

1 VolPy: automated and scalable 2 analysis pipelines for voltage 3 imaging datasets

4 Changjia Cai¹, Johannes Friedrich², Eftychios A Pnevmatikakis², Kaspar
5 Podgorski³, Andrea Giovannucci^{1,4}

*For correspondence:

6 agiovann@email.unc.edu (AG);
7 podgorskik@janelia.hhmi.org (KP)

8 ¹Joint Department of Biomedical Engineering at University of North Carolina at Chapel
9 Hill and North Carolina State University, Chapel Hill, NC, USA; ²Flatiron Institute, Simons
10 Foundation, New York, NY, USA; ³Janelia Research Campus, Howard Hughes Medical
11 Institute, Ashburn, VA, USA; ⁴Neuroscience Center, University of North Carolina at Chapel
12 Hill, Chapel Hill, NC, USA

12 **Abstract** Voltage imaging enables monitoring neural activity at sub-millisecond and
13 sub-compartment scale, and therefore opens the path to studying sub-threshold activity, synchrony,
14 and network dynamics with unprecedented spatio-temporal resolution. However, high data rates
15 (>800MB/s) and low signal-to-noise ratios have created a severe bottleneck for analysis of such
16 datasets. Here we present *VolPy*, the first turn-key, automated and scalable pipeline to pre-process
17 voltage imaging datasets. *VolPy* features fast motion correction, memory mapping, segmentation,
18 and spike inference, all built on a highly parallelized and computationally efficient framework that
19 optimizes memory and speed. Given the lack of single cell voltage imaging ground truth examples,
20 we introduce a corpus of 24 manually annotated datasets from different preparations and voltage
21 indicators. We benchmark *VolPy* against this corpus and electrophysiology recordings,
22 demonstrating excellent performance in neuron localization, spike extraction, and scalability.

24 Introduction

25 While several methods have been developed to process voltage imaging data at mesoscopic scale
26 and multi-unit resolution (*Marshall et al., 2016; Carandini et al., 2015; Akemann et al., 2012*), to
27 date there is no established pipeline for large-scale single cell analysis, which was only recently
28 necessitated by sensitive new voltage indicators (*Knöpfel and Song, 2019; Abdelfattah et al., 2019;*
29 *Adam et al., 2019; Kannan et al., 2018; Piatkevich et al., 2019, 2018; Roome and Kuhn, 2018*).
30 Indeed, voltage imaging datasets present significant new challenges compared to calcium imaging,
31 calling for new approaches. On the one hand, dataset sizes have increased one or two orders of
32 magnitude (Tens of GBs vs TBs per hour), and on the other hand, assumptions of existing calcium
33 imaging analysis methods may be inappropriate. For instance, non-negative matrix factorization
34 (NMF) methods (*Giovannucci et al., 2019*) fail when applied to voltage imaging data for three
35 reasons (*Buchanan et al., 2018*): (i) while good segmentation approaches exist for somatic imaging,
36 these fail for other imaging modalities, (ii) it is difficult to separate weak components from noise
37 using current NMF approaches; (iii) since voltage traces typically display both positive and negative
38 fluctuations around the baseline resting potential, the NMF framework, based on non-negativity in
39 both spatial and temporal domains, is not readily applicable to voltage imaging data.

40 Related work

41 Some relevant methods are beginning to populate the literature. For instance, ad-hoc solutions
42 presented in (*Abdelfattah et al., 2019*) provide interesting starting points to extract and denoise
43 spikes semi-automatically, but suffer from some drawbacks. First, they require manual or semi-
44 manual selection of neurons, which is both labor intensive and prone to irreproducibility. Second,
45 the algorithms do not scale well in computational time and memory. Finally, these algorithms
46 are not embedded into a reusable and well documented format, which hinders their reuse by a
47 broad community (*Teeters et al., 2015*). A more standardized approach is provided by (*Adam et al.,*
48 *2019; Buchanan et al., 2018*). However, this method does not embed an adaptive and automated
49 mechanism for spike extraction and is not integrated in a robust, scalable and multi-platform
50 framework. Further, lack of ground truth datasets has so far hindered the validation of all these
51 approaches. In summary, no validated, complete, scalable and automatic analysis pipeline for
52 voltage imaging data analysis exists to date.

53 Contributions

54 To address these shortcomings, we established objective performance evaluation benchmarks and
55 a new analysis pipeline for pre-processing voltage imaging data, which we named *VolPy*. First, in
56 order to establish a common validation framework and to automate neuron segmentation, we
57 created a corpus of annotated datasets with manually segmented neurons. Second, we used
58 this benchmark to train a supervised algorithm to automatically localize and segment cells via
59 convolutional networks (*He et al., 2017*). Third, we introduced an improved algorithm to denoise
60 fluorescence traces and extract single spikes, which builds upon the SpikePursuit prototype (*Ab-*
61 *delfattah et al., 2019*). We modified the core SpikePursuit algorithm to achieve better performance
62 and scalability, both by speeding up the underlying optimization algorithm, and by building the
63 infrastructure to parallelize it efficiently and with low memory requirements. Notably, the algorithm
64 is automatically initialized using the neural network for localizing and segmenting neurons, a task
65 that was previously performed manually. Fourth, we quantitatively evaluated *VolPy* neuron segmen-
66 tation, spike extraction and scalability. Segmentation was evaluated on 24 datasets, encompassing
67 different brain areas, animal preparations and voltage indicators (Tables 1 and 2). The performance
68 of *VolPy* on the validation set was high for datasets with more training samples, but progressively
69 degraded when less data was available. When compared with electrophysiology data, *VolPy* spike
70 extraction featured F_1 scores mostly above 90% on three example neurons. The computational
71 performance of *VolPy* was evaluated on the largest dataset available to us and showed promising
72 results in terms of computational time (up to 66 frames/sec) and memory requirements (down to
73 1.5X RAM of the original dataset size).

74 We integrated our methods within the *CalmAn* ecosystem (*Giovannucci et al., 2019*), a popular
75 suite of tools for single cell resolution brain imaging analysis. This integration allowed us to use
76 and extend *CalmAn's* tools for motion correction and memory mapping to enable scalability of our
77 algorithms. In particular, we adapted *CalmAn* to perform motion correction (*Pnevmatikakis and*
78 *Giovannucci, 2017*), memory mapping (*Giovannucci et al., 2019*), and run the modified SpikePursuit
79 algorithm on voltage imaging data. Besides the obvious computational advantages, this made
80 *VolPy* immediately available to the research labs already relying on the *CalmAn* ecosystem.

81 In summary, we have developed a validated, scalable, turn-key, documented and easily installed
82 voltage imaging analysis pipeline that has been packaged into a popular open source software suite.
83 This will enable an increasing number of laboratories to exploit the advantages provided by voltage
84 imaging and therefore accelerate the pace of discovery in neuroscience.

85 The paper is organized as follows. We first report the new methods developed in *VolPy*, then we
86 benchmark their performance, and finally we discuss some implications. We leave most of the fine
87 implementation details for the Material and Methods section.

Table 1. Properties of three heterogeneous types of datasets. For each type of dataset the name, organism, brain region, source, imaging rate, voltage indicator, and the total number of neurons selected by the manual annotators are given.

Name	Organism	Brain region	Source	Rate (Hz)	Indicator	# neurons
L1	Mouse	L1 cortex	<i>Abdelfattah et al. (2019)</i>	400	Voltron	523
TEG	Zebrafish	Tegmental	<i>Abdelfattah et al. (2019)</i>	300	Voltron	107
HPC	Mouse	Hippocampus	<i>Adam et al. (2019)</i>	1000	paQuasAr3-s	41

88 Methods

89 Creation of a corpus of annotated datasets

90 To date there is no metric to establish whether voltage imaging algorithms for single cell localization
91 and/or segmentation perform well in practice. To overcome this problem, and with the goal of
92 developing new supervised algorithms, we generated a corpus of annotated datasets (Ground
93 truth, GT) in which neurons are manually segmented. GT is constructed by human labelers from
94 two summary images (mean and local correlation images, Figure 1 B and C) and a pre-processed
95 movie that highlights active neurons (local correlation movie, Suppl Movie 1). More specifically,
96 after motion correction, we generate a mean image, a correlation image and a correlation video as
97 follows:

98 *Mean image.* To compute the mean image, we average the movie across time for each pixel and
99 normalize by the pixel-wise z-score.

100 *Correlation image.* The correlation image is a variation of that implemented in (*Smith and Häusser,*
101 *2010*), which is applied to a baseline-subtracted movie. To estimate the baseline of the movie,
102 frames are first binned according to the window length (a parameter set to 1 second). We compute
103 the 8th running percentile of the signal for each pixel. Intermediate values of the baseline are
104 inferred by spline interpolation, which is a fast approximation of a running window. After removing
105 the baseline of the movie, we compute the correlation image of the movie by averaging the temporal
106 correlation of each pixel with its eight neighbor pixels. We also normalize the correlation image by
107 z-scoring when fed to the neural network.

108 *Correlation movie.* We introduce a novel type of denoising operation, the correlation movie. The
109 correlation movie is essentially a running version of the correlation image computed over over-
110 lapping chunks of video frames. This new type of denoising significantly improves the visibility of
111 spikes in voltage imaging movies (see Movie 1). There are two parameters governing the creation
112 of the correlation movie, the chunk size (number of frames over which each correlation image is
113 computed) and stride (the number of frames to skip between consecutive chunks).

114 We implemented parallelized routines which allow to compute efficiently summary images and
115 correlation movies. These routines need only to load in memory small contiguous chunks of the
116 input movies and can process them efficiently in parallel over multiple cores.

117 Guided by these three visual cues, two annotators marked the contours of neurons using the
118 ImageJ Cell Magic Wand tool plugin (*Walker, 2014*). For neurons to be selected both annotators had
119 to agree on the selection, which had to fulfill the following criteria: (i) neurons were very clear on
120 at least one of the three cues; (ii) Neurons were moderately clear in one of the summary images
121 and exhibited a spatial footprint in selected frames of the local correlations movie (see Figure 8).
122 Summary information about the annotated datasets is reported in Table 1. Examples of manual
123 annotations are reported in Figure 3 (red contours).

