Installation and Usage instructions for the ImageJ plugin calciumImaging

Thomas Braschler, Fatemeh Navee, November 2019

Content

Installation

Prerequisites

Usage of this plugin requires a functional ImageJ[1] or Fiji[2] installation. It was tested on ImageJ version 2.0.0-rc-44/1.50g as part of a Fiji installation on MacOSX but should hopefully also work on a variety of other version and operating systems. While we hope that this plugin and its source code elements will be useful, we can however no guarantee about functionality for any particular purpose or on any particular operation system (see License section).

Download

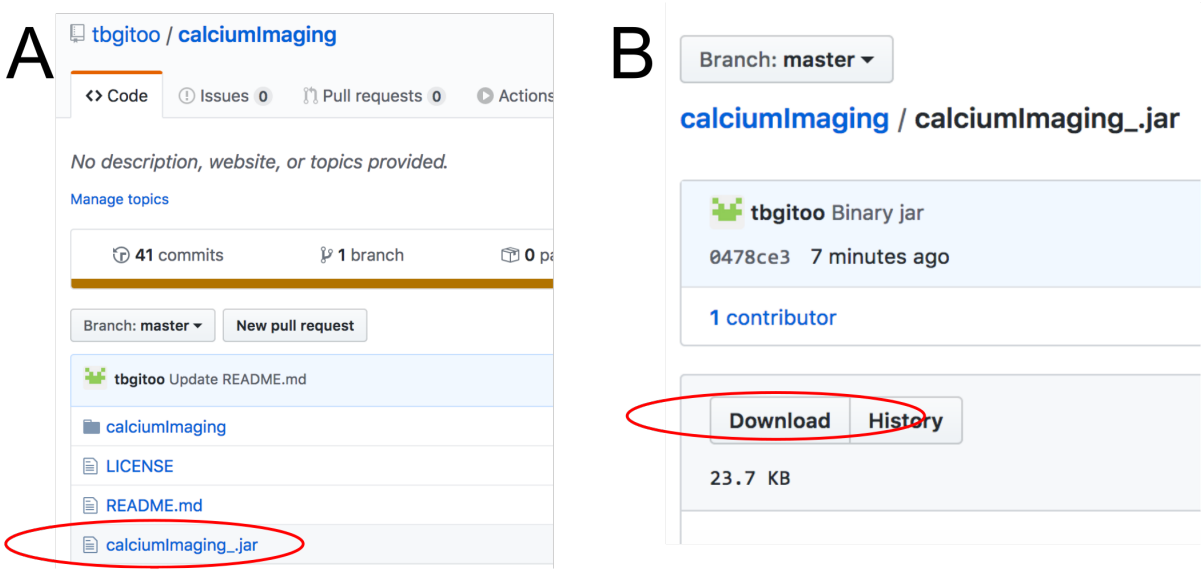

Figure 1. Download calciumImaging_.jar from https://github.com/tbgitoo/calciumImaging

The first step in installation of the calciumImaging plugin is downloading the full .jar file https://github.com/tbgitoo/calciumImaging. The "calciumImaging_.jar" file is the relevant binary (Figure 1A), it can be downloaded by clicking on it on github and then downloading it on the dedicated page (Figure 1B).

Alternatively, a direct link is:

https://github.com/tbgitoo/calciumImaging/blob/master/calciumImaging_.jar

Plugin installation

After downloading, the file "calciumImaging_.jar" needs to placed in the plugins folder of the ImageJ (or Fiji) installation. The location of the ImageJ (or Fiji) installation varies depending on the options chosen during installation. On MacOSX, it can typically be found in /Applications/Fiji or /Applications/ImageJ. On Windows, a typical install location would be C:\\Program Files(x86)\ImageJ respectively Fiji, or also C:\\Program Files\ImageJ respectively Fiji for 64-bit installations or on systems before the introduction of separate program file folders for 32-bit and 64-bit programs.

