Cubical complexes

Meggy-Lie-Anne Chamand, Ana Gelez

This research project focuses on the implementation of homology and persistent homology algorithms for cubical complexes - mathematical structures that provide a combinatorial description of spaces by using cubes as building blocks. Usually, those two concepts are computed for filtrations arising from simplicial complexes, but in certain cases (e.g. 3D models, bitmap pictures etc) the use of cubical decomposition is more appropriate. its applications on different surfaces (e.g. Klein bottle, torus, projective plane). We were able to build the boundary matrices and the persistent homology barcodes for complexes built in $\mathbb{Z}_2$.

Introduction

When talking about homology and persistent homology, it is often assumed that we are studying simplicial complexes and their filtrations. However, in some areas, data is better represented with **cubical complexes**, in which simplices are replaced with cubes. For instance, it is the case when dealing with an image made of square pixels, or a video which can be seen as a three-dimensional image, where the third dimension is time.

For instance, they offer a useful framework for representing pixel-based data, such as bitmap images, where each pixel corresponds naturally to a cube. In this context, the grid-like structure of cubical complexes aligns smoothly with the structure of pixel grids. Moreover, the boundary definition in cubical complexes is particularly advantageous for tasks related to image processing. In a cubical complex, determining the boundary of a cube is straightforward, making it well-suited for identifying neighboring pixels in an image.

More formally, a cube can be defined as a product of elementary intervals. An elementary interval is either degenerate ($[i, i]$, $i \in \mathrm{N}$) or non-degenerate ($[i, i + 1]$). A 2-dimensional cube (square) spanning from (1, 3) to (2, 4) would then be represented as:

$$[1, 2] \times [3, 4]$$

Similarly to a simplicial complex, we can define a cubical complex as a set of cubes such that if a cube is part of the complex, then its boundary is part of it too.

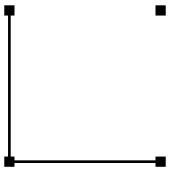


Figure 1: Example of a cubical complex

Using similar definitions as for simplicial complex, it is possible to define homology and persistent homology for this kind of complex.

Method

Representing cubical complexes

We can improve the efficiency of the representation of cubical complexes by taking advantage of their structure. In a d-dimensional complex, each cube can be efficiently represented as an element of $\mathbb{N}^d$. Here, each element of this vector corresponds to a dimension. If the coordinate is even, the corresponding interval in the product is degenerate; if it is odd, it is non-degenerate. The interval corresponding to a coordinate x starts at $x \div 2$ (using Euclidean division).

The complex illustrated in Figure 1 can be represented using this set (assuming the y-axis points down, and (0, 0) is in the upper-left corner):

$$\left\{\binom{0}{0}, \binom{0}{2}, \binom{2}{0}, \binom{2}{2}, \binom{0}{1}, \binom{1}{2}\right\}$$

Each element of this set corresponds to a cube in the complex. The first coordinate in each pair represents the x-coordinate, and the second coordinate represents the y-coordinate. For instance, $\binom{0}{2}$ gives us $[0, 0] \times [1, 1]$, corresponding to the 0-dimensional cube (vertex) (0,1) ; $\binom{1}{2}$ gives us $[0, 1] \times [1, 1]$, correponding to the 1-dimensional cube (edge) [0,1].

This representation is much more compact and less ambiguous than listing the vertices that are part of each cube.

Computing homology

It is then possible to compute homology, and more generaly, persistent homology. To do so, we implemented the algorithm described by Wagner *et al.* [1].

Using a filtration, that is represented as a function that assigns to each vertex an index, we build a generalization of this filtration, that assigns an index to all cubes in the complex. Cubes can then be sorted according to their assigned index. The order of the vertices is preserved, and a non-vertex cube is guaranteed to be after all its vertices.

