SynapCountJ: A Validated Tool for Analyzing Synaptic Densities in Neurons

Gadea Mata[1(✉)], Germán Cuesto[3], Jónathan Heras[1], Miguel Morales[2], Ana Romero[1], and Julio Rubio[1]

[1] Departamento de Matemáticas y Computación, Universidad de La Rioja, Logroño, Spain
{gadea.mata,jonathan.heras,ana.romero,julio.rubio}@unirioja.es
[2] Institut de Neurociéncies, Universitat Autónoma de Barcelona, Barcelona, Spain
miguelmorales@spineup.es
[3] Facultad de Ciencias de la Salud, Centro de Investigaciones Biomédicas de Canarias (CIBICAN) Instituto de Tecnologías Biomédicas (ITB), San Cristóbal de La Laguna, Spain
germancuesto@gmail.com

Abstract. The quantification of synapses is instrumental to measure the evolution of synaptic densities of neurons under the effect of some physiological conditions, neuronal diseases or even drug treatments. However, the manual quantification of synapses is a tedious, error-prone, time-consuming and subjective task; therefore, reliable tools that might automate this process are desirable. In this paper, we present SynapCountJ, an ImageJ plugin, that can measure synaptic density of individual neurons obtained by immunofluorescence techniques, and also can be applied for batch processing of neurons that have been obtained in the same experiment or using the same setting. The procedure to quantify synapses implemented in SynapCountJ is based on the colocalization of three images of the same neuron (the neuron marked with two antibody markers and the structure of the neuron) and is inspired by methods coming from Computational Algebraic Topology. SynapCountJ provides a procedure to semi-automatically quantify the number of synapses of neuron cultures; as a result, the time required for such an analysis is greatly reduced. The computations performed by SynapCountJ have been validated by comparing the results with those of a formally verified algorithm (implemented in a different system).

1 Introduction

Synapses are the points of connection between neurons, and they are dynamic structures subject to a continuous process of formation and elimination. Pathological conditions, such as the Alzheimer disease, have been related to synapse loss associated with memory impairments. Hence, the possibility of changing the number of

This work was supported by the Ministerio de Economía y Competitividad projects [MTM2013-41775-P, MTM2014-54151-P, BFU2010-17537]. G. Mata was also supported by a PhD grant awarded by the University of La Rioja [FPI-UR-13].

A. Fred and H. Gamboa (Eds.): BIOSTEC 2016, CCIS 690, pp. 41–55, 2017.
DOI: 10.1007/978-3-319-54717-6_3

synapses may be an important asset to treat neurological diseases [34]. To this aim, it is necessary to determine the evolution of synaptic densities of neurons under the effect of some physiological conditions, neuronal diseases or even drug treatments.

The procedure to quantify synaptic density of a neuron is usually based on the colocalization between the signals generated by two antibodies [6]. Namely, neuron cultures are permeabilized and treated with two different primary markers (for instance, bassoon and synapsin). These antibodies recognize specifically two presynaptic structures. Then, it is necessary a secondary antibody couple attached to different fluorochromes (for instance red and green; note, that several other combinations of color are possible) making these two synaptic proteins visible under the fluorescence microscope. The two markers are photographed in two gray-scale images; that, in turn, are overlapped using respectively the red and green channels. In the resultant image, the yellow points (colocalization of the code channels) are the candidates to be the synapses.

The final step in the above procedure is the selection of the yellow points that are localized either on the dendrites of the neuron or adjacent to them. Tools like MetaMorph [10] or ImageJ [32]—a Java platform for image processing that can be easily extended by means of plugins—can be used to manually count the number of synapses; however, such a manual quantification is a tedious, time-consuming, error-prone, and subjective task; hence, reliable tools that might automate this process are desirable. In this paper, we present *SynapCountJ*, an ImageJ plugin, that semi-automatically quantifies synapses and synaptic densities in neuron cultures. The program is based on Algebraic Topology techniques and has been validated by comparing some intermediate results with those of Kenzo [12], a (partially) formally verified program.

2 Methodology

SynapCountJ supports two execution modes: individual treatment of a neuron and batch processing—the workflow of both modes is provided in Fig. 1.

2.1 Individual Treatment of a Neuron

The input of SynapCountJ in this execution mode are two images of a neuron marked with two antibodies (an image per antibody), see Fig. 2. SynapCountJ is able to read tiff (a standard format for biological images) and lif files (obtained from Leica confocal microscopes)—the latter requires the Bio-Formats plugin [19]. The following steps are applied to quantify the number of synapses in the given images.

In the first step, from one of the two images, the region of interest (i.e. the dendrites where the quantification of synapses will be performed) is specified using NeuronJ [22]—an ImageJ plugin for tracing elongated image structures. In this way, the background of the image is removed. The result is a file containing the traces of each dendrite of the image.