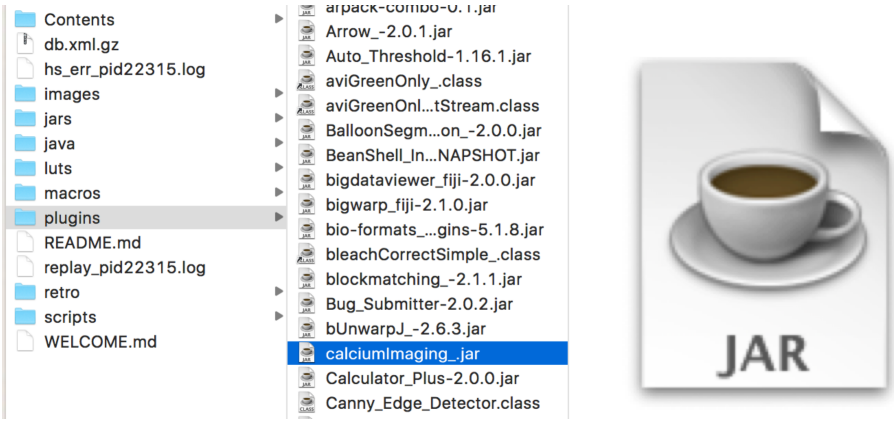

Figure 2. Installation in the plugin folder (here, of Fiji)

In case Fiji or ImageJ was already running during the installation, they need to be restarted after placing the "calciumImaging_.jar" file in the plugins folder of your ImageJ/Fiji installation. This is because the plugins are loaded into memory during startup of ImageJ/Fiji.

Usage

Preparation

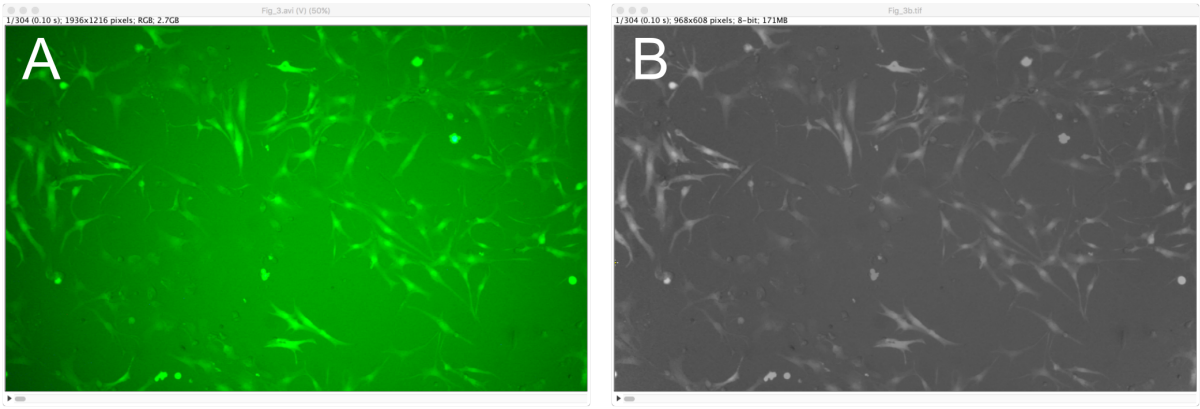

Figure 3. Preparation for peak analysis. A) RGB raw video, loaded in Fiji. B) Green channel as grey scale image, downsampled, and with vignetting and photobleaching corrections

The calciumImaging plugin needs a greyscale, 8-bit z-stack in ImageJ or Fiji to work properly. Therefore, RGB videos need to be processed before to satisfy this requirement.

In many cases, downsampling the xy-dimensions of large videos is also a good idea. This reduces noise which otherwise makes peak detection more difficult, and also accelerates the calculations.

Further, peak detection becomes easier when certain typical microscopy artefacts are corrected. First, many fluorescence objectives show some vignetting – the corners tend to be darker than the center of the image (example: Fig. 3A). This is a purely optical effect and arises from less-than-ideal light transmission far from the image center. We typically correct vignetting by fitting a parabolic background (with the aid of the Plugin "Polynomial Surface Fit", order 2 for x and y direction) on cell-free areas or on control

images in the absence of cells, and use the resulting parabolic smooth vignetting image for correction of the light-collection efficiency (example: Fig. 3B).

Another problem is photobleaching, leading to decreasing fluorescence intensity over time. Like vignetting, this makes application of a global threshold difficult, and it is advisable to adjust the intensity in such a way that there is no global trend over the z-profile. We do so by measuring the time-course of the intensity on a cell-free area, and adjust the overall intensity per z-plane (video frame) such as to keep this intensity constant (not seen in Fig. 3).

These are not the only possibilities, some microscopes provide photobleaching routines, and vignetting can be reduced by narrowing the field of observation or by cropping. It is in any case advisable to analyse such optical and photobleaching effects by tracing in-plane and z-profiles and if necessary to apply appropriate corrections, as the calciumImaging plugin at present takes the data as they are.

