Computation of persistent homology on metric graphs

Advanced Programming for Scientific Computing

Author: Gurrieri Davide
Supervisor: Zunino Paolo
Professors: Formaggia Luca, Africa Pasquale Claudio

September 19, 2023

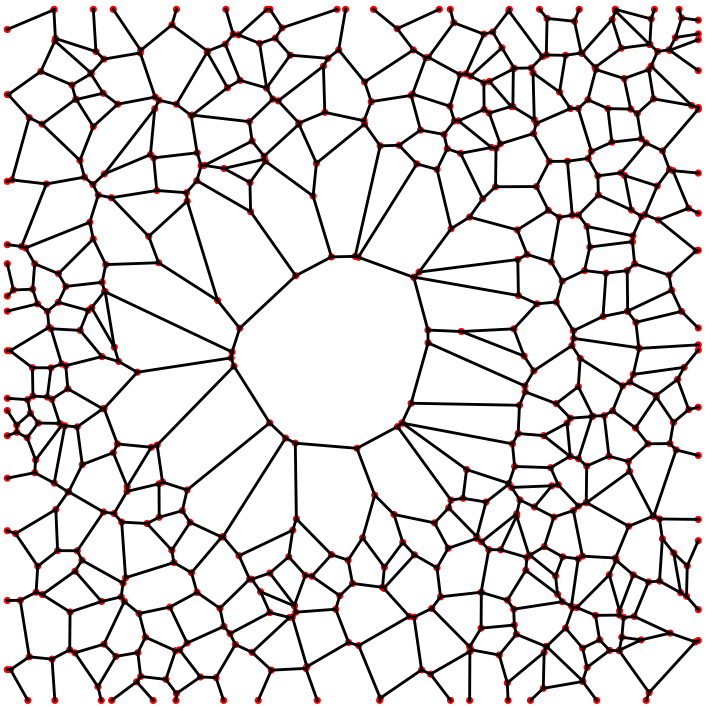
POLITECNICO
MILANO 1863

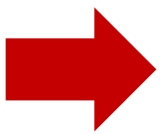
Overview

1. Project context and objective
2. Algebraic topology and persistent homology
3. Methods
4. Implementation
5. Results
6. Conclusions

Project context and objective

Microcirculation characterisation





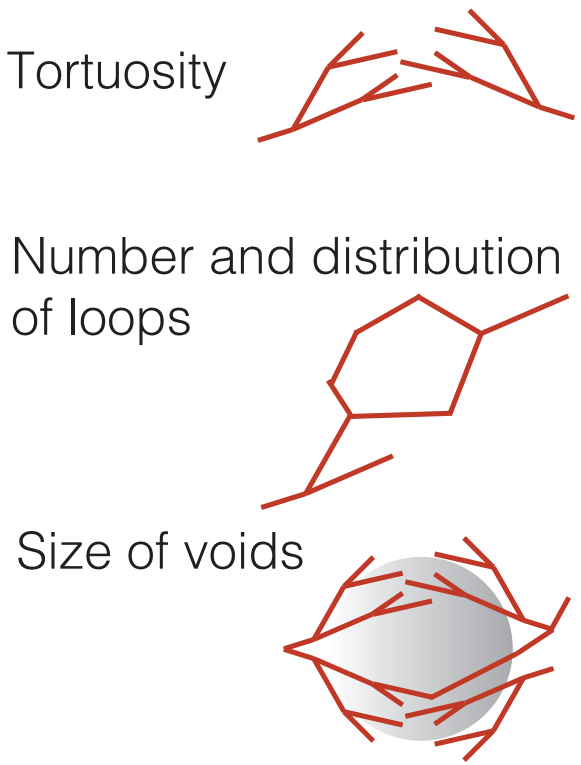


Figure 1: Generated micro-circulation network.

Tumour detection

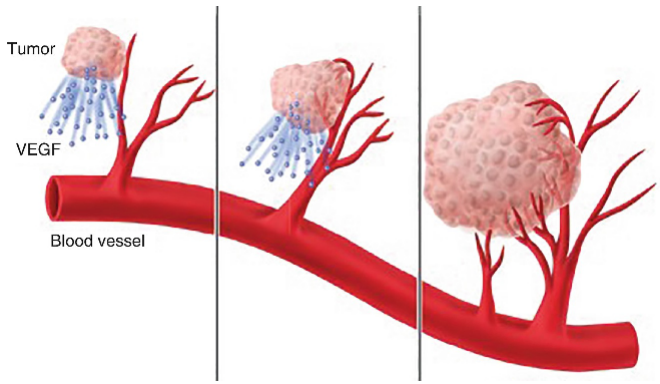


Figure 2: Angiogenesis: growth of new blood vessels from existing ones.

