Software for persistent homology: overview and demonstration[1]

Nina Otter

ATI scoping workshop
and
CAT research school
Mathematical Institute, University of Oxford

11 September 2015

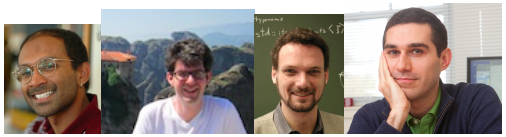
[1] *A roadmap for the computation of persistent homology*, N.O, M. Porter, U. Tillmann, P. Grindrod, H. Harrington, 2015

What is out there?

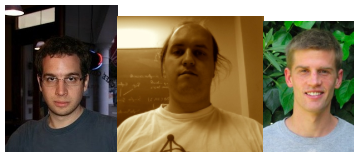
- **Perseus** http://www.sas.upenn.edu/~vnanda/perseus/
- **JavaPlex** http://appliedtopology.github.io/javaplex/
- **Dionysus** http://www.mrzv.org/software/dionysus/
- **pHom** http://cran.r-project.org/web/packages/phom/
- **TDA** http://cran.r-project.org/web/packages/TDA/
- **GAP persistence** http://www-circa.mcs.st-and.ac.uk/~mik/persistence/
- **PHAT** https://code.google.com/p/phat/
- **DIPHA** https://code.google.com/p/dipha/
- **GUDHI** https://project.inria.fr/gudhi/software/
- **Kenzo module** http://www.unirioja.es/cu/anromero/persistent-homology.zip
- **The persistence landscape toolbox**
  https://www.math.upenn.edu/~dlotko/persistenceLandscape.html
- **SimpPers**
  http://web.cse.ohio-state.edu/~tamaldey/SimpPers/SimpPers-software/
- **jHoles** http://cuda.unicam.it/jHoles
- **HoPeS** http://kurlin.org/projects/persistent-skeletons.php
- ...

A brief (biased) history of PH softwares

2000-2004 PH algorithm *H. Edelsbrunner, D. Letscher, G. Carlsson and A. Zomorodian*



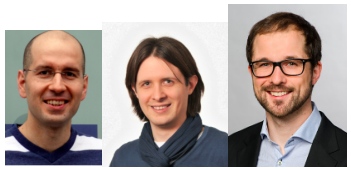
2005 **Plex** *V. de Silva, P. Perry, L. Kettner, A. Zomorodian*



2011 **JavaPlex** *A. Tausz, M. Vejdemo-Johansson, H. Adams*

2012 **Perseus** *V. Nanda*

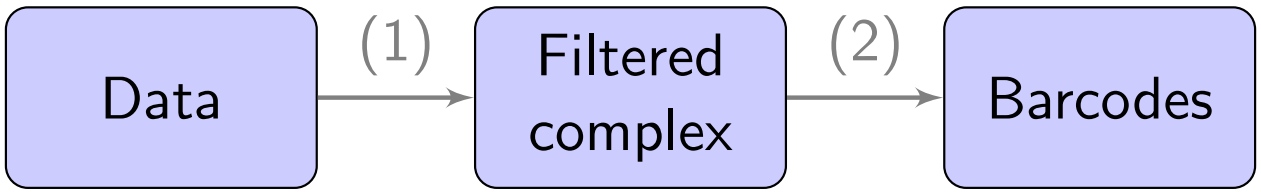
2013 **PHAT** *M. Kerber, J. Reininghaus, U. Bauer, H. Wagner*