Temporal peak detection

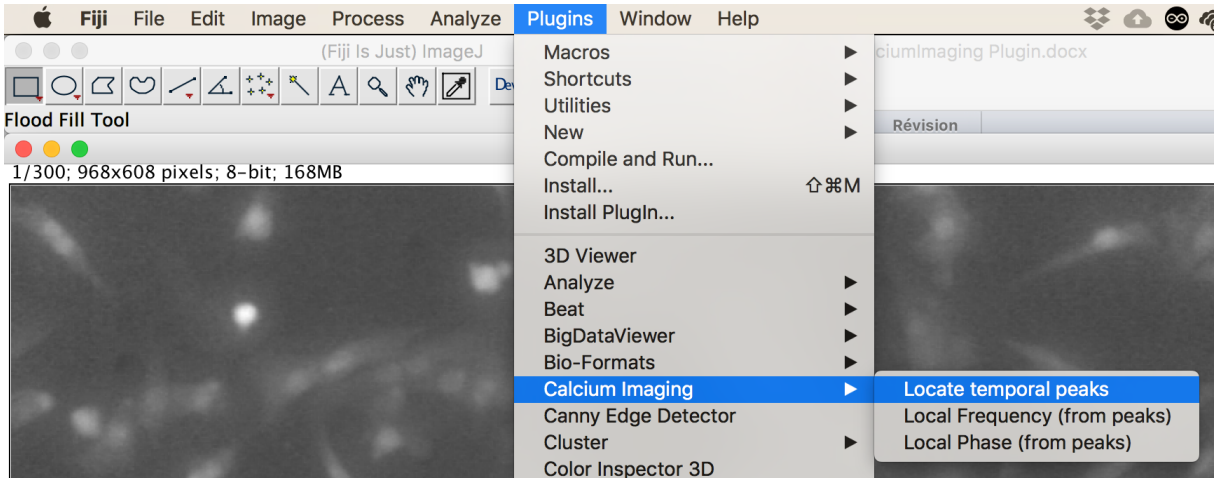

Figure 4. Starting the temporal peak detection.

With a greyscale, 8-bit z-stack loaded, initiate temporal peak detection by clicking ">Plugins>Calcium Imaging>Locate temporal peaks" (Figure 4).

Plugin options

The plugin options appear after plugin start. They are shown schematically in Figure 5, in comparison with a z-profile of intensity, showing temporal activation. We tried to propose reasonable standard values, but experience shows that peak detection can be made more pertinent by examining z-axis profiles in areas containing cells or background only, to more realistically set the plugin parameters.

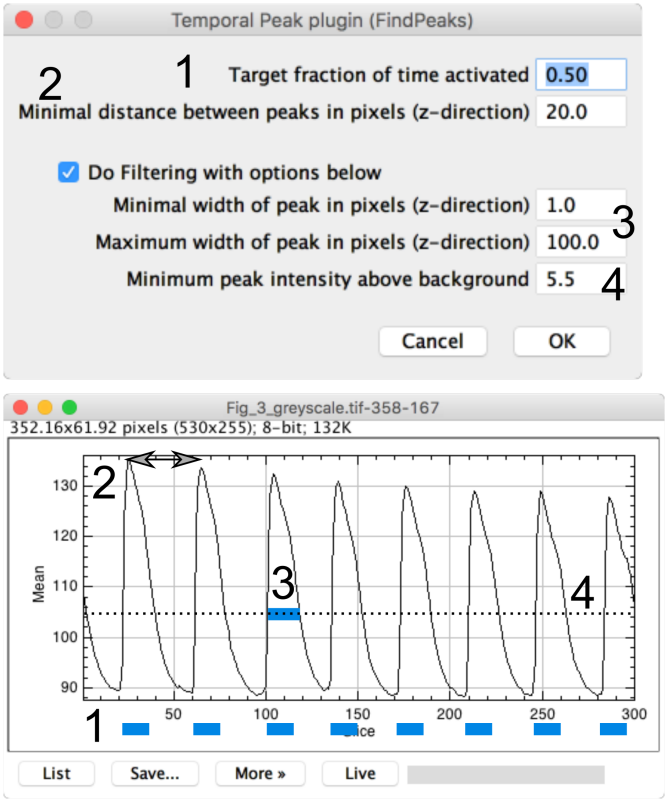

Figure 5. Plugin options for temporal peak detection. The meaning of the different parameters is indicated in the dialog, and illustrated in comparison with the z-axis intensity profile.

Plugin output

The temporal peak detection yields a new image, of the same xy and z dimensions as the stack analyzed. In the new image, white pixel (greyscale value 255) indicate local peaks, black pixels (greyscale value 0) indicate non-peak values (Figure 6).

