On automation and certification of a homological method to process biomedical digital images*

Jónathan Heras, Gadea Mata, María Poza, and Julio Rubio

Department of Mathematics and Computer Science of University of La Rioja
{jonathan.heras, gadea.mata, maria.poza, julio.rubio}@unirioja.es

Abstract. In this paper a methodology to extract and compute homological information from biomedical images is proposed; automating some processes, up to now, manually done. The main features of our approach are the usage of several programming languages (Java, Common Lisp and Haskell) and the application of formal methods (namely theorem provers) to verify the correctness of some of the automated process. As case study to test the suitability of our approach, we have applied it to measure the number of synapses of a neuron.

1 Introduction

In this work a methodology to deal with biomedical digital images is presented. The main steps of our proposal are the following ones. First of all, we detect what kind of homological information is needed in a concrete problem of image processing. Subsequently, we manipulate the image to get an image where the topological information is as explicit as possible. Afterwards, ensuring that no relevant information is lost during the process, we reduce the size of the data. Eventually, applying some computer algebra program, the homological invariants are computed. These homological invariants provide properties of the original image.

The objectives of this kind of research are twofold. On the one hand, we want to automate some tasks (tracing, marking, counting, and so on) made up to now manually (or semi-automatically) by biologists and other experimental scientists. On the other hand, we undertake the challenge of verifying the correctness of the automated process.

Even if the second objective could be considered as excessive, in fields where the accuracy standards are not so high as in mathematics or theoretical computer science, it is necessary to stress that the simplifications done by experimental scientists are based on solid (even if heuristic) previous experience. If they must trust computer programs, it is convenient to produce them in a reliable manner, in such a way that scientists could be confident of the results mechanically obtained.

The next section is devoted to describe an instantiation of this methodology.

* Partially supported by Ministerio de Ciencia e Innovación, project MTM2009-13842-C02-01, and by European Community FP7, STREP project ForMath, n. 243847.

2 A case study: Counting Synapses

In order to test the feasibility of our methodology, we have studied images related to the synaptical structure in neurons [1]. Examples of this kind of images are presented in Figure 1.

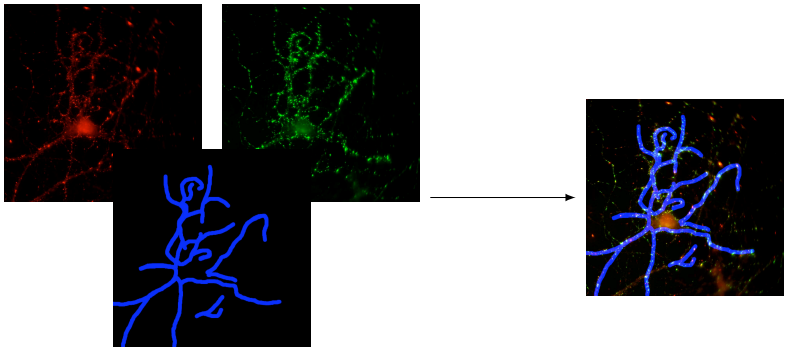

Fig. 1. Images related to the synaptical structure in neurons

As we explained previously, the first step in our methodology consists in determining what kind of homological information is needed. In this particular case the topological invariant to be computed is the number of connected components which will be useful to determine the evolution of the density of the occurrence of synapses in neurons, under the effect of some drugs.

Subsequently, we process the image to get an image where the topological information is as explicit as possible. To this aim, the digital images obtained experimentally are handled by means of *SynapsesCountJ* [9], a plugin of the *ImageJ* Java environment [10], producing a bitmap file where connected components should be counted.

From the previous file an incidence matrix is constructed, which in processed through a Haskell program [8], to obtain a smaller matrix with the same homological information (we are using here Discrete Morse Theory, as explained in [11]). These Haskell programs are being analyzed by using the Coq proof assistant [3,2], and more specifically the *SSReflect* environment [5]. This step corresponds with the reduction process of our methodology.

Finally, the matrices obtained by means of the Haskell programs are given as input to the *fKenzo* system [6,7], a user interface for the *Kenzo* program [4], which actually computes the homological information.

References

1. M. Bear, B. Connors, and M. Paradiso. *Neuroscience: Exploring the Brain.* Lippincott Williams & Wilkins, 2006.

2. Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development, Coq'Art: the Calculus of Inductive Constructions*. Springer-Verlag, 2004.
3. Coq development team. The Coq Proof Assistant Reference Manual, version 8.3. Technical report, 2010.
4. X. Dousson, J. Rubio, F. Sergeraert, and Y. Siret. The Kenzo program. Institut Fourier, Grenoble, 1998. `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`.
5. G. Gonthier and A. Mahboubi. A Small Scale Reflection Extension for the Coq system. Technical report, Microsoft Research INRIA, 2009. `http://hal.inria.fr/inria-00258384`.
6. J. Heras. The *fKenzo* program. University of La Rioja, 2010. `http://www.unirioja.es/cu/joheras/fKenzo/`.
7. J. Heras, V. Pascual, J. Rubio, and F. Sergeraert. *fKenzo*: A user interface for computations in Algebraic Topology. *Journal of Symbolic Computation*, 46:685–698, 2011.
8. S. P. Jones et al. The Haskell 98 language and libraries: The revised report. *Journal of Functional Programming*, 13(1):0–255, 2003. `http://www.haskell.org`.
9. G. Mata. SynapsesCountJ. University of La Rioja, 2011. `http://imagejdocu.tudor.lu/doku.php?id=plugin:utilities:synapsescountj:start`.
10. W. S. Rasband. ImageJ: Image Processing and Analysis in Java, 2003. `http://rsb.info.nih.gov/ij/`.
11. A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. `http://arxiv.org/abs/1005.5685v1`.