Running title

Synapse density quantification using Fiji

Corresponding author

Elena Rebollo

Molecular Imaging Platform (MIP), Molecular Biology Institute of Barcelona IBMB-CSIC, Spain

Baldiri Reixac 15-20, 08028 Barcelona, Spain.

erabmc@ibmb.csic.es

Automated macro approach to quantify synapse density in 2D confocal images from fixed immunolabeled neural tissue sections

E. Rebollo[1,2], J. Boix-Fabrés[1,2], M.L. Arbones[1]*

[1] Molecular Biology Institute of Barcelona (IBMB-CSIC), Spain.

[2] Molecular Imaging Platform IBMB-PCB, Barcelona, Spain.

**Corresponding author: erabmc@ibmb.csic.es*

Summary

This chapter describes an *Image J/Fiji* automated macro approach to estimate synapse densities in 2D fluorescence confocal microscopy images. The main step-by-step imaging workflow is explained, including example macro language scripts that perform all steps automatically for multiple images. Such tool provides a straightforward method for exploratory synapse screenings where hundreds-to-thousands of images need to be analyzed in order to render significant statistical information. The method can be adapted to any particular set of images where fixed brain slices have been immunolabeled against validated presynaptic and postsynaptic markers.

Key Words

Synapse density, ImageJ Macro language, Puncta segmentation, Nuclei segmentation, Chromatic shift correction.

1. Introduction

Synapses are the points of communication between neurons whereby electrical or chemical signals are transmitted from one neuron to another. Anatomically, they are composed of a presynaptic and a postsynaptic terminal, respectively located in different neurons and separated by a gap called synaptic cleft, whose width ranges between 15 and 200 nm. The synaptic vesicles undergo calcium-dependent fusion with the presynaptic membrane, thus releasing their contents to the synaptic cleft, where they interact with receptors located at the postsynaptic membrane. These receptors are held in place by a vast protein scaffold, which contains almost 1500 proteins (reviewed by Harris and Weinberg [1]).

There is an increasing body of evidence suggesting that altered brain network activity underlays many of the most common neurological diseases [2,3]. To a large degree, these advances have been made possible thanks to the development of new image analysis methodologies that facilitate the quantification of synapse morphology, number and distribution in neural cells and tissues visualized under the light microscope [4-7]. Some of the main available quantification techniques focus on different aspects of brain connectivity, such as the distribution of individual synaptic proteins onto a cell of interest, the assessment of cell-to-cell contacts, and the relationship between different synaptic proteins (reviewed in [8]). The first two are usually confined to a small subset of cells, generally stained by a cell fill marker and a single synapse marker. The last, however, tackles the spatial proximity between a presynaptic and a postsynaptic marker, as a way of identifying *bona fide* synaptic sites.

Fluorescence confocal microscopy remains the workhorse technology for fixed tissue imaging up to 100 μm and, for the foreseeable future, will remain one of the dominant tools for studying synapses and their associated proteins in the central nervous system. On the one hand, many antibodies against different presynaptic and postsynaptic components are available, which have been extensively validated to locate synapsis [9]. On the other hand, modern confocal microscopes are nowadays readily available in most laboratories and many different customized routines can be easily established to perform multidimensional data acquisitions. At the light microscope, immunolabeled postsynaptic terminals generally appear as well-defined puncta, whereas presynaptic vesicles deliver a more irregularly shaped pattern (*see* Fig. 1). Addressing their spatial association under the confocal microscope often relies, due to the diffraction-limited resolution, on inferring methodologies such as colocalization [8,10]. However, correlation coefficients between presynaptic and postsynaptic markers tend to be low due to different reasons: i) Presynaptic markers mostly label all synapse vesicles; ii) immature postsynaptic puncta may not yet form synapses [11], and iii) many postsynaptic markers only label a subset of postsynaptic sites. A more direct approach consists in estimating the degree of apposition between the two markers by defining regions of interest (ROIs) around the postsynaptic puncta; the intensity of the presynaptic marker can then be measured within these ROIs in order to select those over an intensity threshold that qualify as synapses. On this line, masking puncta in 3D has probed a highly sensitive technique [5].

This chapter describes an open source automated procedure, developed as a *Fiji* [12] macro language script, to estimate synapse densities in 2D fluorescence confocal microscopy images from fixed brain slices immunostained against validated presynaptic and postsynaptic markers. Although such approach does not render accurate information on synapse distribution or morphology, its straightforwardness makes it suitable to address altered connectivity in exploratory and high content studies where, for instance, different genetic backgrounds or drug treatments are compared. It is known that subtle changes in the structure of neuronal circuits may pass unperceived and nevertheless have seriously detrimental effects [13]. Detecting such slight differences by estimating synapse densities requires systematic random sampling of brain regions involving a high number of images in order to obtain robust comparisons that yield enough statistical information. An automated image analysis procedure is therefore required to extract synapse densities from hundreds to thousands of images.

The image processing workflow (Fig. 2) includes two main segmentation steps. On the first one, the nuclear staining is used to retrieve the nuclear boundaries (Fig. 2B). Since random sampling of tissue sections will most likely generate images having different number of nuclei, this step is necessary to correct the actual area value that will be used to normalize the number of synapses per image (Fig. 2C). The second main segmentation step is focused on postsynaptic spot-like signal. Spot detection is a fundamental procedure for biologists in many imaging applications. One of the best point detection methods is the *Laplacian of Gaussian* (*LoG*) algorithm [14], which is used here to enhance the signal of the postsynaptic puncta. Once detected, further operations between the so obtained binary masks are used to eliminate unspecific nuclear signal particles (Fig. 2E). After segmentation, synapse detection is performed by a double discrimination test. First, low quality puncta are discarded based on the intensity density of the particles detected. This step helps eliminate out of focus puncta as well as weak off-target particles. Second, the remaining puncta are tested for the presence of the presynaptic label. These two steps are performed using global thresholds, so that the same criteria for synapsis discrimination is applied to all the images and conditions compared.

This chapter is developed using as example inhibitory GABAergic synapses [3,11,15], identified by the postsynaptic scaffold protein Gephyrin and the presynaptic vesicular GABA transporter VGAT. The protocol rationale can nevertheless be extended to prospective studies of excitatory glutamatergic synapses [1,16]. Sample images are given for both synapse types, together with their corresponding macros.

2. Materials

2.1 Images

1. 2D confocal images of mice brain slices immunolabeled against validated presynaptic and postsynaptic components and containing DNA staining. 4 sample images are provided with the chapter to test the protocol (Supp. Material and GitHub public repository [17]). The first two ("synapses_inh_01.lsm" and "synapses_inh_02.lsm") show GABAergic synapses as shown by immunostaining using antibodies (Abs) against the postsynaptic scaffold protein Gephyrin and the presynaptic vesicle component VGAT. The other two ("synapses_exc_01.lsm" and "synapses-exc_02.lsm") contain glutamatergic synapses by immunostaining using primary Abs against the postsynaptic scaffolding protein Hommer and the presynaptic vesicular glutamate transporter VGLUT. Alexa Fluor-488 and Alexa Fluor-568 were respectively used as secondary Abs. The DNA was stained using Dapi. *See* **Notes 1 and 2** for sample preparation and acquisition set up tips.
2. 2D confocal image of sub-diffraction fluorescent microspheres labelled with fluorophores that absorb/emit in ranges similar to those of the secondary Abs used in the biological samples and acquired in the same conditions (*see* **Note 3**). A sample image ("Beads.tif") is provided as supplemental material that corresponds to the same

example experiment.

2.2. Software and macros

1. *ImageJ* is an open source image processing and analysis platform [18] originally developed at the National Institutes of Health (Bethesda, Maryland, USA). In this chapter we use the *Fiji Life-line 22th December 2015* distribution [12]. This version can be downloaded at [19] and requires *Java 6*. The description of the *ImageJ* built-in macro functions used can be found at [20].
2. Macros (*see* **Note 4**) developed as indicated in this methods protocol. Two macros are provided as supplemental material: one to calculate the chromatic shift ("Chromatic_Shift_Calculator.ijm") and another that estimates the density of synapses in a set of n images ("Synapse_Counter.ijm"). Both these macros are available at [17].

3. Methods

This method first describes how to calculate the *xy* chromatic shift in a control image, to be used in the correction of the biological images if necessary. Second, the main step-by-step manual workflows aimed to estimate synapse density are provided, together with the rationale to build a customized macro that can be applied to any particular set of images. Finally, a section is dedicated on how to perform the analysis using the main macro provided in this chapter.

3.1. Calculating chromatic shift

In the lateral dimension, chromatic aberration provokes that different wavelengths are imaged at slightly shifted lateral (*xy*) positions. This section explains the manual steps necessary to obtain, from a 2D reference image of fluorescent beads, the *x* and *y* offset values that will later on be used to correct the biological images using translation (see Figure 3). Alternatively, the provided macro ("Chromatic_Shift_Calculator.ijm") can be used to automatically calculate the *x* and *y* offset values on the control image.

1. Open sample image "Beads.tif" in *Fiji* by [File > Open...] or by drag and drop of the file

on the *Fiji* bar (*see* **Note 5**). To follow the manual pipeline, go to step 2, otherwise jump to step 15 to use the provided macro.