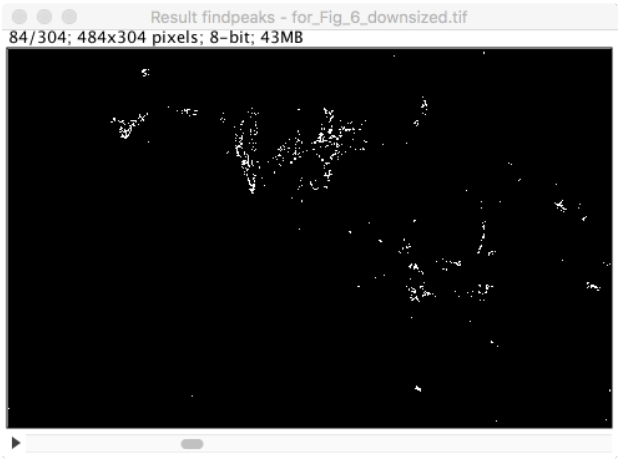

Figure 6. Temporal peak detection output

Evaluation of local frequency

Mean frequency can be evaluated manually, by performing z-projection on a temporal peak stack as shown in Figure 6, and rescaling the values to obtain frequencies in Hz or beats per minute.

For convenience, the calciumImaging plugin offers streamlining the process. To do so, user entry of the actual frame-rate is required (in frames/second), the plugin then does the z-projection and ensures conversion to beats per minutes. To perform conversion to local frequency via the calciumImaging plugin, select Plugins>Calcium Imaging>Local Frequency (from peaks) as shown in Figure 7

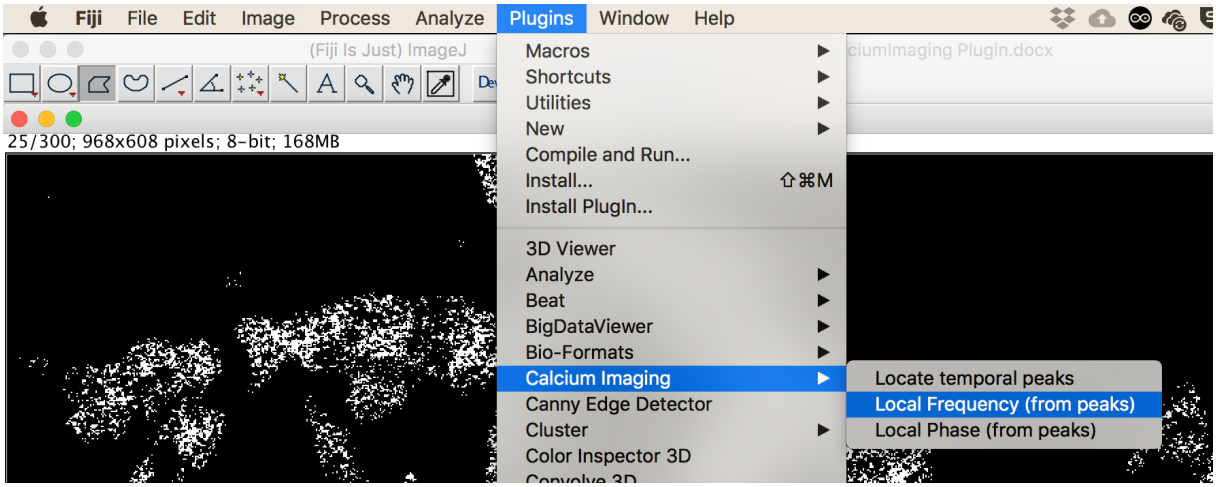

Figure 7. Calculation of local frequency from a temporal peak stack.

The calciumImaging plugin performs the local frequency analysis and shows it as greyscale image with 32-bit (float values). Areas where no peaks were detected are set to 0.

