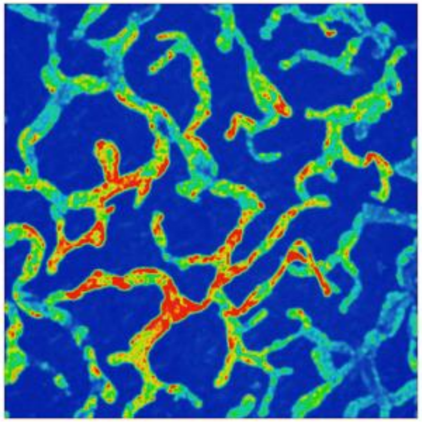
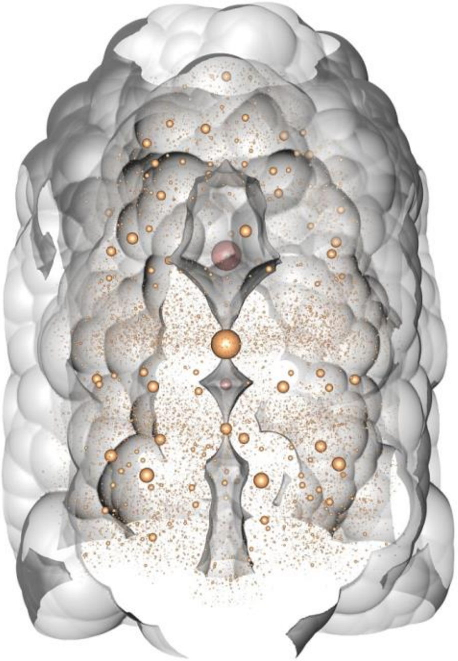
Algorithms & Software for Topological Data Analysis

Jan Reininghaus
(Siemens Industry Software)

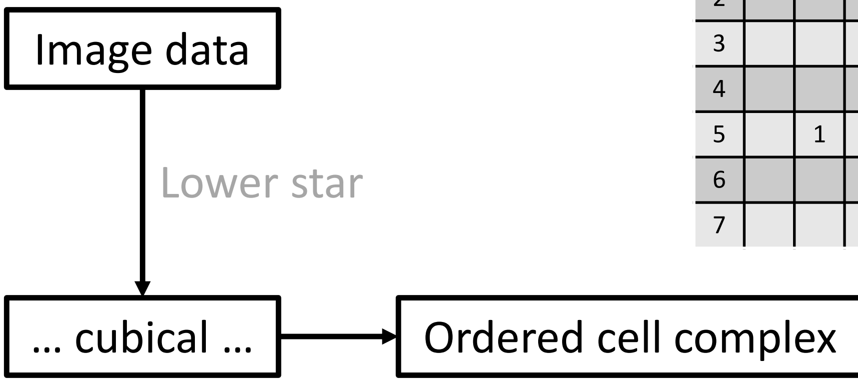
Roadmap

Image data



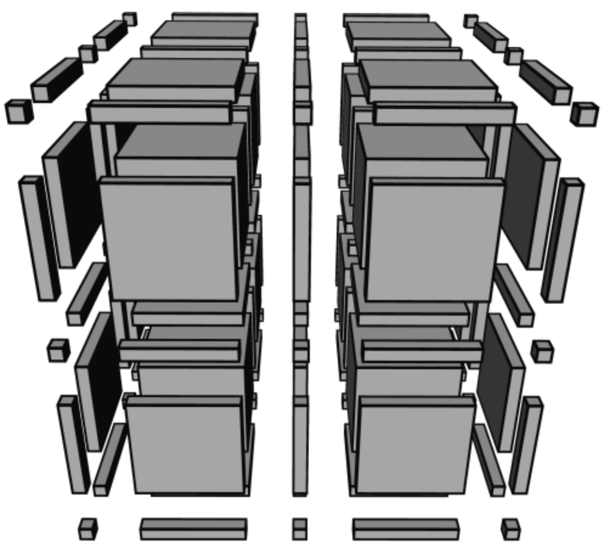


Roadmap

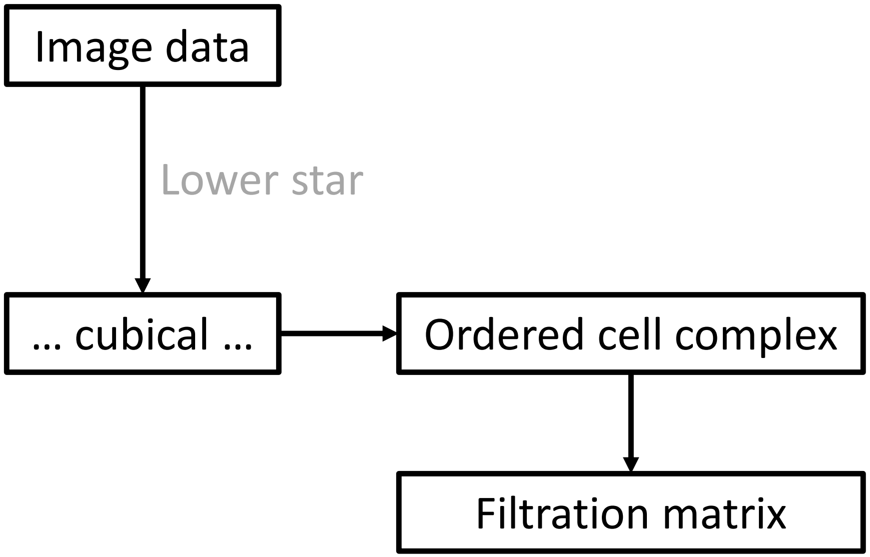


	1	2	3	4	5	6	7
1		1		1			
2						1	
3					1		
4						1	
5		1					
6					1		
7							

1.3
3.2
2.8
6.1
9.1
8.3
5.9



Roadmap



	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

Roadmap

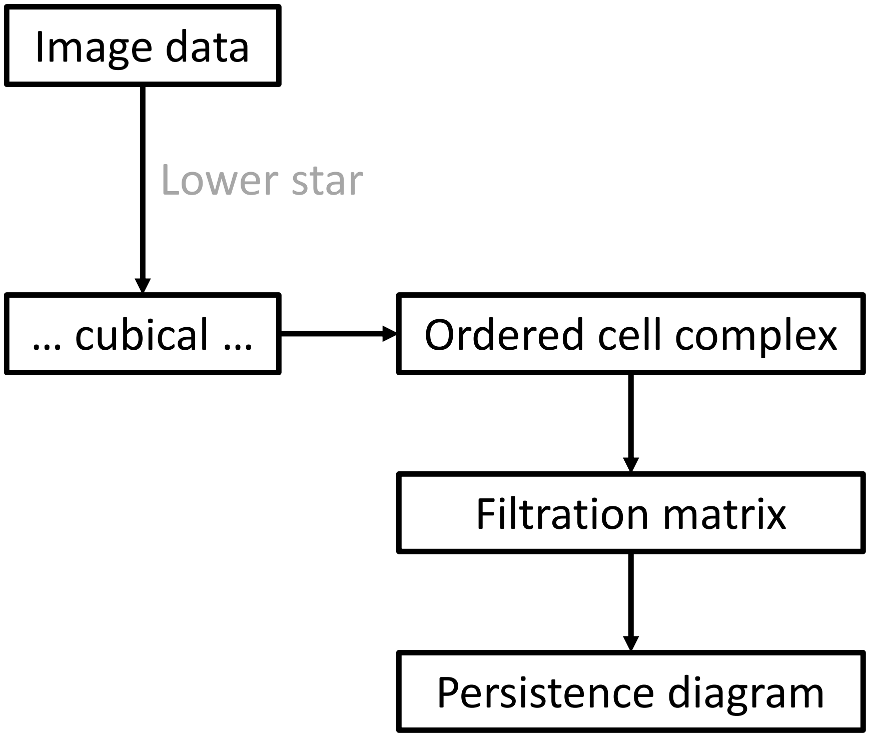
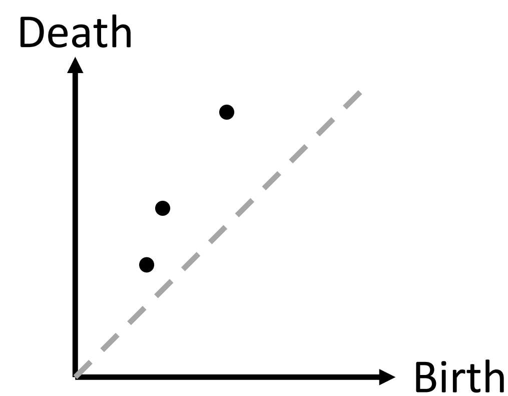


Image data
Lower star
… cubical …
Ordered cell complex
Filtration matrix
Persistence diagram


Death
Birth

Roadmap

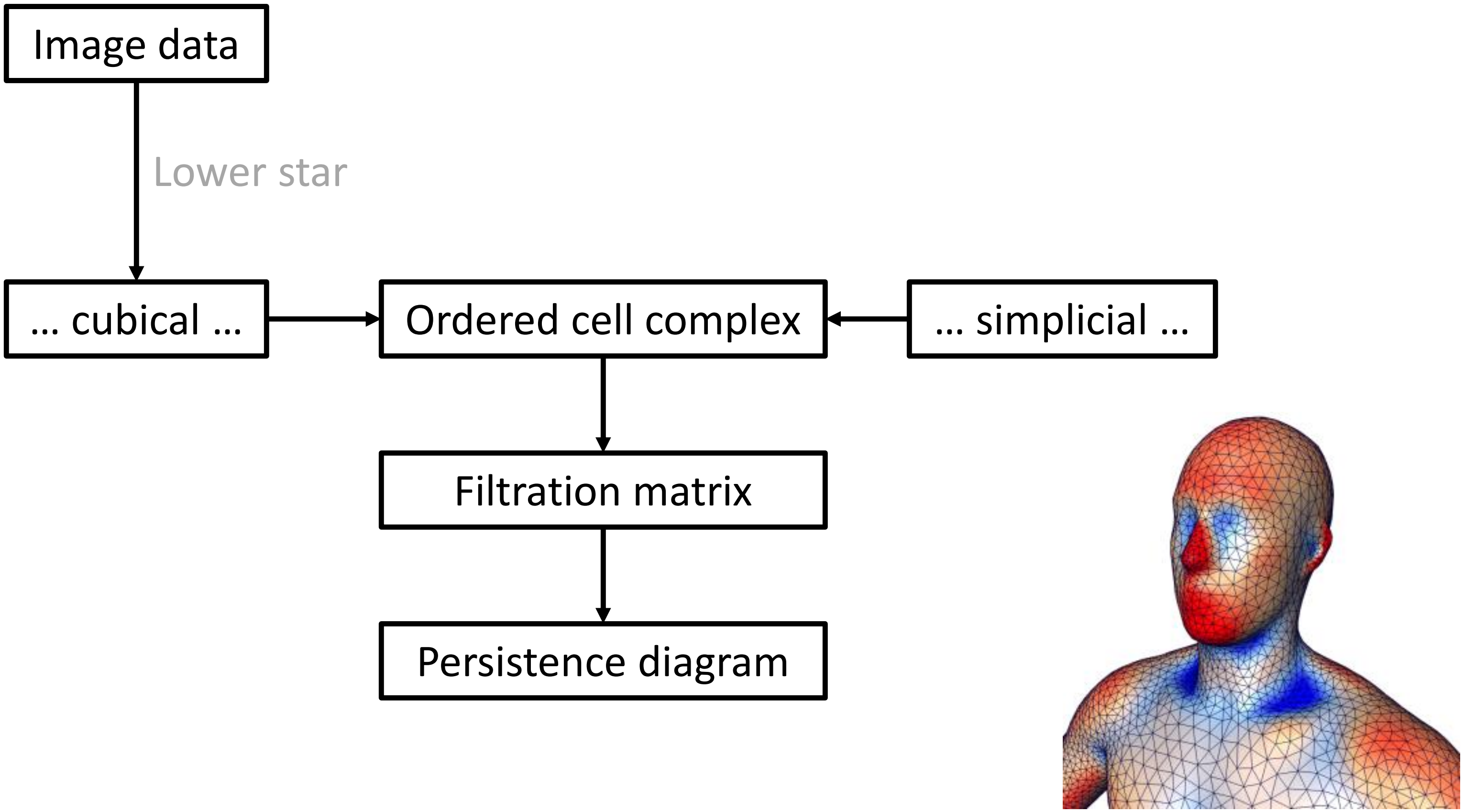
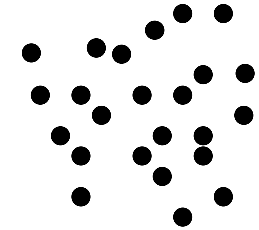