2. Remove image calibration at [Image < Properties…]; set *pixel* as unit of length and use *1* for both pixel width and height.
3. Separate channels at [Images > Color > Split Channels].
4. Rename each independent channel by [Image > Rename…]; we will here name the images "red" and "green", according to their respective channel colors (*see* **Note 6**).
5. Select one channel (e.g. "red") and apply a *LoG* filter (*see* **Note 7**) at [Plugins > Feature Extraction > FeatureJ > FeatureJ Laplacian]; choose a smoothing scale radius of 2.
6. Convert the image to 8-bit at [Image > Type > 8-bit].
7. Threshold the image at [Image > Adjust > Threshold…]; choose an appropriate method (the *Default* method works fine for the provided images, otherwise *see* **Note 8**); unclick the option *Dark background* and hit *Apply*.
8. On the binary mask generated, apply the *Watershed* algorithm at [Process > Binary Watershed] to further separate contiguous objects (*see* **Note 9**).
9. Select *centroid* as parameter to measure at [Analyze > Set Measurements…].
10. Detect objects at [Analyze > Analyze Particles…], using the options *Exclude on edges* and *Add to manager* (*see* **Note 10**).
11. By clicking first *Deselect* and then *Measure* at the *ROI Manager* menu, the results table will open containing the centroid position of all the detected beads. This table can be saved as a text file to be opened later on a spreadsheet for further analysis.
12. Repeat steps 5 to 11 on the other channel (e.g. "green").
13. Once the centroid positions of all the beads have been obtained in both channels, the individual *x* and *y* shifts can be computed by subtracting the values from one channel respect to the other. Then, the *X* and *Y* average shifts will be used to translate one of the channels, thus correcting the chromatic shift (*see* **Note 11**).
14. To test the obtained *x* and *y* shift values, select the original channel "red" and translate it at [Image > Transform > Translate…]; introduce the *x* and *y* offset values in pixels and choose bilinear interpolation (*see* **Note 11**). The translated "red" channel and the

original "green" channel can now be merged at [Image > Color > Merge Channels…]. The result should be similar to that shown in Fig 3.

15. To perform the previous steps automatically, open the macro "Chromatic_Shift_Calculator.ijm" by dragging it to the Fiji bar. With the image opened and selected, hit the macro editor *Run* button. The macro will retrieve the average *x* and *y* shift values directly in pixels. These values will be used to correct the biological images in the next section.

3.2. Create macro to quantify synapse densities in a set of *n* images

The following subsections explain the main image processing steps (Fig. 2) in the form of manual step-by-step imaging protocols; as a general rule, the macro recorder can be kept opened during manual execution, so that preliminary scripts are created (revise **Note 4**). Further developed scripts are provided in each subsection, which sequentially executed, will shape the final macro. Too general or repetitive pipelines have been edited as user-defined functions (*see* **Note 12**). Such scripts can be executed by copy/paste in the *Fiji* script editor.

3.2.1. Channel preparation

At this initial step, the image needs to be manually open and each fluorescent channel must be prepared as an independent image with a short descriptive name that can be easily encoded in the macro script. The refined macro script that automates the steps below is provided in Fig. 4.

1. Open the sample image "Synapses_inh_01.tif" in Fiji by [File > Open...] or by drag and drop of the file on the *Fiji* bar. The image includes calibration.
2. Separate channels at [Images > Color > Split Channels].
3. Rename each independent channel by [Image > Rename…]; we will use the names "red", "green" and "blue" according to the channel´s colors (revise **Note 6**).

3.2.2. Nuclei boundaries segmentation

Nuclei segmentation, based on DNA staining, is widely used to retrieve nuclear boundaries in images from fixed tissues or cells. A pre-processing step is necessary in order to homogenize the background and smooth out the shape of the objects to be defined (*see* **Note 13**). Automatic intensity thresholding is then applied to detect the nuclear objects, which are thereafter converted into a binary image also called mask. Irregular DNA staining, due to variations in experimental conditions or cell cycle stage, may hinder the delimitation of the nuclear boundaries (*see* **Note 14**). In cases where a second nuclear signal exists, such as unspecific nuclear Gephyrin in this example, a segmentation strategy can be used that combines the preliminary masks obtained from both channels, so that a more complete nuclear mask is generated (*see* Fig. 5); small non-nuclear particles can be further filtered by size to create the ultimate mask strictly containing the segmented nuclei. The codes to perform this step are provided in Fig. 6.

1. Duplicate channel "blue" at [Image > Duplicate…] in order to leave the original image untouched; rename it at the *Duplicate* menu window so as to identify its purpose, e.g. "blueNucleiMask".
2. Apply a *Gaussian* filter at [Process > Filters > Gaussian Blur…], using a radius of 2.
3. Threshold the image at [Image > Adjust > Threshold…]; the best result for this example image is obtained using the *Li* method [21]. Set the background option so as to produce a binary image where objects are black and have an intensity value of 255. Hit *Apply*.
4. *Fill holes* at [Process > Binary > Fill Holes…]. This step is used to complete the nuclear mask by removing groups of background level pixels within the selected objects.
5. Perform steps 1 to 4 in channel "green"; we have renamed this image as "greenNucleiMask" (according to the script pipeline in Fig. 6); use a radius of 5 for the *Gaussian* filter.
6. Combine the created binary images "blueNucleiMask" and "greenNucleiMask" at [Process > Image Calculator…]; choose both images as *input 1* and 2 respectively in the popup menu and select *OR* as operator (*see* **Note 15**). By unclicking the option *Create*

new window the result will be generated on top of the first input image; keep this one and close the second input image "greenNucleiMask".

7. On the so modified image "blueNucleiMask", detect the nuclei using the command [Analyze > Analyze Particles…]; select a minimum size for nuclei discrimination (a value of 4 μm^2 is enough for the sample images) in order to discard small particles; untick the option *Exclude on edges* to keep the nuclei that touch an image edge; tick the option *Include holes* to avoid detecting regions within regions; the option *Add to manager* is not needed as the only purpose here is to generate a mask containing just the nuclei; alternatively, select *Mask* as show option and hit *OK*: a new binary image will appear containing only the desired objects; rename this image as e.g. "maskOfNuclei" (see Fig 5e) and close the previous mask.

3.2.3. Working area calculation

The number of synapses must be converted into a biologically meaningful value, such as synapse density, that can be compared between different conditions. Since the number of nuclei per image changes in a random tissue sampling, the actual working area must be calculated per image, excluding the nuclear regions. This subsection describes how to retrieve the actual working area value in μm^2. Fig. 7 provides a script that calculates the area value in μm^2 and stores it to be used at a later step.

1. At [Image > Adjust > Threshold…] threshold the *maskOfNuclei* image created in the previous section; select the *Default* method in the popup menu, and tick *Dark background* so that the objects are now selected; do not hit *Apply* yet.
2. Select *area* as measurement´s parameter at [Analyze > Set Measurements…].
3. At [Analyze > Analyze Particles…] tick the options *Display results* and *Clear results* and hit *OK;* the results window will pop up containing the measurement of the inverse nuclei area in μm^2.

3.2.4. Synaptic signal preprocessing

Before synapse detection, presynaptic ("red" channel) and postsynaptic ("green" channel) signals must be restored. This includes: i) Chromatic shift correction, to undo the mismatch between channels, and ii) background correction and filtering, depending on image quality. The script to perform this step is provided in Fig. 8.

1. Select channel "red" and apply translation at [Image > Transform > Translate…]; use the x and y offsets calculated in section 3.1 and select bilinear interpolations (revise **Note 11**).
2. For background correction use the *Rolling ball* algorithm at [Process > Subtract Background…]; a radius of 15 is fine for both "red" and "green" images.
3. Further filtering using [Process > Filters > Median] may be beneficial for noisy signals, as it is the case for the "red" channel; a radius of 2 is enough.

3.2.5. Postsynaptic puncta detection

Puncta segmentation is based on the dot-enhancing capabilities of the *LoG* algorithm (Fig 9). The idea behind this strategy is to detect as many dots as possible, so that they can be filtered afterwards based on a uniform quality criterion. Binary conversion is directly used to create a mask of the segmented puncta. Also, operations between binary images are explained to eliminate undesired nuclear spots. Further filtering using *size*, *circularity* and positioning criteria is necessary to eliminate artefactual or incomplete particles. The codes to perform this step are provided in Fig. 10.

1. Select channel "green" and apply the *LoG* filter at [Plugins > Feature Extraction > FeatureJ > FeatureJ Laplacian]; choose a smoothing scale radius of 2 (revise **Note 7**).
2. Convert the resulting image, "green Laplacian", into a binary image (*see* **Note 16**) at [Process > Binary > Convert to Mask].
3. Apply the *Watershed* algorithm at [Process > Binary > Watershed] in order to separate touching spots (revise **Note 9**).
4. Use the nuclear mask generated in section 3.2.2 ("maskOfNuclei") to remove nuclear spots. At [Process > Image Calculator…] select the new mask "green Laplacian" as

input 1 image, the previous "maskOfNuclei" as second input image and *Subtract* as operator. All spots located in the nuclear region will become white. The "maskOfNuclei" image can now be closed.

5. Detect particles at [Analyze > Analyze Particles…]. At the *Analyze particles* menu, select 2 μm^2 as upper *size* limit and 0.7 as lower *circularity* limit (revise **Note 10**). Tick the options *Add to manager* and *Exclude on edges* and hit *Apply*. The "green Laplacian" image can now be closed.