124 A novel analysis pipeline for voltage imaging

125 Voltage imaging is characterized by high data rates (up to 800 MB/sec). This often leads to the
126 creation of movies that are difficult to manage using conventional computers. Even though scalable
127 algorithms for calcium imaging exist (*Giovannucci et al., 2019*), they fail when applied to voltage
128 imaging. Here we propose a novel scalable pipeline for automated analysis that performs prepro-

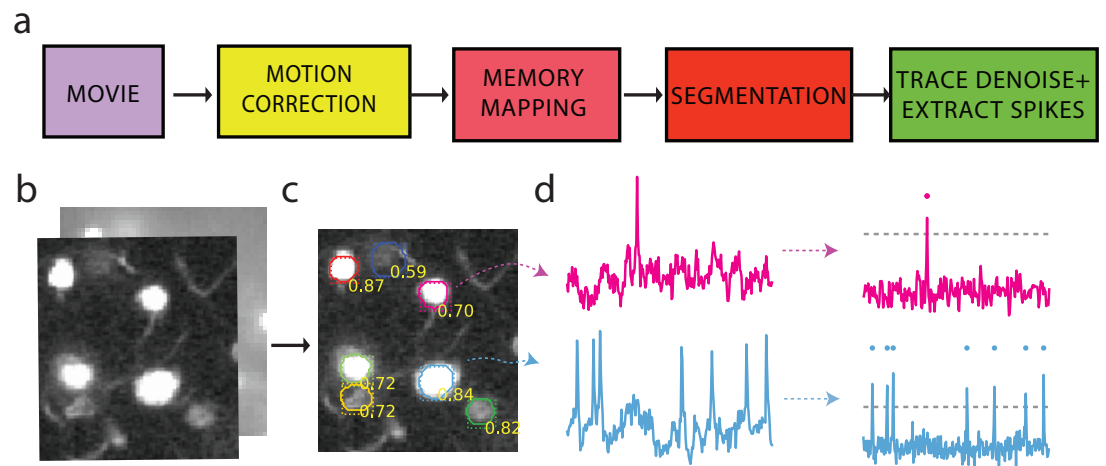


Figure 1. Analysis pipeline for voltage imaging data. (a) Four pre-processing steps are required to extract spikes and neuron locations from voltage imaging movies. (b) Correlation image (front) and mean image (back) of one of the Layer1 neocortex movies as the input of the segmentation step. (c) The segmentation step outputs class probabilities, bounding boxes and contours. The results are overlaid to the correlation image in (b). (d) Result of trace denoising and spike extraction. The gray dashed horizontal line represents the inferred spike threshold.

129 ccessing steps required to extract spikes and sub-threshold activity from voltage imaging movies. In
130 Figure 1 we illustrate the proposed standard pipeline for analyzing voltage imaging data. First, input
131 data is processed to remove motion artifacts with parallelized algorithms, and saved into a memory
132 map file format that enables efficient concurrent access. In a second stage, *VolPy* localizes candidate
133 neurons using supervised algorithms (Figure 1a and c). Finally, *VolPy* denoises fluorescence traces,
134 infers spatial footprints, and extracts neural activity of each neuron through unsupervised learning
135 (Figure 1a and d). Notice that the presented framework is modular, and therefore allows for easy
136 testing of new algorithms by replacing individual components of the pipeline. In what follows we
137 present each stage of the *VolPy* pipeline in detail.

138 Motion correction and memory mapping

139 First, movies need to be corrected for sample movement. We performed this registration relying on a
140 variation of the algorithm described in (*Giovannucci et al., 2019; Pnevmatikakis and Giovannucci,*
141 *2017*), which exploits multi-core parallelization and memory mapping to register frames to a
142 template based on cross-correlation. The only variation with respect the original algorithm is that
143 the new implementation can perform motion correction on a large number of small files containing
144 a single image (a typical output format of fast imaging cameras). This avoids the memory-intensive
145 job of transforming single image files into multi-page files, and limitations of file size. Motion
146 correction, similarly to (*Giovannucci et al., 2019*), is performed in parallel over multiple segments of
147 the same movie and the result is directly stored in a memory mapped file that is efficiently readable
148 frame-by-frame (Fortran (F) order, see Materials and Methods). Relying on the algorithms of *CalmAn*,
149 we then efficiently create a second copy of the file that allows rapid pixel by pixel reads (C order, see
150 Materials and Methods) instead of frame by frame (memory mapping, Figure 1a). This enables a
151 fundamental feature of *VolPy*, that is the ability to quickly read arbitrary portions of the field of view
152 in any direction without having to load the full movie into memory. In summary, the first two steps
153 of the pipeline generate two copies of the motion corrected movie, one efficiently and concurrently
154 read frame-by-frame, and one pixel by pixel. This allows parallelization of all the operations which
155 are required to generate summary images and denoise the signal, as specified below.

Table 2. All annotated datasets for segmentation of *VolPy*. For each dataset the name, size of datasets and number of neurons.

Name	Size	#	Name	Size	#
L1.00.00	20000*512*128	84	HPC.29.04	20000*164*96	2
L1.01.00	20000*512*128	53	HPC.29.06	20000*228*96	2
L1.01.35	20000*512*128	69	HPC.32.01	20000*256*96	4
L1.02.00	20000*512*128	61	HPC.38.05	20000*176*92	4
L1.02.80	20000*512*128	43	HPC.38.03	20000*128*88	2
L1.03.00	20000*512*128	79	HPC.39.07	20000*264*96	5
L1.03.35	20000*512*128	57	HPC.39.03	20000*276*96	5
L1.04.00	20000*512*128	43	HPC.39.04	20000*336*96	4
L1.04.50	20000*512*128	34	HPC.48.01	20000*224*96	2
TEG.01.02	10000*364*320	33	HPC.48.05	20000*212*96	4
TEG.02.01	10000*360*256	29	HPC.48.07	20000*280*96	2
TEG.03.01	10000*508*288	45	HPC.48.08	20000*284*96	3

156 Segmentation

157 The low SNR of voltage imaging data hinders the applicability of the segmentation methods previ-
158 ously devised for calcium imaging data (*Pnevmatikakis et al., 2016*). Here we propose to initialize
159 denoising algorithms with supervised learning approaches. While previous attempts at cell localiza-
160 tion and segmentation have extended U-Net fully convolutional network architectures (*Falk et al.,*
161 *2019*), in our hands this family of methods failed when facing datasets in which neurons overlap
162 (Figure 3a). We hypothesize that this happens since U-Net is a semantic segmentation approach,
163 which aims at separating neurons pixels from the background pixels, and therefore performs poorly
164 in our instance segmentation task of separating overlapping neurons. We approached the problem
165 with Mask R-CNN, a convolutional network for object localization and segmentation (*He et al., 2017*).
166 Mask R-CNN is a particularly promising architecture as it enables to separate overlapping objects in
167 a specific area by providing each object with a unique bounding box.

168 The network, which is trained with a corpus of annotated datasets generated by us, takes sum-
169 mary images as input and outputs contours and bounding boxes of candidate neurons (Figure 1c),
170 along with a class probability. An example of the network inference on a validation dataset by
171 *VolPy* is shown in Figure 2. The resulting network performs well in our task on widely different
172 datasets.

173 Trace denoising and spike extraction

174 Classical algorithms for denoising calcium imaging movies and extracting spikes from the corre-
175 sponding fluorescence traces fail when applied to voltage imaging movies. On the one hand, the low
176 signal-to-noise ratio and the complex background fluorescence require new methods for refining
177 spatial footprints, and on the other hand, substantially different biophysical models underlie the
178 temporal dynamics of the fluorescence associated to spikes. To solve both problems, we build
179 upon and extend the SpikePursuit algorithm (*Abdelfattah et al., 2019*). In particular, we improve
180 SpikePursuit in the following directions (see Material and Methods for details):

- 181 • While the original version of the algorithm required manual selection of candidate neurons,
182 *VolPy* automatically initializes it using the output of the trained Mask R-CNN (Figure 1c and d).
- 183 • A minimal amount of data needs to be loaded in memory thanks to the memory mapping
184 infrastructure, thereby reducing memory requirements.
- 185 • We increase the reliability of the underlying inference algorithm, by introducing a more robust
186 estimate of the background.
- 187 • We scale up the performance by improving the algorithms which perform Ridge regression

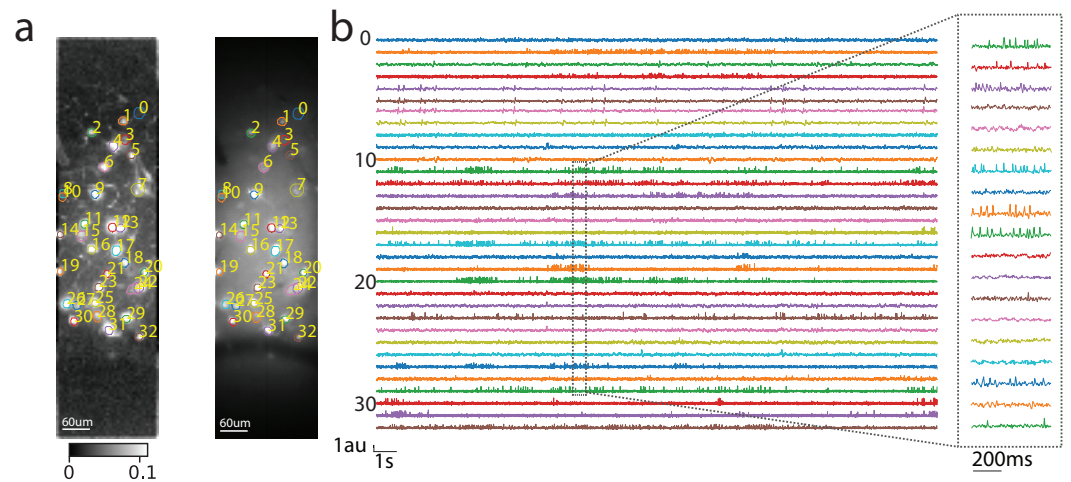


Figure 2. Result of processing a mouse L1 neocortex voltage imaging dataset using the *VolPy* pipeline. (a) Correlation image (left) and mean image (right) overlaid with contours detected by *VolPy*. (b) Temporal traces corresponding to neurons in panel (a) extracted by *VolPy* (left). The dashed gray portion of the traces is magnified on the right.

188 during inference of spikes and spatial masks.

189 Embarrassingly parallel computing in *VolPy*

190 Unlike *CalmAn*, which is based on a Map-Reduce framework to parallelize execution, *VolPy* relies on
191 an embarrassingly parallel paradigm (*Herlihy and Shavit, 2011*). Embarrassingly parallel solutions
192 exploit the lack of dependence among tasks to efficiently deploy concurrency. Indeed the core of
193 *VolPy* algorithms decouples computations so that each neuron is processed independently.