We use this ordering to build a boundary matrix: each column corresponds to a cube in the complex (the i-th column corresponds to the cube with filtration index i to be exact). The j-th element in this column is 1 if the cube with filtration index j is part of the boundary of the i-th cube (otherwise, it is 0).

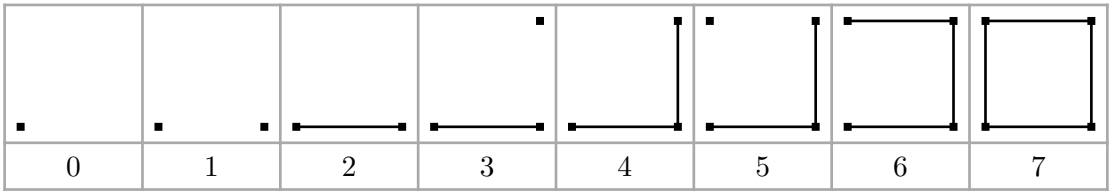
Using the representation described above, it is easy to compute the boundary of any cube: for each coordinate x, if it is non-degenerate, the boundary includes two cubes that both share the same coordinates as the cube excepted for x which is replaced with $x - 1$ and $x + 1$ respectively.

Afterwards, the matrix is reduced, using the algorithm proposed by Chen *et al.* [2]. We can then exploit the equations provided in this same article to build persistence pairs.

Results

A simple example

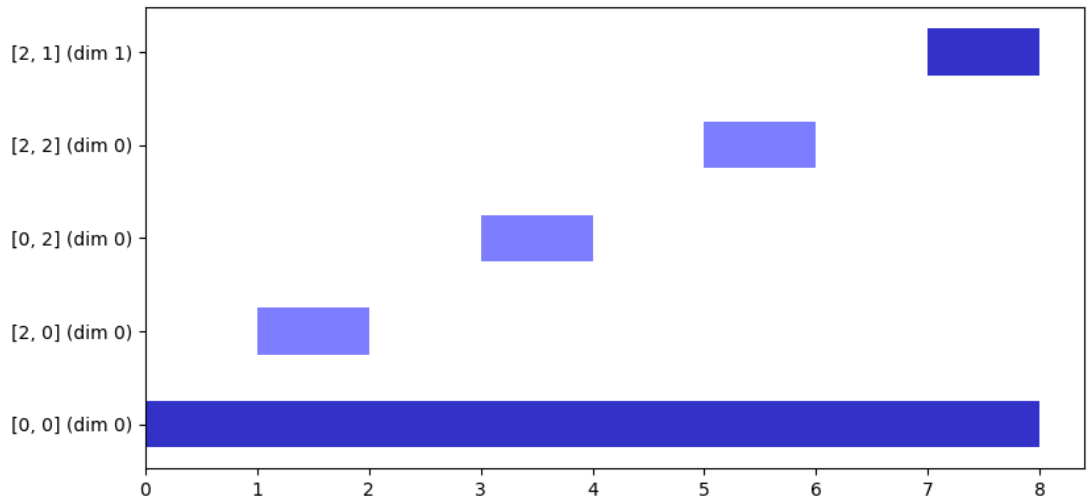
We decided to test our code on a simple example, drawn below (the number below each step is the filtration index of the cube that is added at this step):



Our algorithm gives the following boundary matrix, which is then reduced:

$$\begin{pmatrix} 0&0&1&0&1&0&0&0\\ 0&0&1&0&0&0&0&1\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&1\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{pmatrix} \rightarrow \begin{pmatrix} 0&0&1&0&1&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{pmatrix}$$

If we build the homology pairs and plot them on a bar diagram, this is what we obtain.



The dark blue bars are homology generators: the longest one is for the 0-dimension generator (one connected component) and the short one is for the 1-dimension generator (the loop that appears at the last step). The light blue bars correspond to the addition of a vertex followed by the addition of a line: a new connected component is created and then immediatly killed.