3.2.6. Synapse counting

In this subsection, the selected postsynaptic puncta that do not qualify as potential synaptic sites are discarded using a double discrimination step. First, the ROIs having low quality postsynaptic signal are eliminated; then, among the remaining ROIs, those showing no overlap with presynaptic signal are eliminated. Both discrimination steps are based on the *Integrated density* (*IntDen*) ROI content (*see* **Note 17**). The manual steps below only explain how to extract *IntDen* parameters from a list of ROIs when collated to a given image, in order to inspect for putative discrimination thresholds. However, the actual ROI discrimination necessary at this step cannot be done manually, since it would take too long. A script is provided in Fig. 11 containing two functions, one for threshold selection (*see* **Note 18** and Fig. 12) and another for ROI discrimination; both are first used to discard ROIs where the postsynaptic signal *IntDen* lies below the estimated cut off value; a second round deletes those ROIs where the presynaptic signal *IntDen* lies below the second estimated cut off value. The script can be run from the script editor to inspect for putative thresholds on the biological images.

1. Set *IntDen* as parameter of choice at [Analyze > Set Measurements…].
2. After selecting the image (e.g. "green") hit *Deselect* and then *Measure* at the *ROI Manager* window menu. The results window will pop up containing the integrated density measurements of all ROIs. The obtained values may be inspected in any other software in order to analyze the particle distribution and decide for a cutting threshold.

3.2.7. Verification image

An important issue in image analysis is the visual validation of the protocol. By drawing onto the image the ROIs that remain after synapses discrimination, the outcome of the protocol can be directly observed. This validation step is especially helpful when many images need to be processed, so that any possible outlier result can be tracked back. The steps below explain the manual pipeline to create a verification image, the refined script can be found in Fig. 13.

1. Create a composite image at [Image > Color > Merge Channels…] choosing the three independent channels "red", "green" and "blue" in their corresponding dialogs.
2. Convert the image into an RGB type at [Image > Type > RGB Color] so that the selected ROIs can be drawn in the desired color.
3. Set the foreground color to any desired by double clicking the *Color picker* tool at the tool bar.
4. At the *ROI Manager* window click *Deselect*, display the *more* menu and choose the option *draw*. All selections will be automatically stamped in the composite image.

3.2.8. Automate the protocol for *n* images

By sequentially executing the codes provided from 3.2.1 to 3.2.7 the main script pipeline is created, that allows to process one image. Additional programming features are necessary to be able to automatically process a set of *n* images and retrieve the final results table, where the image names, the areas, the number of selected puncta and the final number of synapsis will be stored. This subsection provides the minimum requirements for such macro automation. The scripts are provided in Fig. 14, where the different code blocks correspond to the following required actions:

1. Create dialog boxes for the user to choose the folder where the images are stored and the folder where the results will be saved.
2. Create an array containing the list of all images; this array will allow obtaining the total number of images later on.

3. Create a dialog for the user to select the threshold modulation cut off value.
4. Create four arrays to store the results (image name, area, number of selected puncta and final number of synapses) obtained per image analyzed.
5. Create a processing loop to apply the image processing workflow to all the images in the folder. This loop must contain: i) A command to open each image; ii) a variable to store its name, to be later on used to identify the verification image; iii) the main body code developed from 3.21 to 3.2.7; iv) a command to save the verification image to the results folder; v) the commands to fill the results' arrays and vi) the command to close all images/windows that remain open before the next loop round starts.
6. Create the table that will contain the results from all the images.
7. Fill the results table with the values stored in the results' arrays.
8. All the functions can be listed at the end of the code.

3.3. Quantify synapse density using the provided macro "Synapse_Counter.ijm".

This section provides the instructions to use the full macro, "Synapse_Counter.ijm", created as explained in section 3.2. For general purposes, we have incorporated two additional features: i) An option to segment the nuclei either based on one ("blue") or two ("blue+green") channels (see **Note 19**), and ii) a dialog box for the user to introduce the *x* and *y* parameters that will grant chromatic shift correction. The following steps explain how to use the macro, using as example the supplementary images provided with the chapter. For particular image sets, the specific parameters required for image preprocessing can be easily adjusted by changing the corresponding values in the variables that call the corresponding function arguments (see Figs 6, 8 and 10). The order of the channels can be also easily corrected in the corresponding function (see Fig 4).

1. Create a folder and copy all the raw images into it; the images should be 3-channel single images in a format that *Fiji* can directly open (*see* **Note 5**). Also, create a folder for the results to be stored.
2. Open the "Synapse_Counter.ijm" supplementary macro by drag and drop on the *Fiji* bar and hit *Run*.

3. In the pop-up browser, select the folder containing the images and hit *Open*. Then, select the folder where the results will be stored and hit *Open*.
4. A dialog box will pop up; introduce here the *x* and *y* shift correction parameters, estimated as explained in section 3.1. For the provided sample images these are x = 0.35 and y = 0.81.
5. Next, a new dialog will ask *In nuclear staining also present at green channel?*. To process the sample images "synapses_inh_01.lsm" and "synapses_inh_02.lsm" choose *Yes*; otherwise choose *No* if only the "blue" channel contains nuclear staining, which should be the option for images "synapses_exc_01.lsm" and "synapses-exc_02.lsm".
6. In the next pop up window two sliders appear, where the cut off values for particle discrimination of both puncta ("green") and vesicular transporter ("red") should be modulated. A range from 1 to 5 ca be selected, which means that 50% to 90% of the particles will be discriminated by the threshold, at 10% increments. For sample images 1 and 2, leave both values to 1; use both 5 for sample images 3 and 4. By hitting *OK* The macro will be executed.
7. Once the macro has finished analyzing all the images, the results folder will contain all the verification images where the synapse selection is drawn (*see* Fig. 11), plus a text file containing the measured parameters: area, number of puncta and number of synapses, all identified by image name. The synapse density can be estimated dividing the number of synapses by the corresponding working area.

4. Notes

1. For immunofluorescence vibratome sections were used, obtained from adult mice that were transcardially perfused with 4% paraformaldehyde. Abs against Gephyrin, Homer, VGAT and VGluT1 required antigen retrieval treatment; sections were treated by boiling for 10 min in sodium citrate buffer (2 mM citric acid monohydrate, 8 mM tri-sodium citrate dihydrate, pH 6.0) before blocking and incubation with primary Abs. Signal amplification was required for Abs against Gephyrin and Homer; after

washing primary Abs, sections were incubated with the corresponding biotinylated secondary Abs, washed and incubated with Alexa-488 conjugated streptavidin.

2. The images were acquired using a confocal LSM Zeiss780 microscope, equipped with an apochromatic 63x oil (NA=1.4) objective (see note 2). The three channels were sequentially acquired using a pinhole aperture of 0.8 AU. Pixel size was adjusted to 50x50 nm. Images (sampling units) of 25.95 x 25.95 micron (512x512 pixels) were randomly taken at the primary somatosensory cortex, within layer 4. Any multichannel confocal microscope is appropriate as long as it contains the laser lines necessary to excite the different dyes and/or fluorophores used. Line sequential acquisition of the different channels is recommended to be able to speed up the image capture while avoiding cross-talk emission contamination; nevertheless, when possible, simultaneous acquisition should be the fastest choice. Pinhole aperture can be minimized in order to reduce the thickness of the sections, as long as the detection efficiency is not severely affected; this should be easier in systems having high quantum efficiency and low noise detectors, such as those based in the Gallium Arsenide Phosphide technology. Pixel size should be at least adjusted according to the Nyquist-Shannon criteria, as a function of the objective numerical aperture (see [22]). A slight oversampling is nevertheless recommended in order to improve the intensity profile of individual spot-like signals.
3. Chromatic aberrations are wavelength-dependent artefacts that occur because the refractive index of every optical glass formulation varies with wavelength. In order to minimize the chromatic aberration: i) Apochromatic objectives are preferred, ii) coverslip thickness should be the one recommended for the particular lens used, unless the latter has the proper thickness correction collar, and iii) the refraction index of the mounting media should match that of the objective immersion medium. The samples used in this chapter were mounted using a mix of 97% 2,2′-Thiodiethanol (TDE, Sigma-Aldrich) and 3 % H2O in order to match the refraction index (IR = 1.52) of the objective immersion oil [23]. In any case, since a slight chromatic *xy* shift may be unavoidable, the use of beads is imperative in order to perform the final shift

correction before collating the channels. Microspheres should be acquired under the exact same detection windows and sampling conditions as the biological samples to be corrected. Any sub-diffraction beads (100-200nm) having labels similar to those used in the biological samples should be all right. Finally, to avoid changes in the orientation of the chromatic aberration, the scanning area should be centred and never rotated, so that the shift of the different images is identical.

4. *ImageJ* macro language (IJM) is a scripting language built-in into *ImageJ*, which allows writing simple programs called *macros* that automate series of image processing actions. By opening the macro recorder at [Plugins > Macros > Record…], the manually performed actions are sequentially recorded into an IJM preliminary script. Hitting the button *create* will open the recorded instructions into the script editor were additional editing can be made in order to make the code usable. For more information about macros see [24].
5. *ImageJ/Fiji* can open many different file formats along with their important metadata, via [File > Import > Bio-formats] or directly at the [File > Open] menu. The list of *ImageJ* supported file formats can be checked at [25]. Original formats will by default contain image calibration, it is important to take this into account when measurements are going to be performed, so that the resulting values can be interpreted in either pixels, micrometers or any other unit. Image calibration can be checked at [Image > Properties…]. In this chapter, for the chromatic shift calculation we prefer to remove image calibration first, as the final units to be used in image translation will be in pixels.
6. Images can be identified in the macro scripts by their names. In general, renaming images will help shortening the names and identifying the current image processing step, thus simplifying the script. The names used in the provided protocol are arbitrary. If other names are chosen, the corresponding code should be changed accordingly.
7. The *LoG* filter is used to enhance spot-like signals. It highlights regions of rapid intensity change, which are the pixels where the *Laplacian* function changes sign, also

called zero crossing points. By smoothing the image first with a *Gaussian* filter, the number of zero crossing points will change according to the inverse of the *Gaussian* radius, being the optimal smoothing scale radius related to the radius of the spots to be enhanced. *ImageJ* contains a *LoG* algorithm located within the *FeatureJ* plugin [26]. The operator retrieves a new 32-bit image where the enhanced dots appear dark on a light background. By default, the term "Laplacian" will be added to the image name in the resulting image window.