194 First, motion correction in *VolPy* is parallelized by processing temporal chunks of movie data
195 on different CPUs while saved in a memory mapped file which is efficiently read frame-by-frame.
196 second, the various summary images and correlation movies can be computed in parallel processing
197 contiguous temporal chunks of the memory mapped movies. Subsequently, the motion corrected
198 file is processed and saved into another memory mapped file which efficiently read pixel-by-pixel.
199 Finally, during trace denoising and spike extraction, candidate neurons can be processed in parallel
200 without significant memory overhead based on the fact that the signal of each neuron is localized
201 in pixels near to the center of the neuron. Exploiting this locality, *VolPy* processes in parallel context
202 regions surrounding each candidate neuron (see Materials and Methods) by reading concurrently
203 from the pixel-by-pixel memory mapped file. Each process extracts denoised fluorescence signals
204 and spikes from the corresponding context region. In conclusion, *VolPy* enables automatic analysis
205 of large scale voltage imaging datasets. In Figure 2, we report the result of preprocessing an
206 example mouse L1 neocortex voltage imaging dataset with the *VolPy* pipeline.

207 Results

208 In what follows we report a systematic evaluation of *VolPy* against ground truth in terms of perfor-
209 mance in identifying neurons, spike extraction and scalability.

210 *VolPy* localizes neurons using a moderate amount of training data

211 We trained a modified version of the Mask R-CNN network architecture (see Material and Methods
212 for details) on three heterogeneous types of datasets (Table 1) and evaluated its performance using
213 3-fold cross validation (see Table 2 and Materials and Methods for details). In Figure 3a, we com-
214 pared the contours predicted by *VolPy* with manual annotations on three example datasets: *VolPy* is
215 able to identify candidate neurons even in conditions of low signal-to-noise and spatial overlap. In

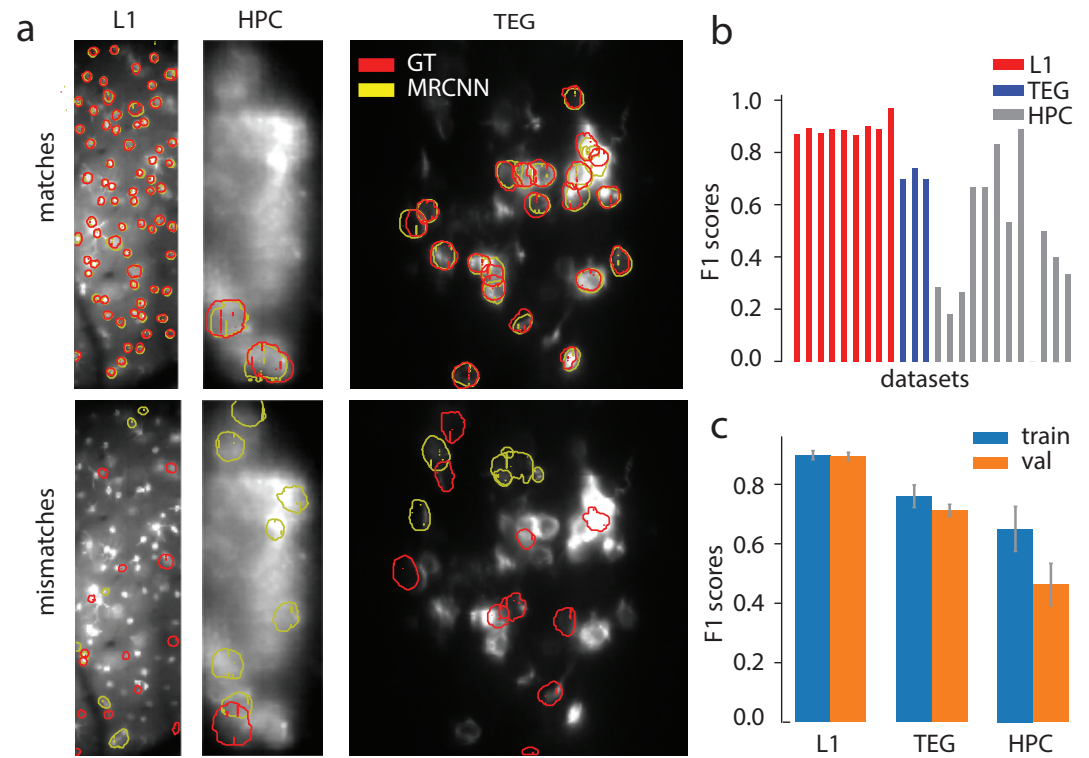


Figure 3. Evaluation of *VoIPy* segmentation. (a) Evaluation of segmentation against three manually annotated datasets including mouse sensory cortex (left, Voltron, dataset L1.00.00), mouse hippocampus (center, paQuasAr3, dataset HPC.48.08), and larval zebrafish (right, Voltron, dataset TEG.01.02). In the upper panels, neurons that are found by both *VoIPy* (yellow contours) and manual annotators (red contours) are displayed over the mean image. The bottom panels display neurons that are found by *VoIPy* but are not present in the ground truth (yellow, False Positives) and neurons that are in the ground truth but are not found by *VoIPy* (red, False Negatives). (b) F_1 score performance of *VoIPy* for all the evaluated datasets. The F_1 score is computed through stratified cross-validation (see Material and Methods). (c) Average performance on training and validation sets grouped by dataset type (see also Table 3). Error bar represents one standard deviation.

216 order to quantify *VoIPy* performance in detecting neurons, we employed a precision/recall frame-
 217 work (see Material and Methods for details), which accounts for the amount of overlap between
 218 predicted and ground truth neurons when assigning matches and mismatches (*Giovannucci et al.,*
 219 **2019**). In Figure 3b and Table 3 we summarize the F_1 score for all the probed datasets. The results
 220 indicate that our segmentation approach performs well provided sufficient neurons are fed to train
 221 the algorithm. Indeed, *VoIPy* obtained F_1 scores of 0.89 ± 0.01 on the L1 dataset (532 neurons in total),
 222 0.71 ± 0.02 on the TEG datasets (107 neurons), and 0.46 ± 0.07 on the HPC dataset (39 neurons). In
 223 case of TEG, the performance of *VoIPy* is fair considering that the network was trained with only two
 224 datasets of this type. In the HPC datasets however the performance on both training and test sets is
 225 relatively inferior. We hypothesize that this is due to the fact that not enough data are available (see
 226 #neurons column in Table 1), possibly combined with the low signal to noise typical of this dataset
 227 type. Note that we used a single neural network trained on the three dataset types simultaneously.
 228 Despite clear differences in neuronal shapes, size, SNRs and data acquisition system the network
 229 performed well across them, suggesting that it will generalize to similar datasets. However, new
 230 datasets deviating substantially from these typologies will need to be added to the training set to
 231 improve generalization performance.

Table 3. Results of *VolPy* for segmentation. For each type of datasets, number of datasets, number of neurons, recall, precision, F_1 score for training and validation computed by stratified cross-validation are provided.

Name	#datasets train/val	#neurons train/val	recall(%) train/val	precision(%) train/val	F_1 (%) train/val
L1	6/3	349 ± 7/174 ± 7	86 ± 4/85 ± 3	94 ± 2/95 ± 1	90 ± 2/89 ± 1
TEG	2/1	71 ± 7/36 ± 7	70 ± 3/67 ± 2	83 ± 7/77 ± 3	76 ± 4/71 ± 2
HPC	8/4	26 ± 1/13 ± 1	88 ± 11/66 ± 7	55 ± 7/40 ± 12	65 ± 8/46 ± 7

232 ***VolPy* detects with fidelity single spikes from voltage imaging data**

233 We validated the *VolPy* SpikePursuit algorithm on three voltage imaging datasets in which electro-
234 physiology was simultaneously recorded with voltage imaging (see Figure 4). We automatically
235 analyzed voltage imaging data *in-vivo* recordings from mouse L1 neocortex and Zebrafish Tegmental
236 area (*Abdelfattah et al., 2019*) with the *VolPy* pipeline. The output of the algorithm are spatial
237 footprints, voltage traces, and corresponding spike timings. Spikes for electrophysiology recordings
238 were obtained by thresholding (see Figure 4). Spikes are matched against ground truth by solving
239 a linear sum assignment problem using the Hungarian algorithm (see Material and Methods for
240 details). The F_1 score of each dataset (see Figure 4b) is computed relying on a precision/recall
241 framework based on matched and unmatched spikes. We observe that *VolPy* performs well on
242 all datasets (the F_1 score across three datasets is 0.94 ± 0.03) and confirms that single spikes from
243 voltage imaging data can be automatically extracted with fidelity.

244 ***VolPy* enables the analysis of large voltage imaging datasets on small and medium 245 sized machines**

246 We examined the performance of *VolPy* in terms of processing time and peak memory for the
247 datasets presented above. We ran our tests on a linux-based desktop (Ubuntu 18.04) with 16
248 CPUs (Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz) and 64 GB of RAM. For segmentation, we used a
249 GeForce RTX 2080 Ti GPU with 11 GB of RAM memory.

250 Figure 5a reports the *VolPy* processing time in function of the number of frames. The results
251 show that the processing time scales linearly in the number of frames. Processing 50 candidate
252 neurons in a 1.5 minutes long video (512*128 pixel FOV) takes about 9 minutes. SpikePursuit (red
253 bar) accounts for most of the processing time.

254 In order to probe the benefits of parallelization, we ran *VolPy* 5 times on the same hardware
255 while limiting the runs to 1, 2, 4, 6 and 8 CPUs respectively (Figure 5b). We observed significant
256 performance gains due to parallelization, especially in the motion correction and SpikePursuit
257 phase, with a maximum speed-up of 2.5X. Simultaneously, we recorded the peak memory usage of
258 *VolPy* while running on a different number of CPUs for each run. Figure 5c shows how the peak
259 memory increases with the number of threads. Therefore, *VolPy* enables speed gains by trading-off
260 execution time for memory usage.

261 **Discussion**

262 **Enabling automated and scalable analysis of voltage imaging data**

263 Recording voltage changes in populations of neurons is necessary to dissect the details of infor-
264 mation processing in the brain. Voltage imaging is currently the only technique that promises to
265 achieve this goal with high spatio-temporal resolution. Indeed, voltage imaging has a long history
266 of development, having been widely used for *in-vivo* studies in the past. However, poor signal-to-
267 noise ratio, photo-toxicity, bleaching, and other difficulties have so far hindered its wider use to
268 answer questions at a cellular level. Recently, however, voltage imaging seems to have reached a
269 point of inflexion and some notable examples are leading the way to new exciting developments
270 (*Abdelfattah et al., 2019; Roome and Kuhn, 2018; Adam et al., 2019; Piatkevich et al., 2019, 2018*).