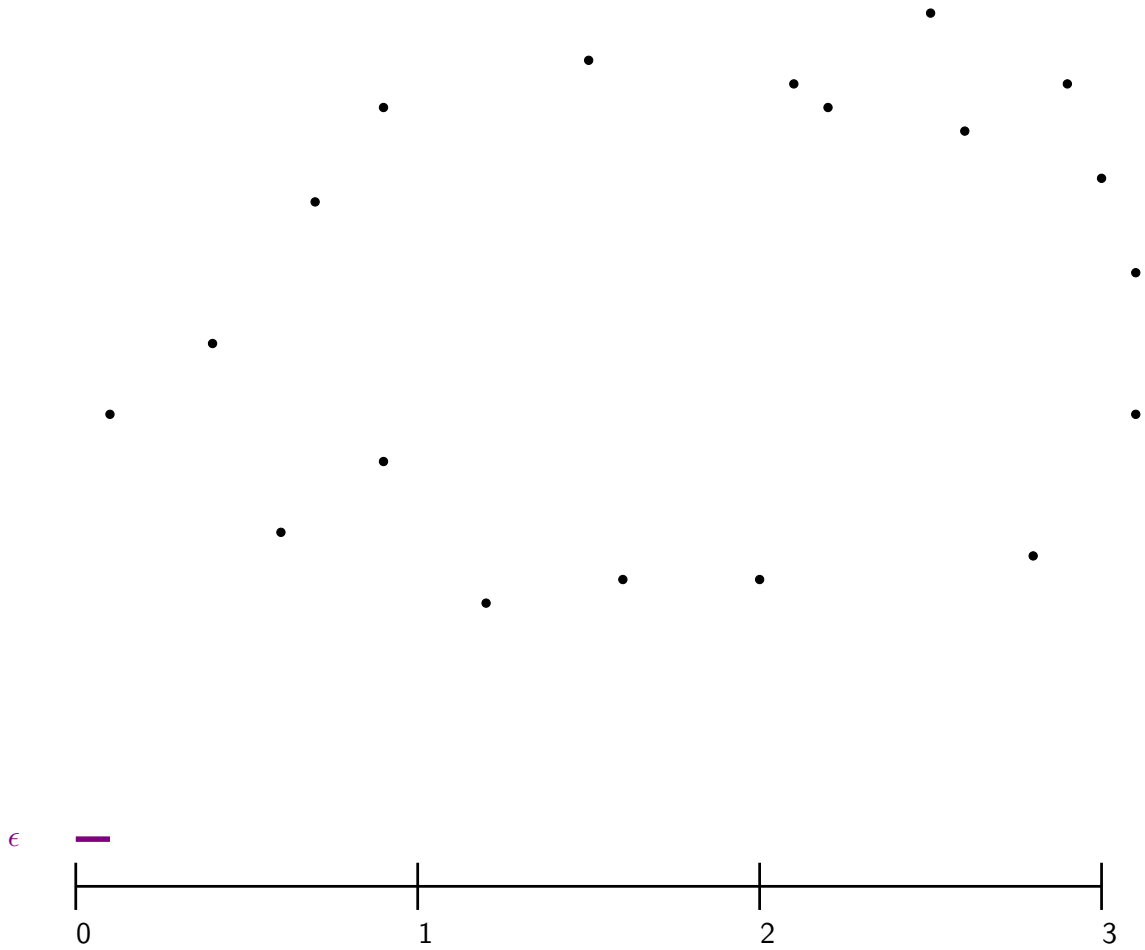
2014 **DIPHA** *M. Kerber, J. Reininghaus, U. Bauer*

2014 **GUDHI** *C. Maria, J.-D. Boissonnat, M. Glisse, M. Yvinec*

Pipeline for PH computation from point clouds



PH in a movie



Optimisations

(1) From data to a filtered complex

Complex K	Size of K	(A) Approximation	(B) Reduction
Čech	$\mathcal{O}(2^n)$	linear size appr. [KS13]	tidy set [Z10] Morse red. [MN13]
Vietoris–Rips (VR)	$\mathcal{O}(2^n)$	linear-size appr. [SH12] (lazy) witness (LW)[dSC14] GIC [DFW13]	tidy set Morse red.
α	$n^{\mathcal{O}(\lfloor d/2 \rfloor)}$ (n points in $\mathbb{R}^d$)	-	tidy set Morse red.

Data structures:

- Simplex tree [Boissonnat, Maria 2012]

[*KS*13] Kerber, Sharathkumar 2013
[*SH*12] Sheehy, 2012
[*Z*10] Zomorodian

[*MN*] Mischaikow, Nanda 2013
[*dSC*14] de Silva, Carlsson 2013
[*DFW*13] Dey, Fankg, Wang 2013

Optimisations: implementations

(1) From data to a filtered complex

Complex K	(A) Approximation	(B) Reduction
Čech $\longrightarrow$ DIONYSUS	linear size appr. $\longrightarrow$?	tidy set $\longrightarrow$ not open source Morse red. $\longrightarrow$ PERSEUS
Vietoris–Rips (VR) $\longrightarrow$ DIONYSUS,PERSEUS, PHAT DIPHA, JAVAPLEX, GUDHI	linear-size appr. $\longrightarrow$? (lazy) witness (LW) $\longrightarrow$ JAVAPLEX GIC $\longrightarrow$?	tidy set Morse red.
α $\longrightarrow$ DIONYSUS, HoPeS (1-D)		tidy set Morse red.

Data structure:

- Simplex tree $\longrightarrow$ GUDHI

Optimisations

(2) From a boundary matrix to barcodes

Sequential:

- Dual algorithm [de Silva, Morozov, Vejdemo-Johansson 2011]
- Twist algorithm [Chen, Kerber 2011]

Parallel:

- Spectral sequence algorithm [Edelsbrunner, Harer 2008]
- Chunk algorithm [Bauer, Kerber, Reininghaus 2013]
- Distributed algorithm [Bauer, Kerber, Reininghaus 2014]

Data structure:

- Bit tree pivot column [Bauer, Kerber, Reininghaus, Wagner 2013]
- Compressed annotation matrix [Boissonnat, Dey, Maria 2013]

Parallel optimisation: example

Spectral sequence algorithm

Let k_j denote the number of simplices in subcomplex K_j.