8. Thresholding is used to extract objects in an image based on their intensity, by setting one or two (upper and lower) cut-off value(s) that separate specific pixel intensities from each other. Most algorithms used for automatic thresholding are developed for a specific purpose or extraction problem. Thus, performance depends on the image content and quality, and the intended use of the pattern extracted. ImageJ provides 16 different methods to automatically compute global thresholds, all of which can be tested at once using the *Autothreshold* plugin at [Image > Adjust > Autothreshold].

9. The *Watershed* algorithm implemented in *ImageJ/Fiji* is a region-based segmentation approach that separates different objects that touch. It first calculates the peaks of local maxima of all objects, based on their euclidian distance map. It then dilates each peak as far as possible, either until the edge of the particle is reached, or the edge of another growing peak is found. For more information about *Watershed* algorithm implementations see [27].

10. The *Analyze particles* plugin counts and measures objects in binary or thresholded images, based on an algorithm that finds the edges. *Size* and *circularity* ranges are generally used to select the objects of interest that will form the binary mask. Size refers to area, in squared micron unless specified, and circularity refers to the calculation $4\pi\times[Area]/[Perimeter]^2$. The option *Exclude on edges* is used the discard incomplete objects that lie on the edge of the image. The option *Add to manager* will load all selected regions into the *ROI Manager* tool, where they can be further analyzed.

11. The *Translate* function at [Image > Transform > Translate…] moves the image in *x* and *y* by a set number of pixels. The input values must be pixels. If the *x* and *y* shifts were calculated on a calibrated image, they should be divided by the pixel size, which can be obtained at [Image > Properties…]. In the present pipeline, the shifts are directly calibrated in pixels. Since the resulting pixel shifts are not integers, a resampling method can be applied during translation to refine the correction; this can be done by selecting an interpolation method, either *bilinear* or *bicubic*, at the translate dialog box.
12. A used-defined function is a block of code that performs a general or repetitive task always in the same way. It can be passed values (defined as arguments) and it can return a value by means of the *return* statement. Functions can be installed or simply written at the end of the macro. They are called from the script by *name(arguments separated by coma);*. For more information about functions visit [28].
13. Uneven image backgrounds can be successfully restored using the rolling ball algorithm implemented in *ImageJ*, which derives from the *Rolling ball* algorithm described in Stanley Sternberg's article [29], modified to use a paraboloid of rotation instead of a ball. The *Rolling ball* radius is the radius of curvature of the paraboloid and should be at least as large as the radius of the largest object in the image that is not part of the background. The *ImageJ* implementation includes some additional code to avoid subtracting object corners (this choice is activated by selecting the *Sliding paraboloid* option at the *Subtract background* dialog box). More information can be found at [30]. Applying filters such as *Gaussian* or *Median* filters may also help denoising, and therefore facilitate object segmentation. Care need to be taken as to choose the filter that better preserves the object properties and details to be analyzed. For more tips on image denoising using filters see [31].
14. Most likely each particular set of images will need a customized approach for nuclei segmentation. Depending on the quality of the labeling, global thresholding methods may not work well enough. Adaptive thresholding may be a good alternative; this method changes the threshold dynamically over the image by computing local

parameters confined to smaller regions, which are more likely to have homogeneous illumination. The *Auto Local Threshold* plugin implemented in *ImageJ* and *Fiji* contains up to 9 different local thresholding methods. The *Try all* option allows for exploring how the different algorithms perform on a particular image. For more information see [32]. A second possibility (the one used in the provided pipeline) is to complement the nuclear mask using a second image where the nucleus is also stained and distinguishable. Note that, for each particular image data set, the nuclear segmentation strategy will have to be adapted and modified in the corresponding script.

15. The *Image Calculator* command performs arithmetic and logical operations between two images selected from popup menus, by applying one of 12 possible operators. Among these, the logical OR operator generates an image where either pixel contained in the source images in included. By selecting the option *Create a new window*, the source images are maintained; otherwise the result is overwritten on the first input image.
16. Direct binary conversion of the *Laplacian* filtered 32-bit image is chosen here. The result is identical to that produced by 8-bit conversion plus thresholding, using the *Default* method. More severe thresholding is not used at this step since the idea is to detect as many spots as possible.
17. Using mean intensity as cut off value helps disregard weak objects in general but may also remove biological meaningful structures. For instance, a tiny but very bright particle may contain less protein content than another particle that is bigger but less bright. Integrated density is calculated as area of the ROI multiplied by the mean intensity of the ROI; using this parameter as cut off value helps remove those detected foci where protein content is lower. The same argumentation is used to discriminate those puncta whose degree of apposition to the presynaptic vesicles lies below a threshold, based on the intensity density of the presynaptic signal.
18. After dot enhancing and segmentation, many detected ROIs will most likely contain low signal intensity values; a mixture of low protein content, out of focus signal or unspecific background most likely account for these low intensities. As a result, the

distribution of *Intensity density* values of the whole ROI population is skewed to the right (Fig 11 b, c). The parameter that better reflects the central tendency is skewed distributions in the median, which is used here as a first guess discrimination threshold. Such statistical approximation provides good results for large-scale comparative studies. However, this strategy may yield biased results when the 50% discarded population contains qualitatively different particles. The possibility to adjust the percentage of "good quality" ROIs has been implemented in the provided threshold selection function; a cutoff value from 1 to 5 modulates the percentage of selected particles, at intervals of 10%, from 50% to 90%. This facilitates the threshold determination to be used in the final comparative study.

19. Nuclei segmentation in the option using only the "blue" channel is based on the local thresholding strategy explained in **Note 14**. Any other strategy can nevertheless be easily implemented in the provided code to adapt it to any particular set of images.

Acknowledgements

We are grateful to Juan Arranz for immunofluorescence experiments and Manel Bosch for useful advice on macro programming. MLA provided samples and biological input. JBF contributed the chromatic shift calculation macro. ER performed the imaging, designed and programmed the macro analysis pipeline and wrote the paper. Images were acquired at the Molecular Imaging Platform IBMB (CSIC) with the support from the Spanish Ministry of Economy and Competitiveness and the European Regional development Fund CSIC13-4E-2065.

References

1. Harris KM, Weinberg RJ (2012) Ultrastructure of synapses in the mammalian brain. Cold Spring Harb Perspect Biol 4 (5). doi:10.1101/cshperspect.a005587

2. Rakic P, Bourgeois JP, Goldman-Rakic PS (1994) Synaptic development of the cerebral cortex: implications for learning, memory, and mental illness. Prog Brain Res 102:227-243. doi:10.1016/S0079-6123(08)60543-9

3. Henstridge CM, Pickett E, Spires-Jones TL (2016) Synaptic pathology: A shared mechanism in neurological disease. Ageing Res Rev 28:72-84. doi:10.1016/j.arr.2016.04.005

4. G. M, Heras J, Morales M, Romero A, Rubio J (2016) SynapCountJ: A Tool for Analyzing Synaptic Densities in Neurons. Proceedings of the 9th International Joint Conference on Biomedical Engineerng Systems and Technologies (BIOSTEC 2016) 2: BIOIMAGING:25-31

5. Fish K, Sweet R, Deo A, Lewis D (2008) An automated segmentation methodology for quantifying immunoreactive puncta number and fluorescence intensity in tissue sections. Brain Res 1240:62-72

6. Danielson E, Lee SH (2014) SynPAnal: software for rapid quantification of the density and intensity of protein puncta from fluorescence microscopy images of neurons. PLoS One 9 (12):e115298. doi:10.1371/journal.pone.0115298

7. Mokin M, Keifer J (2006) Quantitative analysis of immunofluorescent punctate staining of synaptically localized proteins using confocal microscopy and stereology. Journal of Neurosciences Methods (157):218-224

8. Hoon M, Sinha R, Okawa H (2017) Using Fluorescent Markers to Estimate Synaptic Connectivity In Situ. Methods in Molecular Biology 1538:293-320. doi:10.1007/978-1-4939-6688-2_20

9. Weiler NC, Collman F, Vogelstein JT, Burns R, Smith SJ (2014) Synaptic molecular imaging in spared and deprived columns of mouse barrel cortex with array tomography. Sci Data 1:140046. doi:10.1038/sdata.2014.46

10. Cordelières F, Bolte S (2014) Experimenters' guide to colocalization studies: finding a way through indicators and quantifiers, in practice. Methods in Cell Bioloiy 123:395-408

11. Dobie FA, Craig AM (2011) Inhibitory synapse dynamics: coordinated presynaptic and postsynaptic mobility and the major contribution of recycled vesicles to new synapse formation. J Neurosci 31 (29):10481-10493. doi:10.1523/JNEUROSCI.6023-10.2011

12. Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, Preibisch S, Rueder C, Saalfeld S, Schmid B, Tinevez J, White D, Hartenschtein V, Eliceiri K, Tomancak P, Cardona A (2012) Fiji: an open-source platform for biological-image analysis. Nature Methods (9):676–682

13. Dickstein D, Kabaso D, Rocher A, Luebke J, Wearne S, Hof P (2007) Changes in the structural complexity of the aged brain. Aging Cell 6 (3):275-284