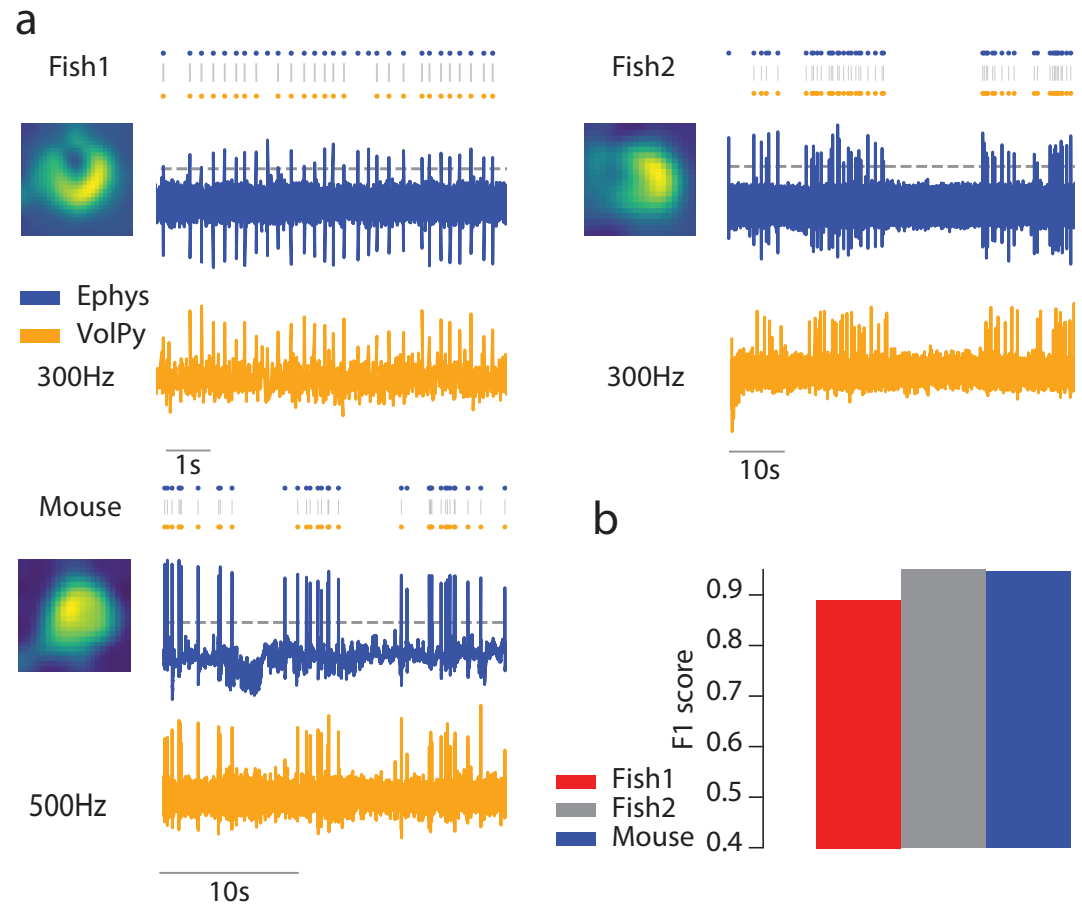


Figure 4. Validation of *VolPy* performance against electrophysiology. (a) Performance of *VolPy* in detecting spikes validated on three datasets from zebrafish (Fish1 and Fish2) and mouse (Mouse). For each dataset, the denoised spatial filter of the target neuron is presented on the left, while electrophysiology (top, blue) and fluorescence signal denoised by *VolPy* (bottom, orange) are reported on the right. Spikes from electrophysiology (blue dots) are obtained by thresholding (gray horizontal dotted line) while spikes from voltage imaging (orange dots) are the output of *VolPy*. Spikes are matched between the two groups by solving a linear assignment problem (see Material and Methods, gray vertical lines). (b) The F_1 scores for each dataset are computed based on the matched and unmatched spikes.

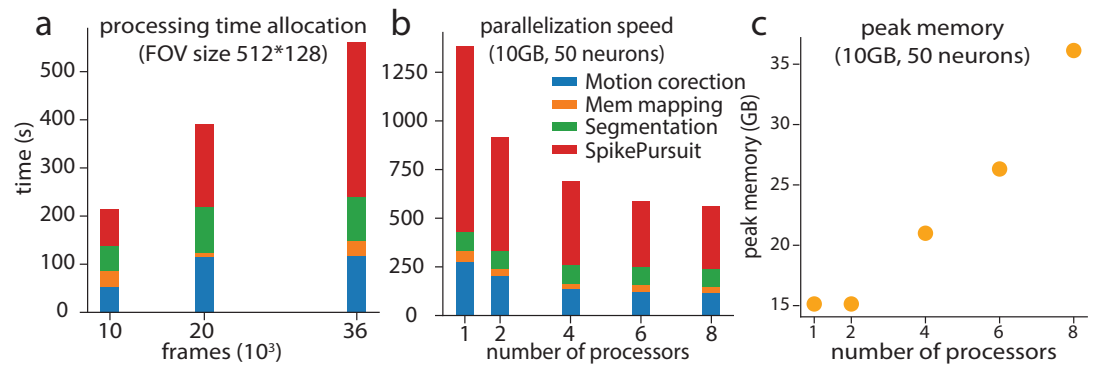


Figure 5. Time and memory performance of *VolPy*. (a) Processing time for *VolPy* as a function of the number of frames on a 512*128 pixels dataset initialized with 50 neurons. Processing time is the sum of motion correction (blue), memory mapping (orange), segmentation (green), and SpikePursuit (red) times. The results indicate a near linear scaling of the processing time with the number of frames. (b) Processing time for *VolPy* utilizing 1, 2, 4, 6 and 8 processors in parallel on a 10 GB dataset and 50 detected neurons. *VolPy* achieves a 2.5-fold speed up when running in parallel on 8 cores. (c) Peak memory usage of *VolPy* in function of the number of processors. Processing in parallel can lead to fair speed gains by regulating the trade-off between time and memory consumption.

271 Despite the recent availability of high quality datasets, there is currently no established and vali-
272 dated pipeline for the analysis of voltage imaging data. The unprecedented data size (one order of
273 magnitude larger than already challenging calcium imaging datasets), low SNR, and high degree
274 of signal mixing have so far limited the development of novel algorithms. Moreover, the lack of a
275 universal benchmark prevents further quantitative comparisons. In this paper we provided both
276 a corpus of manually segmented datasets and *VolPy*, the first turn-key, fully automatic, scalable
277 and reproducible pipeline for the analysis of large scale voltage imaging datasets. *VolPy* equips
278 experimenters with efficient computational routines for data handling, motion correction, memory
279 mapping, neuron localization and segmentation, trace denoising and spike extraction. *VolPy* builds
280 upon several optimized and robust routines of the well-established *CalMan* framework, which it
281 extends to deal with voltage imaging data.

282 In particular, our contributions develop along the following lines. We provided a corpus of
283 24 annotated datasets from different brain areas, collection systems and voltage indicators. We
284 developed an automated segmentation supervised algorithm which relies on a Mask R-CNN neural
285 network architecture. We trained a *single* network for all types of considered datasets and evaluated
286 it using cross-validation. The algorithm performance is excellent when enough training data is
287 provided, but smoothly degrades when input data is scarce for specific types of datasets. Regard-
288 ing trace denoising and spike extraction approaches, we built upon the SpikePursuit algorithm
289 (*Abdelfattah et al., 2019*) and extended it to make it fully automatic, to improve its reproducibility,
290 performance, and to enhance its scalability. We benchmarked the performance of *VolPy* in extract-
291 ing action potentials against ground truth electrophysiology, with results averaging an outstanding
292 F_1 score of 0.94. Scalability is achieved by leveraging the infrastructure previously deployed in
293 *CalMan*, which we adapted to enable the parallel processing of multiple neurons. *VolPy* enables a
294 time-memory trade-off which can be tuned based on the available computing power. We demon-
295 strated that *VolPy* enables voltage imaging data analysis on desktop computers. Towards our
296 goal of providing a single package for dealing with standard problems arising in the analysis of
297 imaging data, *VolPy* is fully integrated into *CalMan* and is therefore immediately available to many
298 laboratories worldwide. The proposed framework is therefore poised to promote the distribution
299 of voltage imaging within the neuroscience community, and in consequence to open the path to a
300 new generation of experiments bridging the gap between electrophysiology and optical imaging.

301 **Future directions**

302 As more data become available and more users adopt *VolPy*, we plan to develop a graphical user
303 interface for experimentalists to manually label datasets and transfer the resulting annotations to a
304 cloud server, which we will periodically use to retrain and improve the performance of our system.

305 SpikePursuit is built upon linear methods with a small number of easily-interpreted parameters.
306 An advantage of this approach is that the parameters for can be tailored to different datasets
307 by end users (for example: context area, number of spikes used for templates, filter bandwidth
308 and confidence in segmentation). A continuing challenge for optical physiology is the limited
309 electrophysiological ground truth available for training complex spike detection models. As more
310 training data become available, we expect machine learning approaches to supersede the spatial
311 and/or temporal filtering steps used by SpikePursuit within *VolPy*. Even without large training
312 datasets, algorithmic improvements may be possible. For example, SpikePursuit implements
313 efficient but approximate spike detection using matched filtering with a single template, but could
314 be extended e.g. to include multiple templates or subtractive interference cancellation (*Franke*
315 *et al., 2015*). *VolPy* and the datasets provided here provide an ideal common ground for comparing
316 such methods.

317 Finally, and similar to our work in calcium imaging (*Giovannucci et al., 2017*), we plan to general-
318 ize our algorithm to real-time scenarios, where activity of neurons needs to be inferred on the fly
319 and frame-by-frame.

320 **Materials and Methods**

321 **Motion correction & Memory mapping**

322 *VolPy* performs motion correction and memory mapping similarly to *CalmAn* (*Giovannucci et al.,*
323 *2019*). For motion correction, *VolPy* uses the NoRMCorre algorithm (*Pneumatikakis and Giovan-*
324 *nucci, 2017*) which corrects non-rigid motion artifacts in two steps. First, motion vectors are
325 estimated with sub-pixel resolution for a set of overlapping patches which tile the FOV. Second,
326 the sub-pixels estimates are upsampled to create a smooth motion field for each frame, which is
327 then applied to correct the original frames. Unlike previously, our new implementation enables to
328 perform motion correction on a large number of small files containing a single image (the typical
329 output of fast imaging cameras). This is achieved by multiple parallel processes reading files incre-
330 mentally and concurrently from the hard drive. This avoids the time- and memory-intensive job of
331 transforming single image files into multi-page or hdf5 files. This modification leads to significant
332 savings in memory, hard drive space and speed.