Let B^j denote the columns numbered $k_{j-1}+1$ to k_j.

Let B_i denote the rows numbered $k_{i-1}+1$ to k_i.

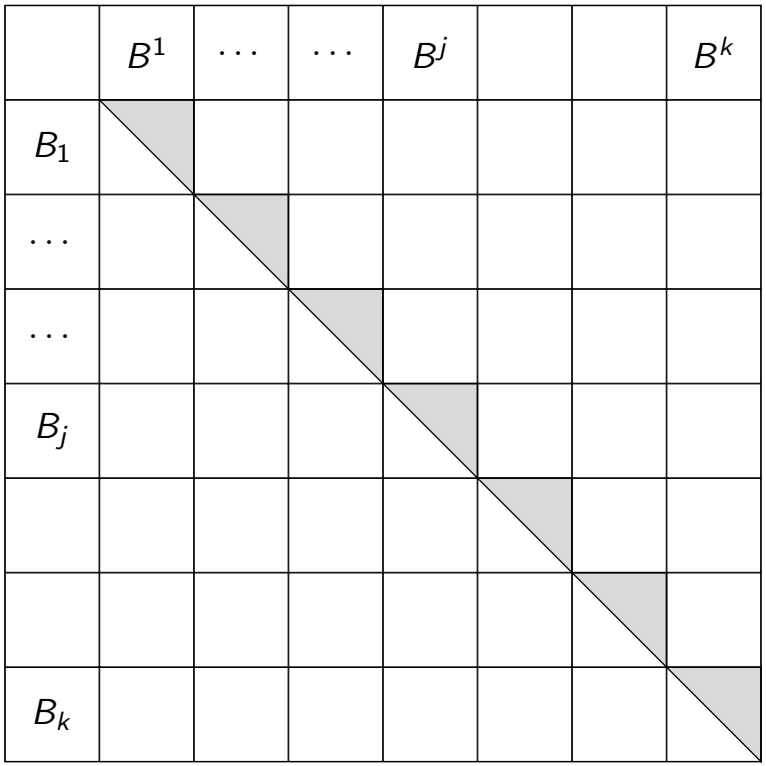
Idea:

- Reduce the matrix in phases: in each phase r, reduce columns in B^j by adding columns in the blocks from B^{j-r+1} to B^j.

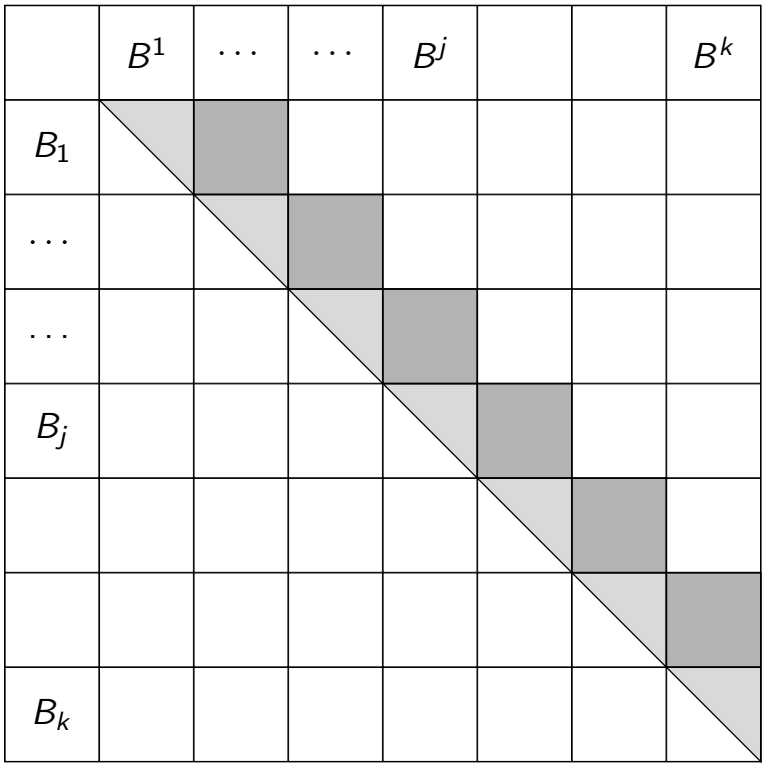
Optimisation:

- The reduction in each block and each phase is independent, and can be executed in parallel.