Characteristics:

- Highly tortous, irregular branching
- Presence of non-vascularized regions and loops

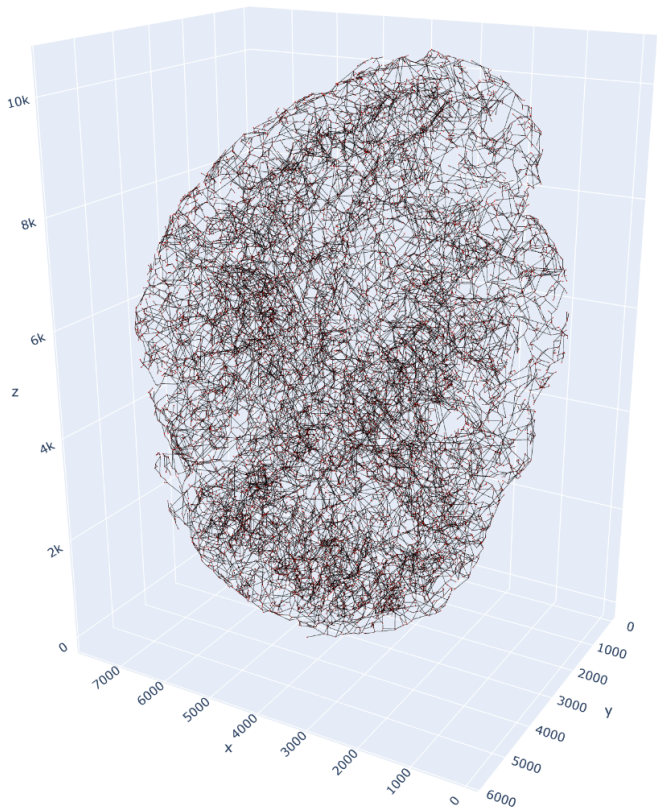


Figure 3: Tumour vascular network of a mouse brain.

Algebraic topology and persistent homology

Homology

Homology measures the topological equivalence of spaces through algebraic structures. Their dimensions are called *Betti numbers*.

β_0 Connected components
β_1 Tunnels
β_2 Voids

Space	β_0	β_1	β_2
Point	1	0	0
Cube	1	0	1
Sphere	1	0	1
Torus	1	2	1

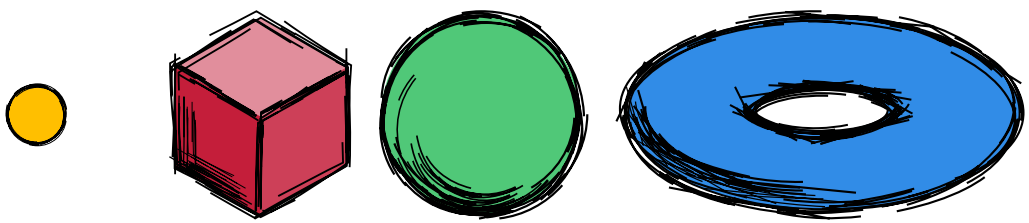


Figure 4: Betti numbers of common shapes.

Simplicial complexes

Simplicial complex

Given a set K_0, a *simplicial complex* is a collection K of non-empty subsets of K_0 such that:

- $v \in K \quad \forall\, v \in K_0$
- $\tau \subset \sigma,\ \sigma \in K \implies \tau \in K$

Elements of K_0 are called *vertices*, elements of K are called *simplices*.

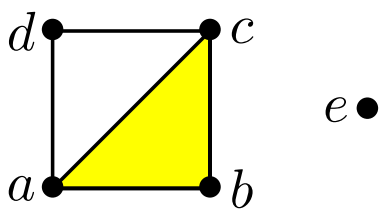
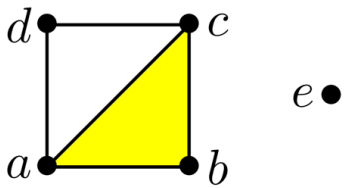


Figure 5:
$K = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{c, d\}, \{a, b, c\}\}$

Homology computation

The computation of Betti numbers boils down to linear algebra.



$$d_1 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} ab & ac & ad & bc & cd \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \qquad d_2 = \begin{array}{c} \\ ab \\ ac \\ ad \\ bc \\ cd \end{array} \begin{array}{c} \begin{array}{c} abc \end{array} \\ \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{array}$$

1. Calculate boundary operator matrices
2. Bring each matrix into Smith normal form (similar to Gaussian elimination)
3. Read off description of homology group

Boundary matrices are reduced in at most cubic time in the number of simplices in K.

Persistent homology

Filtration

Given K a simplicial complex, a *filtration* of K is a sequence of embedded simplicial complexes:

$$\varnothing = K_0 \subseteq K_1 \subseteq K_2 \subseteq \cdots \subseteq K_n = K$$

