: 21909102

39. Jin L, Han Z, Platisa J, Wooltorton JR, Cohen LB, Pieribone VA. Single action potentials and subthreshold electrical events imaged in neurons with a fluorescent protein voltage probe. Neuron. 2012; 75 (5):779–85. doi: 10.1016/j.neuron.2012.06.040 PMID: 22958819

40. Chen T-W, Wardill TJ, Sun Y, Pulver SR, Renninger SL, Baohan A, et al. Ultrasensitive fluorescent proteins for imaging neuronal activity. Nature. 2013; 499(7458):295–300. doi: 10.1038/nature12354 PMID: 23868258

41. St-Pierre F, Marshall JD, Yang Y, Gong Y, Schnitzer MJ, Lin MZ. High-fidelity optical reporting of neuronal electrical activity with an ultrafast fluorescent voltage sensor. Nat Neurosci. 2014; 17(6):884–9. doi: 10.1038/nn.3709 PMID: 24755780

42. Gong Y, Huang C, Li JZ, Grewe BF, Zhang Y, Eismann S, et al. High-speed recording of neural spikes in awake mice and flies with a fluorescent voltage sensor. Science. 2015; 350(6266):1361–6. doi: 10.1126/science.aab0810 PMID: 26586188

43. Sun XR, Badura A, Pacheco DA, Lynch LA, Schneider ER, Taylor MP, et al. Fast GCaMPs for improved tracking of neuronal activity. Nature communications. 2013; 4.

44. Pillow JW, Simoncelli EP. Dimensionality reduction in neural models: an information-theoretic generalization of spike-triggered average and covariance analysis. J Vis. 2006; 6(4):9–.

45. Nickolls J, Dally WJ. The GPU computing era. Micro, IEEE. 2010; 30(2):56–69.

46. Geffen MN, De Vries SE, Meister M. Retinal ganglion cells can rapidly change polarity from Off to On. PLoS Biol. 2007; 5(3):e65. doi: 10.1371/journal.pbio.0050065 PMID: 17341132

47. Krizhevsky A, Sutskever I, Hinton GE, editors. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems; 2012.

48. Strigl D, Kofler K, Podlipnig S, editors. Performance and Scalability of GPU-Based Convolutional Neural Networks. PDP; 2010.

49. Ciregan D, Meier U, Schmidhuber J, editors. Multi-column deep neural networks for image classification. Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on; 2012: IEEE.