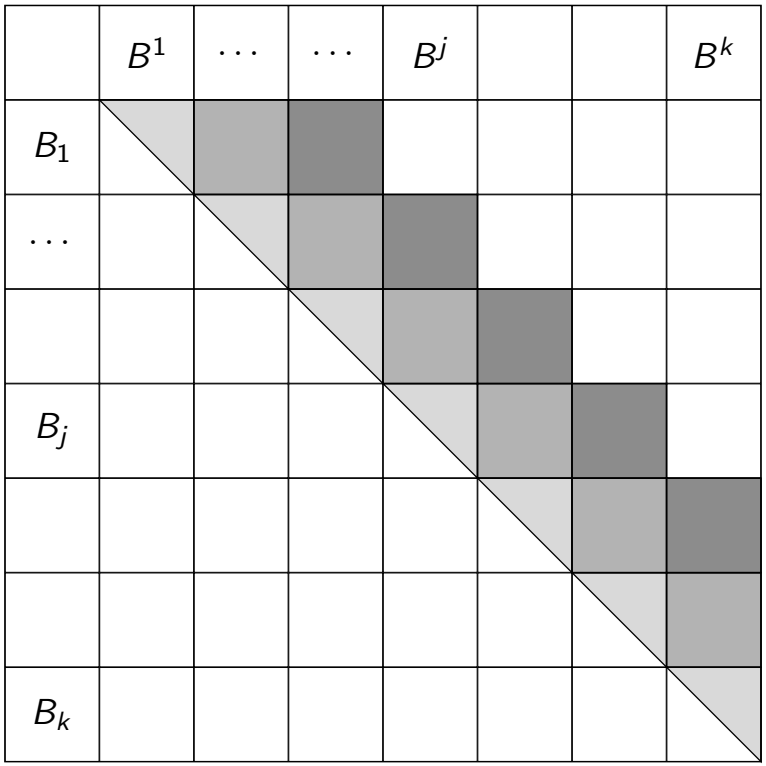
Phase $r = 1$:



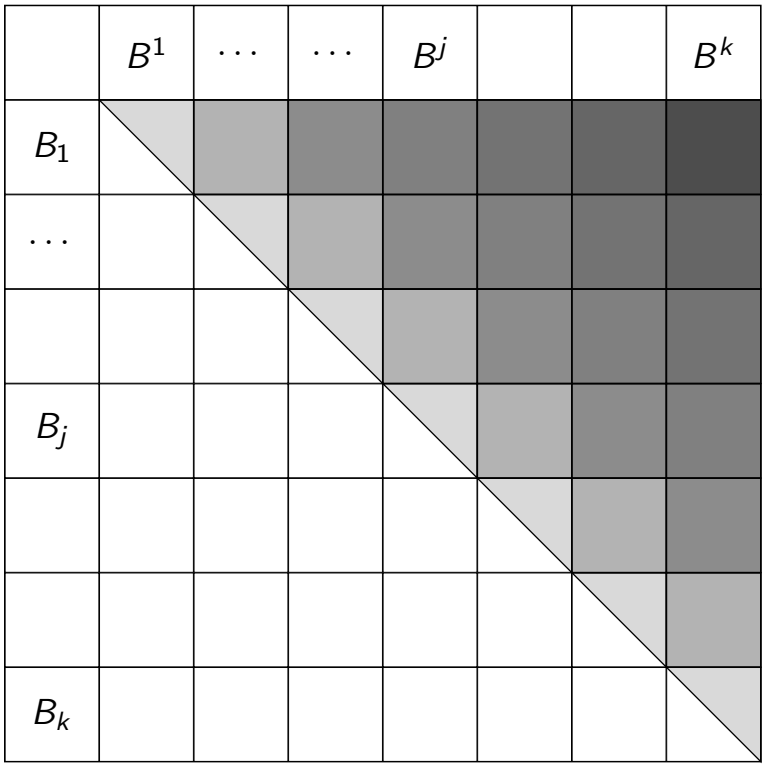
Phase $r = 2$:



Phase $r = 3$:



Phase $r = k$:



Optimisations: implementations

(2) From a boundary matrix to barcodes

Sequential:

- Dual algorithm $\longrightarrow$ DIONYSUS, JAVAPLEX, GUDHI, PHAT, DIPHA
- Twist algorithm $\longrightarrow$ PHAT,DIPHA

Parallel:

- Spectral sequence algorithm $\longrightarrow$ PHAT
- Chunk algorithm $\longrightarrow$ PHAT
- Distributed algorithm $\longrightarrow$ DIPHA

Data structure:

- Bit tree pivot column $\longrightarrow$ PHAT, DIPHA
- Compressed annotation matrix $\longrightarrow$ GUDHI