14. Smal I, Loog M, Niessen W, Meijering E (2010) Quantitative comparison of spot detection methods in fluorescence microscopy. IEEE Trans Med Imaging 29 (2):282-301. doi:10.1109/TMI.2009.2025127

15. Sassoe-Pognetto M, Panzanelli P, Sieghart W, Fritschy JM (2000) Colocalization of multiple GABA(A) receptor subtypes with gephyrin at postsynaptic sites. J Comp Neurol 420 (4):481-498

16. Arranz J, Balducci E, Arato K, Sanchez-Elexpuru G, Najas S, Parras A, Rebollo E, Pijuan I, Erb I, Verde G, Sahun I, Barallobre M, Lucas J, Sanchez M, de la Luna S, Arbones M (2019) Impaired development of neocortical circuits contributes to the neurological alterations in DYRK1A haploinsufficiency syndrome. Neurobiology of Disease In press

17. Github website MI, Synapse Counter https://github.com/MolecularImagingPlatformIBMB/Synapse_Counter.git.

18. Schneider CA, Rasband WS, KW E (2102) NIH Image to ImageJ: 25 years of image analysis. Nature Methods 9 (7):671-675

19. Fiji download website https://imagej.net/Fiji/Downloads). .

20. ImageJ macro functions website https://imagej.nih.gov/ij/developer/macro/functions.html.

21. Li CH, Tam PKS (1998.) An iterative algorithm for minimum cross entropy thresholding. Pattern Recognition Letters 19 (8):771-776

22. Pawley J (2000) The 39 steps: a cautionary tale of quantitative 3-D fluorescence microscopy. Biotechniques 28 (5):884-886, 888

23. Staudt T, Lang MC, Medda R, Engelhardt J, Hell SW (2007) 2,2'-thiodiethanol: a new water soluble mounting medium for high resolution optical microscopy. Microsc Res Tech 70 (1):1-9. doi:10.1002/jemt.20396

24. ImageJ macro programming https://imagej.nih.gov/ij/docs/guide/146-14.html.

25. formats Is https://docs.openmicroscopy.org/bio-formats/5.7.3/supported-formats.html.

26. FeatureJ http://imagescience.org/meijering/software/featurej/.

27. Roerdink J, Meijster A (2001) The Watershed Transform: Definitions, Algorithms and Parallelization Strategies. Fundamenta Informaticae 41 (187-228)

28. functions Iu-d https://imagej.nih.gov/ij/developer/macro/macros.html#functions.

29. Sternberg S (1983) Biomedical Image Processing. Computer 16 (1):22-34

30. ImageJ's Subtract Background
https://imagej.nih.gov/ij/developer/api/ij/plugin/filter/BackgroundSubtracter.html.

31. Singh I, Neeru N (2014) Performance Comparison of Various Image Denoising Filters Under Spatial Domain. International Journal of Computer Applications 96 (19):21-30

32. Auto Local Threshold https://imagej.net/Auto_Local_Threshold.

FIGURES

Figure 1. Synapses identification in confocal microscopy. (A) Confocal microscopy image showing inhibitory synapses as stained with antibodies against the postsynaptic scaffold protein Gephyrin (green arrowhead) and the presynaptic vesicle component V-GAT (red arrowhead). The white line indicates the ROI used for the intensity plot shown in panel B. Bar is 1 μm. (B) Graph showing the intensity plots of the presynaptic (red) and postsynaptic (green) signals along the line drawn in panel A. The distance between the two peaks lies beyond the resolution limit of the confocal microscope. The overlay region is shown in yellow.

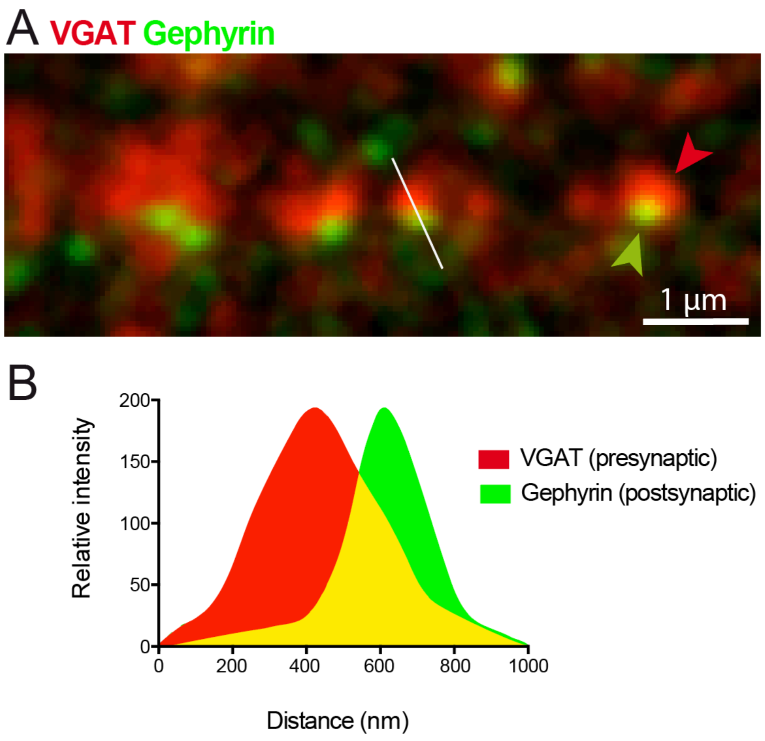
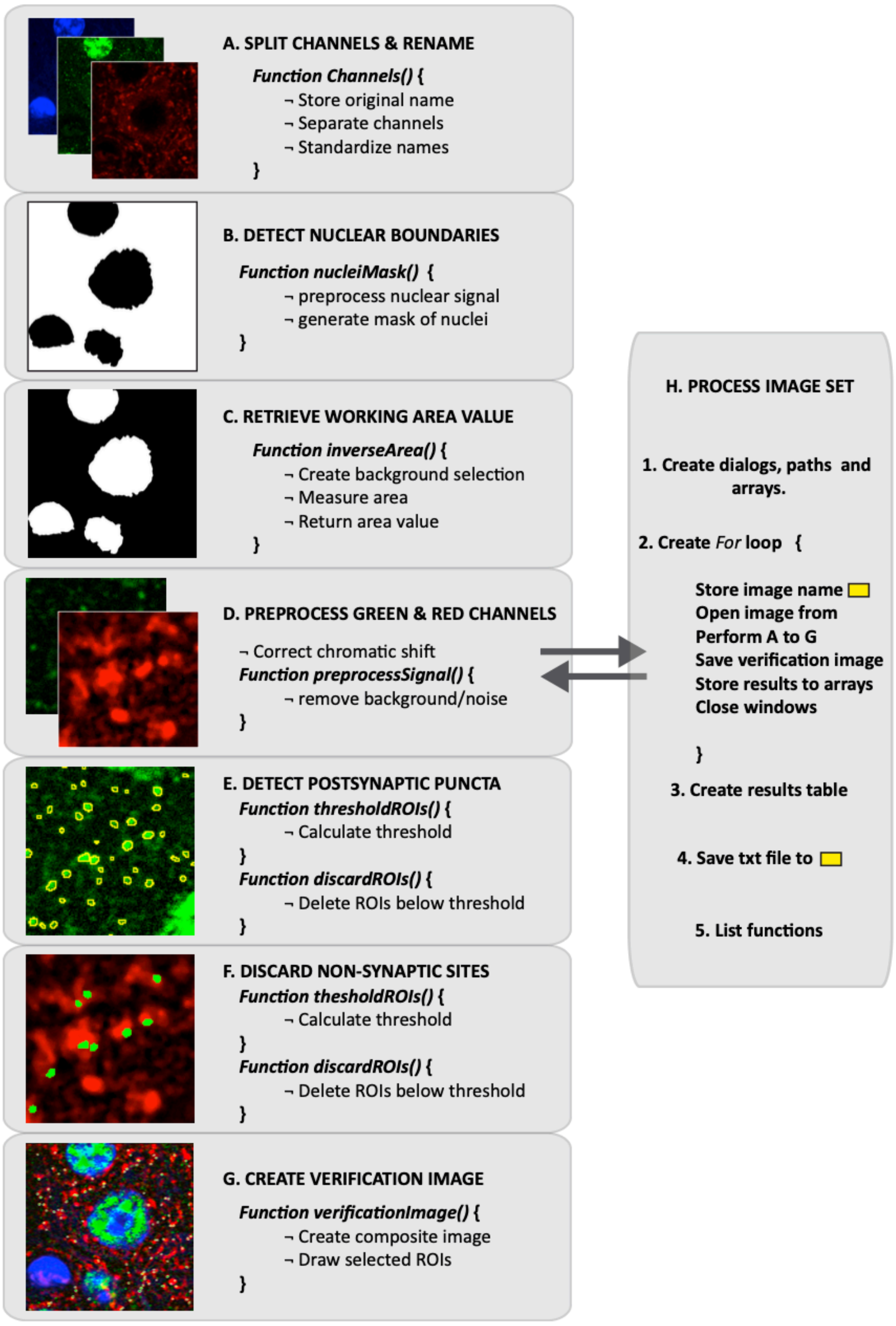


