Supplementary Note 5: “User manual for EXTRACT”

§1. Overview

This user manual aims to provide the basic principles for *how to efficiently and effectively use* EXTRACT, our tractable and robust automated cell extraction tool for Ca^{2+} imaging presented in the main text. For the purpose of this manual, we assume no detailed understanding of robust regression or other technical details presented in the main text. Instead, we will consider EXTRACT as a black box algorithm, controlled by a set of parameters, which takes as an input the Ca^{2+} imaging movie and outputs the Ca^{2+} activity traces of the identified cells in the movie and their spatial profiles. The inner workings of EXTRACT are mainly relevant for those who wish to develop similar algorithms; whereas a phenomenological understanding of EXTRACT’s modules has been enough for many users to date to effectively and efficiently utilize EXTRACT. Below is a summary of the content for each section.

§2 introduces an example code (**Tutorial 1**) to quickly start using EXTRACT. Here, on a simulated Ca^{2+} imaging movie, we showcase some of the most helpful features of EXTRACT, *e.g.*, how to watch the movie before processing, how to set some crucial hyperparameters, and how to perform post-extraction quality controls. **§3** provides an overview of EXTRACT’s modules in a user-friendly way and a full list of key hyperparameters that users need to be familiar with. Moreover, this section also contains a short tutorial regarding parallel computing and graphics processing unit utilization (**Tutorial 2**). **§4-6** introduce new tutorials (**Tutorials 3-6**) to discuss the effect of key hyperparameters of each module in detail. Specifically, we propose a 2-step optimization procedure for EXTRACT’s hyperparameters, in which the modular structure is exploited to optimize the hyperparameters in a modular manner. The tutorials illustrate how to systematically and easily perform this procedure to quickly fine-tune the hyperparameters and include additional tips and tricks for processing challenging, particularly low SNR, movies with EXTRACT. **§7** concludes with final remarks. While working on the tutorials, we recommend keeping **Figure S9** nearby, which provides a visual summary of this user manual.

§2. Code for a quick start

Throughout this user manual, whenever we refer to a variable or a code piece in MATLAB, we denote them with a distinct font. For example, the variable representing the input movie is referred to as `M`. With that, in essence, EXTRACT can be compactly run with a single line of code:

```
output = extractor(M,config);
```

In this section, we provide the basic information needed to quickly start using EXTRACT. Here, we first introduce and discuss EXTRACT's inputs and outputs, and then provide a code walkthrough on a simulated Ca^{2+} imaging movie (Tutorial 1).

2.1. EXTRACT inputs

The EXTRACT algorithm has two inputs. The first input, `M,` corresponds to the movie being processed. It can either be a three-dimensional movie matrix, with the time component being the third dimension, or a command (see below) that points to the location of the movie. The second input is a structure array that contains the configuration parameters for EXTRACT to use. This structure array can be passed as an empty array, in which case EXTRACT uses the default parameters.

Inputting the Ca^{2+} imaging movie into EXTRACT

As noted above, the input movie, `M,` can be provided as a three-dimensional matrix. For the second option of inputting the movie as a command, we will discuss two options. For these discussions, we will use an example h5 dataset, whose file name is `example.h5` and the movie is contained inside the dataset `/mov`.

The first option is to input the movie in a single string format. If it is inputted as a string, the string must be in the format `filepath:dataset`, where the dataset name should not contain a colon (:), though having it in the filepath is fine. For the example movie, the string would be:

```
M = 'example.h5:/mov';
```

As an alternative option, especially if the dataset name contains a colon (:), then `M` can be inputted as a cell such that the first component would include the path, whereas the second component would include the dataset name. In our example, this would be:

```
M = {}; M{1} = 'example.h5'; M{2} = '/mov';
```

If the movie is not within the direct path, but is rather inside a folder, say `C:\movie_folder`, then the file path in both cases would be updated to `'C:\movie_folder\example.h5'` for both options, *e.g.*, `M = 'C:\movie_folder\example.h5:/mov';`

Structure array containing EXTRACT's configurations

The configurations are inputted into the EXTRACT in the form of a structure array in MATLAB. We often refer to this array as `config`. We discuss the most important configuration parameters below, which are also illustrated in **Figure S9** in the main text.

- `avg_cell_radius`: An order of magnitude estimate for an average cell's radius in the movie. This parameter sets the scale on which EXTRACT will operate. For example, this parameter is used for normalization during the application of the high-pass filtering or for several quality metrics (See **§3** for details). Therefore, though setting it to a significantly larger or lower value will impair performance, EXTRACT is fairly robust to the variations in this parameter. A simple and effective way to estimate this parameter would be to plot the maximum projections of the Ca^{2+} imaging movie (**§4**). Default: 6 pixels.
- `num_partitions_x` and `num_partitions_y`: User specified number of spatial partitions in x and y dimensions of the movie, often selected for large-scale Ca^{2+} imaging movies to minimize the RAM usage. If the user does not decide on particular values, EXTRACT performs partitioning automatically (once any side length of the movie exceeds 512 pixels). Practically, we recommend picking the partition sizes such that each spatial chunk contains around thousands of cells, or each partition's file size is around a quarter of the full RAM memory, whichever is smaller. EXTRACT performs the stitching of the results automatically once all partitions are processed. Default: none.
- `cellfind_max_steps`: The maximum number of cells that can be initialized during the cell finding of EXTRACT in each spatial partition. The default is 1000, which may need to be adjusted depending on the particular dataset. Specifically, it can be increased so that EXTRACT finds more cells per partition or decreased to prevent increased false positives.
- `max_iter`: The number of cell refinement iterations (See **Figure 2**). Often, 5-10 iterations are enough to cull out false-positives, whereas too many iterations can lead to unnecessarily increased runtimes (**§5** and **§6**). Default: 6 iterations.
- `cellfind_min_snr` and `thresholds.T_min_snr`: Two different definitions of signal-to-noise ratio for cells' estimated Ca^{2+} activity traces. The former is used only during cell finding, whereas the letter is used for both cell finding and cell refinement. A cell's estimated signal quality should be higher than these values. For `cellfind_min_snr`, a typical value of 1 is reasonable for most datasets. Increase this value to decrease false positives during cell finding with no effect on the cell refinement procedure. For

thresholds.T_min_snr, a default value is 7 (and lowest acceptable value is 3). This parameter is often very effective to cull away false-positives during cell refinement (**§6**).

- trace_output_option: This parameter decides which type of solver EXTRACT will use to produce the final estimated Ca^{2+} activity traces (**§7**). Viable options include: 'nonneg', 'baseline_adjusted', 'no_constraint', 'nonnegative_least_squares', 'least_squares'.

These parameters are perhaps the most immediately relevant ones for an end user to quickly start using EXTRACT, which is what Tutorial 1 focuses on. There are several other parameters relevant to the users, including the robustness parameter $\boldsymbol{\kappa}$, which is adaptively optimized and thus not necessarily needs to be tuned by a first-time user. We will discuss these in the following sections of this user manual with additional tutorials.

2.2. EXTRACT outputs

EXTRACT outputs are compactly saved inside a structure array, which has four major fields: 'config', 'info', 'spatial_weights', and 'temporal_weights'. Config contains the configuration parameters used by EXTRACT, whereas info includes some cell extraction statistics that are not directly relevant for a first time user. The remaining fields are three and two-dimensional matrices, respectively:

- spatial_weights contains the spatial profiles and locations of extracted cells and has the shape [movie_height x movie_width x number_of_cells_found].
- temporal_weights contains the inferred Ca^{2+} activity traces for each cell and has the shape [number_of_movie_frames x number_of_cells_found].