Some classic examples

For these complexes, we used data from SageMath that we converted to our own format. Here are the bar charts that we obtain:

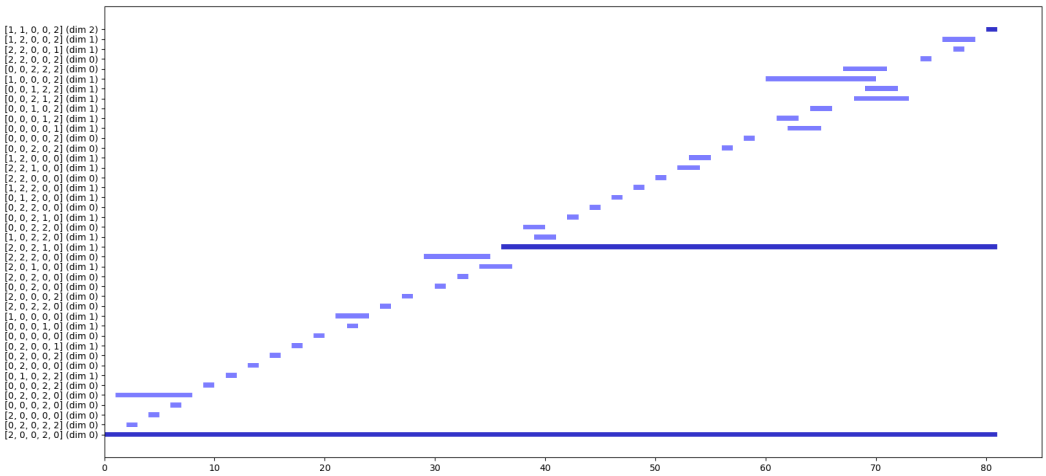


Figure 3: Projective plane bars

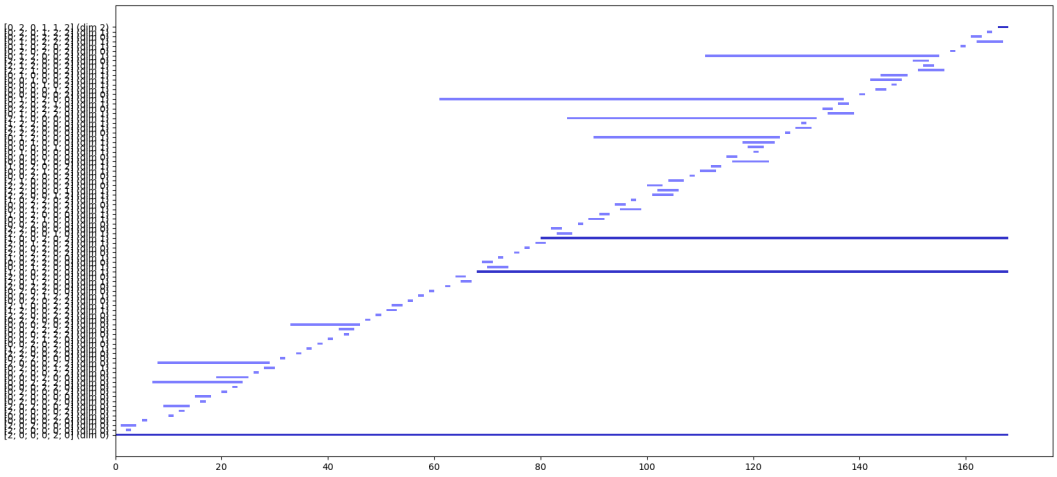


Figure 4: Klein bottle bars

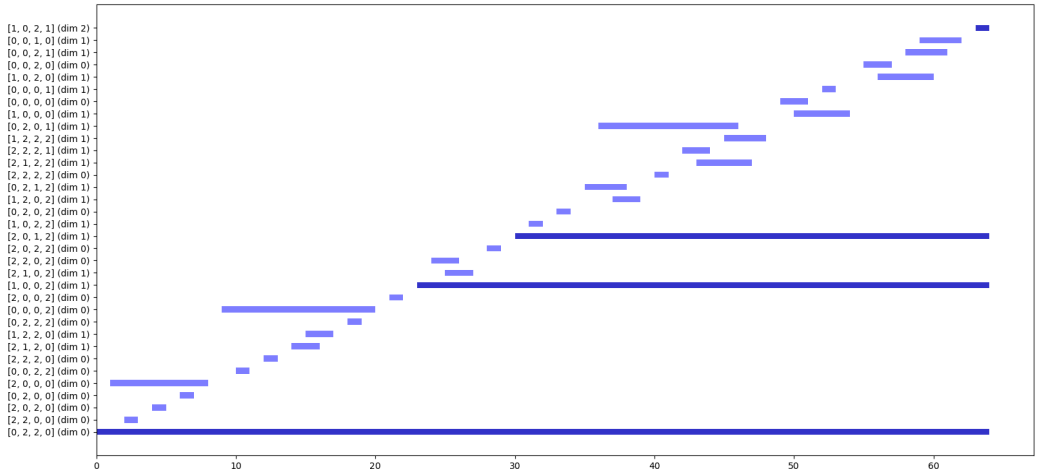


Figure 5: Torus bars

We can see that the number of generator is what we would expect. For instance, the torus has one generator in dimension 0, two in dimension one, and one in dimension two.

Discussion

Our algorithm seems to correctly compute persistent homology for cubical complexes. Unfortunately, due to time constraints, we were unable to implement the persistent homology algorithm using various coefficients. As it is, our code only supports $\mathbb{Z}_2$.

The choice of coefficients in persistent homology are important for the detail of the captured topological features. For instance, using $\mathbb{Z}_2$ coefficients simplifies computations and is well-suited for applications in image processing and computer graphics.

Extending the model to support other coefficients would require we use a more general definition of boundaries (e.g. in $\mathbb{Z}$, cubes in a boundary are oriented). However, this comes at the cost of increased computational complexity. The reduction algorithm should also be adapted to work in a different field, when summing columns together.

Conclusion

In conclusion, this research project delved into the implementation of homology and persistent homology algorithms specifically applied to cubical complexes. While traditional approaches focus on simplicial complexes, cubical complexes provide a more suitable representation for certain types of data, such as pixel-based images or three-dimensional structures incorporating time.