Figure 2. Image-processing workflow. Schematic showing the main steps that compose the imaging pipeline. (A-G) Main steps necessary to count synapses in a single image. (H) Steps 1-5 represent the automation steps required to apply the pipeline to a set of images.



```
A. SPLIT CHANNELS & RENAME
  Function Channels() {
      ¬ Store original name
      ¬ Separate channels
      ¬ Standardize names
  }

B. DETECT NUCLEAR BOUNDARIES
  Function nucleiMask() {
      ¬ preprocess nuclear signal
      ¬ generate mask of nuclei
  }

C. RETRIEVE WORKING AREA VALUE
  Function inverseArea() {
      ¬ Create background selection
      ¬ Measure area
      ¬ Return area value
  }

D. PREPROCESS GREEN & RED CHANNELS
  ¬ Correct chromatic shift
  Function preprocessSignal() {
      ¬ remove background/noise
  }

E. DETECT POSTSYNAPTIC PUNCTA
  Function thresholdROIs() {
      ¬ Calculate threshold
  }
  Function discardROIs() {
      ¬ Delete ROIs below threshold
  }

F. DISCARD NON-SYNAPTIC SITES
  Function thesholdROIs() {
      ¬ Calculate threshold
  }
  Function discardROIs() {
      ¬ Delete ROIs below threshold
  }

G. CREATE VERIFICATION IMAGE
  Function verificationImage() {
      ¬ Create composite image
      ¬ Draw selected ROIs
  }

H. PROCESS IMAGE SET

1. Create dialogs, paths and arrays.

2. Create For loop {

      Store image name 🟨
      Open image from
      Perform A to G
      Save verification image
      Store results to arrays
      Close windows

   }
3. Create results table

4. Save txt file to 🟨

5. List functions
```

Figure 3. Chromatic shift correction. (a) Confocal microscopy image showing one 100 nm microsphere fluorescing in red and green (b) Overlay of the two channels after signal segmentation; notice how the green signal is diagonally shifted downwards and towards the right respect to the red. (c) The red channel has been translated in *x* and *y*, using bilinear interpolation, thus correcting the mismatch. The bar indicates 350 nm

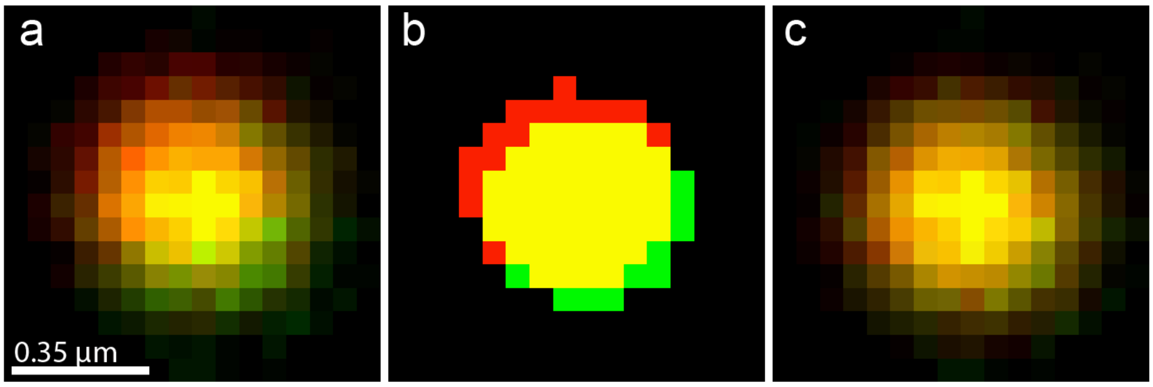


Figure 4. Prepare channels. The script executes a user-defined function called *channels();* the function is described between curly brackets: first, the variable *rawName* stores the original name of the file; the remaining commands split the channels and rename them. At this point the code still requires the image to be opened manually. To execute this code, copy it in the *Fiji* script editor and hit *Run*.

```
//SEPARATE CHANNELS & STANDARDIZE NAMES
channels();
function channels(){
            //Store original file name as a variable
            rawName = getTitle();
            //Separate channels
            run("Split Channels");
            //Rename channels:
            selectWindow("C3-"+rawName);
            rename("blue");
            selectWindow("C2-"+rawName);
            rename("green");
            selectWindow("C1-"+rawName);
            rename("red");
```

Figure 5. Nuclei segmentation using two nuclear signals. (a-a′) Raw "blue" (Dapi) and "green" (Gephyrin) channels respectively. (b-b′) Preprocessed "blue" and "green" channels, after background subtraction and filtering. (c-c′) Binary masks obtained from b and b′. (d) Preliminary mask formed by the sum of c and c′. (e) Final mask containing only the nuclei, including those touching the image edges; small particles (arrowhead) have been filtered by size.

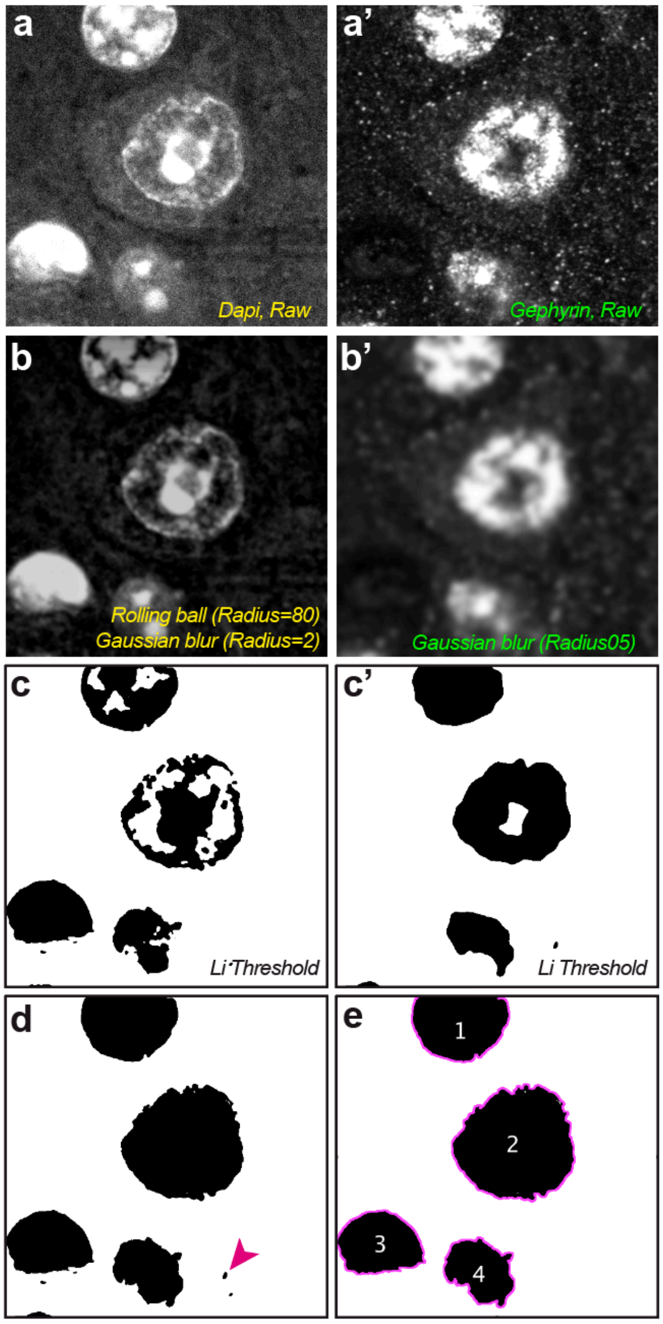


Figure 6. Detect nuclear boundaries. The function *nucleiMask(image, gaussianRadius)* is defined at the end of the script encircled in curly brackets; it duplicates an image, identified by its name through the first argument; then, it applies a *Gaussian* filter using the radius provided as a second argument; finally, it thresholds the image using the *Li* method and fill the binary mask holes to complete the objects. From beginning to end, the script code first applies this function to channels "blue" and "green" using the specified arguments; then, the two masks generated are combined and further filtered to create the ultimate mask containing only the nuclei. To execute this code, copy it in the *Fiji* script editor and hit *Run*.

```
//DETECT NUCLEAR BOUNDARIES
//Preprocess blue & green channels and convert to mask
nucleiMask("blue", 2);
nucleiMask("green", 2);
//Generate complete mask of nuclei
imageCalculator("OR", "blueNucleiMask", "greenNucleiMask");
run("Analyze Particles...", "size=4.00-Infinity show=Masks clear include");
rename("maskOfNuclei");
//Close extra windows
selectWindow("blueNucleiMask ");
run("Close");
selectWindow("greenNucleiMask ");
run("Close");

function nucleiMask(image, gaussianRadius) {
            selectWindow(image);
            run("Duplicate...", "title="+name+"NucleiMask");
            run("Gaussian Blur...", "sigma=gaussianRadius");
            setAutoThreshold("Li dark");
            run("Convert to Mask");
            run("Fill Holes");
}
```

Figure 7. Calculate working area value in µm^2. The function *inverseArea(image)* is defined, its code between curly brackets. This function takes a binary image, specified by the name in the argument, thresholds the background and calculates its area; then, it returns the area value (by means of the return command) to be stored as a variable called *area*, from which the function is executed on the first line. To execute this code, copy it in the *Fiji* script editor and hit *Run*.

```
//RETRIEVE & STORE WORKING AREA VALUE
area = inverseArea(image);
function inverseArea(image) {
            selectWindow(image);
            setAutoThreshold("Default dark");
            run("Set Measurements...", "area redirect=None decimal=2");
            run("Analyze Particles...", "display clear");
            area = getResult("Area", 0);
            return area;
}
```

Figure 8. Preprocess pre and postsynaptic signals. First, translation is applied to one of the channels to correct for chromatic mismatch. Then, a user-defined function called *preprocessSignal(image, rollingRadius, medianRadius)* is used to remove noise in both images; the image is selected by name using the first argument, and the parameters for the *Rolling ball* algorithm and the *Median* filter are selected using the other two arguments. To execute this code, copy it in the *Fiji* script editor and hit *Run*.

```
//PREPROCESS PRE & POSTSYNAPTIC SIGNALS
// Correct chromatic shift
selectWindow("red");
run("Translate...", "x=0.35 y= 0.81 interpolation=Bilinear");
//Preprocess signals
preprocessSignal("green", 15, 0);
preprocessSignal("red", 15, 2);

function preprocessSignal(image, rollingRadius, medianRadius) {
            selectWindow(image);
            run("Subtract Background...", "rolling=”+rollingRadius sliding);
            run("Median...", "radius=”+medianRadius);
}
```

Figure 9. Puncta segmentation. The upper panel contains an example fragment of the channel image containing postsynaptic puncta (a); the *fire* LUT has been applied for better pucta visualization. The background is first removed by a *Rolling ball* algorithm (b); then, the *LoG* transformation is applied (c); the resulting image is binary converted (d) and the *Whatershed* algorithm is further used to separate contiguous objects (e, arrowhead); finally, the segmented puncta are detected *via* the *Analyze particles* menu. The yellow outlines represent the ROI selections, drawn onto the original image. The lower panel shows the mask containing the postsynaptic puncta (g) and the mask containing the nuclei (h); both masks are combined to deliver the final mask containing only the cytoplasmic puncta (i).