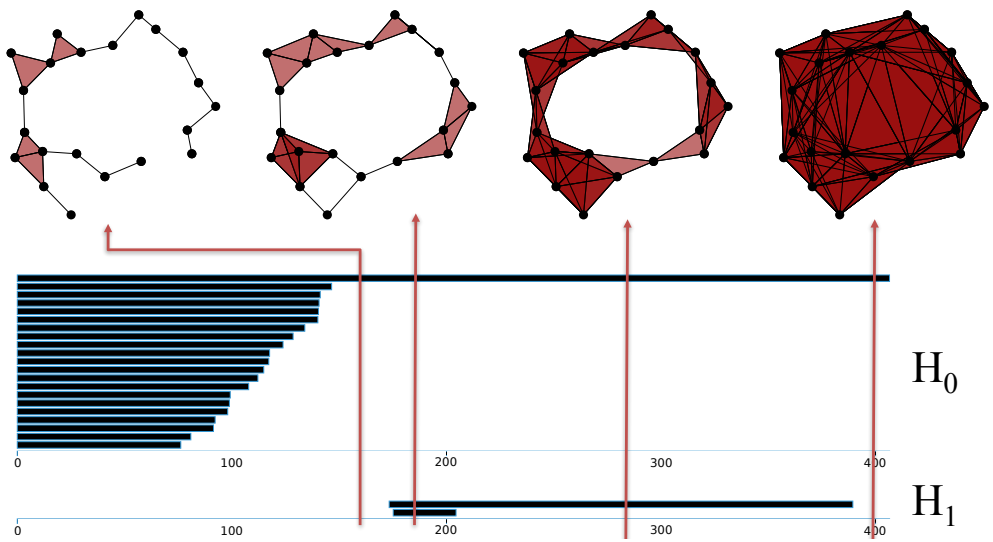
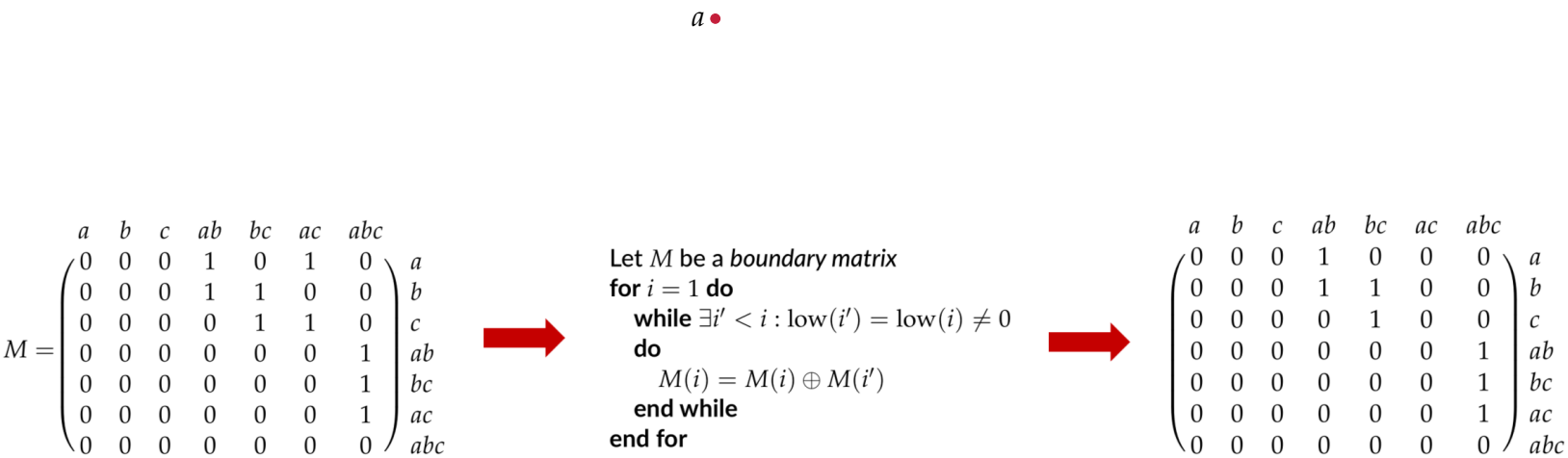


Figure 6: Vietoris-Rips filtration.

Persistent homology computation



- If column i is empty, then σ_i is a positive simplex that creates a topological feature
- If column j is non-empty with $\text{low}(j) = k$, then σ_j is a negative simplex that destroys the topological feature created by σ_k