Image data
Lower star
... cubical ...
Ordered cell complex
... simplicial ...
Filtration matrix
Persistence diagram

Roadmap



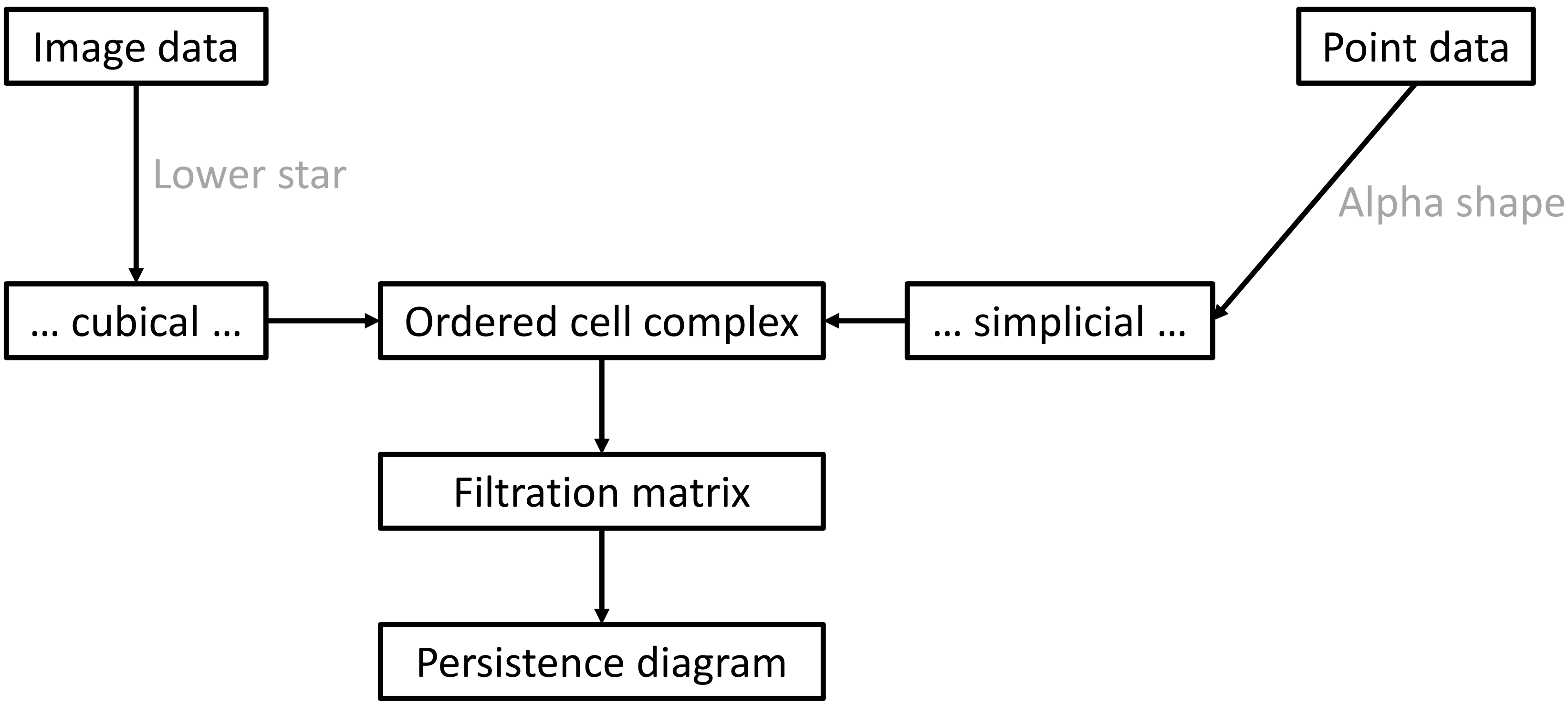
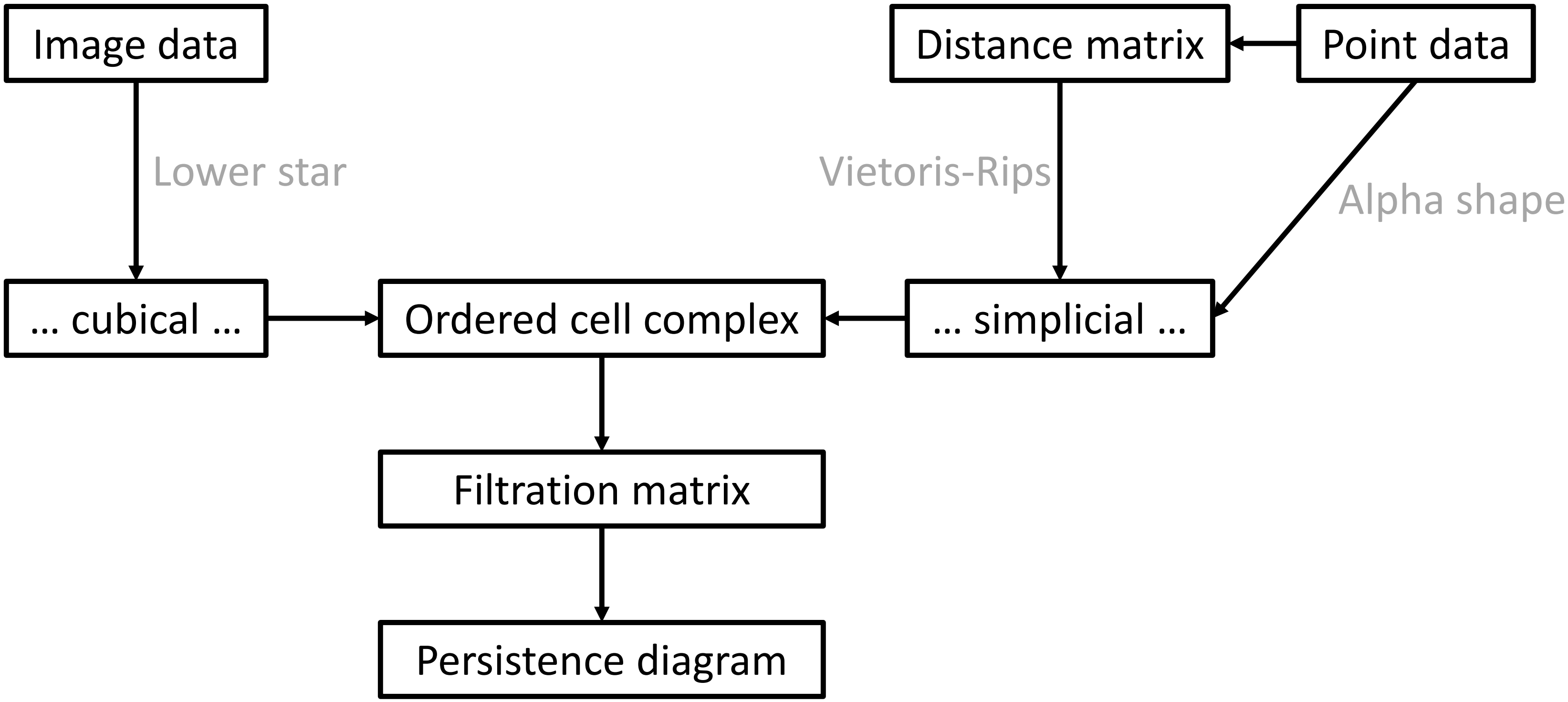


Image data
Lower star
... cubical ...
Ordered cell complex
... simplicial ...
Point data
Alpha shape
Filtration matrix
Persistence diagram