Summary

Software	Precomp.	filt. Compl. [2]	Parallel	Visualiz.	PH algorithms
javaPlex	✓	VR, LW, W, CW	✗	✓	standard, dual zig zag
Perseus	✓	VR	✗	✓	Morse reductions + standard
Dionysus	✗	α, VR, Čech	✗	✗	standard, dual zig-zag
PHAT	✗	✗	✓shared	✗	standard, dual, twist chunk, spectral seq.
DIPHA	✗	VR, lower star	✓distr.	✓	dual, twist distributed
GUDHI	✗	VR	✗	✗	multifield dual

[2] VR=Vietoris Rips complex, W=witness complex, LW=lazy witness complex, CW=CW complex, WRCF=weight rank clique filtration

Methods

We study the software from four different points of view:

1. Performance measured in CPU and real time
2. Memory usage
3. Maximum size of simplicial complex allowed by the software
4. User-friendliness: phases of computation of PH supported by software

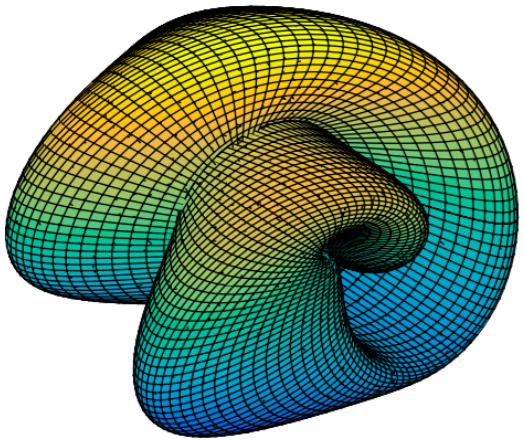
Data

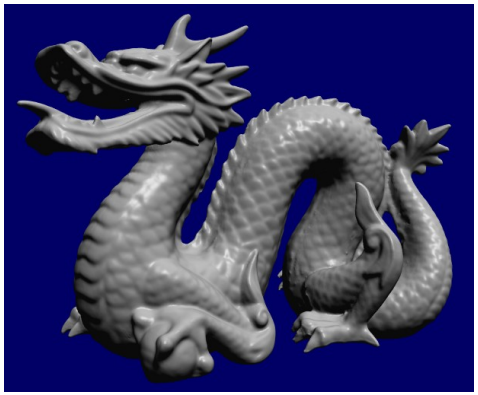
Synthetic data

- Points sampled from Klein bottle
- Random Vietoris Rips complexes

Real world data

- Genomic sequence of HIV virus
- Points sampled from 3D scans of Stanford dragon
- C. Elegans neuronal network
- Human genome network





Machines

- Cluster[3]: 1728 (180*16) cores of 2.0GHz
  RAM : 64 GiB x80 nodes, 128 GiB x 4 nodes

[3] Advanced Research Computing (ARC), Oxford

Results: performance

Data set	C. elegans		Klein		HIV		Dragon 1		Dragon 2	
Size of complex	4.4×10^6		1.1×10^7		2.1×10^8		1.7×10^8		1.3×10^9	
JAVAPLEX st	84	284	747	1031	-	-	-	-	-	-
DIONYSUS st	474	473	1830	1824	-	-	-	-	-	-
DIPHA st	6	68	90	1360	1631	25950	7356	117417	142559	1489615
PERSEUS	543	542	1978	1974	-	-	-	-	-	-
DIONYSUS d	513	513	145	145	-	-	4360	4362	-	-
DIPHA d	4	39	6	73	81	1276	75	1176	2358	37572
GUDHI	6	4	11	10	249	248	285	283	15419	3151

Table: Performance measured in wall-time and CPU seconds for the computation of PH with the Vietoris Rips complex.

Results: memory usage

Data set	C. elegans	Klein	HIV	Dragon 1	Dragon 2
Size of complex	4.4×10^6	1.1×10^7	2.1×10^8	1.7×10^8	1.3×10^9
JAVAPLEX st	< 5	< 15	> 120	> 120	> 120
DIONYSUS st	1.3	11.6	-	-	-
DIPHA st	0.1	0.2	2.7	2.4	4.9
PERSEUS	5.1	12.7	-	-	-
DIONYSUS d	0.5	1.1	-	16.8	-
DIPHA d	0.1	0.2	1.8	1.8	13.8
GUDHI	0.2	0.6	9.9	9.2	64.5

Table: Memory usage in GB. For JAVAPLEX, we indicate the value of the maximum heap size that was sufficient to perform the computation. For DIPHA, we indicate the maximum memory used by a single core.