Methods

Complete pipeline

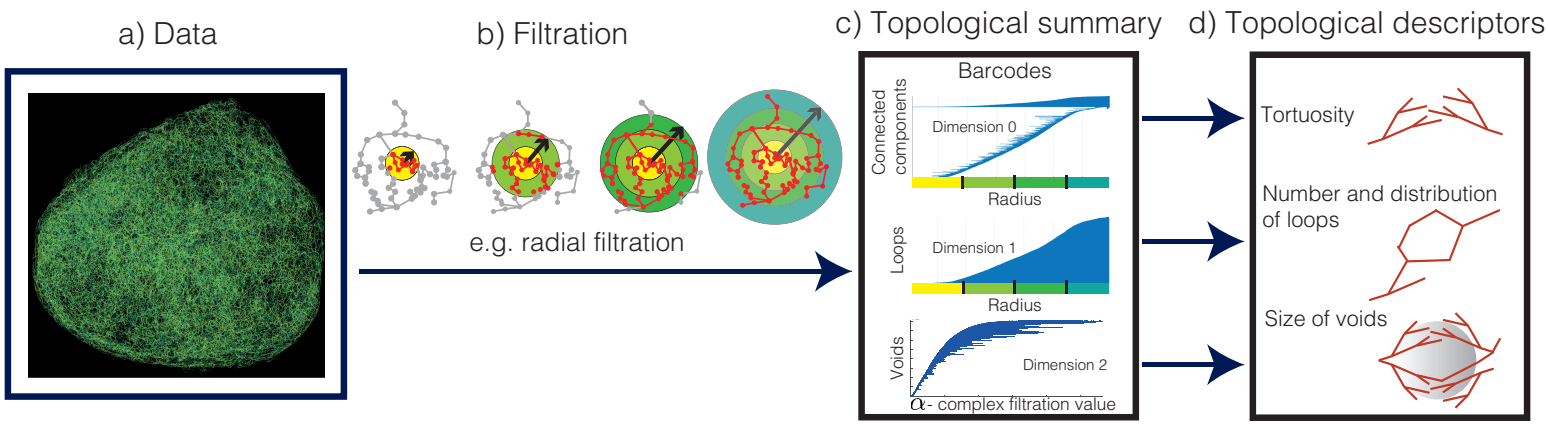
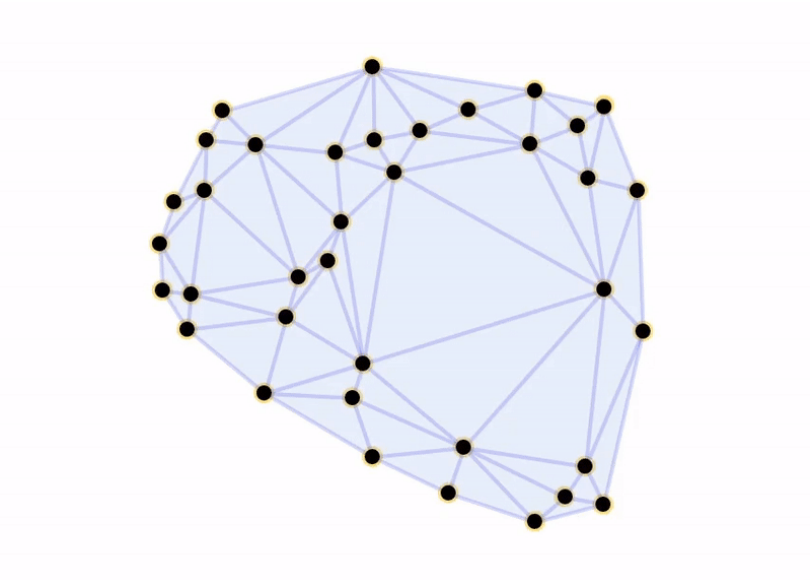


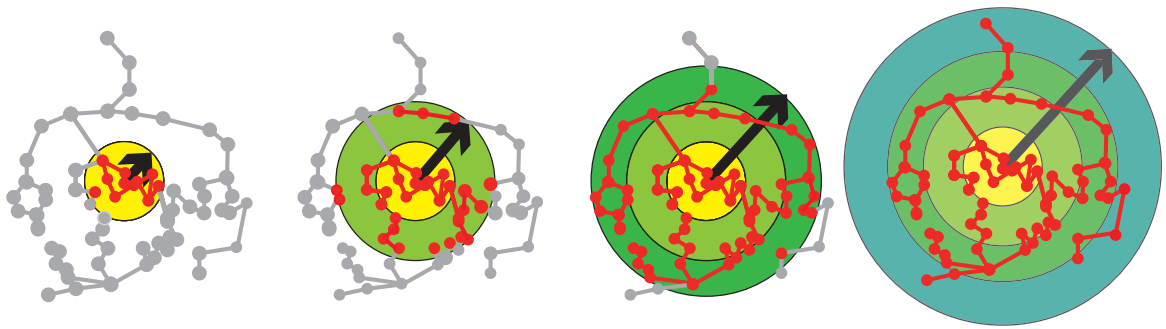
Figure 7: Topological analysis for vascular networks.

Alpha filtration



- Uses only vertices spatial information
- Built by selecting simplices of the *Delaunay complex*
- Allows to identify voids and estimate their size
- Small total number of simplices: grows as $O(n^2)$ in $\mathbb{R}^3$