Roadmap

1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12



Roadmap

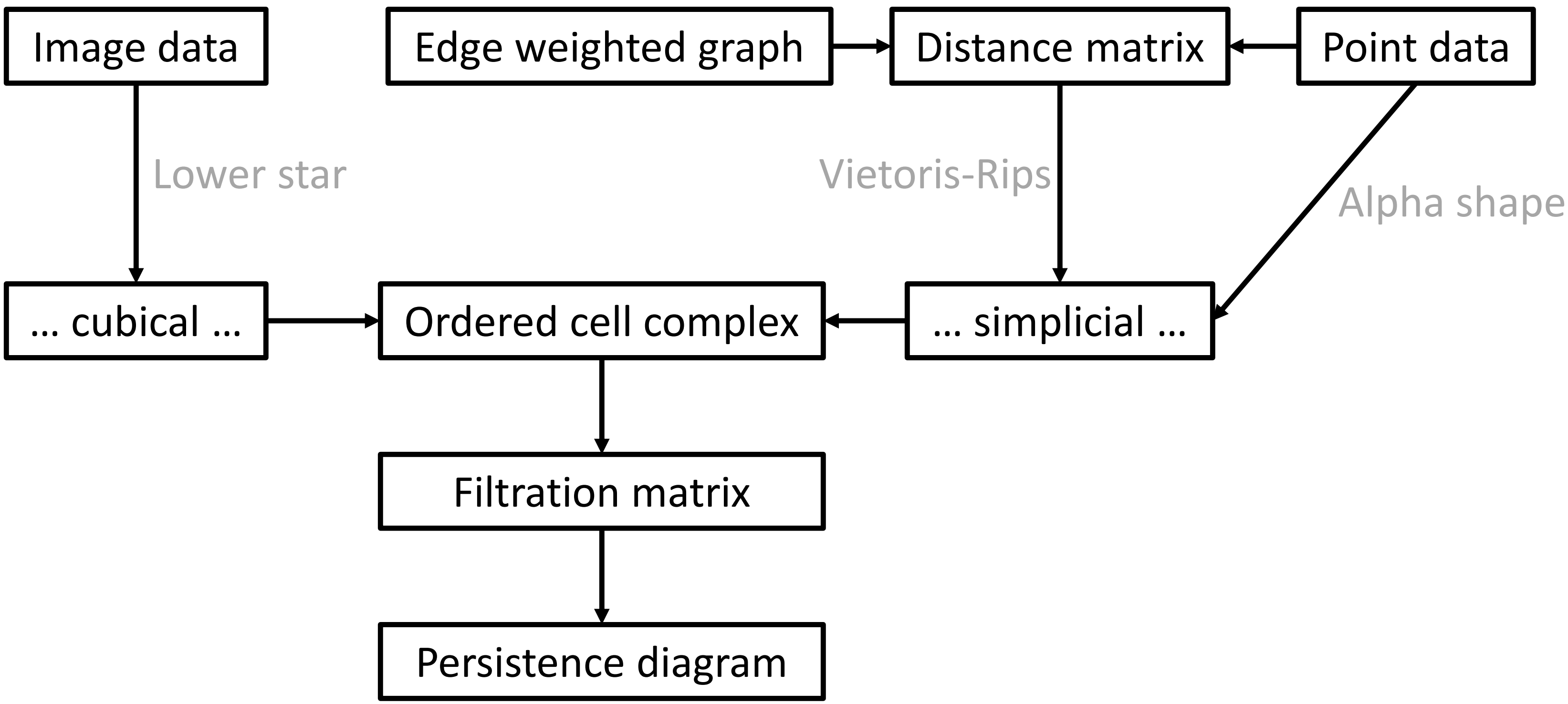


Image data
Edge weighted graph
Distance matrix
Point data
Lower star
Vietoris-Rips
Alpha shape
… cubical …
Ordered cell complex
… simplicial …
Filtration matrix
Persistence diagram

Roadmap

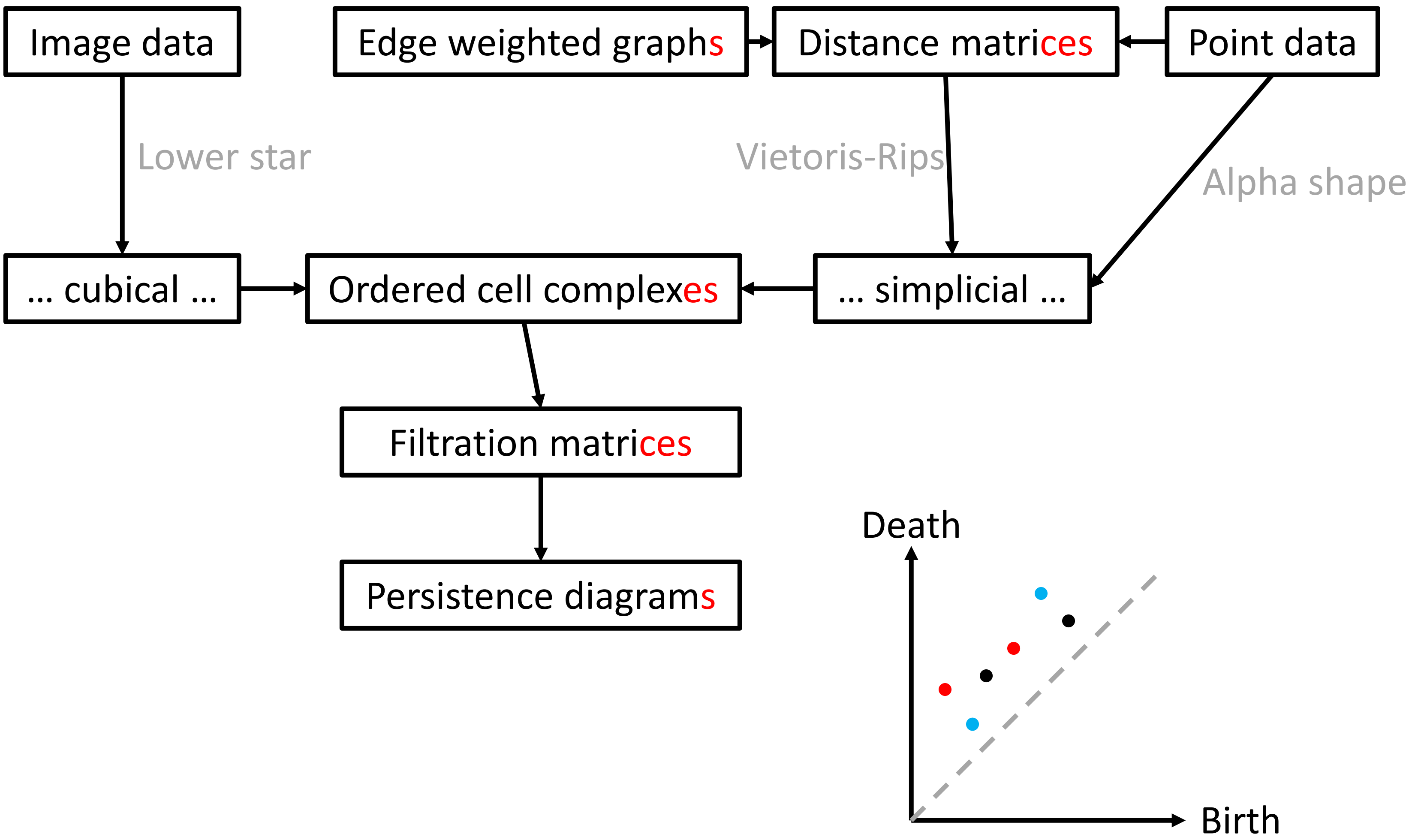


Image data
Edge weighted graphs
Distance matrices
Point data
Lower star
Vietoris-Rips
Alpha shape
... cubical ...
Ordered cell complexes
... simplicial ...
Filtration matrices
Persistence diagrams
Death
Birth

Roadmap

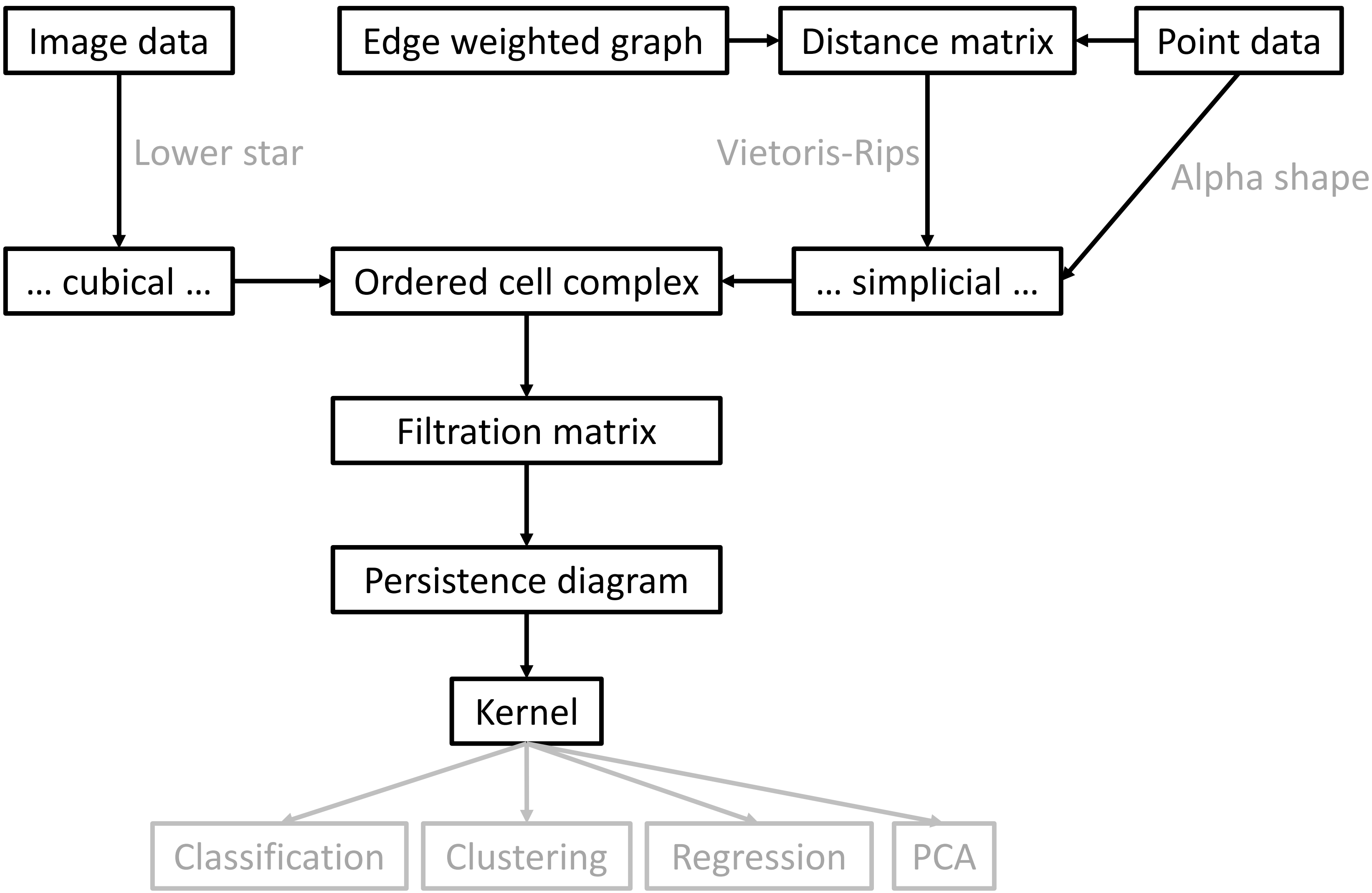
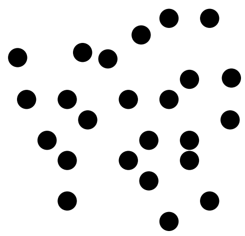
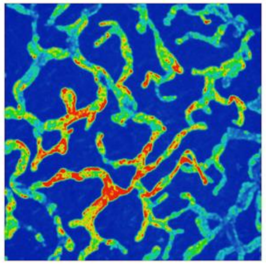


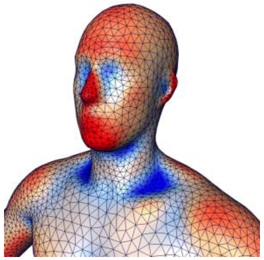
Image data
Edge weighted graph
Distance matrix
Point data
Lower star
Vietoris-Rips
Alpha shape
… cubical …
Ordered cell complex
… simplicial …
Filtration matrix
Persistence diagram
Kernel
Classification
Clustering
Regression
PCA