In the main text and other supplementary notes, (transformed versions of) these matrices have often been referred to with the notations $\boldsymbol{S}$ and $\boldsymbol{T}$, respectively.

2.3. Tutorial 1: A quick start to EXTRACT

In this first tutorial, we will work on a simulated Ca^{2+} imaging movie. Our main goal is to get the first time users started with EXTRACT, though future tutorials will focus on further important hyperparameters. The walkthrough below is provided in the Github as a live script, which also contains the necessary data files for running the tutorial.

Running EXTRACT

Let us start with a simple cell extraction procedure. The code below uses the most important parameters we discussed above and performs a basic cell extraction:

```
load('example.mat');
config=[];
config = get_defaults(config);
config.avg_cell_radius=7;
config.trace_output_option='no_constraint';
config.num_partitions_x=1;
config.num_partitions_y=1;
config.use_gpu=0;
config.max_iter = 10;
config.cellfind_min_snr=0;
config.thresholds.T_min_snr=10;
output=extractor(M,config);
```

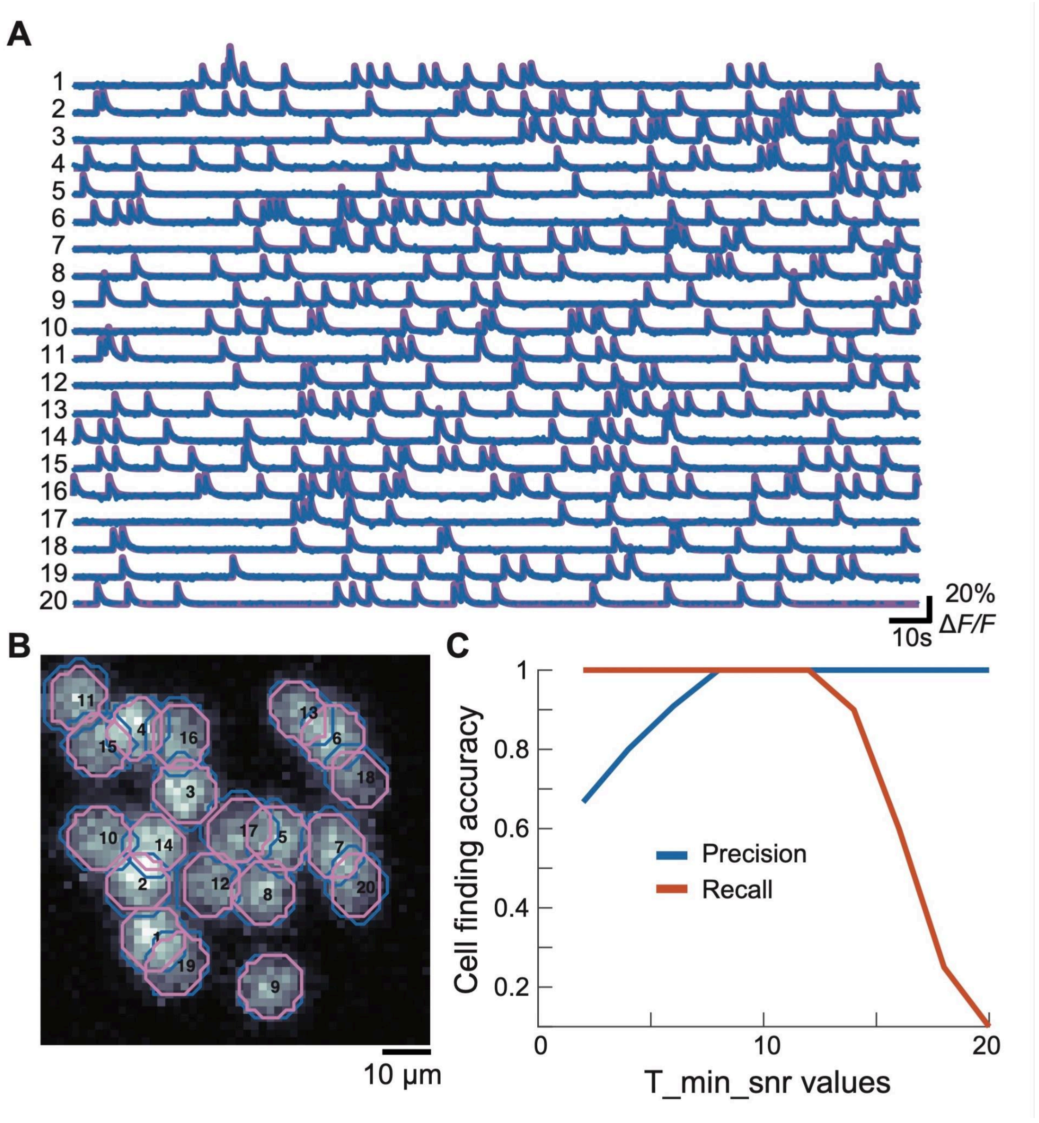
Fortunately, since this is a simulation movie, we can quantify the quality of cell extraction. Within the 'example.mat' file provided as part of the tutorial, we have cells' ground truth Ca^{2+} activities, T_ground, and the spatial profiles, S_ground. To compare the cell extraction results, we first need to compute the same quantities for the extracted cells:

```
T_ex = output.temporal_weights';
S_ex = full(output.spatial_weights);
[h,w,k]=size(S_ex);
S_ex=reshape(S_ex,h*w,k);
```

Note that there are 20 ground truth cells in the provided simulated Ca^{2+} imaging movie. The code above also outputs 20 cells. But, do they match? Did we find all the cells, or did we find some duplicates or spurious ones? To check this, we can use a spatial correlation based function (See **Methods**) that matches the extracted cells to the ground truth:

```
idx_match = match_sets(S_ex, S_ground,0.8);
```

As can be seen from the accompanying tutorial code, EXTRACT indeed finds all the cells.



Appendix Figure 11. Cell extraction results for a simulated Ca2+ imaging movie.
The extracted cells’ Ca^{2+} activity traces (**A**) and spatial profiles (**B**) superposed on top of the ground truth cells. **C** A simple parameter sweep experiment reveals the importance of correctly, but not necessarily finely, tuning the expected SNR value for the cells.