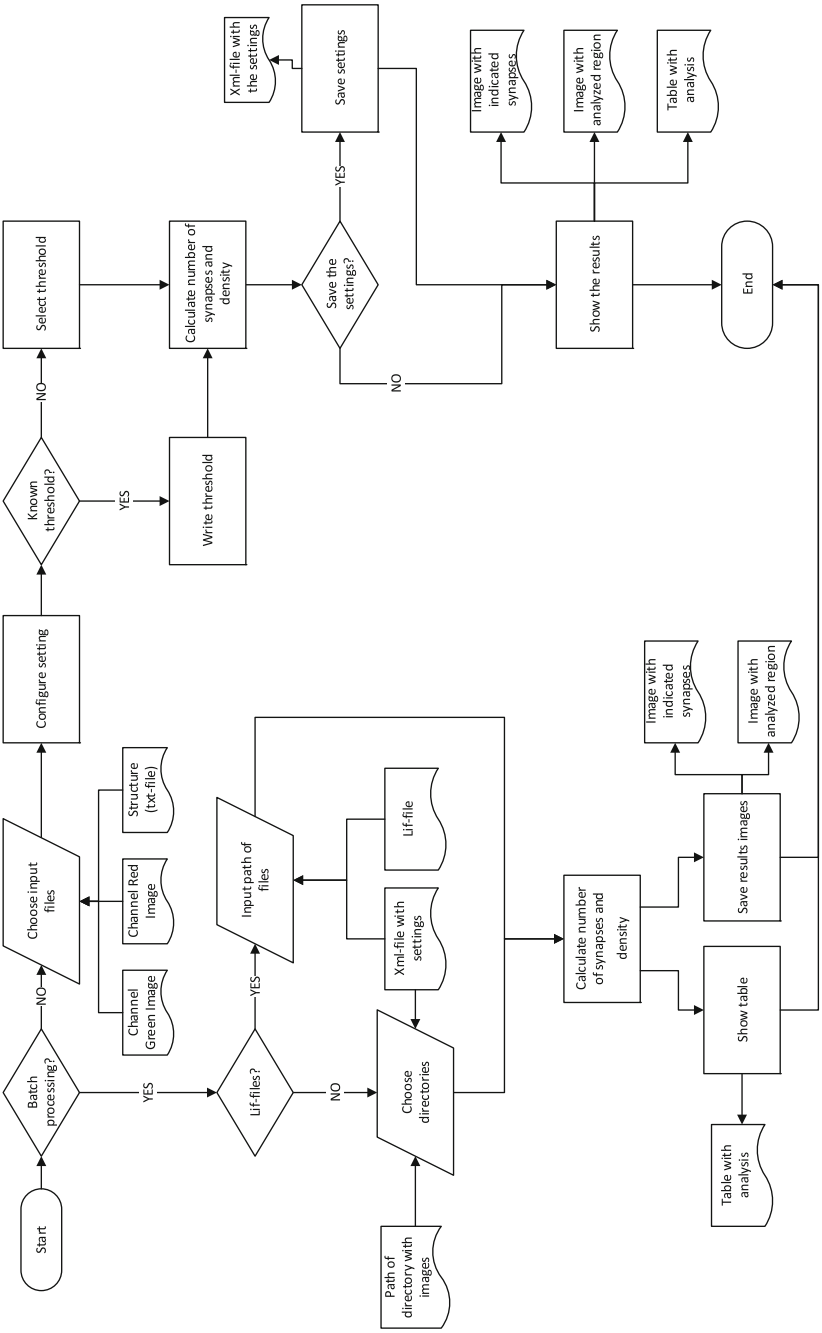

Fig. 1. Workflow of SynapCountJ.

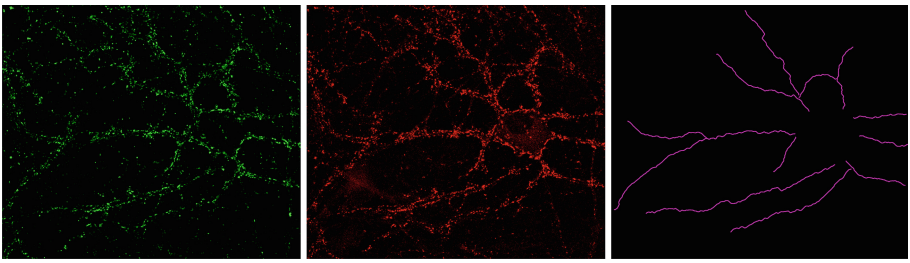

Fig. 2. Neuron with two antibody markers and its structure. *Left.* Neuron marked with the bassoon antibody marker. *Center.* Neuron marked with the synapsin antibody marker. *Right.* Structure of the neuron.

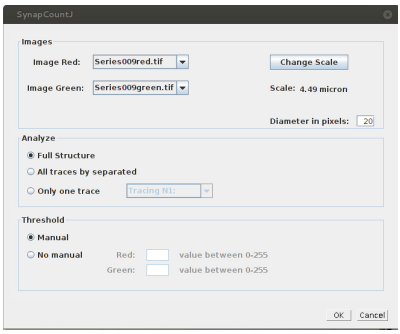

Fig. 3. SynapCountJ window to configure the analysis

Subsequently, the user can decide whether she wants to perform a global analysis of the whole neuron, or a local analysis focused on each dendrite of the neuron. In both cases, SynapCountJ requires additional information such as the scale and the mean thickness (that is determined by the size of the subjacent dendrite) of the region to analyze (see Fig. 3)—these parameters determine the area of the dendrite avoiding the background (i.e. all the non-synaptic marking).