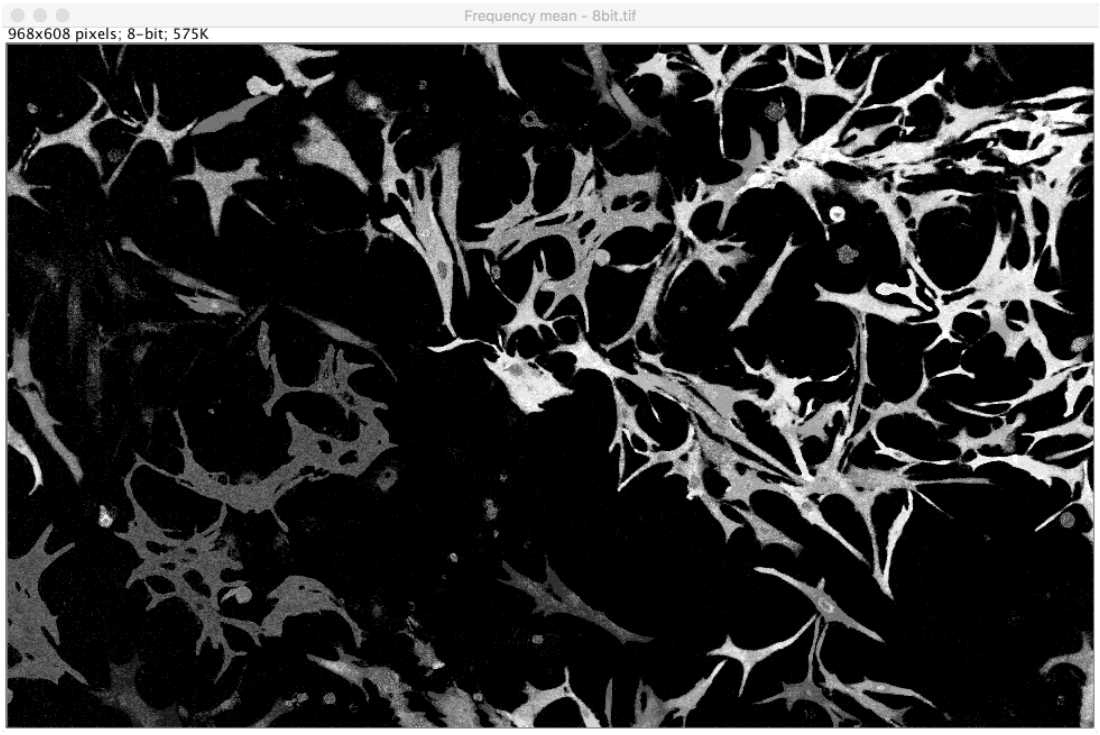

Figure 8. Local frequency output

Local phase

As a last functionality, the calciumImaging plugin allows evaluation of local phase. Like local frequence evaluation, local phase evaluation is performed on a stack with identified peak values (e.g. as in Figure 6). To launch local phase evaluation, select "Plugins>Calcium Imaging>Local Phase (from peaks)" as shown in Figure 9.

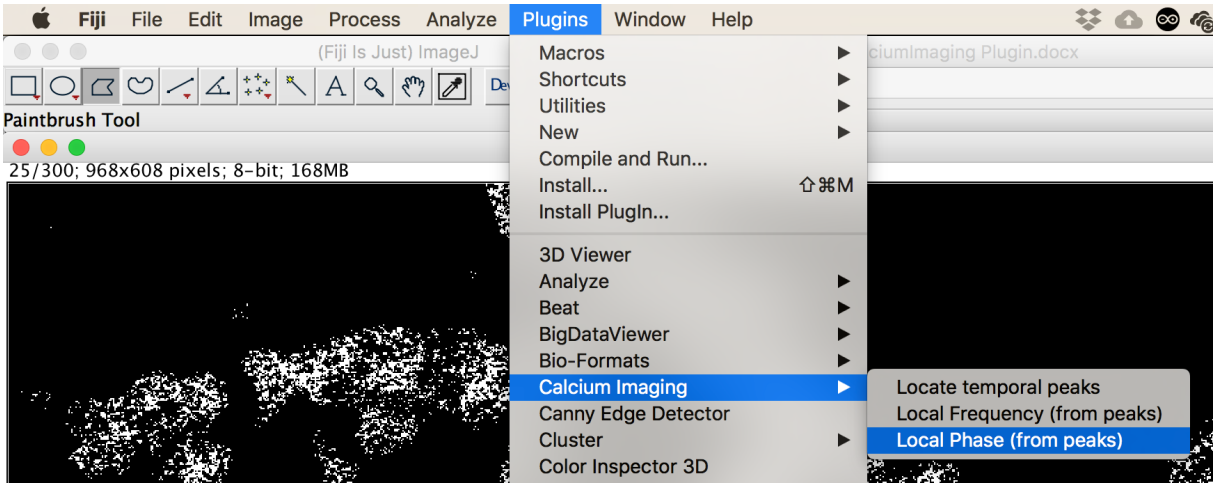

Figure 9. Launching of local phase evaluation.