Computational Pipeline



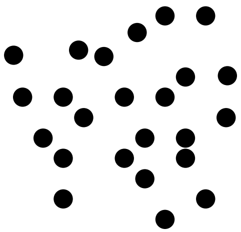
1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12



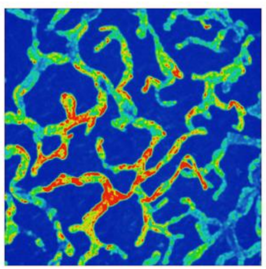


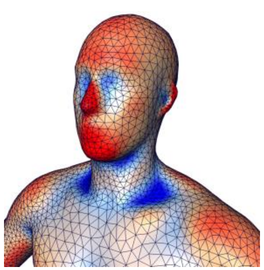
Input data

Computational Pipeline



1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12





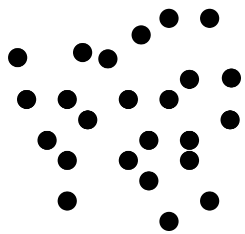
Input data

1.3
3.2
2.8
6.1
9.1
8.3
5.9

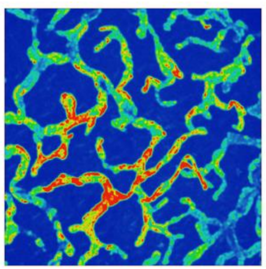
	1	2	3	4	5	6	7
1		1		1			
2						1	
3					1		
4						1	
5		1					
6					1		
7							

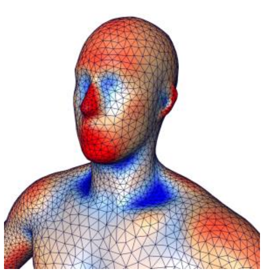
Boundary matrix
+ weights

Computational Pipeline

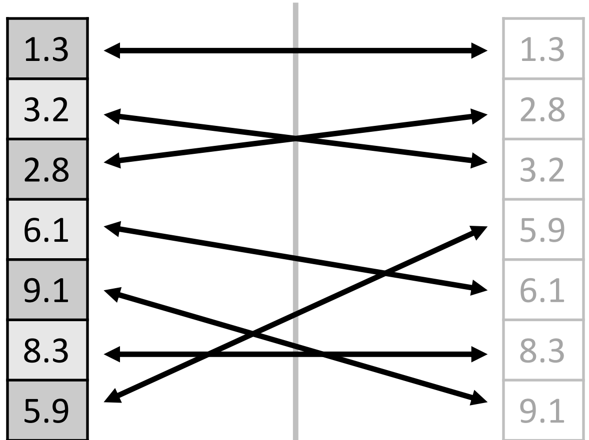


1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12





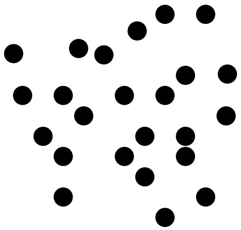
Input data



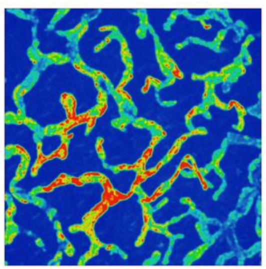
	1	2	3	4	5	6	7
1		1		1			
2						1	
3					1		
4						1	
5		1					
6					1		
7							

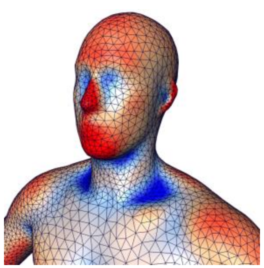
Boundary matrix
+ weights

Computational Pipeline

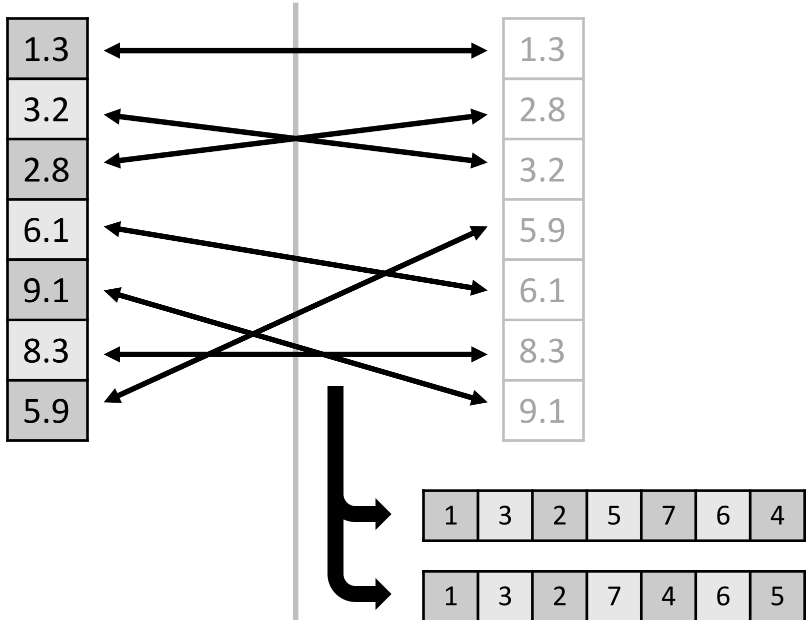


1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12





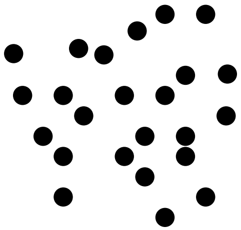
Input data



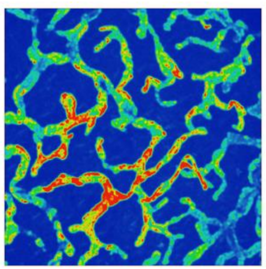
	1	2	3	4	5	6	7
1		1		1			
2						1	
3					1		
4						1	
5		1					
6					1		
7							

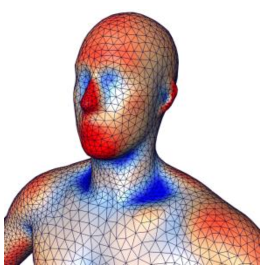
Boundary matrix
+ weights

Computational Pipeline

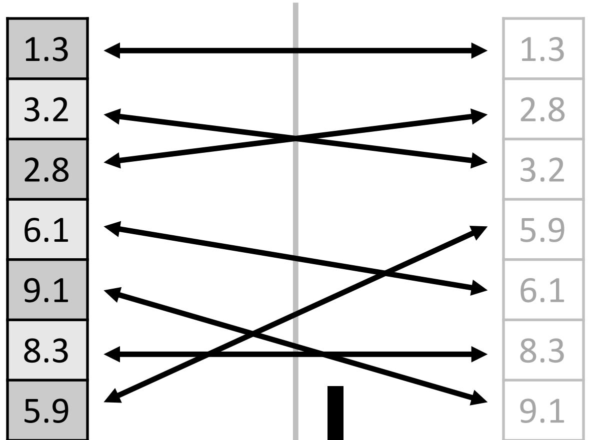


1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12





Input data



	1	2	3	4	5	6	7
1		1		1			
2						1	
3					1		
4						1	
5		1					
6					1		
7							

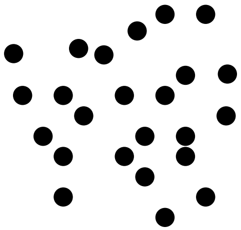
Boundary matrix
+ weights

1	3	2	5	7	6	4

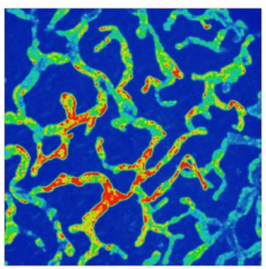
1	3	2	7	4	6	5

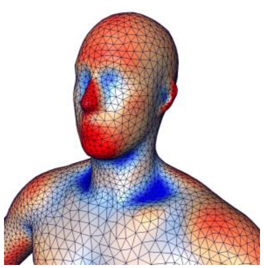
	1	2	3	4	5	6	7
1		1		1			
2		1				1	
3					1		
4				1		1	
5					1		
6					1		
7							

Computational Pipeline

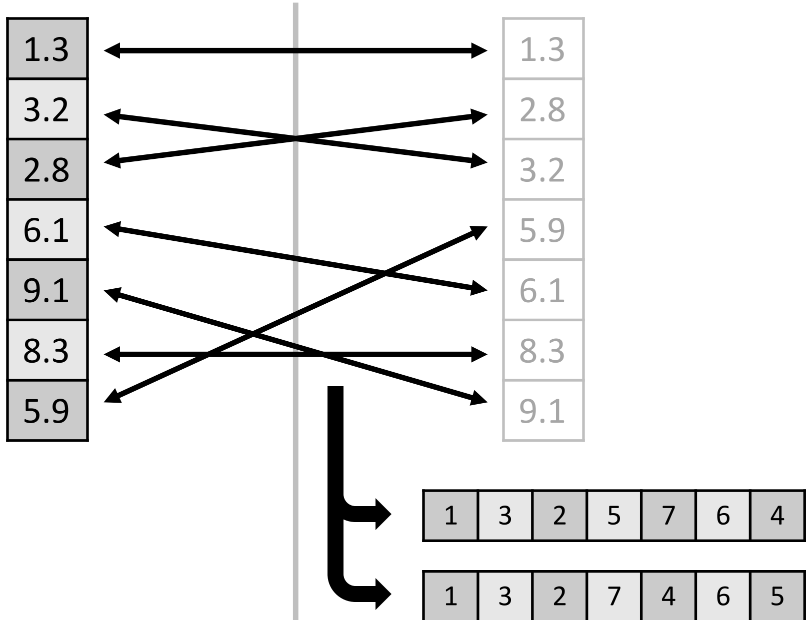


1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12





Input data



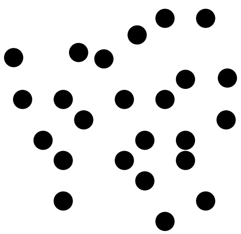
	1	2	3	4	5	6	7
1		1		1			
2						1	
3					1		
4						1	
5		1					
6					1		
7							

Boundary matrix
+ weights

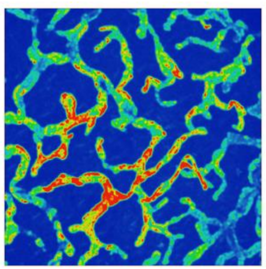
	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