Visualizing cell extraction outputs

Using the following code, we can visualize the cell extraction outputs, which is shown in **Appendix Figure 11 A, B**:

```
color_extract = [0 0.4470 0.7410];
color_gt      = [144 103 167]./255;
color_l2 = [1,0.5,1];
```

```
plot_stacked_traces_double(T_ground(idx_match(2,:),:),...
T_ex(idx_match(1,:),:),1,{color_gt,color_extract},[],[],{5,3});
ims_ex = reshape(S_ex,h,w,[]);
ims_g = reshape(S_ground,h,w,[]);
max_im = max(M,[],3);
plot_simulated_cellmap(ims_g,...
max_im,ims_ex(:,:,idx_match(1,:)),color_extract,color_l2)
```

Importance of SNR parameter

Having shown the results of an example, well optimized, cell extraction, we next show the importance of correctly, not necessarily finely, choosing one of the most important parameters: `thresholds.T_min_snr`. This parameter needs to be adjusted to allow low SNR cells to be picked up or extremely low SNR garbage to be discarded. Using the code below, we can test how very low or very high values of this parameter affect the cell extraction quality.

```
thr_all = linspace(2,20,10);
precision = zeros(1,size(thr_all,2));
recall    = zeros(1,size(thr_all,2));
for i = 1:size(thr_all,2)
   config.thresholds.T_min_snr=thr_all(i);
   config.verbose = 0;
   output=extractor(M,config);
   S_ex=reshape(full(output.spatial_weights),h*w,[]);
   idx_match = match_sets(S_ex, S_ground,0.8);
   precision(i) = size(idx_match,2)/size(S_ex,2);
   recall(i) = size(idx_match,2)/size(S_ground,2);
   fprintf('%d finished.\n',i);
end
```

The results are shown in **Appendix Figure 11C.** As can be seen from the figure, choosing a very low `T_min_snr` leads to high recall, but can lead to diminished precision. This is because garbage/duplicate cells cannot be discarded despite having very low SNR values. On the other side, increasing `T_min_snr` too much can lead to several actual cells being discarded, leading to decreased recall values. Fortunately, the exact ground truth value does not need to be matched, as here flexible choices, *i.e.*, those between 8-12, all provided perfect cell extraction results. Empirically, we find that `T_min_snr` may need to be adjusted all the way down to ~3 for low SNR movies, whereas a value of 7-10 would be preferable to more effectively discard garbage for moderate to high SNR movies.

§3. Introduction to EXTRACT: Understanding the modules and hyperparameters

In this section, we provide a comprehensive list of the hyperparameters that an EXTRACT end user should be broadly familiar with. While the list is arguably long, most hyperparameters are self explanatory (*e.g.*, `skip_dff` parameter that controls whether the $\Delta F/F$ transform should be applied during the preprocessing module). In Figure **2** of the main text, we provided a broad overview of EXTRACT's internal modules: preprocessing, cell finding, cell refinement, and final robust regression. Fortunately, we designed almost all of the hyperparameters by keeping this modularity in mind, which is why we will introduce them below in batches. We note that some parameters were already listed in **§2**, but for completeness we list them here again.

The default values for all hyperparameters can be found inside the function `get_defaults.m`. Similar to this user manual, the parameters there are also listed in batches depending on the module they control (also see **Figure S9**). Whenever in doubt, we recommend checking the exact spelling of the hyperparameters with this script. We advise that users do not change this script, the defaults are picked to be most appropriate for a general set of movies. Finally, please note that this script will not overwrite user provided configurations.

3.1. General control hyperparameters

There are 16 general control parameters immediately relevant for EXTRACT users, which we list below:

- `avg_cell_radius`: An order of magnitude estimate for an average cell's radius in the movie. This parameter sets the scale on which EXTRACT will operate. For example, this parameter is used for normalization during the application of the high-pass filtering or for several quality metrics (See, *e.g.*, the preprocessing module below). Therefore, though setting it to a significantly larger or lower value will impair performance, EXTRACT is fairly robust to the variations in this parameter. A simple and effective way to estimate this parameter would be to plot the maximum projections of the Ca^{2+} imaging movie (**§4**). Default: 6 pixels.

- `downsample_time_by`: For cell finding and refinement purposes only, EXTRACT can use a temporally downsampled version of the movie. This speeds up the cell extraction process and tends to increase the cell finding quality. In general, we recommend setting this parameter such that the downsampled movie has approximately few Hz sampling rate. For example, if the original movie is 20Hz, a reasonable value would be 5. Default: 1.
- `dendrite_aware`: If the movie includes dendrites and/or other non-conventional regions of interest, set this to 1. When this parameter is on, EXTRACT skips several checks regarding the shape and the size of the estimated cell profiles (See Section 3.4 below). Default: 0.
- `Remove_duplicate_cells`, `T_dub_thresh` and `S_corr_thresh`: If the movie is partitioned, it is possible that the same cells are found in two separate slightly overlapping partitions. These parameters control the removal of duplicate cells in these overlap regions. `Remove_duplicate_cells` is a control flag, 'true' by default. The other two parameters set the correlation thresholds for the overlapping cells' activities and spatial profiles. When both are surpassed between two cells, these cells are considered duplicates of each other and the cell with the higher area is retained.
- `use_gpu`: If you have a graphics processing unit (GPU), set this parameter to 1.
- `parallel_cpu`: Boolean flag for parallel processing movie partitions over the CPU cores. EXTRACT either parallelizes across the CPU cores or uses the GPU. Often, GPUs are faster. Default: 0.
- `multi_gpu`: Same as before, but for multiple GPUs. Default: 0.
- `num_workers`: This parameter is relevant when either `parallel_cpu` or `multi_gpu` is on. When using parallel processing, this parameter can be used to set the desired number of CPU cores or GPUs. Default: # of available cores/GPUs to Matlab - 1.
- `use_sparse_arrays`: If this is set to 1, the cells' spatial profiles stored inside the structure array `output.spatial_weights` will be stored as sparse matrices, instead of single format. Turn this flag on when processing large movies. Default: 0.

- `compact_output`: If set to `true`, then the output will not include secondary, but sometimes useful, cell extraction information. Turning this flag on often reduces the size of the output file substantially. Default: 0.
- `hyperparameter_tuning_flag`: If this flag is on, the cell extraction process stops after cell finding and a single cell refinement step. See **§6** and **Tutorial 5** for additional details for how to use this. Default: 0.
- `verbose`: Controls the logs that are outputted to the console. 0: No output at all. 1: EXTRACT emits basic details during the signal extraction process within each spatial patch. 2: Similar to 1, but EXTRACT provides a rather detailed summary (default when serial processing). 3: Only the most basic general information and no particular information from within the spatial patches (default when parallel processing).
- `num_partitions_x` and `num_partitions_y`: User specified number of spatial partitions in x and y dimensions of the movie, often selected for large-scale Ca^{2+} imaging movies to minimize the RAM usage. If the user does not decide on particular values, EXTRACT performs partitioning automatically (once any side length of the movie exceeds 512 pixels). Practically, we recommend picking the partition sizes such that each spatial chunk contains around thousands of cells, or each partition's file size is around a quarter of the full RAM memory, whichever is smaller. EXTRACT performs the stitching of the results automatically once all partitions are processed. Default: none.

3.2. Preprocessing module

Controlling the preprocessing module is rather straightforward, which involves four relevant parameters listed below:

- `preprocess`: Turns the full preprocessing module on or off. Default: 1.
- `skip_dff`: To perform the $\Delta F/F$ transformation of the Ca^{2+} imaging movies in a numerically stable manner, EXTRACT subtracts the static baseline of the movie but does not perform the division explicitly. Instead, we store the static baseline values in a separate array, which is used to normalize the final Ca^{2+} activity traces. Setting this flag to 0 skips the ΔF subtraction and assigns $F = 1$ to all pixels. Default: 0.
- `F_per_pixel`: When the preprocessing module is skipped, this parameter (a matrix) defines the static baseline values for each pixel. Default: none.

- `spatial_highpass_cutoff`: This parameter defines the cutoff values of the spatial high-pass filter applied to the Ca^{2+} imaging movie. The exact cutoff value is proportional to `avg_cell_radius/spatial_highpass_cutoff`, normalized with respect to the average cell radius in the movie. A larger value of this parameter leads to mild filtering, with no filtering being performed when it is set to infinity. Default: 5.