Local phase is evaluated by comparison of the temporal peak sequence to the sequence at a reference point supplied by the user (via the dialog that opens, Figure 10) . The reference section should, where possible, reflect the general beating behavior, and so some care should be taken in its selection.

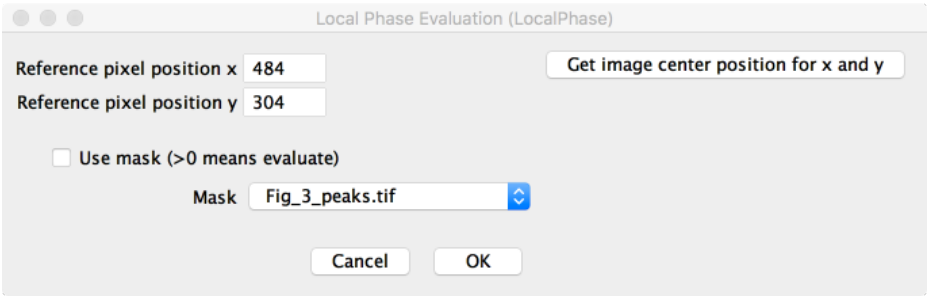

Figure 10. Options for the local phase evaluation.

Figure 11 shows the working principle of local phase evaluation. The analysis compares every point C of the image (see Figure 11A) to the reference point B, as specified in the user dialog (Figure 10). For each point of interest (C) and also for the reference point B, the z-profile is evaluated (see Figure 11).

To evaluate the local phase, comparison proceeds for each peak found in the z-profile of the point of interest (Figure 11C). The relative timing of the peaks in the z-profile of the point of interest to the neareast peak in the reference section (Figure 11B) is determined, and converted to a phase angle between -180° (late) and + 180° (early). For peaks at the point of interest before the first or after the last reference peak, the plugin extrapolates by assuming a constant frequency in the reference.

For averaging of the phase angle, the individual phase angles are converted to unit complex vectors, and the overall phase angle obtained from the mean complex vector; this avoids problems with phase jumps between -180° and 180°.

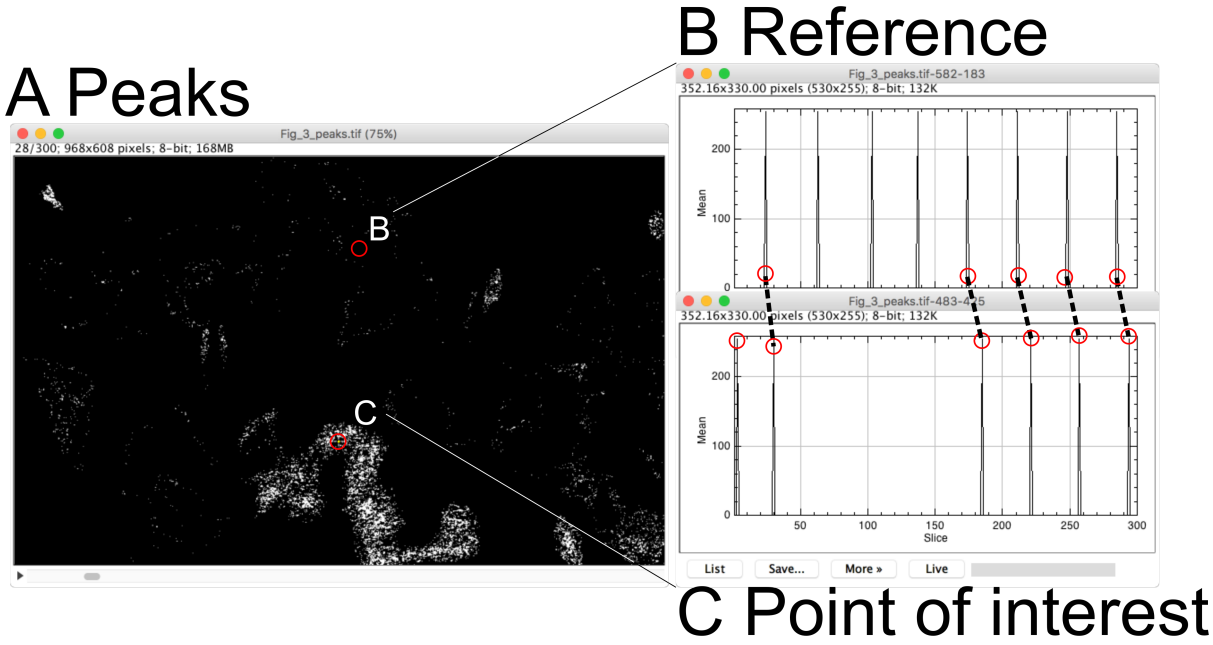

Figure 11 Working principle of local phase evaluation. From the peak localization stack (A), the reference z-profile is extracted at point defined by the user (B). The point of interest is then scanned across the entire image (x and y direction), and for each point, the intensity profile across the stack is evaluated (C, arbitrary example point). Here, the point of interest peaks are generally late compared to reference peaks, so that would be a negative phase value.