Taking into account the settings provided by the user, SynapCountJ overlaps the two original images of the neuron and the structure of the neuron previously defined. From the resultant image, SynapCountJ identifies the almost white points (the result of green, red, and blue combination) as synaptic candidates, and it allows the user to modify the values of the red and green channels in order to modify the detection threshold (see Fig. 4).

Once that the detection threshold has been fixed, the counting process is started. Such a process is inspired by techniques coming from Computational Algebraic Topology. In spite of being an abstract mathematical subject, Algebraic Topology has been successfully applied in digital image analysis [14,20,33]. In our particular case, the white areas are segmented from the overlapped image, and the colors of the resultant image are inverted—obtaining as a result a black-and-white image where the synapses are the black areas. From such an image, the problem

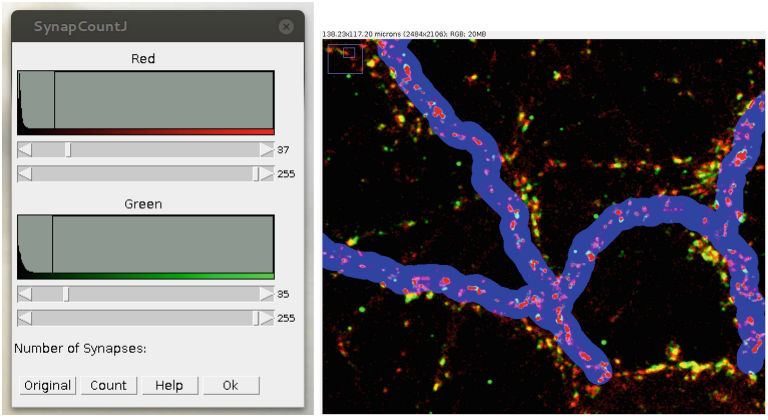

Fig. 4. SynapCountJ window to modify the threshold of the red and green channels. *Left.* Window to fix the threshold of the image. *Right.* Fragment of the neuron image with the synapses indicated as the red areas on the structure of the neuron marked in blue. Moving the scrollbars of left window, the marked areas of the image are changed. (Color figure online)

of quantifying the number of synapses is reduced to compute the homology group in dimension 0 of the image; this corresponds to the computation of the number of connected components of the image. Our algorithm to count synapses can be summarized as presented in Algorithm 1.

Algorithm 1. Counting Synapses in SynapCountJ.

Input: Two images of a neuron marked with two antibodies
Output: Number of synapses of the neuron

1 Create an image with the structure of the neuron using NeuronJ;
2 Overlap the two original images of the neuron and the structure of the neuron;
3 Fix the detection threshold;
4 Segment the white areas of the overlapped image using the fixed threshold;
5 Invert the colours of the segmented image;
6 Count the number of connected components of the image.

Finally, SynapCountJ returns a table with the obtained data (length of dendrites both in pixels and micras, number of synapses, and density of synapses per 100 micron) and two images showing, respectively, the analyzed region and the marked synapses (see Fig. 5).

2.2 Batch Processing

Images obtained from the same biological experiment usually have similar settings; hence, their processing in SynapCountJ will use the same configuration

	Label	Length in pixels	Length in micras	Synapses	Density	Red	Green
1	Tracing N1:	1833.1058	91.6553	71	77.4642	116	164
2	Tracing N2:	867.7840	43.3892	35	80.6652	116	164
3	Tracing N3:	983.5322	49.1766	53	107.7748	116	164
4	Tracing N4:	599.8320	29.9916	41	136.7049	116	164
5	Tracing N5:	437.7388	21.8869	25	114.2234	116	164
6	Tracing N6:	468.8438	23.4422	26	110.9111	116	164
7	Tracing N7:	447.6296	22.3815	31	138.5074	116	164
8	Tracing N8:	574.3691	28.7185	38	132.3191	116	164
9	Tracing N9:	1776.2572	88.8129	69	77.6915	116	164
10	Tracing N10:	1224.7374	61.2369	45	73.4851	116	164
11	Tracing N11:	355.7054	17.7853	26	146.1884	116	164
12	Tracing N12:	905.3750	45.2688	45	99.4063	116	164
13	Total Neuron	10474.9103	523.7455	479	91.4566	116	164

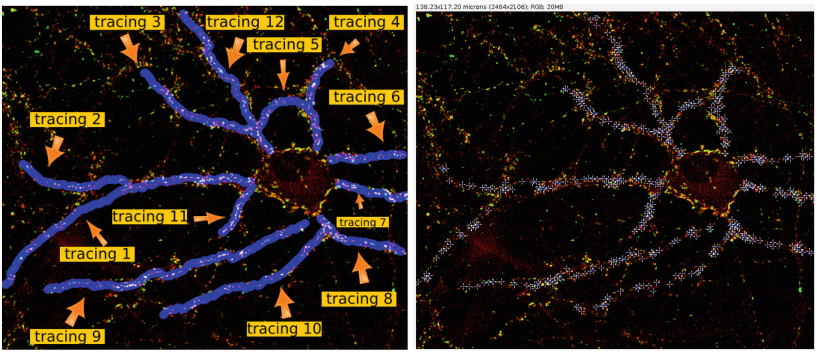

Fig. 5. Results provided by SynapCountJ. *Top.* Table with the results obtained by SynapCountJ. *Bottom Left.* Image with the analyzed region of the neuron. *Bottom Right.* Image with the counted synapses indicated by means of blue crosses.

parameters. In order to deal with this situation, SynapCountJ can be applied for batch processing of several images using a configuration file. It is necessary to study at least one image from experiment to get the optimal settings. The parameters are saved and used to process the set of images from the same experiment.