Results: maximal size of simplicial complex

Software	JavaPlex	Dionysus		DIPHA		Perseus	GUDHI
	st	st	d	st	d	st	d
max. size	$1 \cdot 10^7$	$1.1 \cdot 10^7$	$3.2 \cdot 10^8$	$1.3 \cdot 10^9$	$1.3 \cdot 10^9$	$1 \cdot 10^7$	$1.3 \cdot 10^9$

Table: Maximal size of simplicial complex supported by the software.

Software	javaPlex	Perseus	Dionysus	DIPHA	GUDHI
Installation	✓	✓	–	–	–
Complex	✓	✓	✓	✓	✓
Boundary matrix	✓	✓	✓	✓	✓
Barcodes	✓	✓	✓	✓	✓
Visualisation	✓	✓	–	✓	–

Table: Roughly, one can divide the pipeline for the computation of PH into five steps: (1) installation of the software, (2) computation of the complex from input data, (3) computation of the boundary matrix, (4) computation of barcodes, and (5) visualisation of outputs. For each software package, we indicate which of these steps are supported.

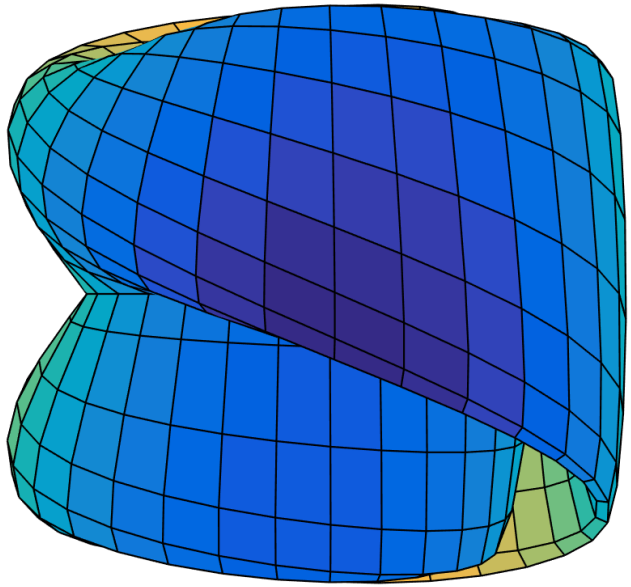
Interlude: Demonstration

Demonstration

Using points sampled from the Klein bottle we will:

- Compute PH using different filtered simplicial complexes
- Plot results as barcodes and persistence diagrams
- Compare results computing bottleneck distance

The Klein bottle



Homology of image of immersion in $\mathbb{R}^3$ and embedding in $\mathbb{R}^4$:

$H_0(K) = F_2$ $\quad$ $H_1(K) = F_2 \oplus F_2$ $\quad$ $H_2(K) = F_2$

Čech, VR, α complexes

Given a point cloud $X \subset \mathbb{R}^d$

- the Čech complex at scale ϵ is

$$C(\epsilon) = \{\sigma \subset X \mid \cap_{x \in \sigma} B_x(\epsilon) \neq \emptyset\}$$

- the Vietoris Rips complex at scale ϵ is

$$V(\epsilon) = \{\sigma \subset X \mid d(x, y) \leq 2\epsilon \text{ for all } x, y \in \sigma\}$$

- the α-complex at scale ϵ is

$$\alpha(\epsilon) = \{\sigma \subset X \mid \bigcap_{x \in \sigma} B_x(\epsilon) \cap V_x \neq \emptyset\}$$

For all $\epsilon > 0$ we have: $\alpha(\epsilon) \subset C(\epsilon) \subset V(\epsilon)$.

Lazy witness complex

Given a point cloud X, choose a set of landmarks $L \subset X$. Let $\nu \in \mathbb{N}$.

For $x \in X$ let $m(x)$ be the distance from x to its ν closest landmark point.

The lazy witness complex $LW_\nu(\epsilon)$ at scale ϵ:

- vertices are L
- edge $\{x_0, x_1\}$ is in $LW_\nu(\epsilon)$ if there exists a witness $z \in Z$ such that $max\{d(x_0, z), d(x_1, z)\} \leq t + m(z)$.

Computations: Čech

With DIONYSUS:

```
./cech-complex < input-file > output-file
```

input-file:

embedding-dimension max-dimension

x_{00} x_{01} ...

x_{10} x_{11} ...

...