The output of the phase angle evaluation is a flat image with the same xy dimensions as the peak stack (Figure 12), with the value of the phase given in degrees at each point. Early timing corresponds to positive phase values (advanced), late timing to negative phase values.

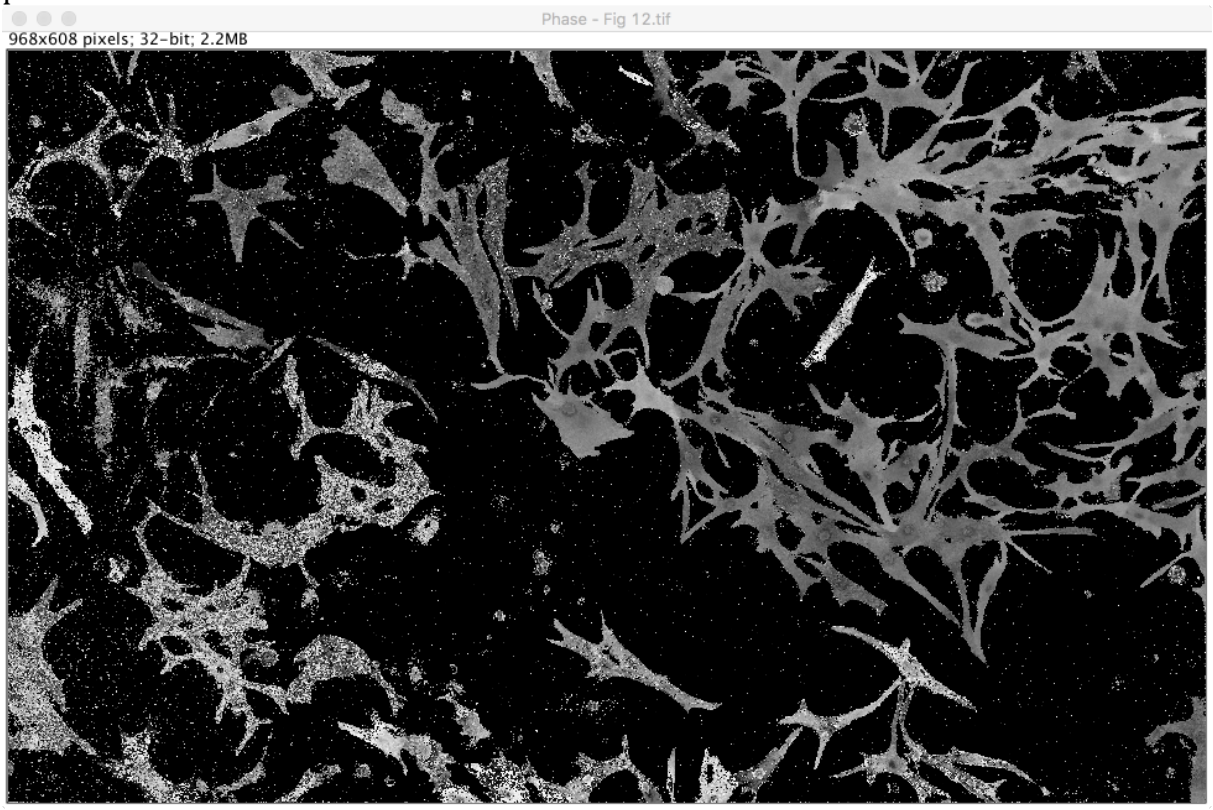

Figure 12. Phase output, grey scale from -180° (late activation) to +180° (early activation). Pixels without activity are given as NaN, which displays black, but can be distinguished from 0 in ImageJ 32-bit images.

Concept and Credits

The calcium imaging data was acquired by Fatemeh Navee, Thomas Braschler wrote the code and this user manual. The screenshots shown here were obtained while treating calcium imaging data, but are not from a single run; the goal is merely to illustrate how to use the plugin, not to characterize any experiment in particular.