For batch processing, SynapCountJ reads tiff files organized in folders or a lif file (the kind of files produced by Leica confocal microscopes), and using the configuration file processes the different images. As a result, a table with the information related to each neuron from the batch (the table includes an analysis for both the whole neuron and from each of its dendrites) is obtained. In addition, in the same directory where the lif-file or tiff-files are stored, the plugin saves all the resultant images for each image from experiment (one of them shows the marked synapses and the other one, the region which has been studied).

3 Experimental Results

The original aim of SynapCountJ was the automatic analysis of synaptic density on neurons treated with SB 415286—an organic inhibitor of GSK3, a kinase which

inhibition was proposed as a therapy in AD treatment [9]—such a treatment, as it was previously demonstrated, promotes synaptogenesis and spinogenesis in primary cultures of rodent hippocampal neurons and in Drosophyla neurons [7,13]. In this setting, a comparative study has been performed in order to evaluate the results that can be obtained with SynapCountJ.

Primary hippocampal cultures were obtained from P0 rat pups (Sprague-Dawley, strain, Harlan Laboratories Models SL, France). Animals were anesthetized by hypothermia in paper-lined towel over crushed-ice surface during 2–4 min and euthanized by decapitation. Animals were handled and maintained in accordance with the Council Directive guidelines 2010/63EU of the European Parliament. in [23]. Briefly, glass coverslips (12 mm in diameter) were coated with poly-L-lysine and laminin, 100 and 4 μg/ml respectively. Neurons at a 10×104 neurons/cm^2 density were seeded and grown in Neurobasal (Invitrogen, USA) culture medium supplemented with glutamine 0.5 mM, 50 mg/ml penicillin, 50 units/ml streptomycin, 4% FBS and 4% B27 (Invitrogen, CA, USA), as described before in [6]. At days 4, 7 and 14 in culture a 20% of culture medium was replace by fresh medium. Cytosine-D-arabinofuranoside (4 μM) was added to prevent overgrowth of glial cells (day 4).

Synaptic density on hippocampal cultures was identified as previously described in [6]. In short, cultures were rinsed in phosphate buffer saline (PBS) and fixed for 30 min in 4% paraformaldehyde-PBS. Coverslips were incubated overnight in blocking solution with the following antibodies: anti-Bassoon monoclonal mouse antibody (ref. VAM-PS003, Stress Gen, USA) and rabbit polyclonal sera against Synapsin (ref. 2312, Cell Signaling, USA). Samples were incubated with a fluorescence-conjugated secondary antibody in PBS for 30 min. After that, coverslips were washed three times in PBS and mounted using Mowiol (all secondary antibodies from Molecular Probes-Invitrogen, USA). Stack images (pixel size 90 nm with 0.5 μm Z step) were obtained with a Leica SP5 Confocal microscope (40x lens, 1. 3 NA). Percentage of synaptic change is the average of different cultures under the same experimental conditions. As a control, we used sister untreated cultures growing in the same 24 well multi plate.

A total of 13 individual images from three independent cultures has been analyzed. In Fig. 6 we can observe that using a manual method to identify and count synapses, we obtain a mean of 24.12 synapses in control cultures and 16.74 in treated cultures. The results obtained with SynapCountJ are similar, there is a mean of 26.03 synapses in control cultures and 16.50 in the ones which have been treated.

Not with standing the differences in the quantification, in both procedures we obtain almost the same inhibition percentage, a 30.51% manually and 36.61% automatically. This shows the suitability of SynapCountJ to count synapses, meaning a considerably reduction of the time employed in the manual process. Namely, the manual analysis of an image takes approximately 5 min; of a batch, 1 h; and, of a complete study, 4 h. Using SynapCountJ, the time to analyze an image is 30 s; a batch, 2 min; and, a complete study, 6 min.