3.3. Cell finding module

The following are the parameters associated with the cell finding module:

- `spatial_lowpass_cutoff`: This parameter defines the cutoff values of the spatial low-pass filter applied to the Ca^{2+} imaging movie for the cell finding purposes only. The exact cutoff value is proportional to `avg_cell_radius*spatial_lowpass_cutoff`, normalized with respect to the average cell radius in the movie. A larger value of this parameter leads to mild filtering, with no filtering being performed when it is set to infinity. Default: 2.
- `cellfind_min_snr` and `thresholds.T_min_snr`: Two different definitions of signal-to-noise ratio for cells' estimated Ca^{2+} activity traces. The former is used only during cell finding, whereas the letter is used for both cell finding and cell refinement. A cell's estimated signal quality should be higher than these values. For `cellfind_min_snr`, a typical value of 1 is reasonable for most datasets. Increase this value to decrease false positives during cell finding with no effect on the cell refinement procedure. For `thresholds.T_min_snr`, a default value is 7 (and lowest acceptable value is 3). This parameter is often very effective to cull away false-positives during cell refinement (**§6**).
- `cellfind_max_steps`: The maximum number of cells that can be initialized during the cell finding of EXTRACT in each spatial partition. The default is 1000, which may need to be adjusted depending on the particular dataset. Specifically, it can be increased so that EXTRACT finds more cells per partition or decreased to prevent increased false positives.
- `cellfind_kappa_std_ratio` and `cellfind_adaptive_kappa`: The robustness parameter, κ, utilized during the cell finding module is initially set to the estimated noise in the movie times `cellfind_kappa_std_ratio`. Large values of κ correspond to assuming low contamination by non-Gaussian noise, setting $\kappa = \infty$ leads to least squares estimates. When `cellfind_adaptive_kappa` is set to one, this parameter is adaptively estimated from the Ca^{2+} imaging movie. Note that these parameters affect only

the cell finding module, there are parameters for the cell refinement and final robust regression modules. Default: $\kappa = 0.7$, the adaptive estimation is off by default.

- `init_with_gaussian`: If true, then during cell finding, each cell is initialized with a gaussian shape prior to robust estimation of their spatial profiles. If false, then initialization is done by computing a correlation image. If cells are of different shape and size, keep this off. Default: 0.
- `avg_yield_threshold`: During cell finding, If the yield in the last few components falls below this threshold, the cell finding module is terminated. Default: 1/10.
- `visualize_cellfinding`: To visualize the cell finding process, turn this flag on. This parameter is particularly useful for optimizing hyperparameters (**§5**), but increases the runtimes significantly. Default: 0.
- `S_init`: Optionally, one can provide cells’ profiles with this parameter and EXTRACT will use these as the initial set of cells, skipping the cell finding module. Default: none.

3.4. Cell refinement module

The main purpose of the cell refinement module is to correctly estimate the true cells’ spatial profiles and delete the spurious and/or duplicate cells. Here, we first provide the list of general hyperparameters of the cell refinement module:

- `kappa_std_ratio`: The robustness parameter, κ, utilized during the cell refinement module is initially set to the estimated noise in the movie times `kappa_std_ratio`. Large values of κ correspond to assuming low contamination by non-Gaussian noise, setting $\kappa = \infty$ leads to least squares estimates. Default: 0.7.
- `adaptive_kappa`: This parameter can take three distinct values. 0: No adaptive estimation of the robustness parameter during cell refinement or final robust regression. 1: Adaptive estimation is performed during the final robust regression only. 2: Adaptive estimation is performed during both cell refinement and final robust regression. Default: 1.
- `max_iter`: The number of cell refinement iterations. Default : 6.
- `l1_penalty_factor`: A scalar that determines the strength of L_1 regularization penalty to be applied when estimating the temporal components. The penalty is applied only to cells that overlap in space and whose temporal components are correlated. Use larger values if spurious cells are observed in the vicinity of high SNR cells, particularly helpful for culling out duplicate cells. Default: 0.

Inside the cell refinement module, EXTRACT computes several quality metrics and discards cell candidates that fail to score sufficiently. The thresholds on these quality metrics are determined by the outside user inside the structure array: `thresholds`. The following parameters controlling these quality metrics are most relevant for EXTRACT users:

- `T_min_snr`: Cells with lower SNR value than this parameter will be eliminated. This parameter is shared with the cell finding module, see above. Default: 7.
- `size_lower_limit` and `size_upper_limit`: These factors are multiplied with the average cell area (determined from `avg_cell_radius`) and any cell with an area outside of these will be eliminated during the cell refinement. Defaults: 0.1 and 10.
- `spatial_corrupt_thresh`: Spatial corruption indices, quantifying the non-uniformity in cells’ spatial profiles, are calculated at each step of the alternating minimization routine. Images that have an index higher than these are eliminated. Default: 1.5.
- `eccent_thresh`: Cells with eccentricity higher than this will be eliminated. One can intuitively think of high eccentricity as cells becoming more elliptic. This parameter eliminates blood vessels in most one-photon Ca^{2+} imaging movies. Default: 6.
- `T_dup_corr_thresh` and `S_dup_corr_thresh`: Through alternating estimation, cells that have higher trace correlation than `T_dup_corr_thresh` and higher image correlation than `S_dup_corr_thresh` are eliminated. Defaults: 0.95 and 0.8.
- `low_ST_index_thresh`: EXTRACT requires cells’ Ca^{2+} activity traces to explain a portion of the fluorescence signal in the pixels encapsulated by the cells. This parameter culls out the identified components, in which the inferred Ca^{2+} activity traces do not explain the pixel activities well. In practice, this parameter may be set to ‘0.01’ and is very helpful for removing garbage at the expense of culling out low SNR cells. In most movies, this parameter is not needed. See **Spatiotemporal match metrics** Section in the **Methods** for additional details. Default: -1 (no checks).

3.5. Final robust regression module

The final robust regression module performs, essentially, one final robust regression to estimate cells’ Ca^{2+} activity traces. Below are the relevant parameters:

- `Regression_only`: When this flag is on, EXTRACT becomes a Ca^{2+} activity trace estimator. Depending on the `trace_output_option`, EXTRACT can be used to perform robust regression, least-squares regression, and/or non-negative least-squares regression. Cell finding and refinement modules will be skipped. `S_init` should be initialized with the cells' spatial profiles, over which the regression will be performed. Default: 0.

- `trace_output_option`: This parameter decides which type of solver EXTRACT will use to produce the final estimated Ca^{2+} activity traces (**§7**). Viable options include: `'nonneg'`, `'baseline_adjusted'`, `'no_constraint'`, `'nonnegative_least_squares'`, `'least_squares'`. Default: `'baseline_adjusted'`.
- `trace_quantile`: When `'baseline_adjusted'` solver is used, EXTRACT uses an adaptive process to compute the cells' activity baselines. This parameter provides the percentile, which will be used as the cells' activity baselines. Default: 0.25.

3.6. Parameters for speed optimization

EXTRACT default parameters are optimized for accuracy. In many cases, additional speed improvements can be achieved by simple tricks that do not affect the cell extraction quality. Below are the relevant parameters:

- `cellfind_max_iter`: Number of alternating estimations during the one-by-one cell finding process. Default is 10, but decreasing down to 3 often leads to little to no performance decreases.
- `max_iter_S` and `max_iter_T`: Maximum number of iterations for S and T estimation steps during the cell finding. Both are 100 by default, but decreasing down to 20 is usually perfectly acceptable, since the robust regression converges fast.
- `max_iter_T_final`: This parameter is used to set the maximum number of ADMM iterations (See **Methods**) at the final regression. Most cases do not require any adjustments, be careful when decreasing to gain speed as final activity traces are obtained through this step. If this number is too small, the cells' Ca^{2+} activity traces may appear distorted at their maximum values due to insufficient estimation iterations. Default: 100.