Radial filtration



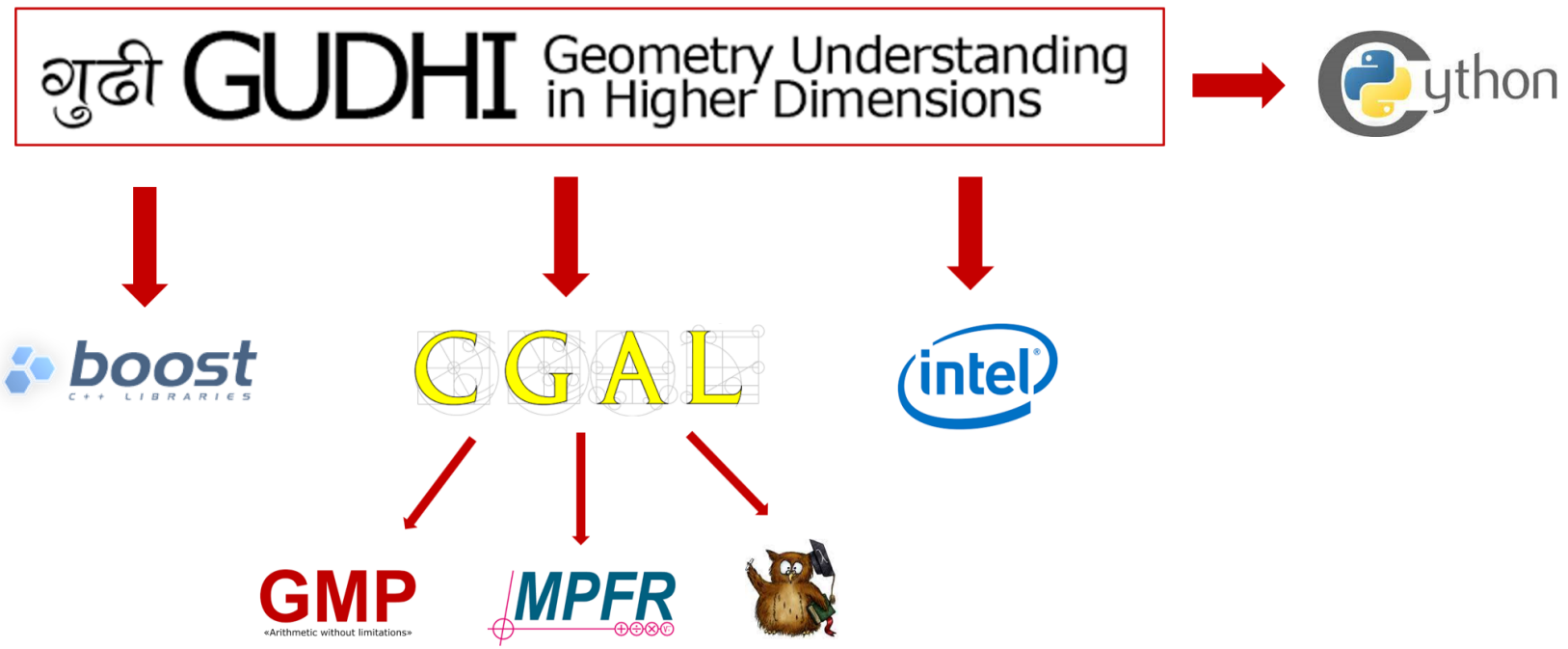
- Uses the complete vascular network, seen as a 1-dimensional simplicial complex
- Vertices and vessels are progressively added growing a sphere from the center
- Characterizes the tortuosity using connected components barcodes
- Identifies loops and their radial position with respect to the center

Implementation

Libraries for TDA

- Efficient implementation of data structures and algorithms is of paramount importance
- There are many well-known libraries for topological data analysis, including: Ripser, Dionysus, JavaPlex, PHAT, Perseus and GUDHI
- GUDHI stands out for the following reasons:
  - Efficient and highly optimized
  - A large range of tools available
  - Well-kept documentation and helpful tutorials
  - Interfaces to Python and R
  - Supported by a Grant of the European Research Council and hosted by INRIA

GUDHI


गुढी GUDHI Geometry Understanding in Higher Dimensions
ython
boost
C++ LIBRARIES
CGAL
intel
GMP
«Arithmetic without limitations»
MPFR

Project structure

```
data/     complex_network/
          synthetic_network/

docs/     Doxyfile

output/   plot/
          plot_python/

python/   requirements.txt
          network_analysis.py
          real_networks_analysis.ipynb
          synthetic_network_analysis.ipynb
          plot_ph_from_cpp_file.py

src/      CMakeLists.txt
          GetPot
          chrono.hpp
          geometry.h
          main.cpp
          network_analysis.cpp
          network_analysis.h
          options1.txt
          options2.txt
```

C++ implementation

```
class Filtration {
public:
  using SimplexTree = Gudhi::Simplex_tree<>;
  using Filtration_value = SimplexTree::Filtration_value;
  using Field_Zp = Gudhi::persistent_cohomology::Field_Zp;
  using Persistent_cohomology =
      Gudhi::persistent_cohomology::Persistent_cohomology<SimplexTree,
                                                          Field_Zp>;
protected:
  SimplexTree simplex_tree;
  unique_ptr<Persistent_cohomology> persistent_cohomology = nullptr;
  Chrono chrono;
public:
  void compute_persistent_cohomology(bool persistence_dim_max);
  void save_persistence(const string &filename) const;
  void print_complex_info();
  void print_range_simplices(unsigned start_index, unsigned length);
  void print_betti_numbers() const;
  void print_elapsed_time() const;
  virtual void make_analysis() = 0;
  virtual void print_analysis() = 0;
};
```

C++ implementation

```
class AlphaFiltration : public Filtration {
public:
  using Kernel = CGAL::Epeck_d<CGAL::Dynamic_dimension_tag>;
  using AlphaPoint = Kernel::Point_d;
  using Alpha_complex = Gudhi::alpha_complex::Alpha_complex<Kernel>;
private:
  vector<double> voids_diameter;
  vector<std::pair<Filtration_value, Filtration_value>>
       voids_persistence;
public:
  AlphaFiltration() = default;
  AlphaFiltration(const vector<Point<CoordType>> &points);
  void compute_voids_diameter();
  void print_diameters(unsigned nmax) const;
  void make_analysis() override;
  void print_analysis() override;
};
```