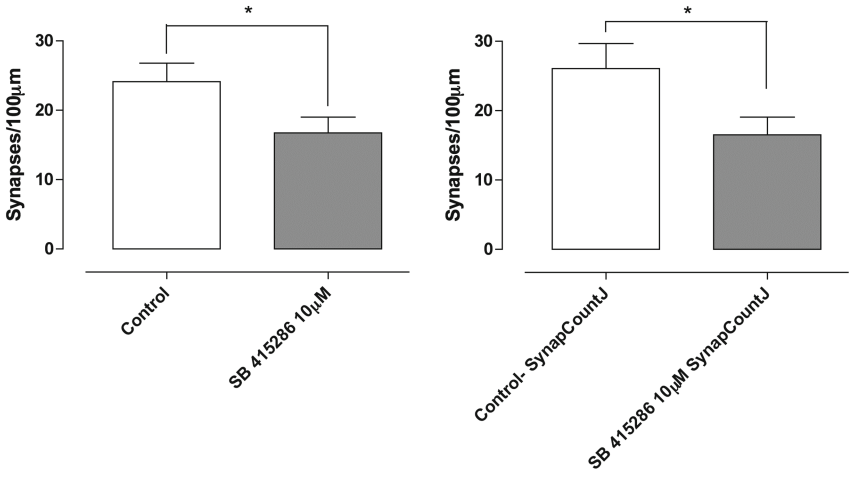

Fig. 6. Quantification of synapses. *Left.* Manual quantification of synapses. *Right.* Quantification of synapses using *SynapCountJ*

4 Scientific Validations of the Computations

Accuracy and reliability are two desirable properties of every software tool, especially in the case of biomedical software. An approach to increase the trust in scientific software is the use of mechanised theorem proving technology to verify the correctness of the programs [2,15]. However, such a formal verification is a challenging task [5]. In our work, we are interested in increasing the reliability of our software; however, due to the difficulty of directly verifying the correctness of our programs, we have followed an indirect approach.

A key component of our algorithm to count synapses is the computation of connected components of a black-and-white image (see Algorithm 1). Such a computation can be performed using two different approaches:

– a direct approach, where the pixels of the image are directly processed; and,
– an indirect approach, where the notion of simplicial complex associated with an image, and techniques from Algebraic Topology (namely, homology groups) are employed to compute the connected components of the image.

The former is efficient and can be easily employed in ImageJ—in fact, it is the one implemented in SynapCountJ—however, its formal verification is a challenging problem. The latter is slower than the former, is difficult to incorporate it into ImageJ; but, it can rely on a previously developed software, the Kenzo system [12], and therefore, it does not require any further development. The formal verification of the Kenzo system is even harder than the verification of the direct approach, but, fortunately, such a task was, at least partially, tackled in the ForMath project [1]—an European project devoted to the development of libraries of formalised mathematics concerning algebra, linear algebra, real number computation, and Algebraic Topology.

In this context, where we have a fast but unverified algorithm, and a slow but verified algorithm, the following strategy can be employed to increase the reliability of the fast version thanks to the verified version. The strategy consists in performing an intensive automated testing checking whether the results obtained with both versions are the same; if that is the case, the reliability of the fast algorithm is increased. In our particular case, we have employed such a strategy to increase the reliability of the computation of connected components of black-and-white images using the fast version implemented in SynapCountJ (the direct approach) thanks to the verified Kenzo system (the indirect approach).

In the rest of this section, we thoroughly explain the two different approaches to compute connected components of a black-and-white image.

4.1 The Direct Approach

The direct approach to compute connected components of a black-and-white image processes directly the pixels of the image by means of an algorithm included in ImageJ which is called *FindMaxima*. This algorithm can be applied to black-and-white, grayscale or color images and determines the local maxima of the image, provided with segmented regions containing all the pixels of the image whose value differs from the corresponding local maxima in less than a chosen threshold. In the case of black-and-white images, the result corresponds to the different connected components.

The algorithm is divided into two steps:

1. First of all, the local maxima of the image are determined, and they are ordered in a decreasing way.
2. Secondly, a filling algorithm is applied for each local maximum to determine its connected region. If a maximum produces a region which was already filled by a previous maximum, the actual local maximum is discarded.

The first step is done by means of a method called *getSortedMaxPoints*. Here, all the pixels in the image are studied comparing them with their adjacent pixels. A pixel is chosen as local maxima if its value is higher than all their adjacent pixels. A threshold is also considered to discard those pixels with value lower than it. The result is an array with the local maxima (with their coordinates) ordered in a decreasing way.

Once the ordered list of local maxima has been obtained, the second step of the algorithm FindMaxima is done by means of a method called *anaylizeAndMarkMaxima*. In this method, a filling algorithm is applied to each local maximum going over the list in a decreasing way. To determine the region associated to a local maximum, an iterative process is applied considering the 8 adjacent pixels to the maximum, selecting those whose difference with the local maximum is lower than a chosen parameter and studying then the adjacent pixels to those selected in the previous step. If a selected pixel is higher than the local maxima then it is stored as the maximum of the region and the previous one is discarded. If it is equal, the new pixel is also stored in order to be able to compute the mean of all

the local maxima in the region as we will explain later. The process finishes when all possible adjacent pixels to the previously selected ones have been studied.

Let us observe that when applying the filling algorithm to a local maximum, we could find other maxima (included in the same connected component as the considered one). In that case, the process stops and the second maximum is discarded. Moreover, in case of having several maxima with the same value in a region, the final maximum is computed as the pixel with the same intensity as the local maximum which is closest to the baricenter of all of them.

For a more complete study of the FindMaxima algorithm in ImageJ see [25].