333 *VolPy* adopts an optimized framework for efficient parallel data read and write. This framework
334 is based on the ipyparallel and memory mapping Python packages (see *Giovannucci et al. (2019)* for
335 more details). In brief, the former enables the creation of distributed clusters across workstations or
336 HPC infrastructures, and the latter enables reading and writing slices of large data tensors without
337 loading the entire file into memory. This is especially important for voltage imaging, considering
338 the larger file sizes compared to calcium imaging. This framework, which in *VolPy* implements an
339 embarrassingly parallel infrastructure, is used across different steps of the pipeline:

- 340 • The output of the motion correction operation is saved into a set of F ordered Python memory
341 map files without creating any other intermediate files. This is done in parallel over all the
342 processed movie chunks.
- 343 • The motion corrected F order files are then consolidated into a single C ordered memory map
344 file. This is also performed in parallel over many processes.
- 345 • During trace denoising and spike extraction, each process loads and processes in parallel a
346 small portion of the field of view.

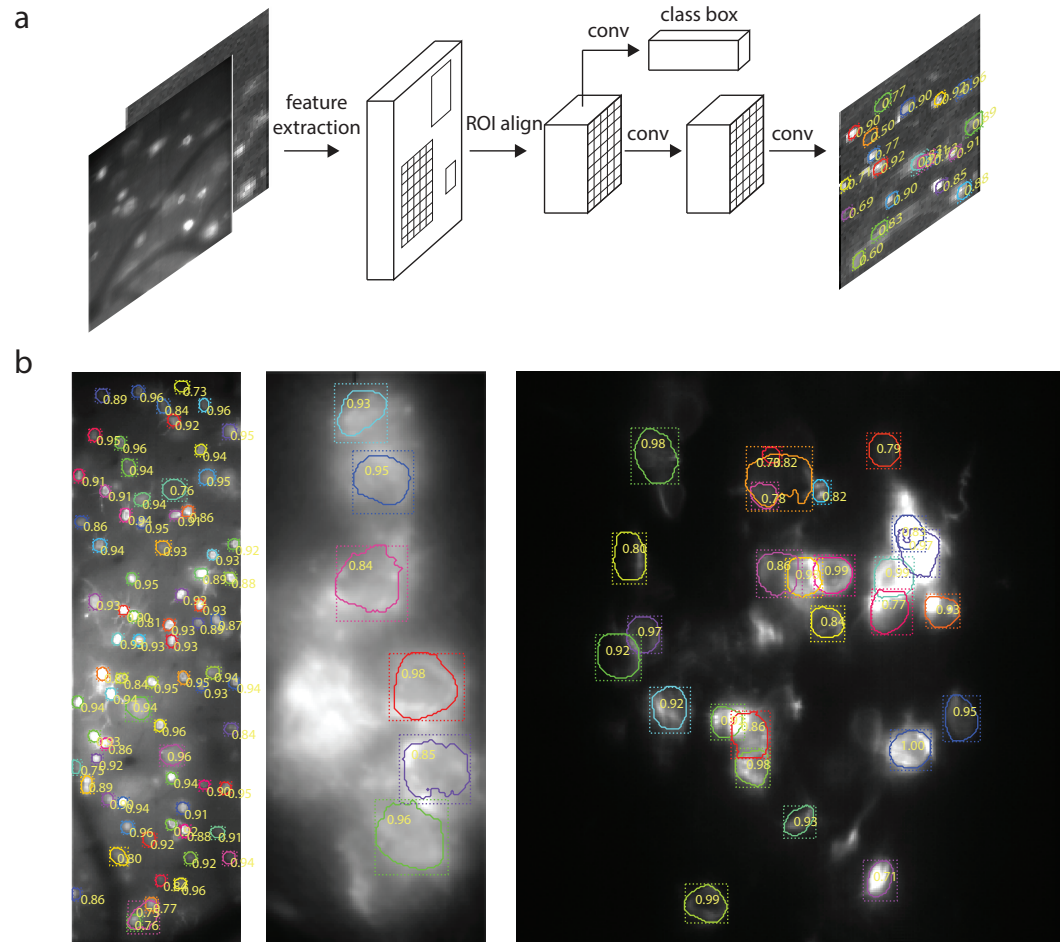


Figure 6. Segmentation algorithm of *VolPy*. (a) Mask R-CNN framework for segmenting neurons with summary images as input to the network. (b) The output of *VolPy* segmentation algorithm run on three example datasets from sensory cortex (left, Voltron, dataset L1.00.00), mouse hippocampus (center, paQuasAr3, dataset HPC.48.08), and larval zebrafish (right, Voltron, dataset TEG.01.02). The mean images are overlaid with contours (solid line), bounding boxes (dotted line) and detection confidence for each candidate neuron. Only neurons with detection confidence greater than 0.7 are displayed.

347 **Creating a corpus of annotated datasets**

348 We generate a corpus of annotated datasets in which neurons are manually segmented. For
349 neurons to be selected at least one of the following criteria needed to be met: (i) Both annotators
350 had to agree on the selection; (ii) Neurons were very clear on at least one of the three cues; (iii)
351 Neurons were moderately clear in one of the summary images and appeared clearly in a few frames
352 of the local correlations movie.

353 Figure 8 shows the process of selecting neurons. Ground truth is inferred by human labelers
354 from mean and local correlation images as well as a local correlation movie which highlights active
355 neurons. Relying on these three visual cues, two annotators marked the contours of neurons
356 (yellow color) using the ImageJ Cell Magic Wand tool plugin (*Walker, 2014*) and saved the result into
357 the ROI manager in ImageJ.

358 **Segmentation via convolutional networks**

359 *VolPy* uses a variation of the Mask R-CNN framework (see Figure 6) to initialize spatial footprints of
360 neurons. In the following section we will introduce the Mask R-CNN framework in the *VolPy* context.

361 Mask R-CNN

362 Mask R-CNN (Figure 6a) is a network architecture which provides simultaneous object localization
363 and instance segmentation via a combination of two network portions: backbone and head. The
364 backbone features a pre-trained convolutional network (such as VGG, ResNet, Inception or others)
365 for extracting features of the input image. Mask R-CNN also exploits another effective backbone:
366 Feature Pyramid Networks (FPN) (Lin et al., 2017), a top-down architecture with lateral connections,
367 which enables the network to extract features on multiple scales from the feature maps. In the
368 head, based on the extracted features, a Region Proposal Network proposes initial bounding boxes
369 for each candidate object, which are fed to two downstream branches. One of them is trained to
370 predict a class label and a bounding box offset which refines the initial bounding box, while the
371 other branch outputs a binary mask for each candidate object.

372 Vo/Py Mask R-CNN

373 We adapt Mask R-CNN to our purpose by introducing the following modifications. We choose a
374 combination of ResNet-50 pre-trained on the COCO dataset and FPN as the backbone. The input
375 of the network is a three channel image: two for the mean images and one for the correlation
376 image. The three channel image is necessary in order to re-use the first few layers which were
377 pre-trained on the COCO dataset. The network is trained to predict only one class, neuron or not
378 neuron (background) instead of a multi-label output.

379 **Training:** We randomly crop the input image into 128x128 crops and apply the following data
380 augmentation techniques using the *imgaug* (Jung et al., 2019) package: flip, rotation, multiply (adjust
381 brightness), Gaussian noise, shear, scale and translation. Each mini-batch contains six cropped
382 images. We train on one GPU the heads (the whole network except the ResNet) of the network for
383 2k iterations with learning rate 0.01 and then train layers after the first three stages of the ResNet
384 (28 layers) for another 2k iterations with learning rate 0.001. We use stochastic gradient descent as
385 our optimizer with a constant learning momentum 0.9. The weight decay is 0.0001.

386 **Validation:** Images are padded with zeroes to make width and height multiples of 64 so that
387 feature maps can be smoothly scaled for the Feature Pyramid Network. We only choose neurons
388 with confidence level greater or equal to 0.7.

389 Trace denoising and spike extraction

390 Trace denoising and spike inference are performed by an improved version of the SpikePursuit
391 algorithm (Abdelfattah et al., 2019), in which we optimized for speed, memory usage, and accuracy.
392 The pseudo-code for the associated computational steps is reported in Algorithm (1) and Figure 7.
393 The algorithm starts by approximating a neuronal signal and the background contamination from
394 the ROI provided by the segmentation step. The algorithm then proceeds iteratively to detect
395 the most prominent spikes, extract a waveform template from detected spikes, use the template
396 to recover similarly-shaped spikes, reconstruct the trace from the recovered spikes, and use the
397 reconstructed trace to improve the spatial filter. These steps are explained in more details below.

398 ROI loading and preprocessing

399 As a result of segmentation, each candidate neuron has an associated binary mask which represents
400 its spatial extent (ROI region R). The ROI is dilated to get a larger region (50x50 pixels by default)
401 centered on the neuron (context region C). Background pixels are defined as all the pixels in the
402 context region at least n_B pixels (12 pixels by default) away from the ROI region (background region).
403 As a first step, all pixels in the context region are efficiently retrieved from the memory mapped file
404 and high-pass filtered as Y_h to compensate for photo-bleaching (Figure 7a-b, Algorithm 1 lines 1–8).
405 The initial temporal trace t_0 associated to a neuron can be approximated either from the mean
406 signal of the ROI region pixels, or as a weighted average across all pixels in the context region when

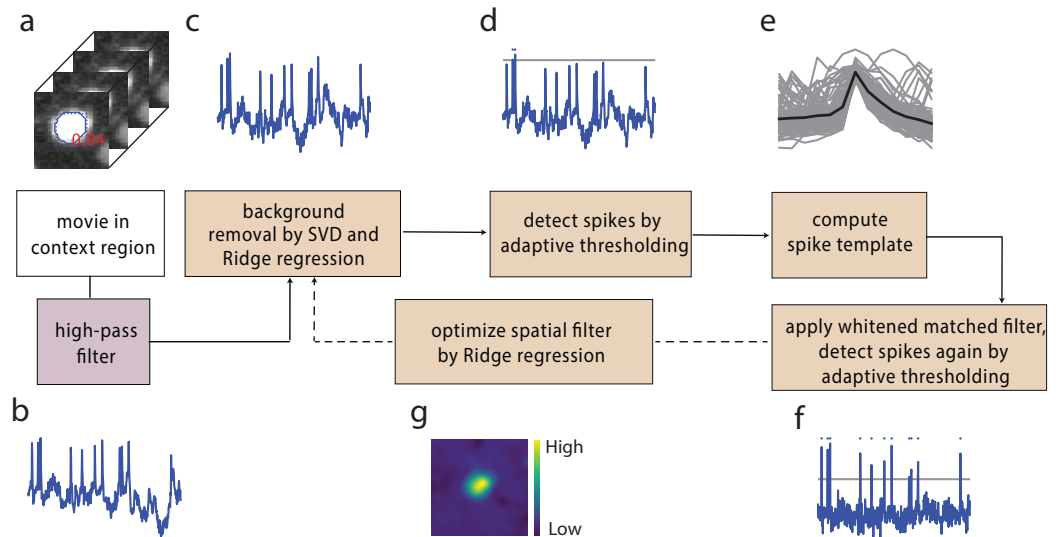


Figure 7. Algorithm for fluorescence trace denoising and spike inference. (a) A small section of the movie (context region) encompassing a candidate neuron and a neighboring area is loaded from the C ordered memory mapped file. (b) After high-pass filtering the movie, the initial temporal trace of the candidate neuron is approximated either from the mean signal of the ROI region pixels, or by applying the spatial filter to the context region if an initial spatial filter is provided. Afterwards, two big steps are executed in loop until convergence (or a maximum of 5 steps). The former ((c),(d),(e) and (f)) estimates spike times, and the latter ((g)) refines the spatial filter. (c) We extract the first 8 principal components of the background pixels using singular value decomposition and then remove the background contamination via Ridge regression. (d) After high-pass filtering the trace, we select spikes with peak larger than an adaptively selected threshold (gray dotted line). The total number of peaks detected in the first round is constrained between 30 and 100. Later rounds of spike detection include all spikes. (e) Waveforms of these spikes (gray) are averaged to obtain a spike template (black line). (f) A whitened matched filter is used to enhance spikes which have a similar shape to the template. (g) Refine spatial filter through Ridge regression. Calculate the weighted average of movie (using the refined spatial filter) as the new temporal trace for the next iteration.