A very important contribution to this calciumImaging plugin is the open source Octave[3] method "findPeaks", by Juan Pablo Carbajal (see https://searchcode.com/codesearch/view/64213481/). This method find peaks in an arbitrary signal, by a combination of value sorting, thresholding, parabola fitting, and filtering, and we applied it to find the location of the peaks. For the sorting part, we also made use of a code snippet in a stackexchange discussion (https://stackoverflow.com/questions/4859261/get-the-indices-of-an-array-after-sorting, see further license details in the source code itself).

Evaluation of beating motion or calcium spiking characteristics from videographic evidence is by no means new[4], and some very sophisticated tools are available. A powerful online analysis tool is for example the Pulsevideoanalysis platform (https://pulsevideoanalysis.com), which can be used to obtain beating frequency and other characteristics from calcium imaging videos. For specific analysis such as the phase analysis proposed here, such closed source software would however be difficult to adapt.

We therefore aimed at providing and using open source software for the calciumImaging plugin. In this context, the idea of using ImageJ[1] to evaluate spikes in calcium imaging is also not new, and there are some published tools available[5][6], although they tend to focus on local spike identification. A sophisticated wavelet analysis for peak detection is available as an open source tool (NA^3[7]), but this requires integration of ImageJ[1] with R[8] on the Bio7[8] platform.

With the present calciumImaging plugin, we pursued the following specific aims:

1) Like other purely ImageJ[1] tools[5][6], the aim was to provide a tool compliant with the simple, usual **ImageJ plugin installation** procedure : By physically placing a binary file into the plugins folder of the local ImageJ installation, the functionality should become available. Hence, we wrote this plugin as a self-contained, Java[10]-only implementation.
2) We nevertheless wanted to be able to carry out efficient **temporal peak identification**[7]. This problem is elegantly addressed in the NA^3 software[7], but it comes at the cost of complexity arising from the connection with R[8] and the use of the Bio7[8] libraries. We therefore resorted to the known highly performant, and open-source findPeaks algorithm of Octave[3] (https://searchcode.com/codesearch/view/64213481/). We re-implemented this in Java[10] to avoid having to use a bridge to Octave[3].

3) And finally, we wanted to be able to carry not only standard **frequency** (« activity » [7]) analysis, but also temporal shift analysis in the form of the **phase shift**.

Bibliography

[1] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "NIH Image to ImageJ: 25 years of image analysis," *Nat Methods*, vol. 9, no. 7, pp. 671–675, Jul. 2012.

[2] J. Schindelin *et al.*, "Fiji: an open-source platform for biological-image analysis," *Nat Methods*, vol. 9, no. 7, pp. 676–682, Jul. 2012.

[3] Eaton, J.W., Bateman, D., Hauberg, S., and Wehring, R., *GNU Octave version 3.8.1 manual: a high-level interactive language for >> numerical computations*. CreateSpace Independent Publishing Platform, 2014.

[4] M. Maddah *et al.*, "A Non-invasive Platform for Functional Characterization of Stem-Cell-Derived Cardiomyocytes with Applications in Cardiotoxicity Testing," *Stem Cell Reports*, vol. 4, no. 4, pp. 621–631, Mar. 2015.

[5] E. M. Steele and D. S. Steele, "Automated Detection and Analysis of Ca2+ Sparks in x-y Image Stacks Using a Thresholding Algorithm Implemented within the Open-Source Image Analysis Platform ImageJ," *Biophysical Journal*, vol. 106, no. 3, pp. 566–576, Feb. 2014.

[6] E. Picht, A. V. Zima, L. A. Blatter, and D. M. Bers, "SparkMaster: automated calcium spark analysis with ImageJ," *Am. J. Physiol., Cell Physiol.*, vol. 293, no. 3, pp. C1073-1081, Sep. 2007.

[7] J. Prada, M. Sasi, C. Martin, S. Jablonka, T. Dandekar, and R. Blum, "An open source tool for automatic spatiotemporal assessment of calcium transients and local 'signal-close-to-noise' activity in calcium imaging data," *PLOS Computational Biology*, vol. 14, no. 3, p. e1006054, Mar. 2018.

[8] R Core Team, *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2015.

[9] Austenfeld, M., "Bio7 User Guide Version 3.0 (work in progress)." .

[10] K. Arnold, J. Gosling, and D. Holmes, *The Java programming language*. Addison Wesley Professional, 2005.

License

You should have received a copy of the GNU General Public License
along with this program; if not, see <http://www.gnu.org/licenses/>.