4.2 The Indirect Approach

The indirect approach to compute connected components of a black-and-white image employs the Kenzo system. Kenzo [12] is a Common Lisp system devoted to Algebraic Topology that was developed by Francis Sergeraert. Kenzo has obtained some results not confirmed nor refuted by theoretical or computational means [35], and also has been used to refute some computations obtained by theoretical means [27,28]. Then, the question of Kenzo reliability arose in a natural way, and several works have been focussed on studying the correctness of Kenzo key fragments and algorithms [3,11,18].

The final aim of Kenzo was not the analysis of digital images, but it was extended with a module that tackles such a problem [16]. In particular, such a module computes homological properties, that measure connected components and holes of

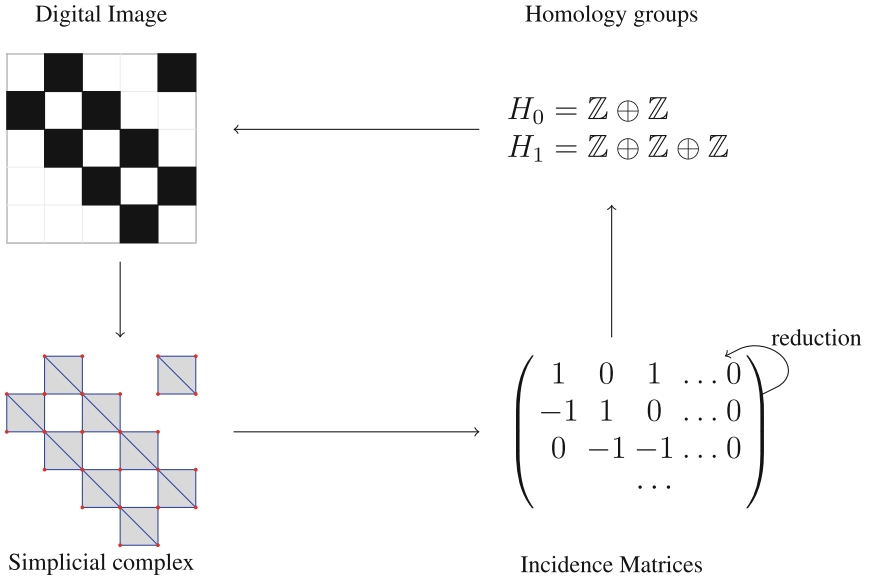

Fig. 7. Workflow to compute homology groups from digital images. The homology groups indicate that the image has two connected components and three holes.

black-and-white images. This Kenzo module for digital images has been employed to validate the results obtained in SynapCountJ using the direct approach.

The Kenzo module for digital images works as follows (see Fig. 7). Given a black-and-white image, a triangulation procedure is employed to obtain a simplicial complex (a generalisation of the notion of graph to higher dimensions)—there are several methods to construct a simplicial complex from a digital image, see [4]. From the simplicial complex, its *boundary (or incidence) matrices* are constructed. Since the size of the boundary matrices coming from biomedical images is too big to be handled directly by Kenzo, a reduction strategy is employed to work with smaller matrices, but preserving their homological properties [29]. From the reduced boundary matrices, homology groups in dimensions 0 and 1 are computed using a diagonalisation process [24]. The homology groups are either null or a direct sum of $\mathbb{Z}$ components, and they should be interpreted as follows: the number of $\mathbb{Z}$ components of the homology groups of dimension 0 and 1 measures respectively the number of connected components and the number of holes of the image. Hence, computing the homology groups associated with a digital image, we can obtain the number of connected components of the image.

The aforementioned workflow to compute homology groups from digital images was fully verified in [17,26].

5 Discussion

Up to the best of our knowledge, 4 tools have been developed to quantify synapses and measure synaptic density: Green and Red Puncta [36], Puncta Analyzer [37], SynD [31] and SynPAnal [8]—a summary of the general features of these tools can be seen in Table 1. The rest of this section is devoted to compare SynapCountJ with these tools—such a comparison is summarized in Table 2.

Table 1. General features of the analyzed software

Software	Language	Underlying technology	Types of images	Technique for detection
Green and Red Puncta	Java	ImageJ	tiff	Colocalization
Puncta Analyzer	Java	ImageJ2	tiff	Colocalization
SynapCountJ	Java	ImageJ	tiff and lif	Colocalization
SynD	Matlab	Matlab	tiff and lsm	Brightness
SynPAnal	Java		tiff	Brightness

There are two approaches to locate synapses in an RGB image either based on colocalization or brightness. In the former, synapses are identified as the colocalization of bright points in the red and green channels—this is the approach followed by Green and Red Puncta, Puncta Analyzer and SynapCountJ—in the latter, synapses are the bright points of a region of an image—the approach employed

Table 2. Features to quantify synapses and synaptic density of the analyzed software

Software	Detection of dendrites	Threshold	Batch processing	Dendrites length	Density	Export	Save
Green and Red Puncta	Not used	✓					
Puncta Analyzer	Manual ROI	✓				✓	
SynapCountJ	Manual	✓	✓	✓	✓	✓	✓
SynD	Automatic	✓	✓	✓		✓	✓
SynPAnal	Manual	✓		✓		✓	✓

in SynD and SynPAnal. In both approaches, it is necessary a threshold that can be manually adjusted to increase (or decrease) the number of detected synapses; such a functionality is supported by all the tools.