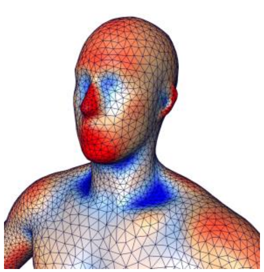
Filtration matrix
+ permutation maps

Computational Pipeline

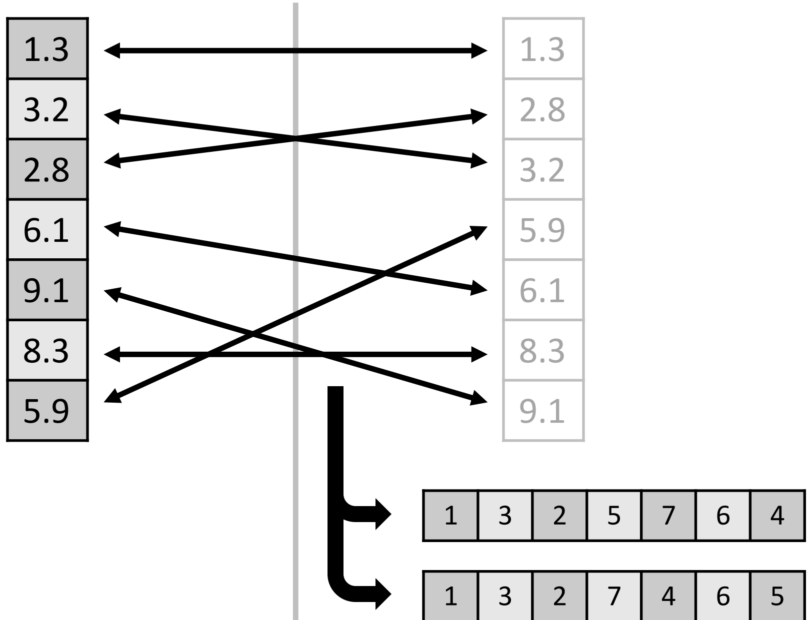


1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12





Input data



	1	2	3	4	5	6	7
1		1		1			
2						1	
3					1		
4						1	
5		1					
6					1		
7							

Boundary matrix
+ weights

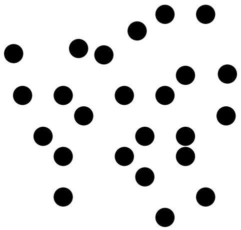
	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

Filtration matrix
+ permutation maps

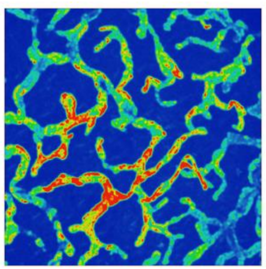
\+ +

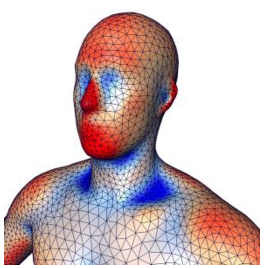
	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

Computational Pipeline

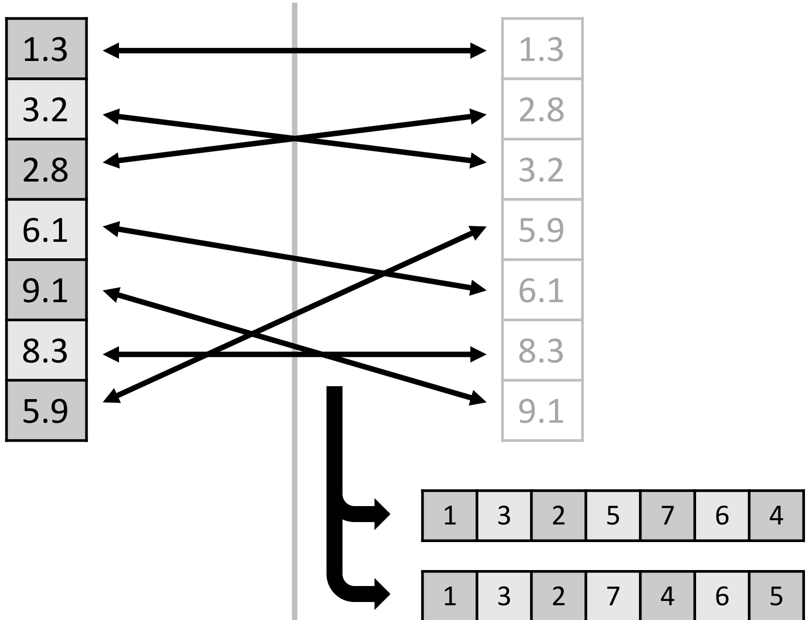


1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12





Input data



	1	2	3	4	5	6	7
1		1		1			
2						1	
3					1		
4						1	
5		1					
6					1		
7							

Boundary matrix
+ weights

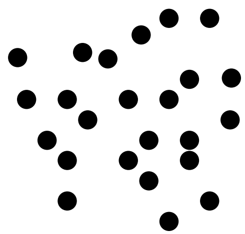
	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

Filtration matrix
+ permutation maps

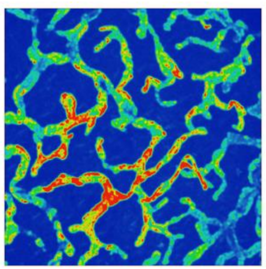
	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

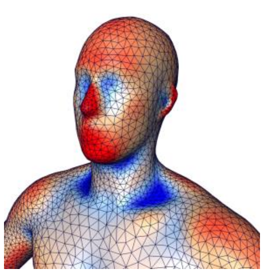
Reduced matrix

Computational Pipeline

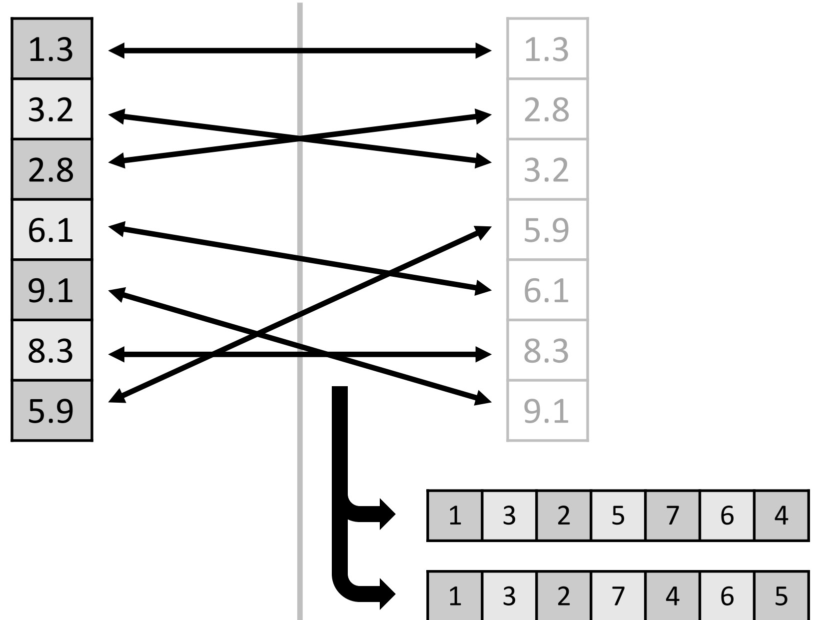


1	3	78	65	3	46	12	1
7	6	23	78	65	3	46	23
5	34	61	2	78	65	3	46
2	6	2	10	2	1	1	51
78	65	3	46	6	1	4	35
1	6	78	65	3	46	24	32
6	6	6	5	78	65	3	46
1	78	65	3	46	2	15	12





Input data

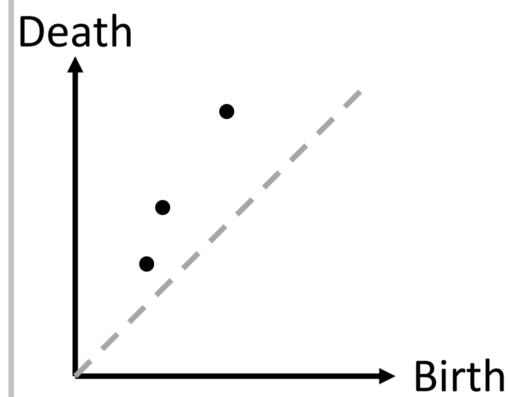


	1	2	3	4	5	6	7
1		1		1			
2						1	
3					1		
4						1	
5		1					
6					1		
7							

Boundary matrix
+ weights

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

Filtration matrix
+ permutation maps

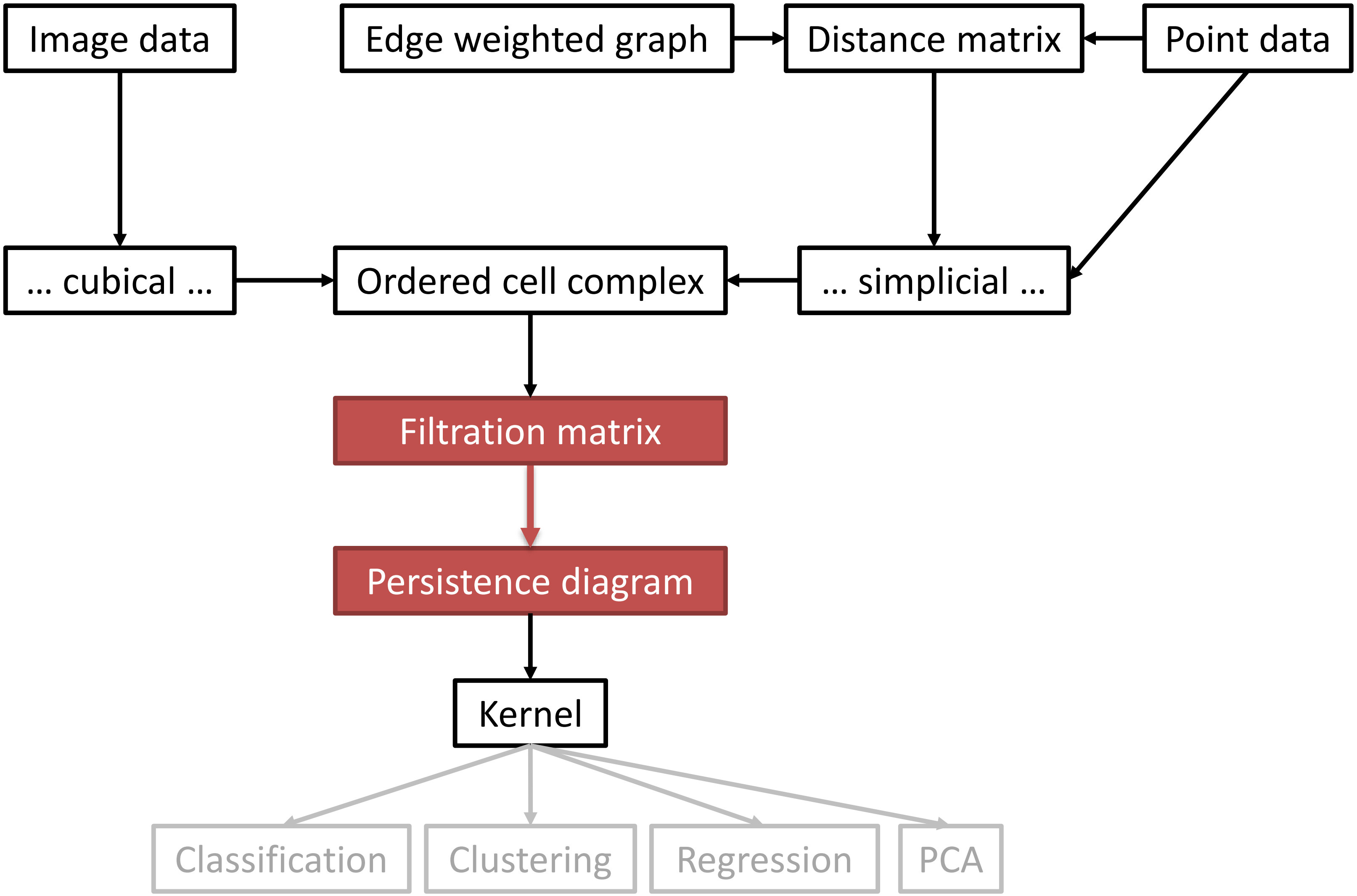


	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

Reduced matrix
→ diagram

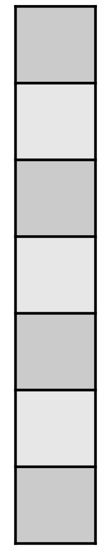
Part 1: Phat - `bitbucket.org/phat-code/phat`