407 a spatial filter \mathbf{w} calculated from previous chunk of data was available (Algorithm 1 lines 9–13):

$$\mathbf{t}_0 = \begin{cases} \frac{1}{n(\mathbf{R})} \sum_{x \in \mathbf{R}} Y_h[:, x] & \text{if } \mathbf{w} \text{ is not given} \\ \sum_{x \in \mathbf{C}} Y_h[:, x] \mathbf{w}[\mathbf{x}] & \text{if } \mathbf{w} \text{ is given} \end{cases} \quad (1)$$

408 where $n(\mathbf{R})$ represents number of pixels in the ROI region.

409 Afterwards, two steps are executed in loop until convergence (or a maximum of 5 steps). The
410 former tries to estimate spike time, and the latter tries to approximate the spatial filter.

411 Spike time estimation

412 In order to estimate spike times from the fluorescence traces (Figure 2c-f) we proceed as follows.

413 First, we compute the singular value decomposition of the background pixels Y_b :

$$Y_b = U \Sigma V \quad (2)$$

414 where U contains the temporal components. This is then used to remove background con-
415 tamination via Ridge regression, in which U_b , the first 8 components of U is the regressor and the
416 temporal trace is the predictor (Algorithm 1 line 15–16).

$$\beta = (U_b^T U_b + \lambda_b \|U_b\|_F^2 I)^{-1} U_b^T \mathbf{t}_0 \quad (3)$$

$$\mathbf{t} = \mathbf{t}_0 - U_b \beta \quad (4)$$

417 We experienced that a very high signal-to-noise ratio neuron with large spatial footprint included in
418 the background pixels led to poor performance due to unregularized linear regression used at this
419 stage in the original SpikePursuit implementation. Use of non-regularized regression to remove
420 the background can allow real signal to be subtracted from neuron traces if the neuron's trace
421 is captured by the background PCs. To ameliorate this issue, we modified the original algorithm
422 by adding an L_2 regularizer to penalize large regression coefficients. This provided more reliable
423 results with respect to the original implementation on multiple datasets.

424 After background removal, the trace is high-pass filtered with a cut-off frequency of 60 Hz and
425 two rounds of spike detection are performed. The first round selects spikes with peak larger than
426 an adaptively selected threshold, while keeping the total number of peaks between 30 and 100
427 (Algorithm 3). A spike template \mathbf{z} is computed by averaging all the peak waveforms:

$$\mathbf{z} = \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{t}[\mathbf{s}[i] - \tau : \mathbf{s}[i] + \tau] \quad (5)$$

428 where \mathbf{s} is the list of spike time, n_s is total number of spikes, τ is the half size of window length.
429 Subsequently, a whitened matched filter (*Franke et al., 2015*) is used to enhance spikes with shape
430 similar to a template. More in details, we use the Welch method to approximate the spectral density
431 of the noise in the fluorescence signal. Second, we scale the signal in the frequency domain to
432 whiten the noise. Finally, we convolve with a time-flipped template. The template we used is the
433 peak-triggered average.

434 The latter round of spike detection incorporates all the spikes detected by applying a newly
435 computed threshold. Then, a reconstructed and denoised trace is computed by convolving the
436 inferred spike train (\mathbf{q}) with the waveform template:

$$\mathbf{r} = \mathbf{z} * \mathbf{q} \quad \text{where } \mathbf{q}[t] = \begin{cases} 1 & \text{if there is a spike at time } t \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

437 Spatial filter refinement

438 The second step, illustrated in Figure 2g, is to refine the spatial filter. The updated spatial filter is
439 computed by Ridge regression, where the reconstructed and denoised trace is used to approximate
440 the high-passed video (Algorithm 1 line 18):

$$\mathbf{w} = (Y_h^T Y_h + \lambda_w \|Y_h\|_F^2 I)^{-1} Y_h^T \mathbf{r} \quad (7)$$

441 Subsequently, the weighted average of movie with the refined spatial filter is used as the updated
442 temporal trace for the following iteration:

$$\mathbf{t} = Y_h \mathbf{w} \quad (8)$$

443 The ridge regression problem was originally solved in SpikePursuit by directly calculating the
444 analytical solution (normal equation). However, the multiplication and inverse of large matrices was
445 computationally inefficient. We decided to apply an iterative and much more efficient algorithm to
446 solve the regression problem (*Paige and Saunders, 1982*) implemented in the Scikit-Learn package
447 ('lsqr').

448 Precision/Recall Framework to measure segmentation performance

449 In order to measure the performance of *VolPy* segmentation, we compared the spatial footprints
450 extracted by *VolPy* with our manual annotations (see (*Giovannucci et al., 2019*) component registra-
451 tion for a detailed explanation). In summary, we computed the Jaccard distance (the inverse of
452 intersection over union) to quantify similarity among ROIs, and then solved a linear assignment
453 problem with the Hungarian algorithm to determine matches and mismatches. Once these were
454 identified, we adopted a precision/recall framework and we defined True Positive (TP), False Positive
455 (FP), False Negative (FN), and True Negative (TN) as follows:

$$\begin{aligned} \text{TP} &= \text{number of matched spatial footprints} \\ \text{FP} &= \text{number of spatial footprints in VolPy but not in GT} \\ \text{FN} &= \text{number of spatial footprints in GT but not in VolPy} \\ \text{TN} &= 0 \end{aligned} \quad (9)$$

456 Next we computed precision, recall and F_1 score of the performance in matching as the following:

$$\begin{aligned} \text{Precision} &= \text{TP}/(\text{TP} + \text{FP}) \\ \text{Recall} &= \text{TP}/(\text{TP} + \text{FN}) \\ F_1 &= 2 \times \text{Precision} \times \text{Recall}/(\text{Precision} + \text{Recall}) \end{aligned} \quad (10)$$

458 Note that the F_1 score is a number between zero and one. The better the performance of
459 matching, the higher the F1 score.

460 Cross-Validation to evaluate segmentation model on limited datasets

461 In order to decrease the selection bias originated from the separation in training and validation
462 datasets and better evaluate Mask R-CNN model on our limited datasets (24 in total), we performed
463 a stratified three-fold cross-validation. The reason we used a stratified three-fold cross-validation
464 rather than a normal three-fold cross-validation is that we want our model train and validate on
465 each type of datasets. We partitioned datasets into three groups so that arbitrary type of data (L1,
466 TEG, HPC) is partitioned equally into three groups without repetition (Figure 2 train/val column
467 shows one group of the partition). During cross-validation two groups were used as training sets
468 while the remaining one as validation set. The cross-validation process was repeated three times
469 with each group used exactly once as validation set.

470 For each run of the cross-validation process, we trained a single network and tested it on both
471 training and validation sets. We then computed the mean and standard deviation of the F1 score
472 for different types of datasets with training and validation sets treated separately.

473 Spike matching

474 In order to validate fidelity of spike extraction algorithm, we needed to match spikes extracted
475 from voltage imaging and electrophysiology datasets. Let v_1, v_2, \dots, v_n be the spike time extracted
476 from voltage imaging traces, and s_1, s_2, \dots, s_m be the spike times from electrophysiology ground truth,
477 where n and m are the total number of spikes respectively. We formulate the problem as a linear
478 sum assignment problem. Let \mathbf{D} be a distance matrix where $D[i, j]$ is the cost of matching spikes v_i
479 and s_j . When the difference of spike-times is larger than a threshold t , we assign a large distance
480 value M :

$$\mathbf{D}[i, j] = \begin{cases} \|v_i - s_j\|, & \text{if } \|v_i - s_j\| < t \\ M, & \text{otherwise} \end{cases} \quad (11)$$

481 Let \mathbf{X} be the Boolean matrix where $\mathbf{X}[i, j] = 1$ if v_i and s_j are matched and 0 otherwise. Each
482 spike can be matched at most once, i.e. at most one element for each row (or column) of \mathbf{X} can be
483 one. The optimal assignment has the cost:

$$\min \sum_i \sum_j \mathbf{D}_{i,j} \mathbf{X}_{i,j} \quad (12)$$

484 We solve this optimization problem using the Hungarian algorithm implemented in the Scipy
485 package and delete matched spikes whose costs are equal to M . After identifying matches and
486 mismatches, we proceeded similarly to what explained above to extract the F_1 score. We define TP,
487 FP, FN, TN similar to Equation 9:

$$\begin{aligned} \text{TP} &= \text{number of matched spikes} \\ \text{FP} &= \text{number of spikes in VolPy but not in GT} \\ \text{FN} &= \text{number of spikes in GT but not in VolPy} \\ \text{TN} &= 0 \end{aligned} \quad (13)$$

488 Then we calculated F_1 score same as Equation 10.

489 Acknowledgments

490 We thank K Svoboda, A Singh, M Ahrens, T Kawashima, Y Shuai, A Cohen, M Xie for providing voltage
491 imaging datasets. We thank H Eybposh for useful discussions. We thank A Singh for the initial
492 version of Python code, and for insightful comments and discussions.