In the quantification of synapses from RGB images, it is instrumental to determine the region of interest (i.e. the dendrites of the neurons where the synapses are located); otherwise, the analysis will not be precise due to noise coming from irrelevant regions or the background of the image—this happens in the Green and Red Puncta tool since it considers the whole image for the analysis. Puncta Analyzer allows the user to fix a rectangle containing the dendrites of the neuron, but this is not completely precise since some regions of the rectangle might contain points considered as synapses that do not belong to the structure of the neuron. SynD is the only software that automatically detects the dendrites of a neuron; however, it can only be applied to neurons with a cell-fill marker, and does not support the analysis from specific regions, such as soma or distal dendrites. SynapCountJ and SynPAnal provide the functionality to manually draw the dendrites of the image; allowing the user to designate the specific areas where quantification is restricted.

The main output produced by all the available tools is the number of synapses of a given image; additionally, SynapCountJ, SynD and SynPAnal provides the length of the dendrites; and, SynapCountJ is the only tool that outputs the synaptic density per micron. All the tools but Green and Red Punctua can export the results to an external file for storage and further processing.

Finally, as we have explained in Subsect. 2.2, images obtained from the same biological experiment usually have similar settings; hence, batch processing might be useful. This functionality is featured by SynapCountJ and SynD, and requires a previous step of saving the configuration of an individual analysis. SynPAnal does not support batch processing, but the configuration of an individual analysis can be saved to be later applied in other individual analysis.

As a summary, SynapCountJ is more complete than the rest of available programs. It can use different types of synaptic markers and can process batch images. Furthermore, a differential feature of SynapCountJ is that it is based on a topological algorithm (namely, computing the number of connected components in a

combinatorial structure), allowing us to validate the correctness of our approach by means of formal methods in software engineering.

6 Conclusions and Further Work

SynapCountJ is an ImageJ plugin that provides a semi-automatic procedure to quantify synapses and measure synaptic density from immunofluorescence images obtained from neuron cultures. This plugin has been tested not only with neurons in development, but also with the neuromuscular union of Drosophila; therefore, it can be applied to the study of images that contain two synaptic markers and a determined structure. The results obtained with SynapCountJ are consistent with the results obtained manually; and SynapCountJ dramatically reduces the time required for the quantification of synapses. Moreover, the realiability of SynapCountJ has been increased by validating some of its computations using the formally verified module for digital images of Kenzo.

As further work, it remains the tasks of improving the usability of the plugin and including post-processing tools to manually edit the obtained results. Additionally, and since the final aim of our project is the complete automation of the whole process, it is necessary a procedure to automatically detect the neuron morphology, and also to automatically fix the threshold for the segmentation of neurons. For such an automation, machine learning techniques like the ones presented in [21] might be employed.

7 Availability and Software Requirements

SynapCountJ is an ImageJ plugin that can be downloaded, together with its documentation, from http://imagejdocu.tudor.lu/doku.php?id=plugin:utilities:synapsescountj:start. SynapCountJ is open source and available for use under the GNU General Public License. This plugin runs within both ImageJ and Fiji [30] and has been tested on Windows, Macintosh and Linux machines.

References

1. Formath: formalisation of mathematics (2010–2013). http://wiki.portal.chalmers.se/cse/pmwiki.php/ForMath/ForMath
2. Amorim, A., et al.: A verified information-flow architecture. In: 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2014) (2014)
3. Aransay, J., Ballarin, C., Rubio, J.: A mechanized proof of the Basic Perturbation Lemma. J. Autom. Reasoning **40**(4), 271–292 (2008)
4. Ayala, R., Domínguez, E., Francés, A., Quintero, A.: Homotopy in digital spaces. Discrete Appl. Math. **125**, 3–24 (2003)
5. Benton, N.: Machine Obstructed Proof: how many months can it take to verify 30 assembly instructions? (2006)