3.7. Tutorial 2: Parallelizing cell extraction across multiple CPU cores

In this brief tutorial, we show the importance of running EXTRACT on multiple GPUs and/or CPU cores. The computer we ran this tutorial on, whose results will be shown as part of the live script in our Github[1], has two NVIDIA Geforce RTX 3090s and an Intel i9-10980XE CPU with 18 physical cores. To start with, we first simulate an example two-photon Ca^{2+} imaging movie using the code:

```
if ~isfile('Example_2p_movie.h5')
   [opts_2p] = get_2p_defaults();
   opts_2p.ns = 500;
   rng(1)
   create_2p_movie(opts_2p,'Example_2p_movie');
End
```

This code simulates a two-photon Ca^{2+} imaging movie with a 500×500 pixels^2 field-of-view. In the upcoming tutorials, we will show how to optimize EXTRACT to achieve optimal cell extraction. For this tutorial, our goal is to showcase the speed of EXTRACT under various conditions. Thus, we will use the already optimized parameters:

```
M = 'Example_2p_movie.h5:/mov';
config = get_defaults([]);
config.adaptive_kappa = 2;
config.spatial_highpass_cutoff = inf;
config.downsample_time_by = 4;
config.num_partitions_x = 4;
config.num_partitions_y = 4;
config.thresholds.T_dup_corr_thresh = 0.99;
config.thresholds.spatial_corrupt_thresh = 0.1;
config.thresholds.eccent_thresh = 3;
config.max_iter = 10;
config.thresholds.size_upper_limit = 3;
config.cellfind_min_snr = 0;
config.verbose = 0;
config.trace_output_option = 'no_constraint';
```

To start with, we run this movie on a single CPU core using the code:

```
config.use_gpu = 0;
output = extractor(M,config);
```

This code, on our computer, took around 15 minutes to complete. In a similar manner, by using the parameters `use_gpu`, `parallel_cpu`, `multi_gpu`, and `num_workers`, we ran the same movie on 16 CPU cores (7 mins), a single GPU (5 mins), and two GPUs (3.8 mins).

Overall, GPU performance was best, which was followed by parallelization across CPU cores. Please note that the runtimes are mainly due to some fixed costs (also shown in **Figure 5K**). These extra costs are negligible for large scale movies (as in **Figure 4**), but prevent major speedups with multiple GPUs for short movies like the one we considered here.

§4. Prerequisites: Pre-processing of movies

In this section, we will mainly be discussing the **Tutorial 3**, which provides a brief introduction to EXTRACT’s preprocessing module. Here, we discuss loading (and watching) Ca^{2+} imaging movies from the h5 files, fast spatial high-pass filtering using EXTRACT’s GPU accelerated bandpass filter, and other supporting functions for the preprocessing pipeline.

As before, this tutorial is provided in our Github repository[1]. We start by simulating a one-photon Ca^{2+} imaging movie using the following code:

```
if ~isfile('Example_1p_movie.h5')
   [opts_2p,opts_back] = get_1p_defaults();
   opts_2p.ns = 100;
   opts_2p.n_cell = 100;
   opts_back.ns = 100;
   opts_back.n_cell = 20;
```

```
    opts_back.cell_radius = [20,40];
    rng(1)
    create_1p_movie(opts_2p,opts_back,'Example_1p_movie');
end
```

Throughout this tutorial, we use this simulated movie to introduce the most important aspects of the preprocessing module.

4.1. Reading and watching the Ca^{2+} imaging movies

For EXTRACT, we recommend to use the '.h5' format for storing and reading Ca^{2+} imaging movies. Each '.h5' file is accompanied by at least one dataset, which are conventionally named with a backslash, *e.g.*, '`/example_dataset`'. In our example, the dataset that contains the movie is called '`/mov`'. If unsure, one can always check the dataset name and size using the following command:

```
info = h5info('Example_1p_movie.h5');
```

The movie can be read using MATLAB's native `h5read` function. The `info` array above tells us that the movie matrix has the shape $100 \times 100 \times 5000$. We can read the first 1000 frames using the following code:

```
M = h5read('Example_1p_movie.h5','/mov',[1,1,1],[100,100,1000]);
```

Here, the first entry is the name of the h5 file, the second entry dataset, the third entry the start coordinates, and the final entry is the how much to include in that coordinate, reading in rectangular chunks. If the third and forth entries are not given, h5read reads the full movie into RAM memory.

Once loaded on the RAM memory, the Ca^{2+} imaging movie can be watched using the native EXTRACT function:

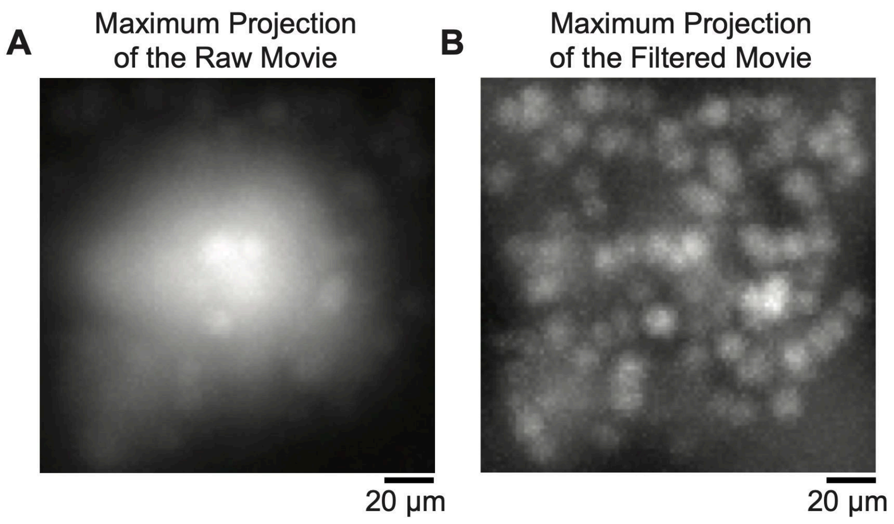
```
view_movie(M)
```

It is often good practice to watch the raw Ca^{2+} imaging movies for potential motion and/or experimental artifacts before starting the cell extraction process.

4.2. Spatial high-pass filtering in EXTRACT

Prior work utilizes distinct strategies to either model the background contamination[2] or explicitly remove neuropil[3]. With EXTRACT, we use a principled approach that can remove spatially correlated neuropil activities while retaining the cells' Ca^{2+} activity signals, *i.e.*, utilize spatial high-pass filtering. Within EXTRACT's preprocessing module (`preprocess_movie.m`), we have coded a GPU accelerated band-pass filter, which can be applied as a spatial high-pass filter with the following code:

```
avg_cell_radius = 6;
spatial_highpass_cutoff = 5;
spatial_lowpass_cutoff = inf;
use_gpu = 0;
M_proc = spatial_bandpass(M, avg_cell_radius, ...
 spatial_highpass_cutoff, spatial_lowpass_cutoff, ...
 use_gpu);
view_movie(M_proc)
```


A
Maximum Projection of the Raw Movie
B
Maximum Projection of the Filtered Movie
20 μm
20 μm

Appendix Figure 12. Spatial high-pass filtering removes global neuropil contamination.
A The maximum projection of the raw simulated one-photon Ca^{2+} imaging movie shows signs of global contamination. **B** Performing a spatial high-pass filtering on the full movie removes the global background and allows visual identification of the cells' spatial profiles.