Joint work with: U. Bauer, M. Kerber, H. Wagner



Classic Matrix Reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							



Pivot-to-Column Map

Classic Matrix Reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

3

Pivot-to-Column Map

Classic Matrix Reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

3
5

Pivot-to-Column Map

Classic Matrix Reduction

	1	2	3	4	5	6	7
1			1		1	1	
2			1			1	
3							1
4					1		
5							1
6							1
7							

3
5

Pivot-to-Column Map

Classic Matrix Reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

Pivot-to-Column Map

3
5

Classic Matrix Reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

3
5
7

Pivot-to-Column Map

Phat features

- Algorithms:
  - Standard
  - Row
  - Twist (best sequential performance/complexity ratio)
  - Chunk (parallel)
  - Spectral sequence (best parallel performance/complexity ratio)
  - Dualized versions of all above

- Column data structures:
  - vector_vector
  - vector_heap (best performance/complexity ratio)
  - vector_set
  - vector_list (purely educational)
  - sparse_pivot_column
  - heap_pivot_column
  - full_pivot_column
  - bit_tree_pivot_column (fastest, requires N memory)

Phat performance

	List	Vector	Set	Heap	A-Heap	A-Set	A-Full	**A-Bit-Tree**
standard	17.1	2.8	7.5	5.9	5.6	8.6	2.3	1.62
standard*	2580.0	168.0	17.3	13.5	14.6	16.0	3.9	0.57
twist	16.3	2.7	7.0	5.7	5.4	6.5	2.2	1.59
twist*	0.23	0.03	0.04	0.02	0.03	0.03	0.02	**0.02**
row	39.9	4.3	20.3	7.2	21.4	37.9	15.5	13.8
row*	0.25	0.06	0.06	0.05	0.08	0.09	0.05	0.05
chunk	0.63	0.19	0.6	0.50	0.35	0.5	0.24	0.24
chunk*	2.9	0.42	0.1	0.11	0.17	0.14	0.07	0.04
spectral	10.7	1.8	4.03	3.3	3.4	4.3	2.1	1.2
spectral*	0.35	0.03	0.05	0.04	0.04	0.04	0.02	**0.01**

Dataset: 3-skeleton of the Vietoris–Rips filtration of 64 points on a 2-sphere (7e5 columns)

	List	Vector	Set	Heap	A-Heap	A-Set	A-Full	**A-Bit-Tree**
twist*	2635.4	339.9	4.9	2.0	2.5	6.1	2.1	**1.0**
spectral*	2644.8	349.2	5.2	1.9	3.3	6.6	3.1	**1.0**

Dataset: 3-skeleton of the Vietoris–Rips filtration of 192 points on a 2-sphere (6e7 columns)

Phat performance

	List	Vector	Set	Heap	A-Heap	A-Set	A-Full	**A-Bit-Tree**
standard	141.1	16.1	23.9	17.5	19.4	21.5	10.8	10.7
standard*	460.2	39.6	27.5	18.8	20.8	22.8	14.0	9.8
twist	9.8	0.54	0.30	0.11	0.13	0.13	0.09	**0.07**
twist*	337.1	18.6	0.99	0.52	0.52	0.72	0.24	0.11
row	9.9	1.5	0.50	0.48	1.5	2.1	1.2	0.78
row*	350	43.8	1.0	0.93	24.5	44.0	15.5	7.0
chunk	1.8	0.19	0.19	0.12	0.09	0.09	0.08	0.08
chunk*	5.7	0.53	0.27	0.22	0.19	0.17	0.13	0.14
spectral	8.4	0.78	0.19	0.11	0.11	0.11	0.08	**0.06**
spectral*	339.2	21.9	0.90	0.65	0.74	0.74	0.35	0.12

Dataset: 64^3 image dat set (2e6 columns)

	List	Vector	Set	Heap	A-Heap	A-Set	A-Full	**A-Bit-Tree**
twist	2080.2	101.7	26.4	11.3	11.1	12.3	10.4	**8.8**
chunk	894.5	156.1	9.8	6.5	6.3	6.2	5.6	**4.7**
spectral	1197.7	261.7	11.9	8.5	8.5	8.4	7.1	6.1

Dataset: 256^3 image dat set (1e8 columns)

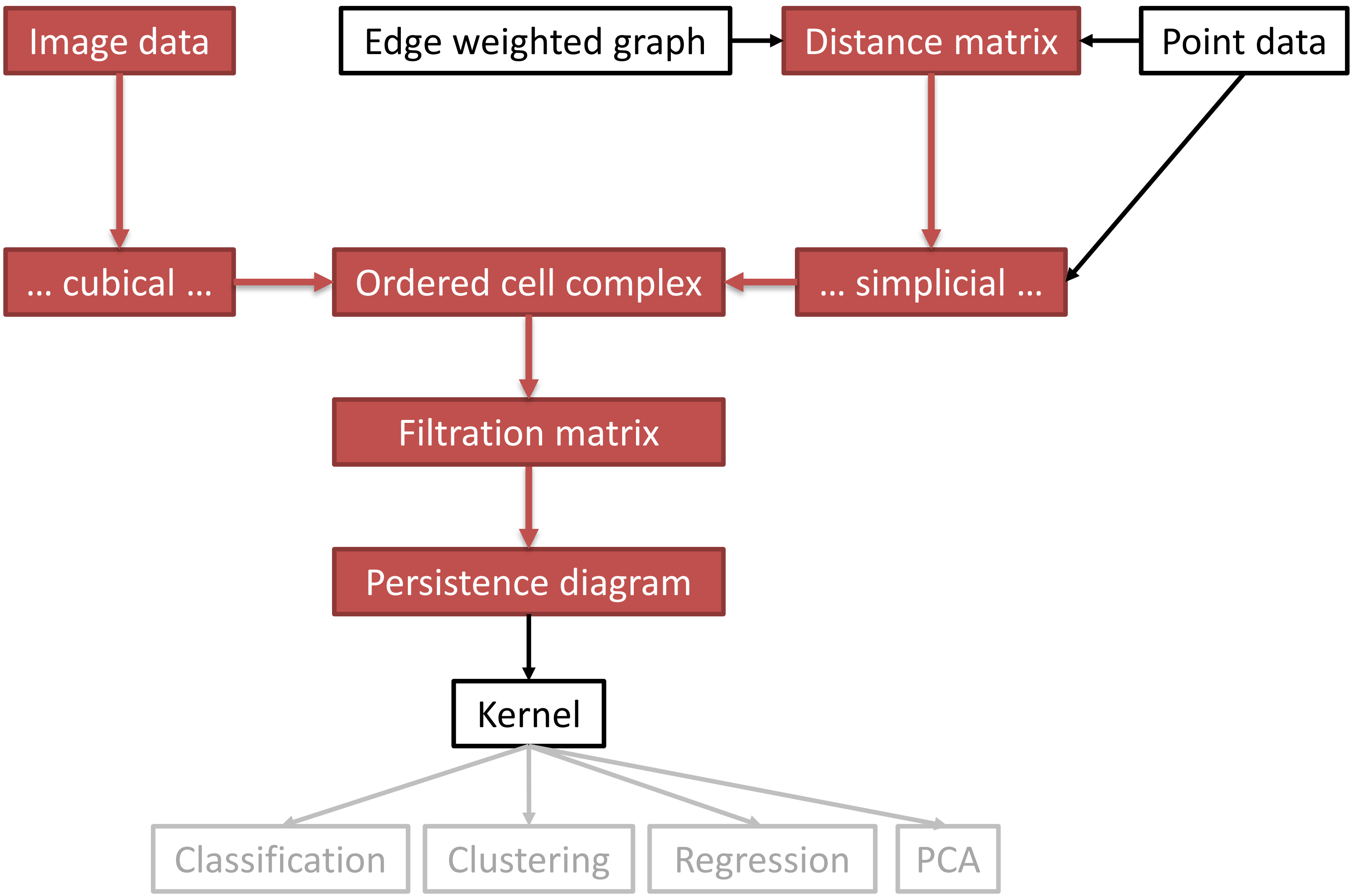
Phat comparison

	Dionysus	JavaPlex	Perseus	Gudhi	Phat (simple)	Phat (opt)
Rips 64	2.6	4.4	18.0	0.15	0.02	0.01
Rips 192	359	465	(m)	9.8	2.0	1.0
Image 64^3		163	11 139		0.11	0.06
Image 256^3		(m)	(m)		11.3	4.7

	Dionysus	JavaPlex	Perseus	Gudhi	Phat (simple)	Phat (opt)
Rips 64	60 MB	270 MB	718 MB	44 MB	53 MB	61 MB
Rips 192	4.9 GB	12.3 GB	(m)	3.1 GB	3.6 GB	3.8 GB
Image 64^3		2.04 GB	1.5 GB		0.16 GB	0.16 GB
Image 256^3		(m)	(m)		10.2 GB	10.3 GB

Part 2: Dipha - `github.com/DIPHA/dipha`

Joint work with: Ulrich Bauer, M. Kerber



Why Distributed?

- Fast algorithms and data structures: Twist + Bit-tree + Dualization

→ Can process millions of simplices per second

→ Bottlenecked by memory consumption

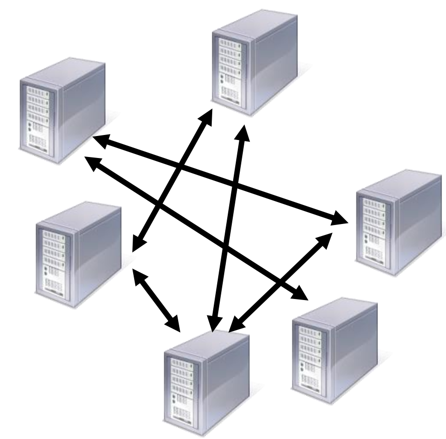
- Memory efficient approaches only for special cases

Why Distributed?