6. Cuesto, G., Enriquez-Barreto, L., Caramés, C., et al.: Phosphoinositide-3-kinase activation controls synaptogenesis and spinogenesis in hippocampal neurons. J. Neurosci. **31**(8), 2721–2733 (2011)
7. Cuesto, G., Jordán-Álvarez, S., Enriquez-Barreto, L., et al.: GSK3β inhibition promotes synaptogenesis in Drosophila and mammalian neurons. Plos One **10**(3), e0118475 (2015). doi:10.1371/journal.pone.0118475
8. Danielson, E., Lee, S.H.: SynPAnal: software for rapid quantification of the density and intensity of protein puncta from fluorescence microscopy images of neurons. PLoS ONE **9**(12), e115298 (2014). doi:10.1371/journal.pone.0115298
9. DaRocha-Souto, B., Scotton, T.C., Coma, M., et al.: Brain oligomeric β-amyloid but not total amyloid plaque burden correlates with neuronal loss and astrocyte inflammatory response in amyloid precursor protein/tau transgenic mice. J. Neuropathol. Exp. Neurol. **70**(5), 360–376 (2003)
10. Devices, M.: Metamorph research imaging (2015). http://www.moleculardevices.com/systems/metamorph-research-imaging
11. Domínguez, C., Rubio, J.: Effective homology of bicomplexes, formalized in Coq. Theor. Comput. Sci. **412**, 962–970 (2011)
12. Dousson, X., Rubio, J., Sergeraert, F., Siret, Y.: The Kenzo program. Institut Fourier, Grenoble (1998). https://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/
13. Franco, B., Bogdanik, L., Bobinnec, Y., et al.: Shaggy, the homolog of glycogen synthase kinase 3, controls neuromuscular junction growth in Drosophila. J. Neurosci. **24**(29), 6573–6577 (2004)
14. González-Díaz, R., Real, P.: On the Cohomology of 3D digital images. Discrete Appl. Math. **147**(2–3), 245–263 (2005)
15. Hales, T.: The Flyspeck Project fact sheet (2005). Project description available at http://code.google.com/p/flyspeck/
16. Heras, J., Pascual, V., Rubio, J.: A certified module to study digital images with the Kenzo system. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) EUROCAST 2011. LNCS, vol. 6927, pp. 113–120. Springer, Heidelberg (2012)
17. Heras, J., Dénès, M., Mata, G., Mörtberg, A., Poza, M., Siles, V.: Towards a certified computation of homology groups for digital images. In: Ferri, M., Frosini, P., Landi, C., Cerri, A., Fabio, B. (eds.) CTIC 2012. LNCS, vol. 7309, pp. 49–57. Springer, Heidelberg (2012)
18. Lambán, L., Martín-Mateos, F.J., Rubio, J., Ruiz-Reina, J.L.: Verifying the bridge between simplicial topology and algebra: the Eilenberg-Zilber algorithm. Logic J. IGpPL **22**(1), 39–65 (2013)
19. Linkert, M., Rueden, C.T., Allan, C., et al.: Metadata matters: access to image data in the real world. J. Cell Biol. **189**(5), 777–782 (2010)
20. Mata, G., et al.: Zigzag persistent homology for processing neuronal images. Pattern Recogn. Lett. **62**(1), 55–60 (2015)
21. Mata, G., et al.: Automatic detection of neurons in high-content microscope images using machine learning approaches. In: Proceedings of the 13th IEEE International Symposium on Biomedical Imaging (ISBI 2016). IEEE Xplore (2016)
22. Meijering, E., Jacob, M., Sarria, J.C.F., et al.: Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. Cytometry Part A **58**(2), 167–176 (2004)
23. Morales, M., Colicos, M.A., Goda, Y.: Actin-dependent regulation of neurotransmitter release at central synapses. Neuron **27**(3), 539–550 (2000)
24. Munkres, J.R.: Elements of Algebraic Topology. Addison-Wesley, Reading (1984)
25. de Greñu de Pedro, J.D.: Análisis Matemático de rutinas de procesamiento de imágenes digitales en Fiji/ImageJ. Technical report, Universidad de La Rioja (2014)

26. Poza, M., Domínguez, C., Heras, J., Rubio, J.: A certified reduction strategy for homological image processing. ACM Trans. Comput. Logic **15**(3), 23 (2014)
27. Romero, A., Heras, J., Rubio, J., Sergeraert, F.: Defining and computing persistent Z-homology in the general case. CoRR abs/1403.7086 (2014)
28. Romero, A., Rubio, J.: Homotopy groups of suspended classifying spaces: an experimental approach. Math. Comput. **82**, 2237–2244 (2013)
29. Romero, A., Sergeraert, F.: Discrete Vector Fields and Fundamental Algebraic Topology (2010). http://arxiv.org/abs/1005.5685v1
30. Schindelin, J., Argand-Carreras, I., Frise, E., et al.: Fiji: an open-source platform for biological-image analysis. Nat. Methods **9**(7), 676–682 (2012)
31. Schmitz, S.K., Hjorth, J.J.J., Joemail, R.M.S., et al.: Automated analysis of neuronal morphology, synapse number and synaptic recruitment. J. Neurosci. Methods **195**(2), 185–193 (2011)
32. Schneider, C., Rasband, W., Eliceiri, K.: NIH Image to ImageJ. Nat. Methods **9**, 671–675 (2012)
33. Ségonne, F., Grimson, E., Fischl, B.: Topological correction of subcortical segmentation. In: Ellis, R.E., Peters, T.M. (eds.) MICCAI 2003. LNCS, vol. 2879, pp. 695–702. Springer, Heidelberg (2003)
34. Selkoe, D.J.: Alzheimer's diseases is a synaptic failure. Science **298**(5594), 789–791 (2002)
35. Sergeraert, F.: Effective homology, a survey. Technical report, Institut Fourier (1992). http://www-fourier.ujf-grenoble.fr/sergerar/Papers/Survey.pdf
36. Shiwarski, D.J., Dagda, R.D., Chu, C.T.: Green and red puncta colocalization (2014). http://imagejdocu.tudor.lu/doku.php?id=plugin:analysis:colocalization_analysis_macro_for_red_and_green_puncta:start
37. Wark, B.: Puncta analyzer v2.0 (2013). https://github.com/physion/puncta-analyzer