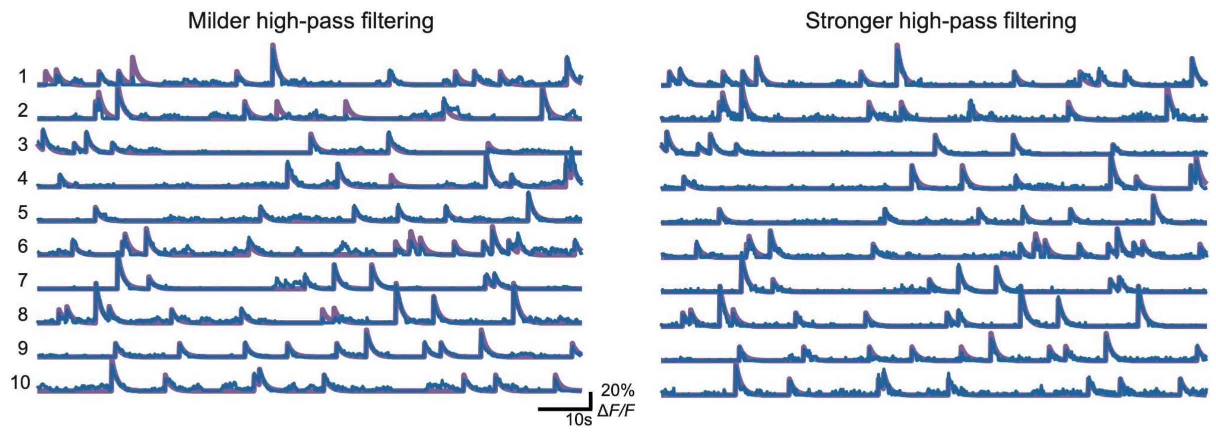
As shown in **Appendix Figure 12**, the spatial filtering is particularly useful for removing global background. As we show next, this process does not remove cells' Ca^{2+} events (**Appendix Figure 13**), which is theoretically consistent with the choice of the cutoff threshold.

4.3. Preprocessing and saving movies in batches

When large Ca^{2+} imaging movies need to be processed in batches, the preprocessing module of EXTRACT can be run in an offline manner, and by looping through the movie frames to prevent memory errors. This can be achieved with a few lines of code:

```
config = get_defaults([]);
config.partition_size_time = 500;
preprocess_save('Example_1p_movie.h5:/mov',config)
```

We conclude the **Tutorial 3** with an example cell extraction with this preprocessed movie, whose results are shown in the accompanying live script (See our Github repository) and **Appendix Figure 13**.



Appendix Figure 13. Spatial high-pass filtering does not remove cells’ Ca^{2+} events.
Since the cutoff value for the spatial high-pass filtering is chosen far away from the spatial frequencies containing cells, the spiking events are not removed from the Ca^{2+} activity traces. In fact, in certain cases, stronger high-pass filtering may allow higher quality cell extraction, and therefore better recovery of the spiking events in noisy Ca^{2+} imaging movies. Here, we are using a simulated movie with extensive global neuropil contamination.

§5.Two-step hyperparameter optimization of the cell finding and refinement modules

EXTRACT’s cell finding and refinement modules can be optimized in a two-step optimization This section primarily discusses **Tutorials 4** and **5**, where we discuss the optimization of hyperparameters for EXTRACT’s cell finding and refinement modules, respectively. For these tutorials, we use a low SNR two-photon Ca^{2+} imaging movie, courtesy of Parker Lab[4].

5.1. Step one: Optimizing the cell finding module

Given the modularity of EXTRACT’s cell finding module, we can optimize its hyperparameters in a modular and decoupled manner in **Tutorial 4**. We start by preprocessing the raw Ca^{2+} imaging movie and watching the resulting filtered version for potential artifacts via the code:

```
M = h5read('jones.h5','/data');
config = get_defaults([]);
M_proc = preprocess_movie(M,config);
view_movie(M_proc(:,:,1:100))
```

This particular movie has rather low signal and high levels of noise, making it challenging to process with the default values. To test this, we can turn off the cell refinement process and watch the cell extraction real-time as it is happening with following the code:

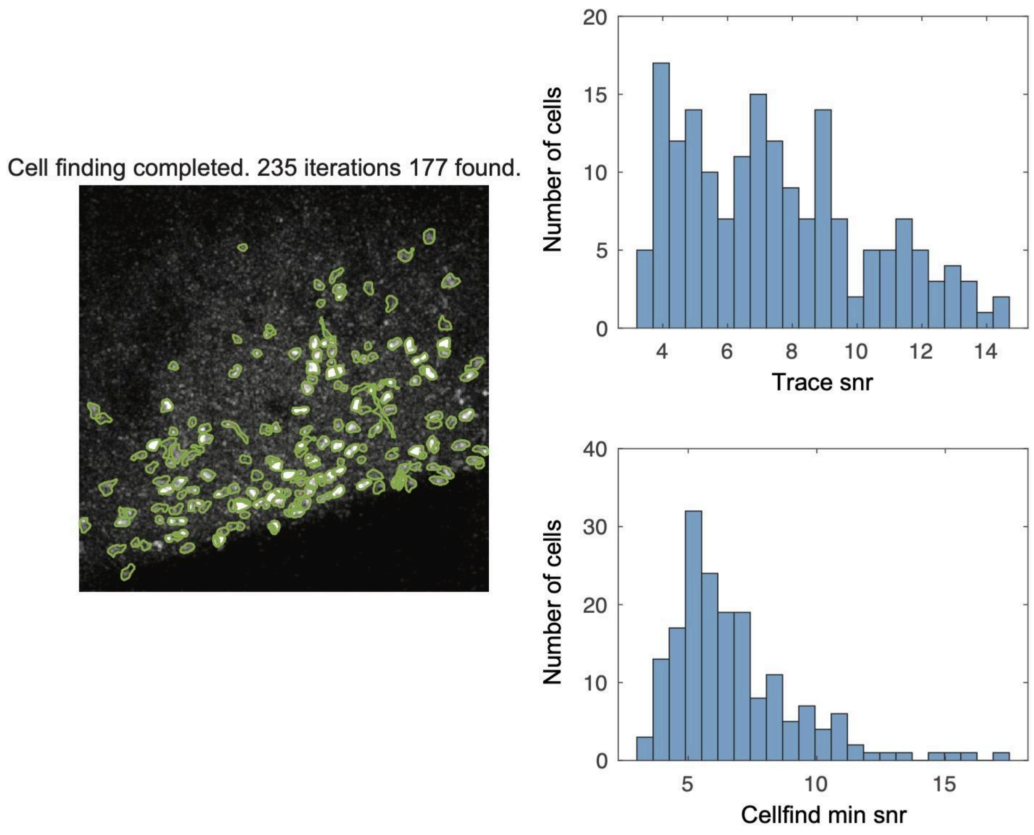
```
config.max_iter = 0;
config.visualize_cellfinding = 1;
output = extractor(M,config);
```

As shown in the accompanying live script[1], EXTRACT's cell finding module fails to identify several cell candidates that are visible to the naked eye.