493 References

- 494 **Abdefattah AS**, Kawashima T, Singh A, Novak O, Liu H, Shuai Y, Huang YC, Campagnola L, Seeman SC, Yu J,
495 Zheng J, Grimm JB, Patel R, Friedrich J, Mensh BD, Paninski L, Macklin JJ, Murphy GJ, Podgorski K, Lin BJ, et al.
496 Bright and photostable chemigenetic indicators for extended in vivo voltage imaging. *Science*. 2019 Aug;
497 365(6454):699–704. <https://science.sciencemag.org/content/365/6454/699>, doi: 10.1126/science.aav6416.
- 498 **Adam Y**, Kim JJ, Lou S, Zhao Y, Xie ME, Brinks D, Wu H, Mostajo-Radji MA, Kheifets S, Parot V. Voltage imaging and
499 optogenetics reveal behaviour-dependent changes in hippocampal dynamics. *Nature*. 2019; 569(7756):413.
- 500 **Akemann W**, Mutoh H, Perron A, Park YK, Iwamoto Y, Knöpfel T. Imaging neural circuit dynamics with a
501 voltage-sensitive fluorescent protein. *Journal of Neurophysiology*. 2012 Jul; 108(8):2323–2337. <https://www.physiology.org/doi/full/10.1152/jn.00452.2012>, doi: 10.1152/jn.00452.2012.
- 502 **Buchanan EK**, Kinsella I, Zhou D, Zhu R, Zhou P, Gerhard F, Ferrante J, Ma Y, Kim S, Shaik M, Liang Y, Lu R, Reimer
503 J, Fahey P, Muhammad T, Dempsey G, Hillman E, Ji N, Tolia A, Paninski L. Penalized matrix decomposition for
504 denoising, compression, and improved demixing of functional imaging data. arXiv:180706203 [q-bio, stat].
505 2018 Jul; <http://arxiv.org/abs/1807.06203>, arXiv: 1807.06203.
- 506 **Carandini M**, Shimaoka D, Rossi LF, Sato TK, Benucci A, Knöpfel T. Imaging the Awake Visual Cortex with a
507 Genetically Encoded Voltage Indicator. *Journal of Neuroscience*. 2015 Jan; 35(1):53–63. <https://www.jneurosci.org/content/35/1/53>, doi: 10.1523/JNEUROSCI.0594-14.2015.

- 510 **Falk T**, Mai D, Bensch R, Cicek O, Abdulkadir A, Marrakchi Y, Böhm A, Deubner J, Jäckel Z, Seiwald K, Dovzhenko
511 A, Tietz O, Dal Bosco C, Walsh S, Saltukoglu D, Tay TL, Prinz M, Palme K, Simons M, Diester I, et al. U-Net:
512 deep learning for cell counting, detection, and morphometry. *Nature Methods*. 2019 Jan; 16(1):67–70.
513 <https://www.nature.com/articles/s41592-018-0261-2>, doi: 10.1038/s41592-018-0261-2.
- 514 **Franke F**, Quian Quiroga R, Hierlemann A, Obermayer K. Bayes optimal template matching for spike sorting –
515 combining fisher discriminant analysis with optimal filtering. *Journal of Computational Neuroscience*. 2015;
516 38(3):439–459. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4420847/>, doi: 10.1007/s10827-015-0547-7.
- 517 **Giovannucci A**, Friedrich J, Gunn P, Kalfon J, Brown BL, Koay SA, Taxidis J, Najafi F, Gauthier JL, Zhou P, Khakh BS,
518 Tank DW, Chklovskii DB, Pnevmatikakis EA. CalmAn an open source tool for scalable calcium imaging data
519 analysis. *eLife*. 2019 Jan; 8:e38173. <https://doi.org/10.7554/eLife.38173>, doi: 10.7554/eLife.38173.
- 520 **Giovannucci A**, Friedrich J, Kaufman M, Churchland A, Chklovskii D, Paninski L, Pnevmatikakis EA. Onacid:
521 Online analysis of calcium imaging data in real time. In: *Advances in Neural Information Processing Systems*;
522 2017. p. 2381–2391.
- 523 **He K**, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer*
524 *vision*; 2017. p. 2961–2969.
- 525 **Herlihy M**, Shavit N. The art of multiprocessor programming. Morgan Kaufmann; 2011.
- 526 **Jung AB**, Wada K, Crall J, Tanaka S, Graving J, Yadav S, Banerjee J, Vecsei G, Kraft A, Borovec J, Vallentin C,
527 Zhydenko S, Pfeiffer K, Cook B, Fernández I, Chi-Hung W, Ayala-Acevedo A, Meudec R, Laporte M, others,
528 imgaug; 2019. <https://github.com/aleju/imgaug>.
- 529 **Kannan M**, Vasan G, Huang C, Haziza S, Li JZ, Inan H, Schnitzer MJ, Pieribone VA. Fast, in vivo voltage imaging
530 using a red fluorescent indicator. *Nature methods*. 2018; 15(12):1108.
- 531 **Knöpfel T**, Song C. Optical voltage imaging in neurons: moving from technology development to practical tool.
532 *Nature Reviews Neuroscience*. 2019; p. 1–9.
- 533 **Lin TY**, Dollár P, Girshick R, He K, Hariharan B, Belongie S. Feature pyramid networks for object detection. In:
534 *Proceedings of the IEEE conference on computer vision and pattern recognition*; 2017. p. 2117–2125.
- 535 **Marshall JD**, Li JZ, Zhang Y, Gong Y, St-Pierre F, Lin MZ, Schnitzer MJ. Cell-type specific optical recording of
536 membrane voltage dynamics in freely moving mice. *Cell*. 2016 Dec; 167(6):1650–1662.e15. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5382987/>, doi: 10.1016/j.cell.2016.11.021.
- 538 **Paige CC**, Saunders MA. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM*
539 *Transactions on Mathematical Software (TOMS)*. 1982; 8(1):43–71.
- 540 **Piatkevich KD**, Bensussen S, Tseng Ha, Shroff SN, Lopez-Huerta VG, Park D, Jung EE, Shemesh OA, Straub C,
541 Gritton HJ. Population imaging of neural activity in awake behaving mice in multiple brain regions. *bioRxiv*.
542 2019; p. 616094.
- 543 **Piatkevich KD**, Jung EE, Straub C, Linghu C, Park D, Suk HJ, Hochbaum DR, Goodwin D, Pnevmatikakis E, Pak N.
544 A robotic multidimensional directed evolution approach applied to fluorescent voltage reporters. *Nature*
545 *chemical biology*. 2018; 14(4):352.
- 546 **Pnevmatikakis EA**, Giovannucci A. NoRMCorre: An online algorithm for piecewise rigid motion correction of
547 calcium imaging data. *Journal of Neuroscience Methods*. 2017 Nov; 291:83–94. <http://www.sciencedirect.com/science/article/pii/S0165027017302753>, doi: 10.1016/j.jneumeth.2017.07.031.
- 549 **Pnevmatikakis E**, Soudry D, Gao Y, Machado TA, Merel J, Pfau D, Reardon T, Mu Y, Lacefield C, Yang W, Ahrens
550 M, Bruno R, Jessell TM, Peterka D, Yuste R, Paninski L. Simultaneous Denoising, Deconvolution, and Demixing
551 of Calcium Imaging Data. *Neuron*. 2016 Jan; 89(2):285–299. <http://www.sciencedirect.com/science/article/pii/S0896627315010843>, doi: 10.1016/j.neuron.2015.11.037.
- 553 **Roome CJ**, Kuhn B. Simultaneous dendritic voltage and calcium imaging and somatic recording from Purkinje
554 neurons in awake mice. *Nature communications*. 2018; 9(1):3388.
- 555 **Smith SL**, Häusser M. Parallel processing of visual space by neighboring neurons in mouse visual cortex. *Nature*
556 *Neuroscience*. 2010 Sep; 13(9):1144–1149. <https://www.nature.com/articles/nn.2620>, doi: 10.1038/nn.2620.

557 **Teeters JL**, Godfrey K, Young R, Dang C, Friedsam C, Wark B, Asari H, Peron S, Li N, Peyrache A, Denisov G, Siegle
558 JH, Olsen SR, Martin C, Chun M, Tripathy S, Blanche TJ, Harris K, Buzsáki G, Koch C, et al. Neurodata Without
559 Borders: Creating a Common Data Format for Neurophysiology. *Neuron*. 2015 Nov; 88(4):629–634. doi:
560 [10.1016/j.neuron.2015.10.025](https://doi.org/10.1016/j.neuron.2015.10.025).

561 **Walker T**. Cell magic wand tool. Cell Magic Wand Tool; 2014.

562 Description of Supplemental Movies

563 **Video 1**. Example of voltage imaging data on mouse neocortex data. Left: Raw data. Right: Local correlation
564 video.

565 Description of Supplemental Images

566 Algorithmic Details

567 In the following section we present the pseudocode for several of the routines introduced and used
568 by *VolPy*. Note that the pseudocode descriptions do not aim to present a complete picture and may
569 refer to other work for some of the steps.

Algorithm 1 SPIKEPURSUIT

Require: Input data matrix M , binary matrix for region of interest R , number of background principal components n_b , rest of parameters

```
1:  $R_c = \text{DILATION}(R, \text{params})$ 
2:  $x_{min}, x_{max}, y_{min}, y_{max} = \text{FINDBORDER}(R_c)$  ▷ Find border of context region
3:  $Y = M[:, x_{min} : x_{max}, y_{min} : y_{max}]$  ▷ Extract pixels in context region
4:  $R = R[x_{min} : x_{max}, y_{min} : y_{max}]$ 
5:  $\mathbf{p} = \text{FIND}(R == 1)$  ▷ Pixels for ROI
6:  $R_b = \text{DILATION}(R, \text{params})$ 
7:  $\mathbf{p}_b = \text{FIND}(R_b == 0)$  ▷ Pixels for background region
8:  $Y_h \leftarrow \text{HIGHPASSFILTER}(Y, \text{params})$ 
9: if  $\mathbf{w}$  is None then
10:    $\mathbf{t}_0 = \text{MEAN}(Y_h[:, \mathbf{p}])$  ▷ Mean of movie across all pixels in ROI
11: else
12:    $\mathbf{t}_0 = \text{WEIGHTEDAVERAGE}(Y_h, \mathbf{w})$  ▷ Weighted average of movie
13: end if
14:  $Y_b = Y_h[:, \mathbf{p}_b]$  ▷ Background signal
15:  $U_b = \text{SVD}(Y_b, n_b)$  ▷ Find top  $n_b$  background components
16:  $\beta = \text{RIDGREGRESSION}(U_b, \mathbf{t}, \lambda_b)$ 
17:  $\mathbf{t} \leftarrow \mathbf{t}_0 - U_b \beta$  ▷ Remove background components
18:  $\mathbf{t}, \mathbf{s}, \mathbf{r}, \mathbf{z} \leftarrow \text{DENOISESPIKES}(\mathbf{t}, \text{params})$  ▷ Compute optimized trace, spike times, reconstructed signal, temporal template  $\mathbf{t}, \mathbf{s}, \mathbf{r}, \mathbf{z}$ 
19: for  $k = 1, \dots, K$  do
20:    $\mathbf{w} = \text{RIDGREGRESSION}(Y_h, \mathbf{r}, \lambda_w)$  ▷ Calculate spatial filter
21:    $\mathbf{t} \leftarrow \text{WEIGHTEDAVERAGE}(Y_h, \mathbf{w})$ 
22:    $\beta \leftarrow \text{RIDGREGRESSION}(U_b, \mathbf{t}, \lambda_b)$ 
23:    $\mathbf{t} \leftarrow \mathbf{t} - U_b \beta$ 
24:    $\mathbf{t}, \mathbf{s}, \mathbf{r}, \mathbf{z} \leftarrow \text{DENOISESPIKES}(\mathbf{t}, \text{params})$ 
25: end for
26: return  $\mathbf{t}, \mathbf{s}$ 
```

