SynapseLocator application

Repeated imaging of brain tissue at high spatial resolution is faced with minute movements and non-concordant positional changes of synapses. SynapseLocator is a versatile tool that combines many steps of image processing, non-rigid image registration, and spot localisation. Input data are typically 3D images from pre- and post-stimulation (square input images in form of interleaved two-channel data, scanimage format, scanimage.vidriotechnologies.com, 16-bit greyscale) for which numerical and graphical result data are produced. Input images are filtered and deconvolved which considerably improves registration and spot detection. By default, results are reported based on processed images. However, channel intensities for minimally (or non) pre-processed images are also reported.

SynapseLocator Installation

SynapseLocator has been tested with Matlab version 2017b and higher (Windows 7 and Windows 10 operating system). In order to run the software ImageJ and elastix must be installed.

Prepare ImageJ/Fiji

Install Fiji (https://fiji.sc/), note that Fiji should not be installed to "Program Files". Choose rather “C:\fiji.app” to make it system-wide available or chose a directory somewhere in your user space (e.g., "C:\Users\[your name]\Fiji.app"). Download code for DeconvolutionLab2 (http://bigwww.epfl.ch/deconvolution/deconvolutionlab2/) and place the jar file into the plug-ins folder of Fiji. Start Fiji and run the “Update...” command from the Help drop down menu. Click on “Manage update sites” and add update sites “ImageScience” and “3D ImageJ Suite”. Click Apply changes, restart Fiji.

Prepare elastix

Install elastix transformation software (http://elastix.isi.uu.nl/). Download the latest version (elastix-4.9.0) from github (https://github.com/SuperElastix/elastix/releases/tag/4.9.0). Unpack the zipped file and place all files into folder “~\SynapseLocator\elastix\elastixProgram”.

Prepare Matlab

Add the “SynapseLocator” directory with all its subfolders to the Matlab search path. Test images are also included (see sub-directory “Data”). Use Matlab (“Open as text”) to open the parameter file “synapseLocatorParams.csv” from the SynapseLocator directory and edit line 19 “IJ_exe, C:\Fiji.app\ImageJ-win64.exe” to update the path setting for your installed Fiji.

Running SynapseLocator requires Matlab and “Image Processing Toolbox” licenses.

Prepare Test Input

To check the software functionalities, load the test data (“~\Data”, stack1.tif and stack2.tif) which contain typical experimental data (G1 holds the “green” data from time point 1, R1 the corresponding “red” channel data and so forth). Alternatively, load your own input data (check for the required format, square images, interleaved two channel data). It is recommended to start with small image stacks (256×256 pixels and a height of approximately 5-10x the spot diameter) to limit the computation time. If necessary, crop larger image stacks using ImageJ and search for a region that contains a number of spots in various environments (e.g., isolated spots and spots that have neighbouring spots at a close distance). If necessary, region masking (i.e. soma) should also be done with ImageJ. Save stacks to a separate analysis directory.

It is advised to process images including a deconvolution step prior to registration and spot finding and this is part of SynapseLocator’s workflow. However, preprocessing steps can be tested independently by starting ImageJ macro “preprocess.ijm” (found in the SynapseLocator directory “~/fiji/IJ1Macros”) from ImageJ’s macro editor. Preprocessing sequentially applies median filter/Gaussian filter, bandpass filter, background subtraction, and deconvolution (DeconvolutionLab2). Parameters for the individual steps, PSF size, and voxel size are set interactively (default values apply to the test data set). Strictly avoid settings that distort the image. Adjustment of the PSF parameters and the voxel size according to the resolution of your microscope may be needed.

Using Synapse Locator

Start SynapseLocator GUI from Matlab command window (type command “SynapseLocator”). The program reads parameters from the enclosed “synapseLocatorParams.csv” file and starts with default settings and loads a model for spot detection.

Click the “Params” button (upper right) to access the parameter settings An extended list of parameters is accessible through the “Expert” button but do not modify “Expert Settings” at this stage (see description …). Keep default settings from the “Image (Pre)Processing” panel (note that setting in the “Image (Pre)Processing” panel must be set before loading images). Click on “Load Data” and select image stacks from time-point #1 and #2. Accept the suggested output folder. (Click “No” in the “Load feature set” pop-up, see…). Adjust parameters in the “Preprocess params” pop-up to match the current experiment. In particular, check PSF and voxel size (default settings work for the test data set). SynapseLocator starts loading images and runs preprocessing. This may take a few minutes depending on PC features. The status indicator in the GUI header (next to “Params” button) displays information on the current processing step. When finished, an image from G1 (at the approximate stack centre section, see “z Level”) is displayed.

Inspect images by changing between channels and time-points using buttons from the “Image” panel (i.e. G1, G2, R1, R2). Use Matlabs Data Cursor button (upper corner of GUI) to activate the Datatip function (i.e. little cross icon) and inspect the image intensity in image G1 and G2 around spot positions. Adjust the threshold settings (parameter fields, “Threshold Image” panel); suggested values might be too low. Insert a number higher than background pixel values. Options in “Apparent Similarity” and “Label Density” allow to tune the parameters to match the actual image stacks appearance (i.e. image similarity judged by manual inspection and sparsity of labelling). The “Iterations N” panel allows to switch between a quick exploratory run and more elaborated registration. Check “quick” for a first trial.