- Fast algorithms and data structures: Twist + Bit-tree + Dualization
→ Can process millions of simplices per second
→ Bottlenecked by memory consumption

- Memory efficient approaches only for special cases

vs.



IST Austria: 256 GB | 50 x 64 GB = 3.2 TB

Why Distributed?

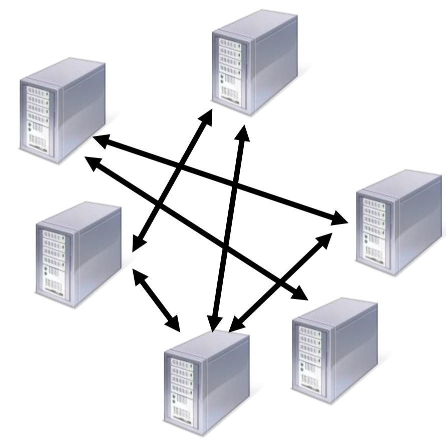
- Fast algorithms and data structures: Twist + Bit-tree + Dualization

→ Can process millions of simplices per second

→ Bottlenecked by memory consumption

- Memory efficient approaches only for special cases

vs.

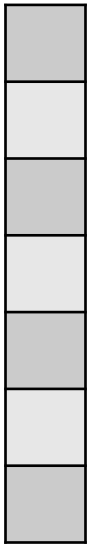


IST Austria: 256 GB 50 x 64 GB = 3.2 TB

Challenge:	High latency (> 1000x)