To optimize the cell finding module, we first note that the earlier run had shown an early cutoff for the trace snr values, indicating that more cells might have been found if `thresholds.T_min_snr` had a lower SNR threshold. Second, the `cellfind_min_snr` value was centered around 5 (also see **Appendix Figure 14** below), which is often at the order of tens to hundreds. Thus, as a first step, we pick very low values for both parameters. Next, since our main concern is being able to identify low SNR cells, we also decrease `spatial_lowpass_cutoff` value, which performs a low-pass filtering and smooths the movie for cell finding purposes only:

```
M = 'jones.h5:/data';
config = get_defaults([]);
config.downsample_time_by = 4;
config.spatial_lowpass_cutoff = 1;
config.use_gpu = 0;
config.max_iter = 0;
config.visualize_cellfinding = 1;
config.cellfind_min_snr = 0;
config.thresholds.T_min_snr = 3.5;
output = extractor(M,config);
```

The results, shown in **Appendix Figure 14,** contain several duplicate and/or spurious cell candidates, which need to be removed by the cell refinement module that we discuss next.

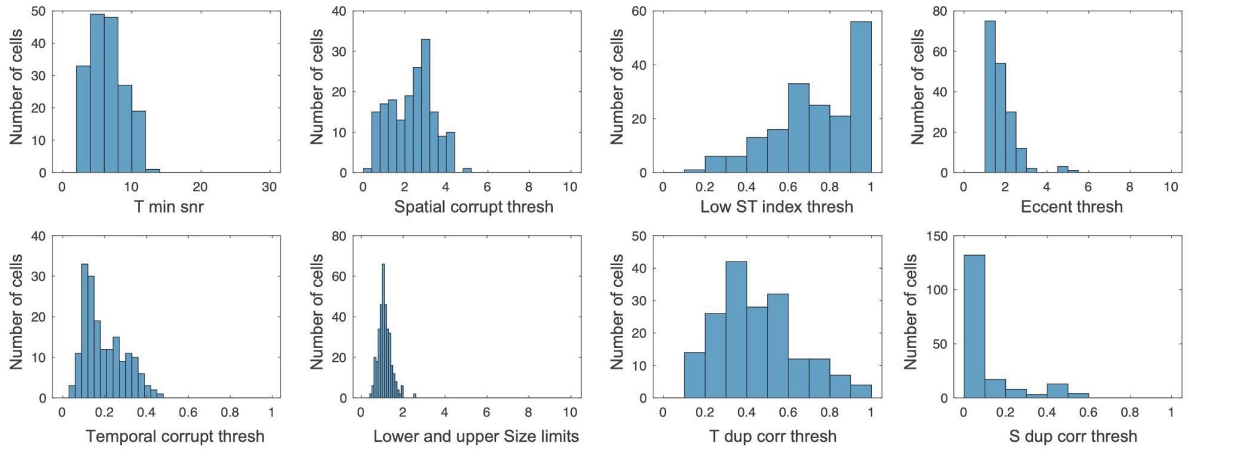


Appendix Figure 14. The dynamical visualization of the cell finding module.
If `visualize_cellfinding = 1`, EXTRACT provides a dynamical visualization of the cell finding process. Here, the cells are identified on a projection map on the left and the cell finding metrics are presented real-time on the right. In this tutorial, multiple low SNR cells (with SNR < 7) were identified that would have been rejected by the default values.

5.2. Step two: Optimizing the cell refinement module

Having discussed the first step of the optimization routine, with **Tutorial 5**, we now focus on the cell refinement module. Within EXTRACT, we have a flag for optimizing the cell refinement module (`hyperparameter_tuning_flag`). When this parameter is true, EXTRACT performs a single cell refinement step and saves the quality metrics. Then, the function, `plot_hyperparameter_curves.m`, can be used to plot the quality metrics:

```
M = 'jones.h5:/data';
config = get_defaults([]);
config.downsample_time_by = 4;
config.spatial_lowpass_cutoff = 1;
config.use_gpu = 0;
config.hyperparameter_tuning_flag = 1;
config.cellfind_min_snr = 0;
config.thresholds.T_min_snr = 3.5;
config.cellfind_min_snr = 0;
config.adaptive_kappa = 2;
output = extractor(M,config);
plot_hyperparameter_curves(output)
```



Appendix Figure 15. A summary of the trace quality metrics for hyperparameter tuning. When `hyperparameter_tuning_flag = 1`, EXTRACT performs only a single cell refinement step and saves the computed trace quality metrics. Using the helper function `plot_hyperparameter_curves.m`, the user can visualize these metrics and broadly decide on the first guess of the thresholding parameters.

The resulting plots (shown in **Appendix Figure 15**) constitute the starting points for the tuning of the quality metric thresholds:

- Firstly, many data points in the `T_min_snr` curve seem to be right at the threshold. We might wish to lower it further (here down to 3.2) and check if it prevents discarding true positive cells.
- Next, the spatial corruption values in these plots are at ~ 4-5, much higher than the default value of 1.5. This is expected for very noisy movies. We need to increase this threshold value if we wish to keep the true cells (here up to 5).
- Next, the extracted cell areas seem to be rather small, usually a sign that either our average cell radius is quite off or that `kappa_std_ratio` is too strict. We changed the latter to 1 in this example, which helped recover some incorrectly discarded cells.

- We picked the number of iterations to be 10, which was empirically when the cell profiles stabilized (you can check this by setting `visualize_cellfinding = 1`).
- Moreover, we kept the adaptive estimation for cell refinement (`adaptive_kappa = 2`), which was particularly helpful for this low SNR movie.
- Finally, we observe a long tail in the `T_dup_corr_threshold` curves, which is why we set the correlation threshold to 0.8. We finetuned this value while checking the final outputs for a few runs and ensuring that duplicate cells were properly discarded.

With that, we were able to optimize the cell refinement module with a few runs, each took around 2-3 minutes. For larger movies, the same procedure can be performed on a smaller chunk of the movie for faster optimization.