$|X| = 100 \subset \mathbb{R}^3$
max-dimension 3 ✓

$|X| = 400 \subset \mathbb{R}^3$
max-dimension 2 ✓
max-dimension 3 ✗

Computations:VR

With DIONYSUS:

```
./rips-pairwise -s max-dimension -m max-distance \
-d output-file input-file
```

input-file:

$$x_{00}\ x_{01}\ \dots$$
$$x_{10}\ x_{11}\ \dots$$
$$\dots$$

$|X| = 100 \subset \mathbb{R}^3$
max-dimension 3 ✓

$|X| = 400 \subset \mathbb{R}^3$
max-dimension 2 ✓
max-dimension 3 ✗

Computations:VR

With PERSEUS:

```
./perseus distmat input-file output-file
```

input-file:

number-points

ϵ_0 ϵ-increment number-steps max-dimension

d_{00} d_{01} ...

d_{10} d_{11} ...

...

$|X| = 100 \subset \mathbb{R}^3$
max-dimension 3 ✓

$|X| = 400 \subset \mathbb{R}^3$
max-dimension 2 ✓
max-dimension 3 ✗

Computations:VR

With GUDHI:

```
./rips_persistence input-file -r max-distance \
-d max-dimension -p prime -o output-file input-file
```

input-file:

$$x_{00}\ x_{01}\ \dots$$
$$x_{10}\ x_{11}\ \dots$$
$$\dots$$

$|X| = 100 \subset \mathbb{R}^3$
max-dimension 3 ✓

$|X| = 400 \subset \mathbb{R}^3$
max-dimension 2 ✓
max-dimension 3 ✓

Computations:VR

With DIPHA:

```
./dipha [options] input-file output-file
```

input-file in binary format:

number-points

$d_{00}\ d_{01}\ \ldots$

$d_{10}\ d_{11}\ \ldots$

$\ldots$

$|X| = 100 \subset \mathbb{R}^3$
max-dimension 3 ✓

$|X| = 400 \subset \mathbb{R}^3$
max-dimension 2 ✓
max-dimension 3 ✓

Computations:VR

With JAVAPLEX in MATLAB:

```
vietoris_rips_javaplex(input-file, max-dimension,...
max-filtration-value, number-steps, plot-name)
```

input-file:

$$x_{00}\ x_{01}\ \dots$$
$$x_{10}\ x_{11}\ \dots$$
$$\dots$$

$|X| = 100 \subset \mathbb{R}^3$
max-dimension 3 ✓

$|X| = 400 \subset \mathbb{R}^3$
max-dimension 2 ✓
max-dimension 3 ✗

Computations: α

With DIONYSUS from command line:

```
./alphashapes3d-cohomology  input-file output-file
```

input-file:

$$x_{00}\ x_{01}\ \dots$$
$$x_{10}\ x_{11}\ \dots$$
$$\dots$$

$|X| = 100 \subset \mathbb{R}^3$
max-dimension 3 ✓

$|X| = 400 \subset \mathbb{R}^3$
max-dimension 3 ✓

Computations: lazy witness

with JAVAPLEX in MATLAB:

```
lazy_witness_javaPlex(input-file, max_dimension,...
num_landmark_points, type, nu, filtration_steps, plot_name)
```

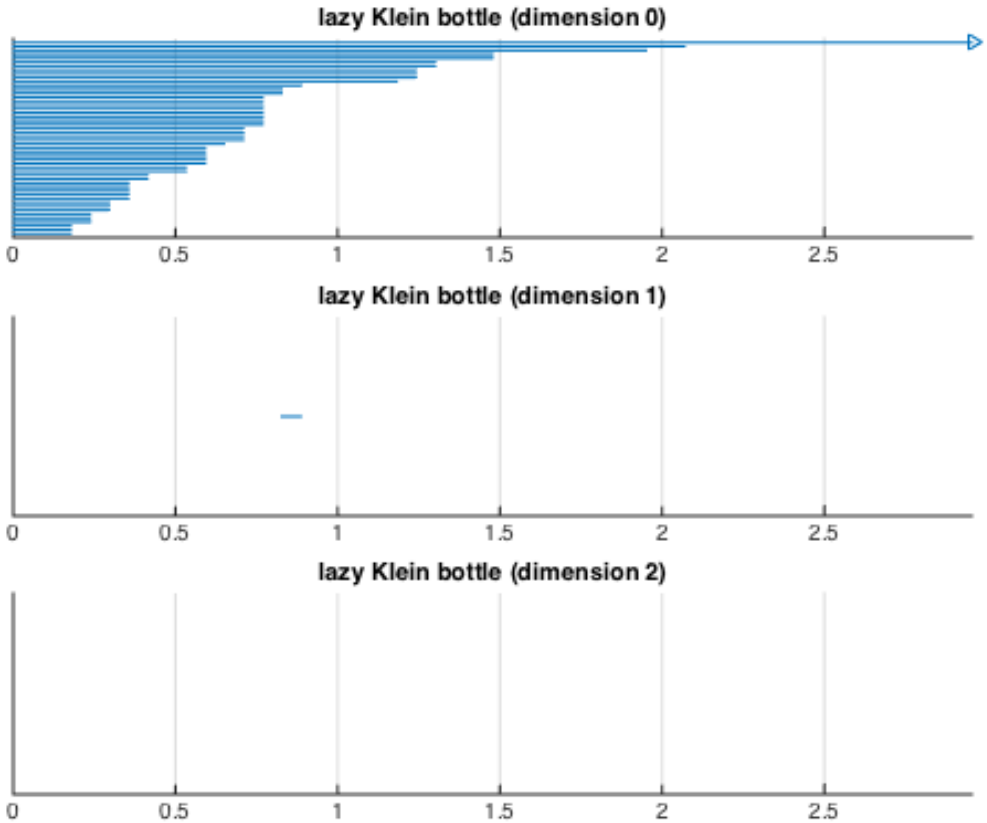
input-file:

$$x_{00}\ x_{01}\ \dots$$
$$x_{10}\ x_{11}\ \dots$$
$$\dots$$

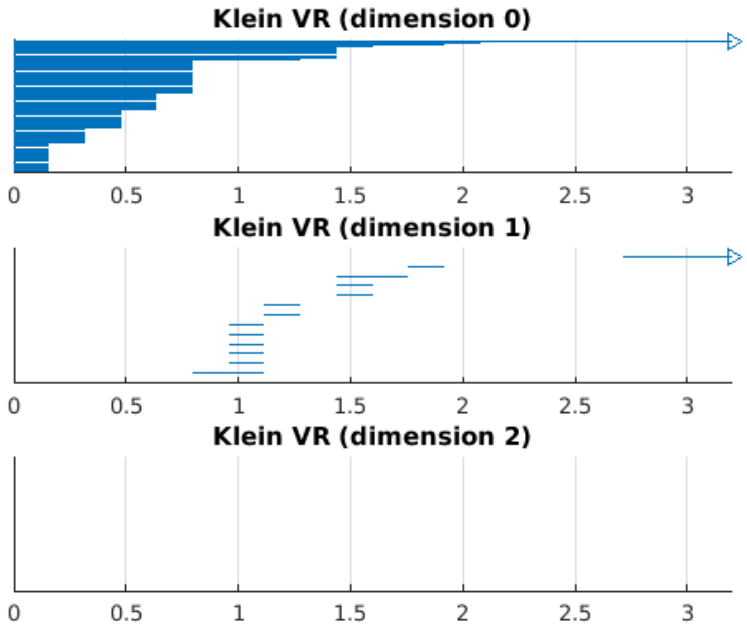
$|X| = 100 \subset \mathbb{R}^3$ $L = 50$
max-dimension 3 ✓

$|X| = 400 \subset \mathbb{R}^3$ $L = 200$
max-dimension 2 ✓
max-dimension 3 ✗

Barcodes for $|X| = 100$



Barcodes for $|X| = 100$



Other barcodes and bottleneck distance: exercise!

Other barcodes and bottleneck distance: exercise!

- Scripts will be uploaded to `http://people.maths.ox.ac.uk/otter/` this weekend.

Other barcodes and bottleneck distance: exercise!

- Scripts will be uploaded to `http://people.maths.ox.ac.uk/otter/` this weekend.
- In the meantime you can install the libraries..

Conclusions

Conclusions

Challenges:

- User-friendly interfaces.
- Creation of a computational topology library.
- Definition and construction of benchmarking datasets.
- Uniformization of input type across different implementations.
- Focus on step (1) of pipeline: implementation and construction of simpler simplicial complexes.
- Stream Processing: new techniques are needed to compute PH for streams of data.

Software

- **Perseus** http://www.sas.upenn.edu/~vnanda/perseus/
- **JavaPlex** http://appliedtopology.github.io/javaplex/
- **Dionysus** http://www.mrzv.org/software/dionysus/
- **pHom** http://cran.r-project.org/web/packages/phom/
- **TDA** http://cran.r-project.org/web/packages/TDA/
- **GAP persistence** http://www-circa.mcs.st-and.ac.uk/~mik/persistence/
- **PHAT** https://code.google.com/p/phat/
- **DIPHA** https://code.google.com/p/dipha/
- **GUDHI** https://project.inria.fr/gudhi/software/
- **Kenzo module** http://www.unirioja.es/cu/anromero/persistent-homology.zip
- **The persistence landscape toolbox**
  https://www.math.upenn.edu/~dlotko/persistenceLandscape.html
- **SimpPers**
  http://web.cse.ohio-state.edu/~tamaldey/SimpPers/SimpPers-software/
- **jHoles** http://cuda.unicam.it/jHoles
- **HoPeS** http://kurlin.org/projects/persistent-skeletons.php
- ...