Towards a certified computation of homology groups for digital images[⋆]

Jónathan Heras[1], Maxime Dénès[2], Gadea Mata[1], Anders Mörtberg[3], María Poza[1], and Vincent Siles[3]

[1] Department of Mathematics and Computer Science of University of La Rioja
[2] INRIA Sophia Antipolis - Méditerranée
[3] University of Gothenburg
jonathan.heras@unirioja.es, Maxime.Denes@inria.fr, gadea.mata@unirioja.es, mortberg@chalmers.se, maria.poza@unirioja.es, siles@chalmers.se

Abstract. In this paper we report on a project to obtain a verified computation of homology groups of digital images. The methodology is based on programming and executing *inside* the Coq proof assistant. Though more research is needed to integrate and make efficient more processing tools, we present some examples partially computed in Coq from real biomedical images.

Keywords: Homology; Discrete Morse Theory; Proof assistant tools; Coq; SSReflect; Synapses.

1 Introduction

The discipline of Algebraic Digital Topology, or more specifically, the computation of homology groups from digital images is mature enough (see, for instance, [27], one among many good references) to go one step further and investigate the possibility of a *certified computation* (i.e., formally verified by proving correctness using an *interactive* proof assistant) in digital topology, as it happens in other areas of computer mathematics (see [8]).

In a very rough manner, the process to be verified is reflected in Figure 1. Putting it into words, from the black pixels of a monochromatic image a simplicial complex is obtained (by means of a triangulation procedure); subsequently, from the simplicial complex, its *boundary (or incidence) matrices* are constructed, and finally, *homology* can be computed. If we work with coefficients over a field (and it is well-known that it is enough to take as coefficients the field $\mathbb{Z}/2\mathbb{Z}$, when we work with 2D and 3D digital images) and if only the *dimensions* of the homology groups (as vector spaces) are looked for, then having a program able to compute the rank of a matrix is sufficient to accomplish the whole task.

[⋆] Partially supported by Ministerio de Educación y Ciencia, project MTM2009-13842-C02-01, and by the European Union's 7th Framework Programme under grant agreement nr. 243847 (ForMath).

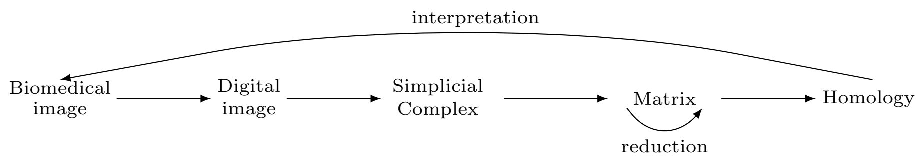


Fig. 1. Computing homology from a digital image

This architecture is particularized in this paper with a real problem that appeared in an industrial application and with the Coq proof assistant as programming and verifying tool.

The rest of this paper is organized as follows. Section 2 is devoted to present an example, coming from the biomedical context, as a test-case for our formal development. The formalization process is explained in Section 3, focusing on the link between boundary matrices and homology groups. Section 4 explains how the certified programs can be used to effectively compute homology of images. A way to deal with the management of the huge matrices produced by biomedical images is presented in Section 5. The paper ends with a section of Conclusions and Further work, and the bibliography.

2 Motivation

When developing formal proofs, a major issue is ensuring that concepts are defined in a way that will be applicable to concrete use. In our case, we are developing a general theory of effective simplicial homology as part of the Formath project [1]. We decided to validate our design choices on biomedical digital images obtained from synaptical structures.

Synapses are the points of connection between neurons. The relevance of synapses comes from the fact that they are related to the computational capabilities of the brain.

The possibility of changing the number of synapses may be an important asset in the treatment of neurological diseases, such as Alzheimer, see [26]. Therefore, we can claim that an efficient, reliable and automatic method for counting synapses is instrumental in the study of the evolution of synapses in scientific experiments.

Up to now, the method to count synapses was manual, see [6]. This was impractical since it implies a considerable time investment. In order to improve this process, a plug-in called SynapCountJ [17] for the ImageJ environment [22] has been developed.

The procedure implemented in this software to handle neuron images can be split into two steps. First, taking as input three images of a neuron, namely the neuron with two different antibody markers and the structure of the neuron, SynapCountJ produces a bitmap where synapses are the connected components, see Figure 2. Then the second step consists in counting the connected components of the bitmap. A detailed explanation of the procedure was given in [13].