C++ implementation

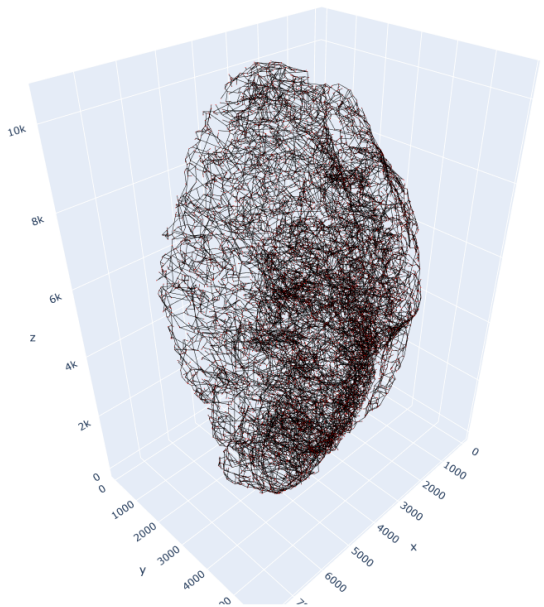
```
class RadialFiltration : public Filtration {
public:
  unsigned n_edges;
  double max_radius;
  double tortuosity_descriptor;
  double loops_descriptor;

  RadialFiltration() = default;
  RadialFiltration(const vector<Point<CoordType>> &points,
                   const vector<Edge> &edges);
  void compute_descriptors();
  void make_analysis() override;
  void print_analysis() override;
};
```

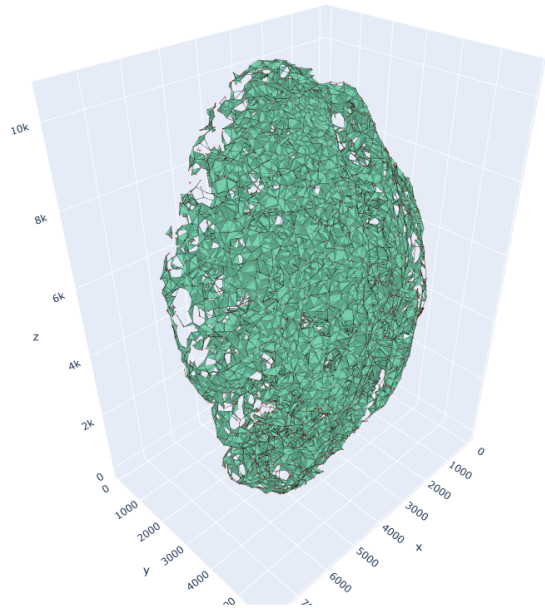
C++ implementation

```
class NetworkAnalysis {
private:
  const NetworkData *data;
  std::vector<std::unique_ptr<Filtration>> filtrations;
public:
  NetworkAnalysis() = default;
  NetworkAnalysis(const NetworkData *data_);
  void analyse() const;
  void print_global_analysis() const;
  void save_persistence(const std::string &name) const;
};
```

Python implementation

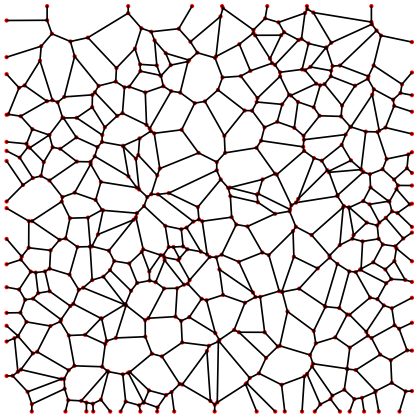
- Easier way to take advantage of GUDHI, mantaining its performance
- Pre-compiled module available: `pip install gudhi`
- Useful functions for visualizing vascular networks, filtrations and the results of persistent homology computation