The whole process comprising registration and spot finding can be started by clicking “Run”. However, first time users should run the two steps separately (use the “single/two step process” toggle just beside the “Run” buttons to toggle between combined and consecutive two-step processing) to speed up potential troubleshooting. For now, click on “RUN elastix”. SynapseLocator opens a Windows console in which elastix registration is started. The calculated transformation field is then used to register images from the second time-point (G2/R2) to the first time-point. Watch the status indicator to see when registration is finished. Check the “Image Match” panel to evaluate the success of the registration. Reported is the fraction of voxels with intensities above mean image intensity that are present at identical positions in both image stacks (green channel, pre and post registration). The absolute numbers vary with image quality, but in general, an increase of 10 percentage points indicates a successful registration. Visually check for successful registration and continue with spot localization.

The standard spot model (“genericSpotModel”, in “Spot Detection” panel) is a good point to start. Adjust pre/post threshold values for the green channel signal (“Spot Threshold“ panel). Again, visually inspect typical spot areas, choose values well above background (the localization step can be rerun easily with modified thresholds), and click on “RUN locator”. Processing steps can be followed at the SynapseLocator progress bar. SynapseLocator opens a table showing relevant features of identified spots and displays green and red channel data graphically. To inspect the spots in the main GUI click on Reset (“R”, GUI upper left corner) and toggle visibility of identified spot locations (clicking on panel word “Spot” panel). Use “Class” and “Probs” button to show either ROIs finally assigned as spots and total inspected voxels, respectively. Note that “Class” represents spots as ellipsoids to increase readability.

In case, registration or spot localization seem not convincing, changes made in either “Registration Params” or “Spot Detection” panel allow to re-run the respective process with updated settings. Changes in the image pre-processing step require reloading of images.

Saving SynapseLocator results

The program creates a directory “SynapseLocator_{date}_{time}” into which the resulting data are saved. Additionally, directory “elastix” is created and contains log files from the registration step. Directory “featureData” holds the features necessary to build a model and locate spots, these data can be kept and reloaded, if the location process should be repeated. Clicking “Save Results” writes results from current analysis state to spreadsheet (identified spot positions and channel intensities) and saves images including spot location (tif format). Images from image pre-processing are saved at individual step (unprocessed saved as “~_raw.tif”, just median filtered saved as “~_mf.tif”, after applying all filtering saved as “~_prepro.tif”).

Build a Model

A good spot model is crucial for automatic spot detection. If the default spot models do not provide satisfactory results, a customized model can be easily obtained after very few iterations thanks to the implemented Weka machine learning algorithm (https://www.cs.waikato.ac.nz/~ml/index.html). It classifies voxels as being either “Spot” or “No Spot” based on user-defined locations containing the respective classes. To do so, zoom into a region that contains representative spots. On “Add Class ROIs” panel, click on “Add ROI to” button. Draw a small ROI at the spot centre and click the “SPOT” button. This adds the location to the spot list. Further add 3-4 spots. Now add ROIs for positions (3 or 4) between spots by adding those as “NO SPOT”. Train the machine (click on “Train Classifier” in “ML” panel). A text summary of model performance is printed on Matlab console (percentage value in line “Correctly Classified Instances...” should approach 100). Try the new model by clicking on “RUN locator” and compare performance to previous results. Remove current locations from the list (right mouse-click) if they seem to deteriorate the model performance. Add further ROIs at positions where spot localization was not satisfactory. Train model again and check performance. Finally save your model (at default directory in the SynapseLocator tree).

The model building step can be performed after registration but also prior to registration.

Using SynapseLocator with previously registered images

The spot location process can be run independently of the registration if registered image stacks are available from an earlier run. The relevant files (“G1R1_SLready” and “G2R2_SLready”) are created together with result files from the “Save Results” command. To load registered images, the “Load transformed” button from the “Expert Settings” panel has to be activated. This forces SynapseLocator to skip input processing and to directly enter the spot location procedure. In this context, when asked for “Load feature set” an already calculated set of features may be loaded to reduce computation time.

SynapseLocator makes use of freely available software and code snippets from other sources which we hereby aknowledge. From Matlab’s File Exchange we used wekalab (https://de.mathworks.com/matlabcentral/fileexchange/58675-wekalab-bridging-weka-and-matlab), saveastiff (https://de.mathworks.com/matlabcentral/fileexchange/35684-multipage-tiff-stack), matlab_elastix (https://de.mathworks.com/matlabcentral/fileexchange/52982-matlab-elastix), KronProd (https://de.mathworks.com/matlabcentral/fileexchange/25969-efficient-object-oriented-kronecker-product-manipulation). For image stack loading we used ScanImage Tiff Reader (https://scanimage.gitlab.io/ScanImageTiffReaderDocs/index.html). Image handling tools were accessed through ImageJ/Fiji (https://fiji.sc/). For image deconvolution a plugin was loaded from DeconvolutionLab2 (http://bigwww.epfl.ch/deconvolution/deconvolutionlab2/). For image segmentation we made use of Weka machine learning tools (https://www.cs.waikato.ac.nz/~ml/index.html) available as InageJ plugin.