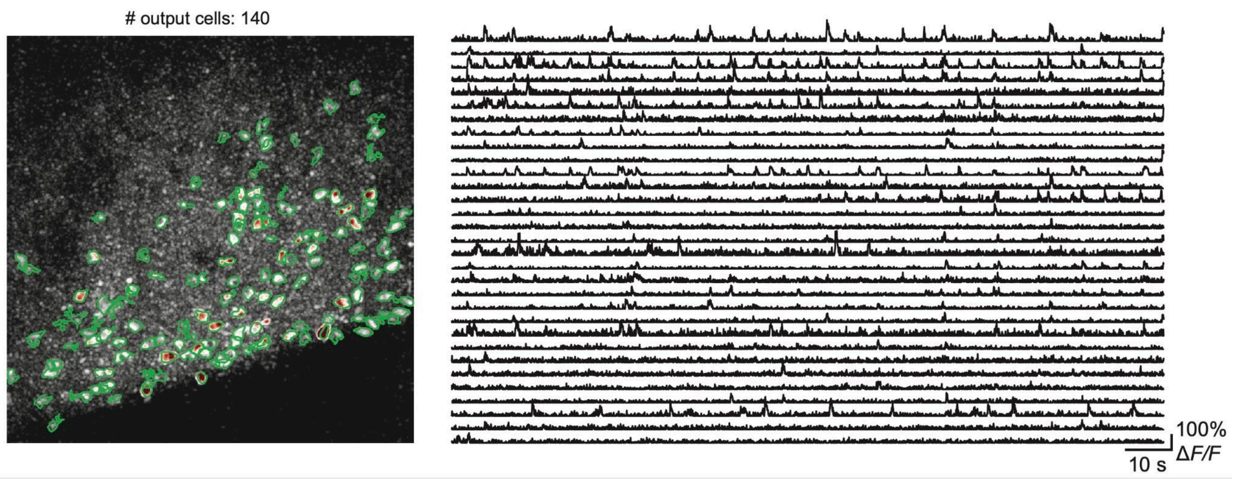
5.3. The final cell extraction result

Bringing all together, we ran EXTRACT on the full movie after a quick optimization, which gave rise to the following code:

```
M = 'jones.h5:/data';
config = get_defaults([]);
config.downsample_time_by = 4;
config.spatial_lowpass_cutoff = 1;
config.use_gpu = 0;
config.max_iter = 10;
config.cellfind_min_snr = 0;
config.thresholds.T_min_snr = 3.2;
config.thresholds.spatial_corrupt_thresh = 5;
config.thresholds.T_dup_corr_thresh = 0.8;
config.adaptive_kappa = 2;
config.kappa_std_ratio = 1;
output = extractor(M,config);
figure
plot_output_cellmap(output,[],[],'clim_scale',[0.2, 0.999])
figure
T_ex = output.temporal_weights';
```

```
plot_stacked_traces_double(T_ex(1:31,:),[],0)
```

The cell extraction results are shown in **Appendix Figure 16**. In our experience, these optimized parameters are robust and often perform well across days and mice, as long as the imaging and experimental conditions are not significantly different.



Appendix Figure 16. Visualization of the final cell extraction results with EXTRACT.
After optimization, EXTRACT provided high quality cell extraction results with little to no spurious or duplicate cells (left) and the Ca^{2+} activity traces from the first 30 extracted cells (right).

§6. How to use the final robust regression module as a stand-alone tool

Apart from being utilized as a cell extraction tool, EXTRACT can also be used as a post-processing tool to obtain Ca^{2+} activity traces via robust regression given the cells' spatial profiles. In **Tutorial 6**, we quickly show how to configure EXTRACT to do this by using the movie from the first tutorial:

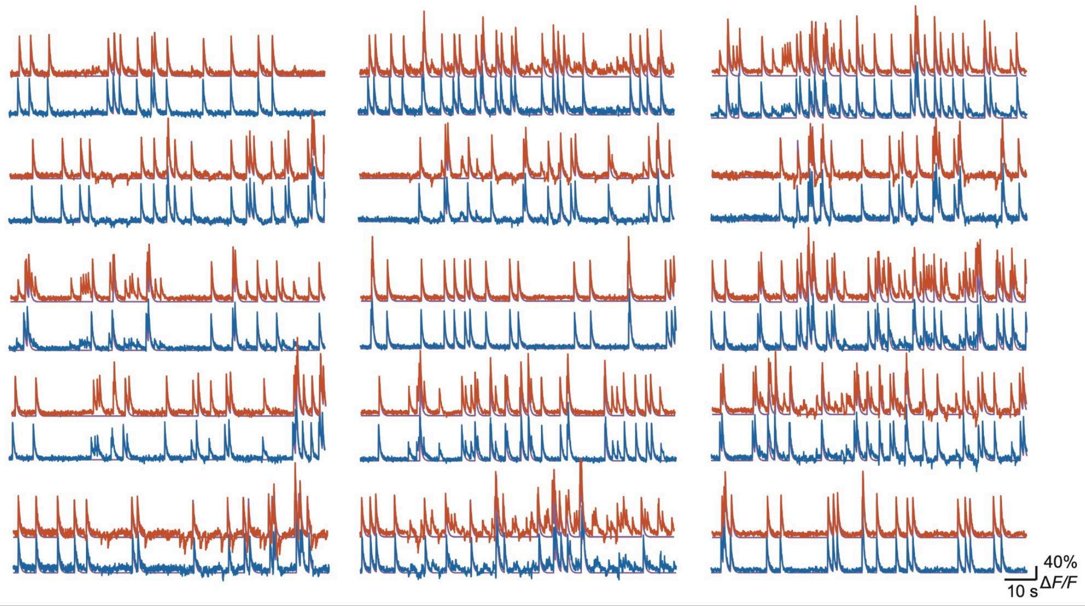
```
load('example.mat');
M = M-1; % ground truth movie with F = 0;
```

We initialize EXTRACT with the first 15 cells, whereas the other 5 are undetected, simulating the existence of neuropil and other contamination sources. Then, we can compute both the robust and the least-squares estimates of Ca^{2+} activity traces with the following code:

```
config=[];
```

```
config = get_defaults(config);
config.avg_cell_radius=7;
S = S_ground(:,1:15);
config.S_init = S;
config.preprocess = 0; %using a preprocessed movie
config.F_per_pixel = ones(50,50); % F values are all one
config.trace_output_option='no_constraint';
config.use_gpu=0;
config.regression_only = 1;
output_rb=extractor(M,config);
T_ex = output_rb.temporal_weights';
config.trace_output_option='least_squares';
output_ls=extractor(M,config);
T_ls = output_ls.temporal_weights';
```

Here, once `regression_only=1`, EXTRACT skips the cell finding and refinement modules. In this scenario, `S_init` should contain the spatial profiles (2D or 3D, both acceptable). The results are shown in **Appendix Figure 17**. Note that both estimates are imperfect, though robust regression preferentially suppresses the contamination from unidentified cells.



Appendix Figure 17. EXTRACT as a post-processing tool for estimating Ca^{2+} activity traces.
Using the movie from **Tutorial 1**, we estimated cells’ Ca^{2+} activity traces by using EXTRACT as a post-processing tool. We initialized EXTRACT with 15 out of 20 cells and estimated the Ca^{2+} activity traces with robust (blue) and least-squares (red) regression. Though both algorithms showed occasional crosstalk, robust regression led to preferential suppression of contamination and consequently to more accurate Ca^{2+} activity traces.

§7. Conclusion

In this document, we provided a quick start guide for EXTRACT and introduced the two-step optimization routine that we regularly use to quickly optimize its hyperparameters. We have shared six tutorials in our accompanying repository and a descriptive figure (**Figure S9**), which should hopefully help first-time users get started with EXTRACT. For any additional questions or comments, please open an issue in the Github repository or contact via email extractneurons@gmail.com. We also occasionally hold tutorial sessions to help new labs get started, which you can request via the same email.

References for Supplementary Note 5

1. GitHub - schnitzer-lab/EXTRACT-public: EXTRACT is a tractable and robust automated cell extraction tool for calcium imaging, which extracts the activities of cells as time series from both one-photon and two-photon calcium imaging movies. *GitHub* https://github.com/schnitzer-lab/EXTRACT-public.
2. Giovannucci, A. *et al.* CaImAn an open source tool for scalable calcium imaging data analysis. *Elife* **8**, (2019).
3. Pachitariu, M. *et al.* Suite2p: beyond 10,000 neurons with standard two-photon microscopy. *bioRxiv* 061507 (2017) doi:10.1101/061507.
4. Parker, J. G. *et al.* Diametric neural ensemble dynamics in parkinsonian and dyskinetic states. *Nature* **557**, 177–182 (2018).