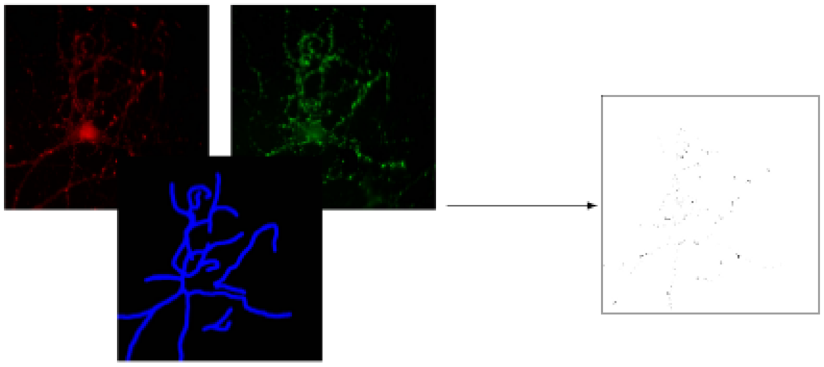


Fig. 2. Example of the results produced by SynapCountJ

To test the suitability of this program, biologists consider, on the one hand, control cultures and, on the other hand, cultures under the effect of some drugs; in this way, the evolution of the density of the occurrence of synapses under the effect of those drugs can be determined. For instance, using the chemical inhibitor GSK3, the evolution percentage manually obtained is 36% and the one obtained with SynapCountJ is 36.6%. Thus, the experimental results obtained with SynapCountJ were considered (by the biologists) very satisfactory.

The former step of the procedure implemented in SynapCountJ, the extraction of a bitmap with the synapses from three images of the neurons, is carried out based on solid previous experience of experimental scientists; therefore, they consider it as a safe process. The latter step, the computation of connected components, can be solved with many algorithms and is an interesting test case for our framework where we can compute the homology in dimension 0 of such images. This is a well known procedure to measure the amount of connected components of an image, even if more elementary methods are also applicable.

3 Verification in Coq/SSReflect

In the introduction we have explained a method, based on simplicial homology, to study the homology of a digital image which consists of: (1) building a simplicial complex from the image, (2) generating the boundary matrices associated with the simplicial complex, and (3) computing the homology from the boundary matrices.

The correctness of the programs in charge of both the construction of a simplicial complex from an image and the generation of the boundary matrices associated with a simplicial complex have been formally proved using proof assistant tools as can be seen in [21] and [14] respectively. Then, there only remains the verification of the third point, the computation of homology groups from the boundary matrices.

In our formalization, we have used the Coq proof assistant [5]. This system provides a formal language to write mathematical definitions, executable algorithms and theorems together with an environment for semi-interactive development of machine-checked proofs. In addition, we take advantage of the features included in SSReflect [9], an extension for Coq whose development was started by G. Gonthier during the formal proof of the *Four Color Theorem* [8]. The SSReflect libraries include enough ingredients to undertake the task of defining and computing homology from matrices. Some details of the proofs will be omitted; the interested reader can consult the original and complete source code at `http://wiki.portal.chalmers.se/cse/pmwiki.php/ForMath/ProofExamples`.

First of all, we define the notion of homology in Coq. Let K be a field, $V1, V2, V3$ vector spaces on K, and $f : V1 \to V2, g : V2 \to V3$ linear applications; then, the *Homology of* f, g is the quotient between the *kernel* of g and the *image* of f. This is translated into Coq in the following way.

```
Variable (K : fieldType) (V1 V2 V3 : vectType K)
         (f : linearApp V1 V2) (g : linearApp V2 V3).
Definition Homology := ((lker g) :\: (limg f)).
```

Nevertheless, we do not usually work with linear applications when trying to compute homology but with the matrices representing those linear applications. In particular, as we are working on a field K, given two matrices with coefficients in this field, let us called them, mxf and mxg of sizes $v1 \times v2$ and $v2 \times v3$ respectively and such that their product is the null matrix, the dimension of the corresponding homology vector space is given by the formula: $v2 - rank(mxg) - rank(mxf)$. This definition is introduced in Coq as follows.

```
Definition dim_homology (mxf:'M[K]_(v1,v2)) (mxg:'M[K]_(v2,v3)) :=
     v2 - \rank mxg - \rank mxf.
```

Now, the correctness of `dim_homology` can be shown by proving that given two matrices `mxf` and `mxg` whose product is the null matrix (`mxf *m mxg = 0`), then the result obtained using `dim_homology` is the dimension of the homology group associated with the linear applications defined from `mxf` and `mxg` (`(LinearApp mxf)` and `(LinearApp mxg)`).

```
Lemma dimHomologyrankE: mxf *m mxg = 0 ->
      \dim Homology (LinearApp mxf) (LinearApp mxg) =
      dim_homology mxf mxg.
```

However the use of SSReflect libraries may trigger heavy computations during deduction steps, that would not terminate within a reasonable amount of time. To handle this issue, some definitions like matrices are locked in a way that do not allow direct computations.