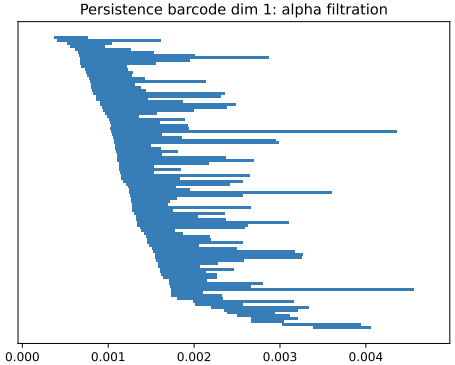


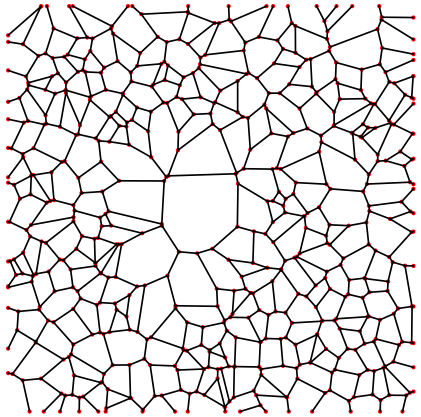
Results

Synthetic networks: Alpha filtration

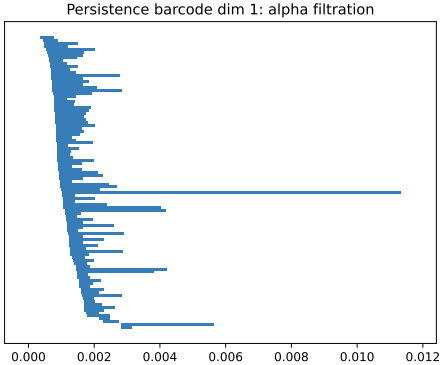


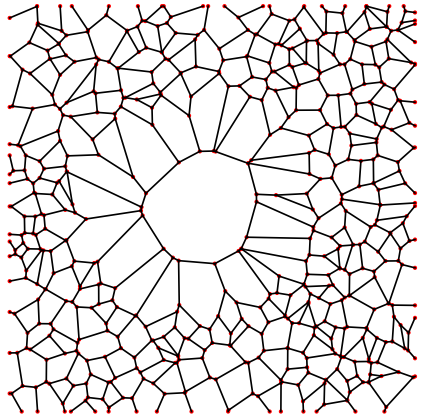
(birth, death)	$\sqrt{p}$	diameter
(0.00104, 0.00436)	0.05762	0.13208
(0.00173, 0.00456)	0.05319	0.13503
(0.00125, 0.00360)	0.04852	0.12004
(0.00067, 0.00287)	0.04689	0.10711
(0.00108, 0.00299)	0.04372	0.10934



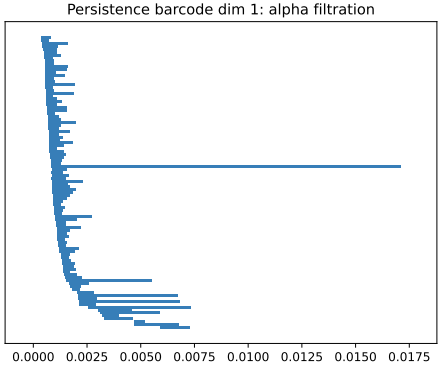


(birth, death)	$\sqrt{p}$	diameter
(0.00103, 0.01132)	0.10143	0.21275
(0.00111, 0.00417)	0.05533	0.12913
(0.00106, 0.00402)	0.05441	0.12681
(0.00282, 0.00563)	0.05295	0.15004
(0.00146, 0.00423)	0.05260	0.13002





(birth, death)	$\sqrt{p}$	diameter
(0.00084, 0.01711)	0.12755	0.26161
(0.00257, 0.00735)	0.06913	0.17146
(0.00212, 0.00682)	0.06856	0.16512
(0.00206, 0.00671)	0.06818	0.16382
(0.00156, 0.00552)	0.06294	0.14862



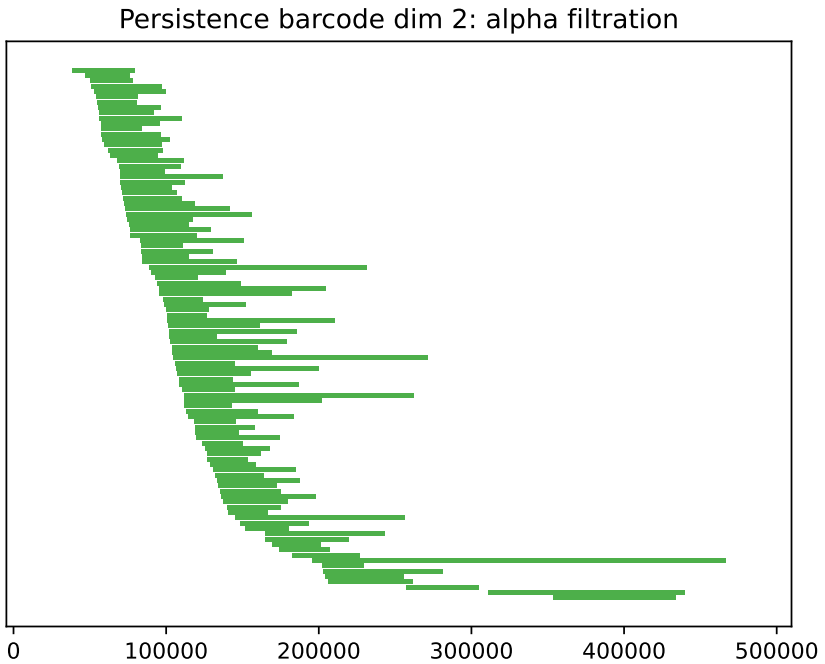
Synthetic networks: Radial filtration

	Network 0	Network 1	Network 2
Loops number	210	223	229
Loops descriptor	0.2811	0.2802	0.2799
Tortuosity descriptor	0	0.0013	0.0012