The method presented efficiently represents cubical complexes, and allow for a more effective representation compared to listing individual vertices. The implemented algorithms for computing homology and persistent homology were based on the work of Wagner et al. [1], with an emphasis on using filtrations to assign indices to cubes and building boundary matrices.

Results from testing the code on both a simple example and classic mathematical surfaces, such as the Klein bottle, torus, and projective plane, gave expected outcomes. The homology generators and their corresponding dimensions align with the theoretical expectations. However, our current implementation supports only $\mathbb{Z}_2$ coefficients. It can be noted that $\mathbb{Z}_2$ coefficients are suitable for applications in image processing and computer graphics. Also, extending the model to accommodate other coefficients would involve a more general definition of boundaries.

Further work could involve refining the model to support additional coefficients, adapting the reduction algorithm accordingly, and exploring applications in various fields where cubical complexes play a crucial role.

Division of work

Ana Gelez : researches on cubical complexes and persistent homology, implementation of the algorithm, tests, report Meggy-Lie-Anne Chamand : researches on cubical complexes and persistent homology, report

References

[1] H. Wagner, C. Chen, and E. Vuçini, "Efficient Computation of Persistent Homology for Cubical Data", in *Topological Methods in Data Analysis and Visualization II: Theory, Algorithms, and Applications*, R. Peikert, H. Hauser, H. Carr, and R. Fuchs, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 91–106. doi: 10.1007/978-3-642-23175-9_7.

[2] C. Chen and M. Kerber, "Persistent Homology Computation with a Twist", 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:18232748