To overcome this pitfall, we use the matrix representation and the rank algorithm developed in [4] to define `ex_homology` which takes as argument two such matrices (represented by means of lists of lists) `mxf` and `mxg` which dimensions are `v1`×`v2` and `v2`×`v3` respectively, and computes the homology.

```
Definition ex_homology (v1 v2 v3:nat) (mxf mxg : seqMatrix K) :=
    v2 - (rank v2 v3 mxg) - (rank v1 v2 mxf).
```

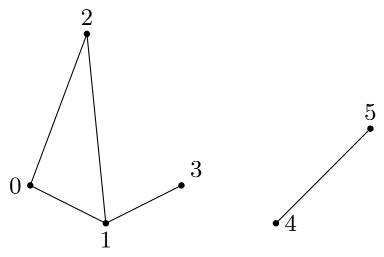
Finally, we prove the correctness of `ex_homology` by showing its equivalence to `dim_homology` up to a change of representation (this domain transformation is given by `seqmx_of_mx`).

```
Lemma ex_homology_rankE: forall (mxf: 'M[K]_(w1,w2)) (mxg : 'M[K]_
    (w2,w3)), ex_homology (seqmx_of_mx mxf) (seqmx_of_mx mxg) =
    dim_homology mxf mxg.
```

Then, we have an executable program to compute homology, for any dimension, whose correctness has been verified in Coq; therefore, we can claim that its results will always be correct.

4 Computing homology with Coq

An example is presented in this section in order to clarify how we can compute homology groups in Coq. Let us consider the simplicial complex of the left side of Figure 3. If we impose a lexicographical order on the simplices of the same dimension of this simplicial complex, its boundary matrix in dimension 1 is the one presented in the right side of Figure 3; it is worth noting that the rest of boundary matrices are empty, in particular we do not consider the empty set as an element of dimension -1.



$$\begin{array}{c} \\ (0)\\(1)\\(2)\\(3)\\(4)\\(5) \end{array}\begin{array}{c}\begin{array}{ccccc}(0,1)&(0,2)&(1,2)&(1,3)&(4,5)\end{array}\\ \left(\begin{array}{ccccc} 1&1&0&0&0\\ 1&0&1&1&0\\ 0&1&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1\\ 0&0&0&0&1 \end{array}\right)\end{array}$$

Fig. 3. Simplicial complex and its boundary matrix

The procedure to compute the homology (note that it only makes sense to compute homology in dimensions 0 and 1) of the simplicial complex of Figure 3 is as follows. Firstly, we define the boundary matrices.

```
Definition d0_ex1 := [::].
Definition d1_ex1 := [::[::1;1;0;0;0];
                        [::1;0;1;1;0];
                        [::0;1;1;0;0];
                        [::0;0;0;1;0];
                        [::0;0;0;0;1];
```

```
                                   [::0;0;0;0;1]].
Definition d2_ex1 := [::].
```

Eventually, we can compute the homology using the following instructions.

```
Eval vm_compute in (ex_homology 0 6 5 d0_ex1 d1_ex1).
Eval vm_compute in (ex_homology 6 5 0 d1_ex1 d2_ex1).
```

obtaining 2 and 1 respectively. In the same way, we could compute homology from the boundary matrices associated with the simplicial complex generated from a digital image. However, if we try to compute the homology from the images produced by SynapCountJ (see Figure 2), Coq is not able to handle those images yet, due to the size of data involved.

It is worth noting that Coq is a Proof Assistant and not a Computer Algebra system. Efficient implementations of mathematical algorithms running inside Coq is an ongoing effort, as shown by recent works on efficient real numbers [16], machine integers and arrays [2] or a previous approach to compiled execution of internal computations [10].

We devise a couple of ways to achieve better efficiency:

- Improve the runtime system using the extraction mechanism which translates Coq code to a functional programming language like OCaml or Haskell. However, this would not allow us to reuse the result of our homological computations for further proofs. Indeed, output of external programs are untrusted so they cannot be imported. Instead, we are using a recent intermediate approach consisting in internally compiling Coq terms to OCaml with performance comparable to extracted code [18].
- Optimize algorithms and representations using sparse matrices, which is well suited to simplicial complexes obtained from digital images. We have developed an Haskell implementation of such an algorithm but we still need to formally verify its correctness.