- Synthetic networks have approximatively the same number of loops
- Patological networks have higher tortuosity descriptor

Real network: Alpha filtration

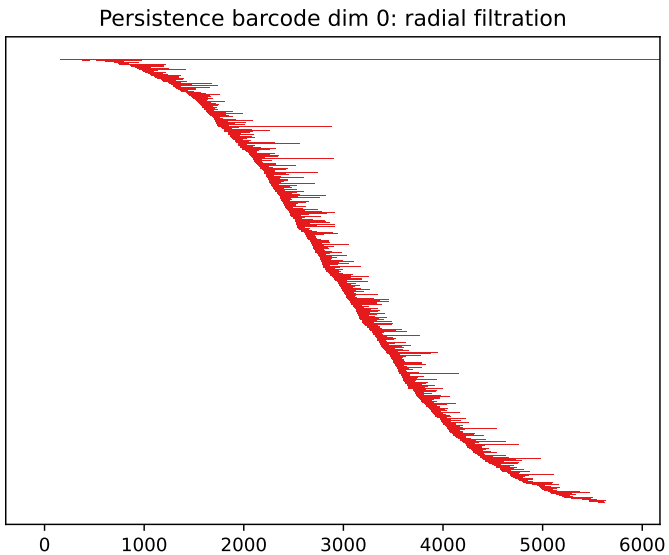
(birth, death)	$\sqrt{p}$	diameter
(195758, 466836)	520.652	1366.51
(104306, 271075)	408.374	1041.3
(111517, 261906)	387.801	1023.54
(89133.9, 231368)	377.139	962.014
(311026, 439792)	358.84	1326.34
(145373, 256354)	333.138	1012.63
(100880, 210850)	331.617	918.369
(95344.2, 204643)	330.604	904.75
(106776, 200246)	305.729	894.976
(111536, 201720)	300.307	898.266

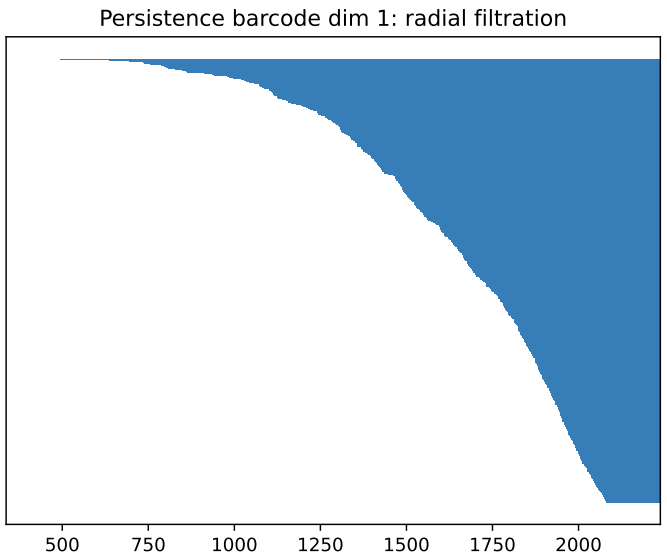


- Most persistent and large voids are highlighted
- Possible interpretation: areas of necrosis due to tumor's rapid growth
- Voids descriptor could also be defined

Real network: Radial filtration

Descriptor	Value
Loops number	5857
Loops descriptor	0.2446
Tortuosity descriptor	0.1106





- Intervals of connected components are used to define the tortuosity descriptor
- Loops birth gives their radial position with respect to the center

Conclusions

Conclusions

The primary contribution lies in an unified, structured and efficient implementation of a framework for topological analysis of vascular networks.

This work can lead to important advantages in these two fields:

- **Research:** incorporates topological information to simulate vascular networks more realistically
- **Clinical:** obtains targeted therapies for patients

References i


J.-D. Boissonnat, T. K. Dey, and C. Maria.
The compressed annotation matrix: An efficient data structure for computing persistent cohomology.
In *European Symposium on Algorithms*, pages 695–706. Springer, 2013.


J.-D. Boissonnat and C. Maria.
The simplex tree: An efficient data structure for general simplicial complexes.
Algorithmica, 70:406–427, 2014.


T. K. Dey, F. Fan, and Y. Wang.
Computing topological persistence for simplicial maps.
In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 345–354, 2014.

References ii


T. K. Dey and Y. Wang.
Computational topology for data analysis.
Cambridge University Press, 2022.


H. Edelsbrunner and J. L. Harer.
Computational topology: an introduction.
American Mathematical Society, 2022.


N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington.
A roadmap for the computation of persistent homology.
EPJ Data Science, 6:1–38, 2017.


B. Stolz, J. Kaeppler, B. Markelc, F. Mech, F. Lipsmeier, R. Muschel, H. Byrne, and H. Harrington.
Multiscale topology characterises dynamic tumour vascular networks. arxiv.
preprint, 2020.

References iii


P. W. Sweeney, A. d'Esposito, S. Walker-Samuel, and R. J. Shipley. **Modelling the transport of fluid through heterogeneous, whole tumours in silico.** *PLoS computational biology*, 15(6):e1006751, 2019.