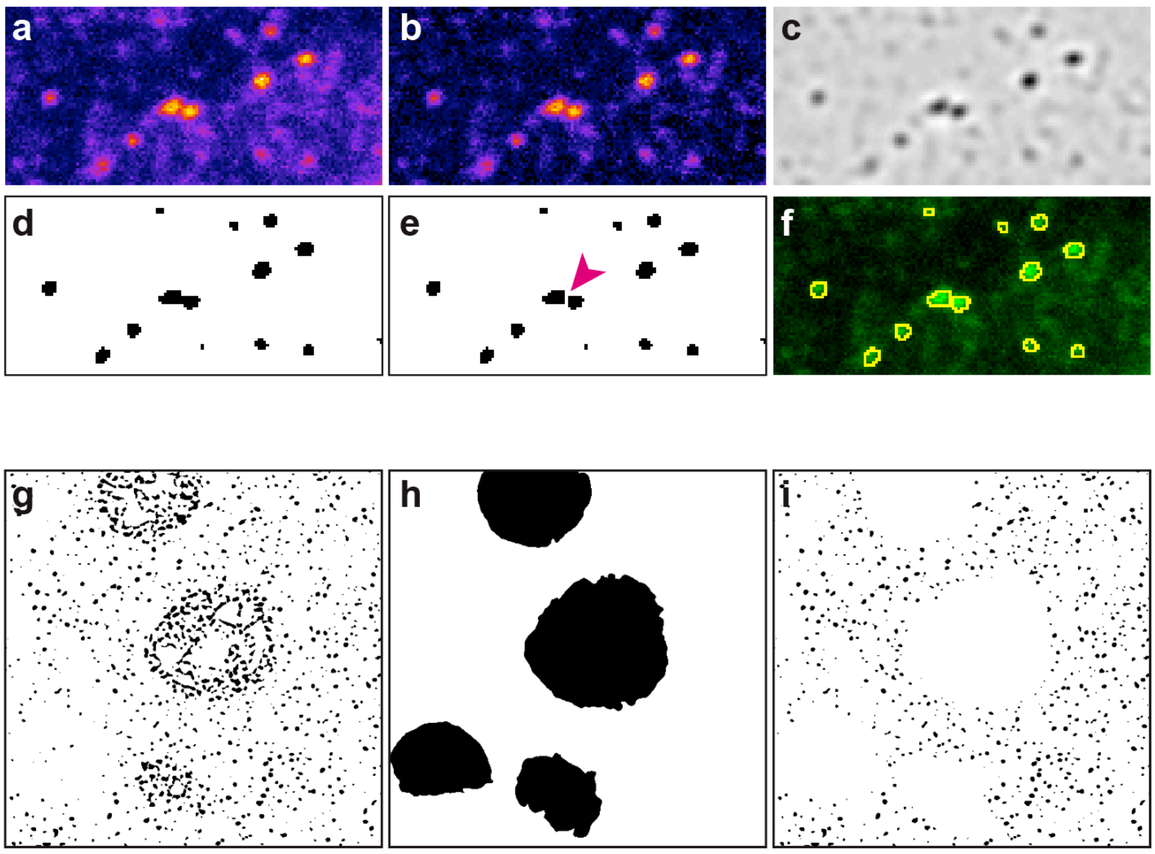


Figure 10. Detect post-synaptic puncta. The function *detectPuncta(image1, image2, logRadius)* is described between curly brackets; the two first arguments call the input images by name, being the third the radius to be specified for the *LoG* transformation. Within the function, the *LoG* algorithm is applied to enhance spot-like signals. After binary conversion, the *Whatershed* algorithm is used to further separate contiguous dots. The image “maskOfNuclei”, previously created, is subtracted to remove nuclear spots. Automatic detection is utilized to filter the puncta and collect their shapes to the *ROI Manager*. The previous masks are then closed, and the detected foci kept in the *ROI Manager* to be filtered in the next step. To execute this code, copy it in the *Fiji* script editor and hit *Run*.

```
//DETECT POSTSYNAPTIC PUNCTA
detectPuncta( “green”, “maskOfNuclei” , 2);

function detectPuncta(image1, image2, logRadius) {
          selectWindow(image1);
          run("FeatureJ Laplacian", "compute smoothing=" +logRadius);
          setOption("BlackBackground", false);
          run("Convert to Mask");
          run("Watershed");
          //Remove nuclear spots
          imageCalculator("Subtract", image1+"  Laplacian", image2);
          roiManager("reset");
          run("Analyze Particles...", "size=0.00-2 circularity=0.7-1.00 exclude clear add");
          //Close masks
          selectWindow(image1+"  Laplacian");
          run("Close");
          selectWindow(image2);
          run("Close");
}
```

Figure 11. Discard non-synaptic puncta and retrieve synapse number. The function *thresholdROIs(image, cutoff)* extracts the median value of the *IntDen* ROI distribution on a given image, and returns the threshold value corrected by a cutoff value, provided as argument; such cutoff is specified from 1 to 5, which set the threshold low threshold at 50% to 90% of the distribution respectively, at intermediate increments of 10%; the function *discardROIs(image, threshold)* discards those ROIs whose *IntDen* value on a given image (first argument) lies below a certain threshold (second argument); the function returns the number of ROIs that remain in the *ROI Manager* after its execution. The whole code performs the following tasks: 1) It calculates the median of the postsynaptic *IntDen* ROI distribution (*thresholdGreen*), 2) It uses the obtained value as threshold to discard low quality puncta and counts the number of remaining puncta (*noPuncta*); 3) It then calculates the median of the presynaptic *IntDen remaining ROI* distribution (*thresholdRed*); 4) Finally, it uses this second threshold to discard those ROIs that do not qualify as synaptic sites, and retrieves the final number of synapses (*noSynapses*). To execute this code, copy it in the Fiji script editor and hit *Run*.

```
//CALCULATE THRESHOLD FOR PUNCTA DISCRIMINATION
thresholdGreen= thresholdROIs("green", 1);
//DISCARD LOW QUALITY PUNCTA & COUNT SELECTED PUNCTA
noPuncta = discardROIs("green", thresholdGreen);
//CALCULATE THRESHOLD FOR SYNAPSE DISCRIMINATION
thresholdRed= thresholdROIs("red", 1);
// DISCARD NON SYNAPTIC PUNCTA & COUNT SYNAPSES
noSynapses = discardROIs("red", thresholdRed);

function thesholdROIs(image, cutoff) {
        selectWindow(image);
        run("Set Measurements...", " integrated redirect=None decimal=2");
        roiManager("deselect");
        roiManager("measure");
        //Create IntDen array & obtain median
        noRois = roiManager("count");
        intDensities = newArray(noRois);
        for (j=0; j<noRois; j++){
                intDensities [j] =  getResult("IntDen", j);
                }
        Array.sort(intDensities);
        lengthArray = lengthOf(intDensities);
        if (lengthArray%2==0) {lengthArray = lengthArray+1}
        t = round(lengthArray*(0.6 - 0.1*cutoff));
        threshold = intDensities[t] ;
        return threshold;
}

function discardROIs(image, threshold) {
        selectWindow(image);
        run("Select All");
        run("Set Measurements...", "mean integrated redirect=None decimal=2");
        //Discard ROIs below threshold value
        noRois = roiManager("count");
        for (j=0; j<noRois; j++){
                roiManager("select", j);
                roiManager("measure");
                value = getResult("IntDen", 0);
                if (value < threshold) {
                roiManager("delete");
                j--;
                noRois--;
                }
        run("Clear Results");
        }
        return noRois;
}
```

Figure 12. Synapses threshold calibration and counting. (a) Postsynaptic (Gephyrin) channel showing all puncta (yellow circles) detected after segmentation; the *fire* LUT has been applied to the image in order to better show low intensity signals. (b) The upper graph shows the postsynaptic signal *IntDen* distribution histogram of all puncta ROIs detected in panel A; (c) the lower graph shows the presynaptic signal *IntDen* distribution histogram of the ROIs that remain after puncta discrimination; the main descriptive statistics, mean and median, are shown in each graph. The middle panel contains an inset of the same image (depicted by the white square in a) taken as example to show synapse discrimination; (d) is the original three channels image inset; (e) shows all the ROIs detected during puncta segmentation (white circles); (f) depicts the ROIs that remain after puncta discrimination, using as threshold the median value of the green *IntDen* distribution; the yellow arrowheads depict signal spots that have been discarded during this discrimination step; (g) contains the final ROIs that have been selected as synapses after the second round of discrimination, using as threshold the median value of the red *IntDen* ROI distribution; the arrowheads depict spots that do not overlap with red signal. The lower panel contains the output verification images from the supplementary sample images "Synapses_inh_01" (h) and "Synapses_exc_02" (i), corresponding to inhibitory and excitatory synapse counts respectively, processed as explained in section 3.3.