In the next section we describe another method to overcome the efficiency drawback, based on reducing the size of matrices while keeping the same homological information.

5 Computing discrete vector fields

The method that we are using for the reduction process is based on *Discrete Morse Theory* [7]; namely, we work in the algebraic setting of this theory which was described in [25]. Roughly speaking, the aim of Discrete Morse Theory consists of finding *simplicial collapses* which transform a simplicial complex $\mathcal{K}$ into a smaller one but keeping its homological properties. In this context, the instrumental tool are *admissible discrete vector fields* which allows one to reduce the amount of information removing “useless” information but keeping the homological properties of the original object.

The use of these techniques from Discrete Morse Theory has been welcomed in the study of homological properties of digital images, see [3,11,15], for instance.

This is due to the fact that the size of the cellular object associated with an image can be huge, but the choice of an appropriate vector field can produce a much smaller object.

So, the question now is given a cellular complex how we can produce a vector field as large as possible (the larger the vector field, the smaller the reduced object). Several approaches to solve this problem have been studied as can be seen in [24,12,23,19], the strategy that we have chosen was explained in [25]. It is not the aim of this paper to describe that algorithm (from now on, called RS's algorithm; RS stands for Romero–Sergeraert); but, we just introduce some ideas. This algorithm takes as input one of the boundary matrices associated with the cellular complex and provides an admissible discrete vector field (subsequently, from the matrix and the vector field a reduced matrix can be obtained).

The algorithm has been implemented in Haskell; and, some remarkable results have been obtained in the reduction process. As benchmark to test our programs, we have considered matrices coming from, on the one hand, 500 randomly generated images; and, on the other hand, biomedical images. In the former case, the size of the matrices was initially around 100×300, and after the reduction process the average size was 5×50. Using the original matrices Coq takes around 12 seconds to compute their rank; on the contrary, using the reduced matrices Coq only needs milliseconds. In the latter case, the matrices coming from biomedical images, the size of matrices is reduced from around 690×1400 to 97×500. In this case, Coq cannot deal with the original matrices; on the contrary, it is able to handle matrices as the ones obtained after applying the reduction programs and compute the results in, approximately, 25 seconds.

As a final remark, let us explain the main reason for using Haskell to implement the RS algorithm. The use of this language is due to the fact that Haskell is quite close to Coq; and, therefore, algorithms implemented in Haskell can be verified using Coq, a question which is, as we have seen, instrumental in our developments. In particular, the formalization of the correctness of the algorithm in charge of constructing an admissible discrete vector field given a matrix is ongoing work; and, up to now, we have certified that our programs build a discrete vector field. The proof of the admissibility property remains as further work.

6 Conclusions and further work

In this paper, we have presented how we can use Algebraic Topology techniques to study biomedical images in a reliable manner. The first step consists in processing the biomedical images to obtain an image where homological information is as explicit as possible. Subsequently, using programs whose correctness has been verified in the Coq/SSReflect proof assistant, homological properties from the pre-processed image are obtained, which in turn are interpreted as features of the original image.

This methodology has been applied in this paper to the problem of determining the number of synapses of a neuron. In this case, the problem is reduced to measure the number of connected components of a monochromatic image. An issue which can be solved, even if it is not the straightforward manner, thanks to the computation of the homology group in dimension 0 of the image.

The use of certified tools able to compute homology groups will be important in the future; for instance, to recognize the structure of a neuron; a problem which seems to involve the homology group in dimension 1, see [20]. Other techniques, like the ones of persistent homology, could be applied in stacks of neurons to remove the noise of the images and help to the detection of the dendrites (the branches of the neuron).

Some formalization aspects also remain as future work. We have already mentioned the on-going work around proving the correctness of the admissible discrete vector fields programs. Moreover, certifying the correctness of integer homology computation is also further work (some results about the formalization of the Smith Normal Form are already encoded in Coq, see [4]).

As we previously mentioned, we are still working on efficiency issues but switching to better representations and more efficient algorithms will not require to redo the proofs related to homology.