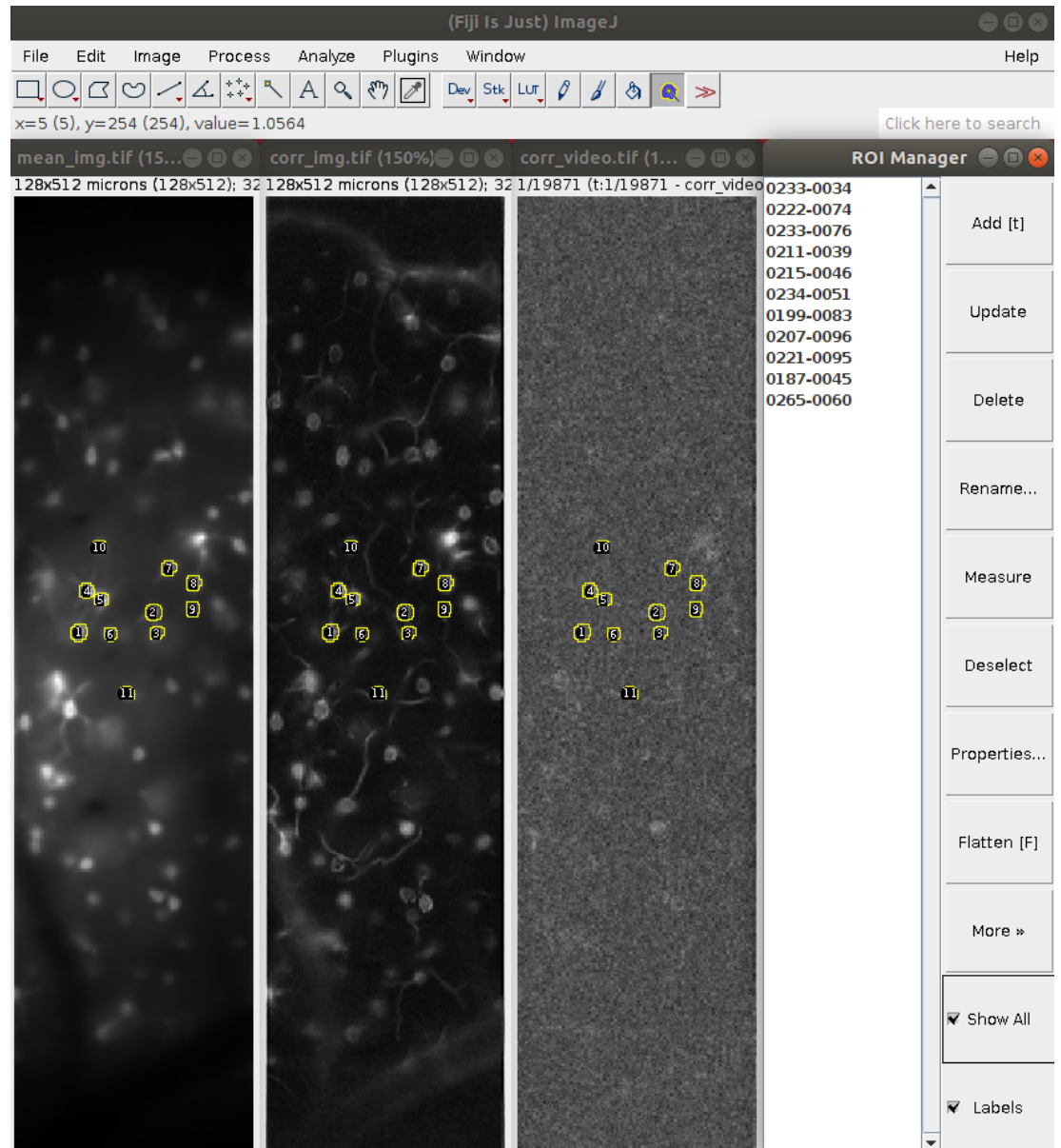


Figure 8. Create manual annotations of voltage imaging datasets with ImageJ. We selected neurons based on mean image (left), correlation image (mid-left) and local correlation movie (mid-right). Two annotators marked the contours of neurons using ImageJ Cell Magic Wand tool plugin and saved selections in ROI manager (right).

Algorithm 2 DENOISESPIKES

Require: Temporal trace \mathbf{t} , window length τ , max number of spikes picked n_{max}

```

1:  $\mathbf{t} \leftarrow \text{HIGHPASSFILTER}(\mathbf{t})$ 
2:  $\mathbf{p}, \mathbf{s}, \mathbf{q} = \text{LOCALMAXIMUM}(\mathbf{t})$            ▷ Compute peak heights  $\mathbf{p}$ , spike time  $\mathbf{s}$  and spike train  $\mathbf{q}$ 
3:  $h = \text{GETTHRESH}(\mathbf{p}, n_{max})$                  ▷ Only detect large spiking events
4:  $\mathbf{s} \leftarrow \mathbf{s}[\mathbf{p} > h]$ 
5:  $\mathbf{z} = \frac{1}{n(s)} \sum_{i=1}^{n(s)} \mathbf{t}[\mathbf{s}[i] - \tau : \mathbf{s}[i] + \tau]$            ▷ Compute temporal template
6:  $\mathbf{t} \leftarrow \text{WHITENEDMATCHEDFILTER}(\mathbf{t}, \mathbf{s}, \tau)$ 
7:  $\mathbf{p}, \mathbf{s}, \mathbf{q} \leftarrow \text{LOCALMAXIMUM}(\mathbf{t})$ 
8:  $h \leftarrow \text{GETTHRESH}(\mathbf{p}, \infty)$            ▷ Detect all spikes that can be found
9:  $\mathbf{s} \leftarrow \mathbf{s}[\mathbf{p} > h]$ 
10:  $\mathbf{r} = \text{CONVOLVE}(\mathbf{q}, \mathbf{z})$                    ▷ Compute reconstructed signal  $\mathbf{r}$ 
11: return  $\mathbf{t}, \mathbf{s}, \mathbf{r}, \mathbf{z}$ 

```

Algorithm 3 GETTHRESH

Require: peak heights \mathbf{p} , max number of spikes picked n_{max} , norm number p_{norm} , min number of spikes detected n_{min} , rest of parameters

```

1:  $\mathbf{x} = \text{Linspace}(\text{MIN}(\mathbf{p}), \text{MAX}(\mathbf{p}), \text{params})$    ▷ Evenly spaced samples between min and max of peak heights
2:  $\mathbf{f} = \text{KDE}(\mathbf{p}, \mathbf{x})$                            ▷ Estimate distribution of peak heights
3:  $\mu = \text{MEDIAN}(\mathbf{p})$ 
4:  $j = \text{FIND}(\mathbf{x}[i] < \mu, \mathbf{x}[i + 1] > \mu)$ 
5:  $\mathbf{f}_{\text{noise}}[1 : j] = \mathbf{f}[1 : j]$ 
6:  $\mathbf{f}_{\text{noise}}[j + 1 : \text{end}] = \mathbf{f}[j : 1]$            ▷ Approximate noise distribution
7:  $\mathbf{F} = \text{CUMSUM}(\mathbf{f})$                                ▷ Cumulative distribution
8:  $\mathbf{F}_{\text{noise}} = \text{CUMSUM}(\mathbf{f}_{\text{noise}})$ 
9:  $\mathbf{F} = \mathbf{F}[\text{end}] - \mathbf{F}$ 
10:  $\mathbf{F}_{\text{noise}} = \mathbf{F}_{\text{noise}}[\text{end}] - \mathbf{F}_{\text{noise}}$ 
11:  $\mathbf{g} = \mathbf{F}^{p_{norm}} - \mathbf{F}_{\text{noise}}^{p_{norm}}$ 
12:  $k = \text{ARGMAX}(\mathbf{g})$                                ▷ Adaptive thresholding
13:  $h = \mathbf{x}[k]$ 
14: if  $\text{SUM}(\mathbf{p} > h) < n_{min}$  then                 ▷ Too few spikes are found, adjust to  $n_{min}$ 
15:    $h = \text{PERCENTILE}(\mathbf{p}, 100 * (1 - n_{min}/\text{LEN}(\mathbf{p})))$ 
16: else if  $\text{SUM}(\mathbf{p} > h) > n_{max}$  then           ▷ Too many spikes are found, adjust to  $n_{max}$ 
17:    $h = \text{PERCENTILE}(\mathbf{p}, 100 * (1 - n_{max}/\text{LEN}(\mathbf{p})))$ 
18: end if
19: return  $h$ 

```

Algorithm 4 WHITENEDMATCHEDFILTER

Require: Temporal trace \mathbf{t} , spike train \mathbf{q} , window length τ

```

1:  $\mathbf{q}' = \text{CONVOLVE}(\mathbf{q}, \text{ONES}(2\tau + 1))$ 
2:  $\mathbf{t}_{\text{noise}} = \mathbf{t}[\mathbf{q}' < 0.5]$ 
3:  $\mathbf{s}_n = \text{SQRT}(\text{WELCH}(\mathbf{t}_{\text{noise}}))$            ▷  $\mathbf{s}_n$  is scaling factor in frequency domain
4:  $\mathbf{t}' = \text{IFFT}(\text{FFT}(\mathbf{t})/\mathbf{s}_n)$ 
5:  $\mathbf{z} = \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{t}'[\mathbf{s}[i] - \tau : \mathbf{s}[i] + \tau]$            ▷ Compute temporal template
6:  $\mathbf{t}' \leftarrow \text{CONVOLVE}(\mathbf{t}', \text{FLIP}(\mathbf{z}))$            ▷ Template matching
7: return  $\mathbf{t}'$ 

```