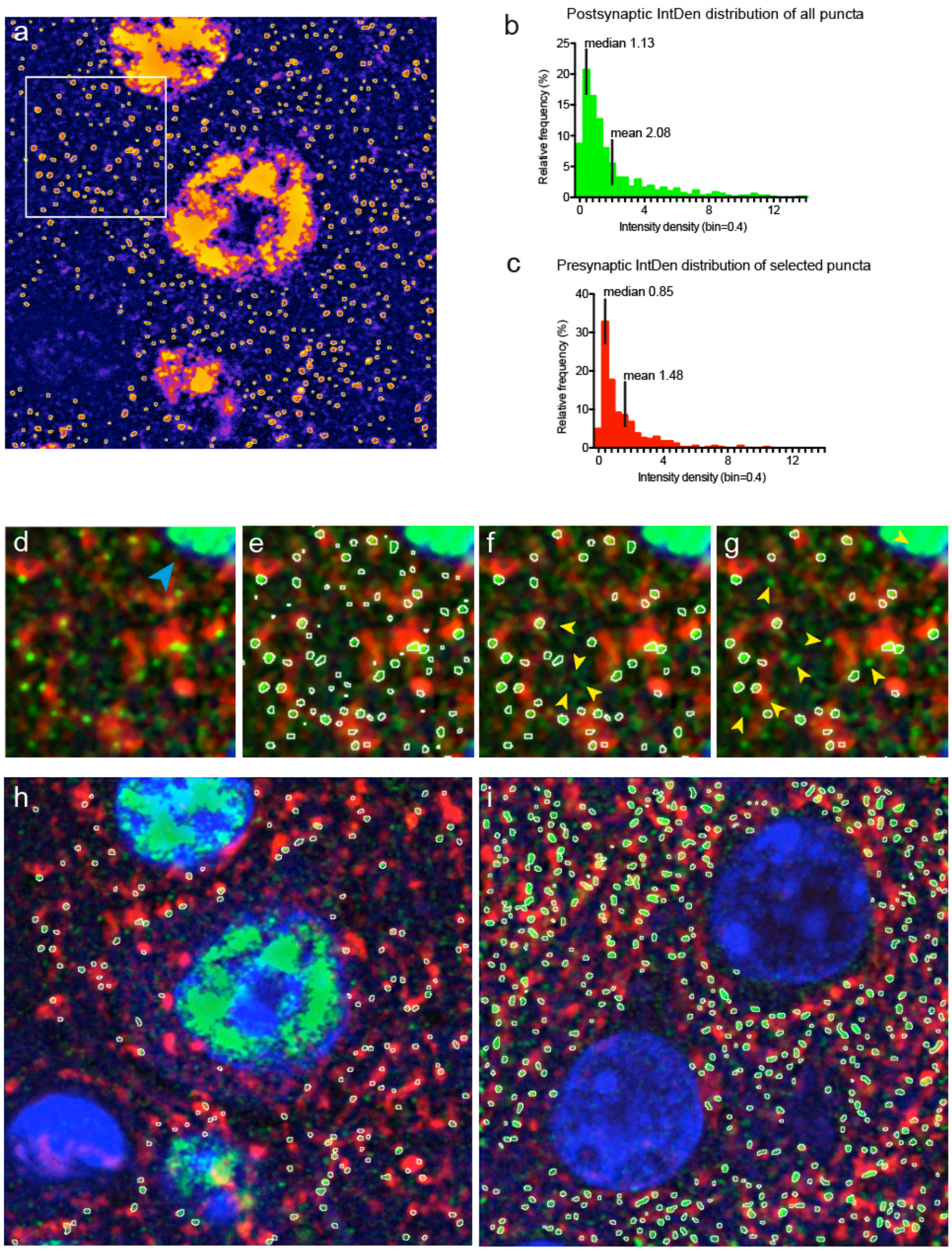
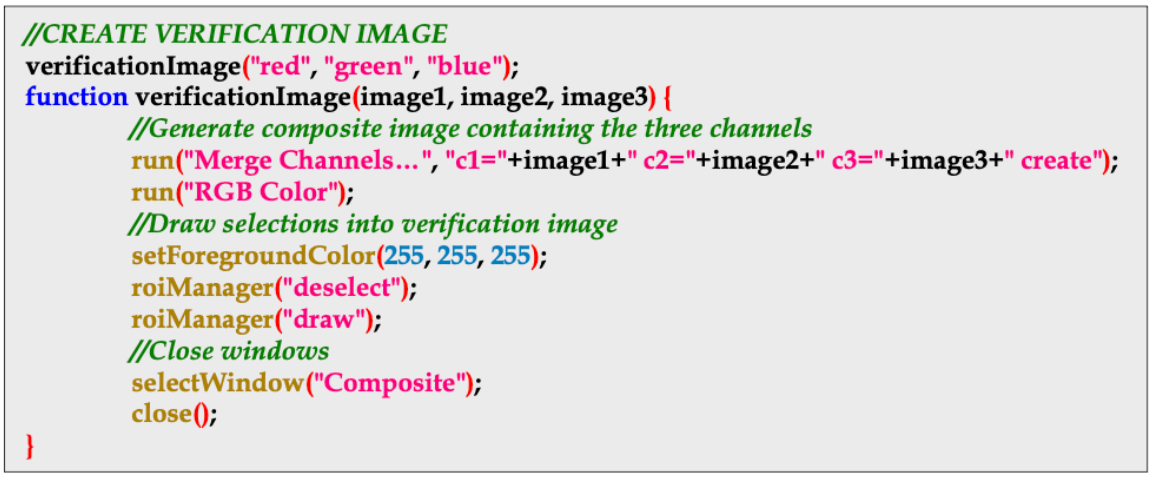

a
b
Postsynaptic IntDen distribution of all puncta
median 1.13
mean 2.08
Relative frequency (%)
Intensity density (bin=0.4)
c
Presynaptic IntDen distribution of selected puncta
median 0.85
mean 1.48
Relative frequency (%)
Intensity density (bin=0.4)
d
e
f
g
h
i

Figure 13. Create verification image. The function *verificationImage(image1, image2, image3)* creates a composite image by merging three channels, provided as arguments; then it draws, onto the RGB converted image, all the ROIs contained in the *ROI Manager*, using white color. To execute this code, copy it in the Fiji script editor and hit *Run*.



```
//CREATE VERIFICATION IMAGE
verificationImage("red", "green", "blue");
function verificationImage(image1, image2, image3) {
            //Generate composite image containing the three channels
            run("Merge Channels...", "c1="+image1+" c2="+image2+" c3="+image3+" create");
            run("RGB Color");
            //Draw selections into verification image
            setForegroundColor(255, 255, 255);
            roiManager("deselect");
            roiManager("draw");
            //Close windows
            selectWindow("Composite");
            close();
}
```

Figure 14. Minimum codes to analyze n images and retrieve results. The codes below will allow automating the main image-processing pipeline for a set of *n* images; the user-defined functions are not included in the figure, and should be listed at the end of the code.

```
//CALL ORIGEN & DESTINATION FOLDERS
imagesFolder = getDirectory("Choose directory containing images");
resultsFolder = getDirectory("Choose directory to save results");

//CREATE ARRAY LISTING ALL IMAGES IN THE FOLDER
list = getFileList(imagesFolder);

//CREATE DIALOG FOR THRESHOLD MODULATION
Dialog.create("Indicate cut-off value for puncta assingment:");
Dialog.addMessage("1=50% 2=60% 3 =70% 4=80% 5=90%");
Dialog.addSlider("Green cut-off: ", 1, 5, 1);
Dialog.addSlider("Red cut-off: ", 1, 5, 1);
Dialog.show;
cutoffG = Dialog.getNumber();
cutoffR = Dialog.getNumber();

//CREATE ARRAYS TO STORE RESULTS
names = newArray(list.length);
areas = newArray(list.length);
puncta = newArray(list.length);
synapses = newArray(list.length);

//CREATE LOOP TO ANALYSE ALL IMAGES
for(i=0; i<list.length; i++){
        showProgress(i+1, list.length);
        //Open image and get name
        open(imagesFolder+list[i]);
        name = File.nameWithoutExtension;
        // Add main code here  (3.2.1. to 3.2.7)
        // Save verification image
        saveAs("TIFF", resultsFolder+name+"_processed.tif");
        // Close windows
        close("\\Others");
        roiManager("Reset);
        // Fill arrays
        names[i] = name;
        areas[i] = area;
        puncta[i] = noPuncta;
        synapses[i] = noSynapses;
}

//CREATE RESULTS TABLE
run("Table...", "name=[Results] width=400 height=300 menu");
print("[Results]", "\\Headings:"+"Image\t Area\t No Puncta\t No Synapses");
for(i=0; i<list.length; i++){
        print("[Results]",""+names[i]+"\t"+areas[i]+ "\t"+puncta[i] "\t"+synapses[i]);
}

//SAVE RESULTS TEXT FILE IN RESULTS FOLDER
selectWindow("Results");
saveAs("Text", resultsFolder+"Results.txt");
run("Close");

//LIST ALL FUCNTIONS BELOW
```