References

1. ForMath: formalisation of mathematics. `http://wiki.portal.chalmers.se/cse/pmwiki.php/ForMath/ForMath`.
2. M. Armand, B. Grégoire, A. Spiwack, and L. Théry. Extending coq with imperative features and its application to SAT verification. In M. Kaufmann and L. C. Paulson, editors, *Interactive Theorem Proving*, volume 6172, pages 83–98. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
3. F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon Morse-Smale complex and the Connolly function. In *Proceedings 19th ACM Symposium on Computational Geometry (SCG'03)*, pages 351–360, 2003.
4. C. Cohen, M. Dénès, A. Mörtberg, and V. Siles. Smith Normal form and executable rank for matrices. `http://wiki.portal.chalmers.se/cse/pmwiki.php/ForMath/ProofExamples`.
5. Coq development team. The Coq Proof Assistant Reference Manual, version 8.3. Technical report, 2010.
6. G. Cuesto et al. Phosphoinositide-3-Kinase Activation Controls Synaptogenesis and Spinogenesis in Hippocampal Neurons. *The Journal of Neuroscience*, 31(8):2721–2733, 2011.
7. R. Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998.
8. G. Gonthier. *Formal proof - The Four-Color Theorem*, volume 55. Notices of the American Mathematical Society, 2008.
9. 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`.
10. B. Grégoire and X. Leroy. A compiled implementation of strong reduction. In *Proceedings of the seventh ACM SIGPLAN international conference on Functional programming*, ICFP '02, page 235–246, New York, NY, USA, 2002. ACM.
11. A. Gyulassy, P. Bremer, B. Hamann, and V. Pascucci. A practical approach to Morse-Smale complex computation: Scalability and generality. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1619–1626, 2008.

12. S. Harker et al. The Efficiency of a Homology Algorithm based on Discrete Morse Theory and Coreductions. In *Proceedings 3rd International Workshop on Computational Topology in Image Context (CTIC'2010)*, volume 1 of *Image A*, pages 41–47, 2010.
13. J. Heras, G. Mata, M. Poza, and J. Rubio. Homological processing of biomedical digital images: automation and certification. Technical report, 2010. `http://wiki.portal.chalmers.se/cse/uploads/ForMath/hpbdiac`.
14. J. Heras, M. Poza, M. Dénès, and L. Rideau. Incidence simplicial matrices formalized in Coq/SSReflect. In *Proceedings 18th Symposium on the Integration of Symbolic Computation and Mechanised Reasoning (Calculemus'2011)*, volume 6824 of *Lectures Notes in Computer Science*, pages 30–44, 2011.
15. G. Jerse and N. M. Kosta. Tracking features in image sequences using discrete Morse functions. In *Proceedings 3rd International Workshop on Computational Topology in Image Context (CTIC'2010)*, volume 1 of *Image A*, pages 27–32, 2010.
16. R. Krebbers and B. Spitters. Computer certified efficient exact reals in coq. In J. H. Davenport, W. M. Farmer, J. Urban, and F. Rabe, editors, *Intelligent Computer Mathematics*, volume 6824, pages 90–106. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
17. G. Mata. SynapsCountJ. University of La Rioja, 2011. `http://imagejdocu.tudor.lu/doku.php?id=plugin:utilities:synapsescountj:start`.
18. Mathieu Boespflug, Maxime Dénès, and Benjamin Grégoire. Full reduction at full throttle. In Jean-Pierre Jouannaud and Zhong Shao, editors, *Certified Programs and Proofs*, volume 7086, pages 362–377. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
19. H. Molina-Abril and P. Real. A homological-based description of subdivided nD objects. In *Proceedings of the 14th international conference on Computer analysis of images and patterns (CAIP'2011)*, volume 6854 of *Lectures Notes in Computer Science*, pages 42–50, 2011.
20. M. Mrozek et al. Homological methods for extraction and analysis of linear features in multidimensional images. *Pattern Recognition*, 45(1):285–298, 2012.
21. F. L. R. node. From Digital Images to Simplicial Complexes: A report. Technical report, 2011. `http://wiki.portal.chalmers.se/cse/uploads/ForMath/fditscr`.
22. W. S. Rasband. ImageJ: Image Processing and Analysis in Java, 2003. `http://rsb.info.nih.gov/ij/`.
23. P. Real and H. Molina-Abril. Towards Optimality in Discrete Morse Theory through Chain Homotopies. In *Proceedings 3rd International Workshop on Computational Topology in Image Context (CTIC'2010)*, volume 1 of *Image A*, pages 33–40, 2010.
24. V. Robins, P. Wood, and A. Sheppard. Theory and algorithms for constructing discrete Morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1646 – 1658, 2011.
25. A. Romero and F. Sergeraert. Discrete Vector Fields and Fundamental Algebraic Topology, 2010. `http://arxiv.org/abs/1005.5685v1`.
26. D. J. Selkoe. Alzheimer's disease is a synaptic failure. *Science*, 298(5594):789–791, 2002.
27. D. Ziou and M. Allili. Generating Cubical Complexes from Image Data and Computation of the Euler number. *Pattern Recognition*, 35:2833–2839, 2002.