Spectral Sequence Matrix Reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							



Pivot-to-Column Map

Spectral Sequence Matrix Reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

3

Pivot-to-Column Map

Spectral Sequence Matrix Reduction

	1	2	3	4	5	6	7
1			1		1		
2			1			1	
3							1
4					1	1	
5							1
6							1
7							

Pivot-to-Column Map
3
5

Spectral Sequence Matrix Reduction

	1	2	3	4	5	6	7
1			1		1	1	
2			1			1	
3							1
4					1		
5							1
6							1
7							

3
5

Pivot-to-Column Map

Spectral Sequence Matrix Reduction

	1	2	3	4	5	6	7
1			1		1	1	
2			1			1	
3							1
4					1		
5							1
6							1
7							

3
5
7

Pivot-to-Column Map

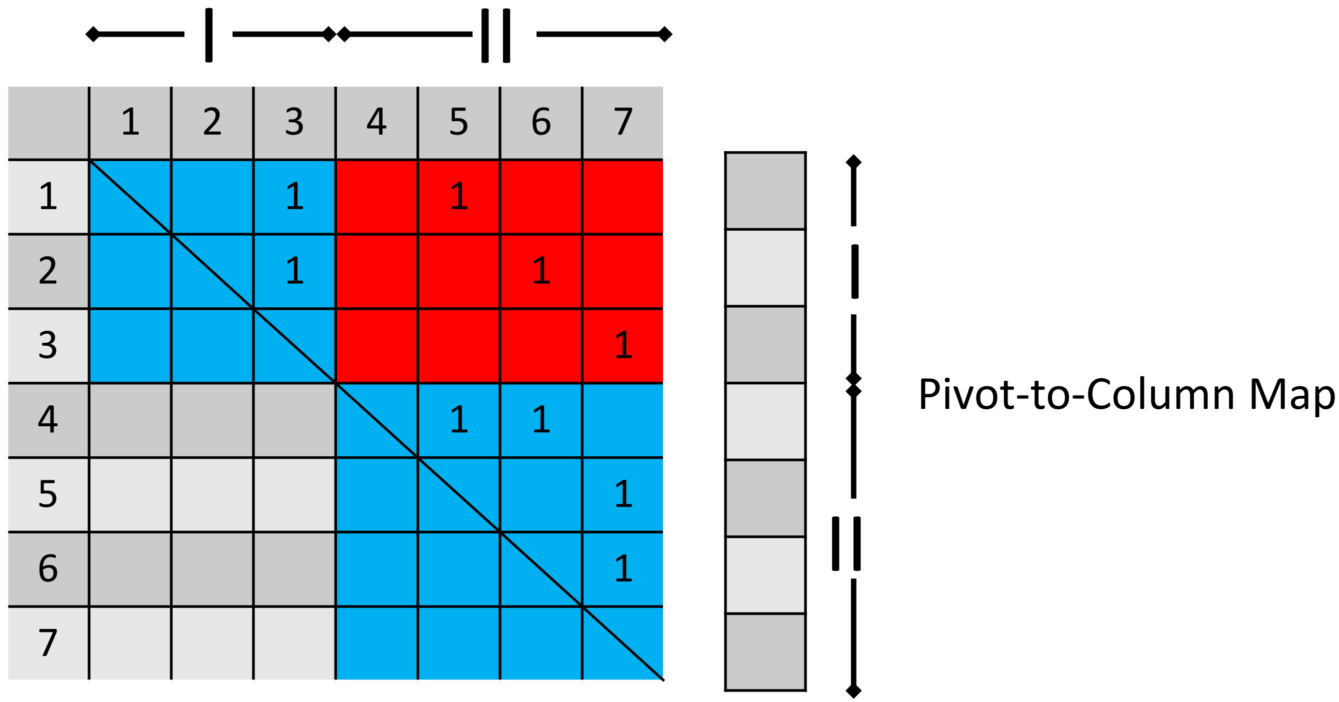
Spectral Sequence Matrix Reduction

	1	2	3	4	5	6	7
1			1		1		
2			1				
3							1
4					1		
5							1
6							1
7							

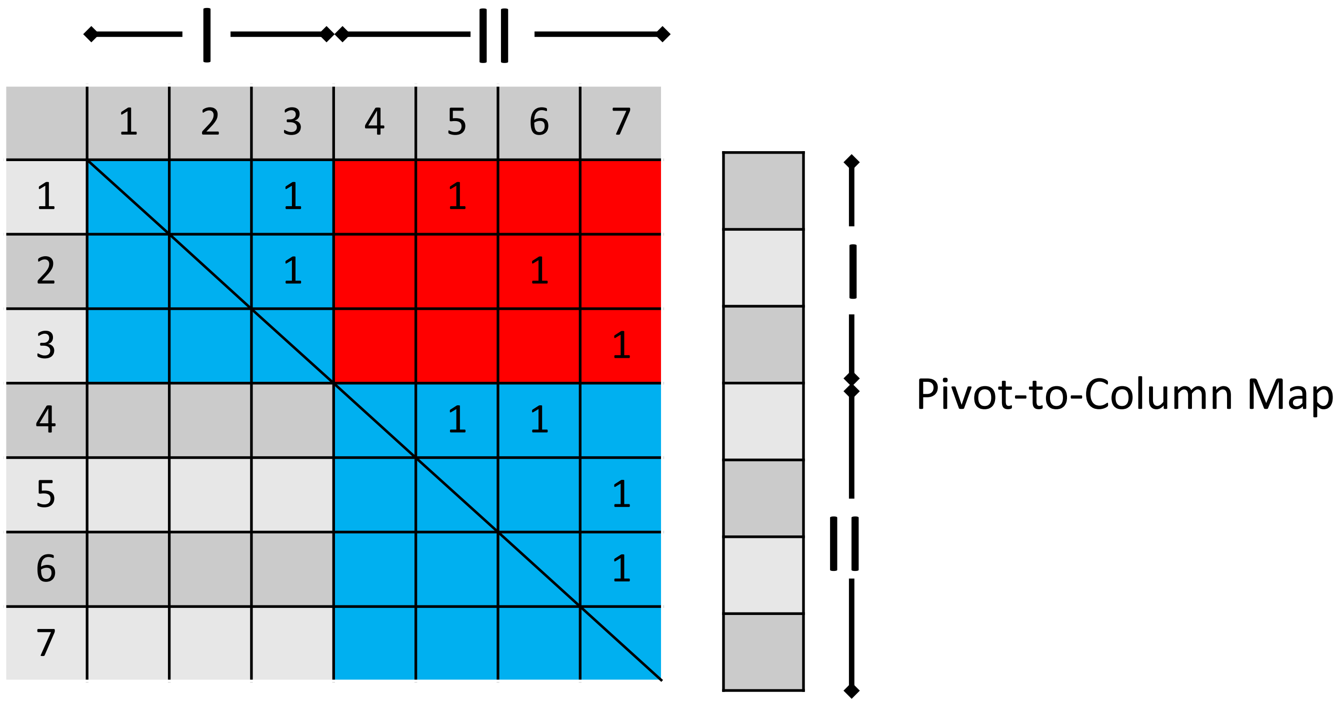
3
5
7

Pivot-to-Column Map

Inefficient Distributed Matrix Reduction

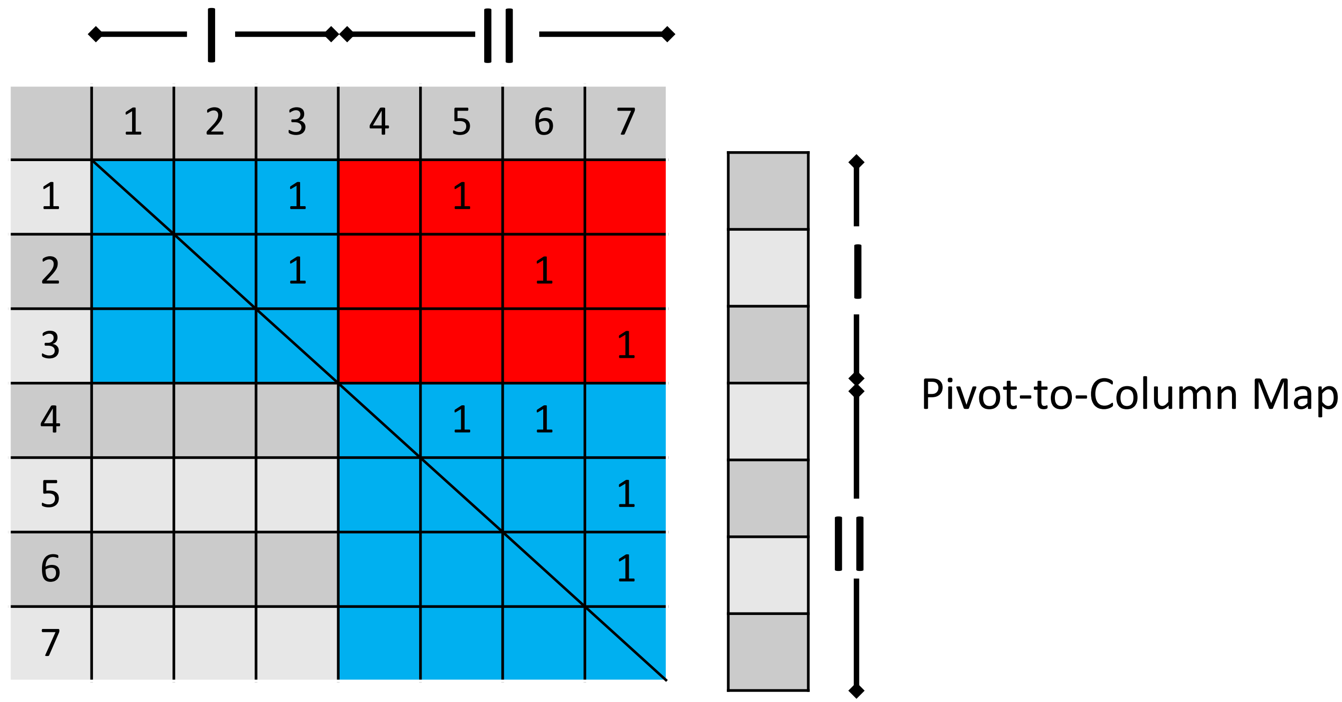


Inefficient Distributed Matrix Reduction



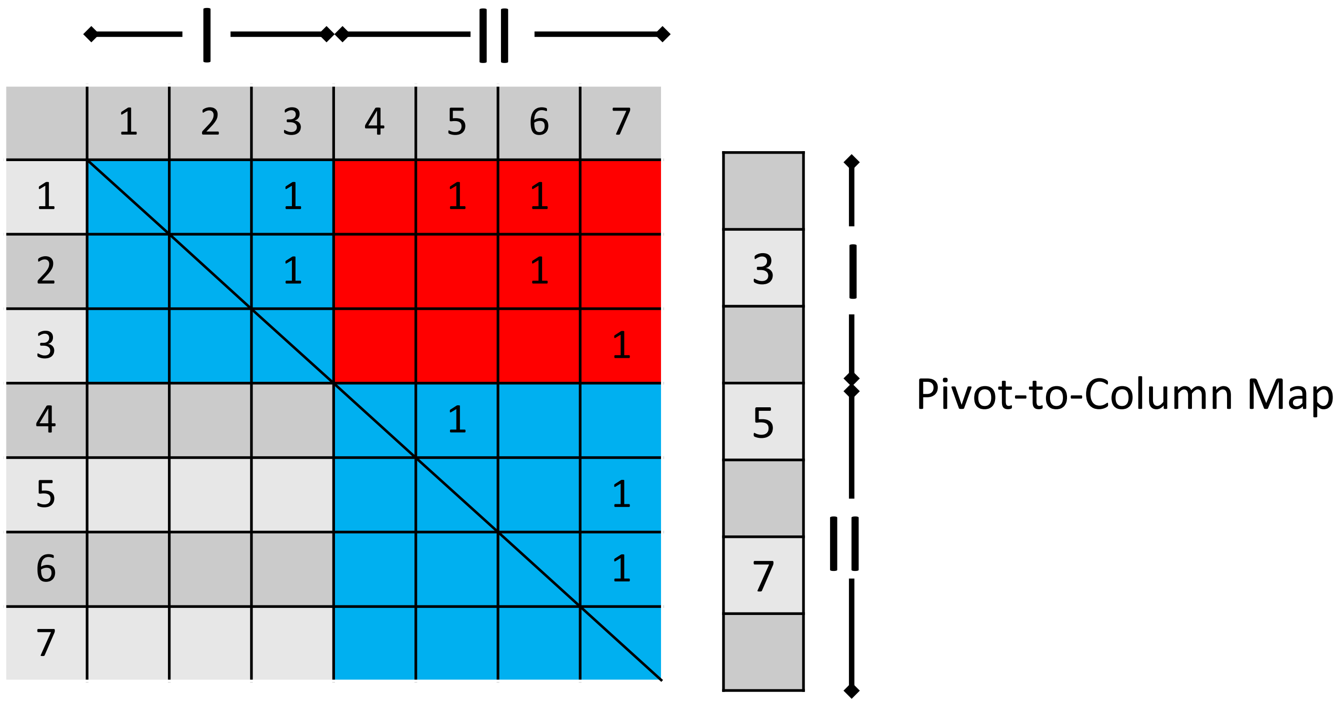
	Step 1
I	Load cols 1-3
II	Load cols 4-7

Inefficient Distributed Matrix Reduction



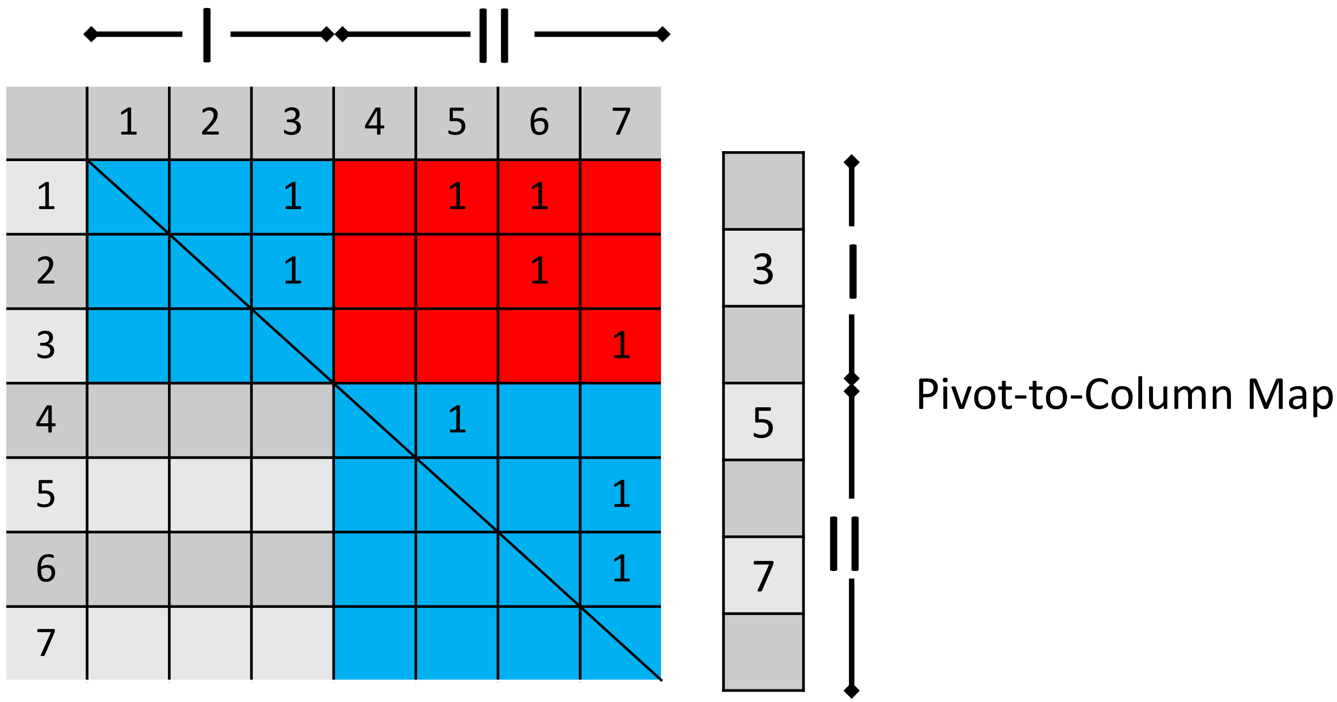
	Step 1	Step 2
I	Load cols 1-3	Reduce to idx 1
II	Load cols 4-7	Reduce to idx 4

Inefficient Distributed Matrix Reduction



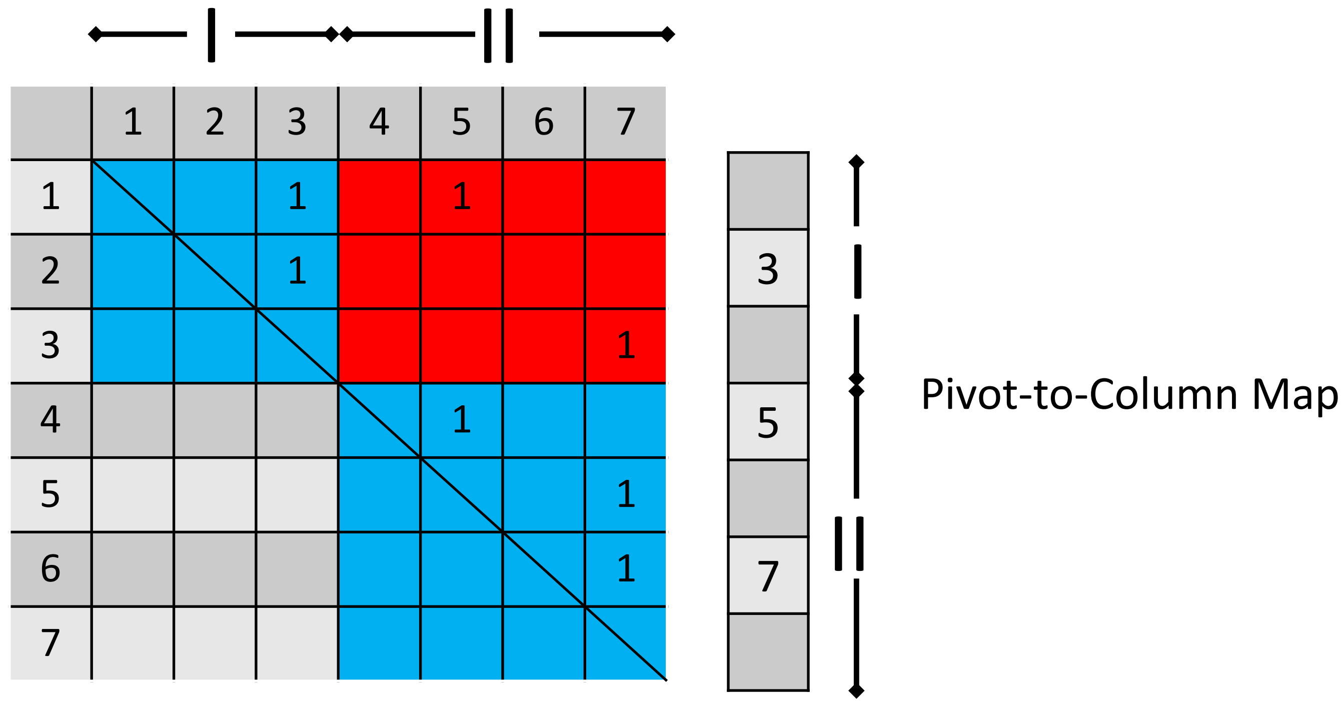
	Step 1	Step 2
I	Load cols 1-3	Reduce to idx 1
II	Load cols 4-7	Reduce to idx 4

Inefficient Distributed Matrix Reduction



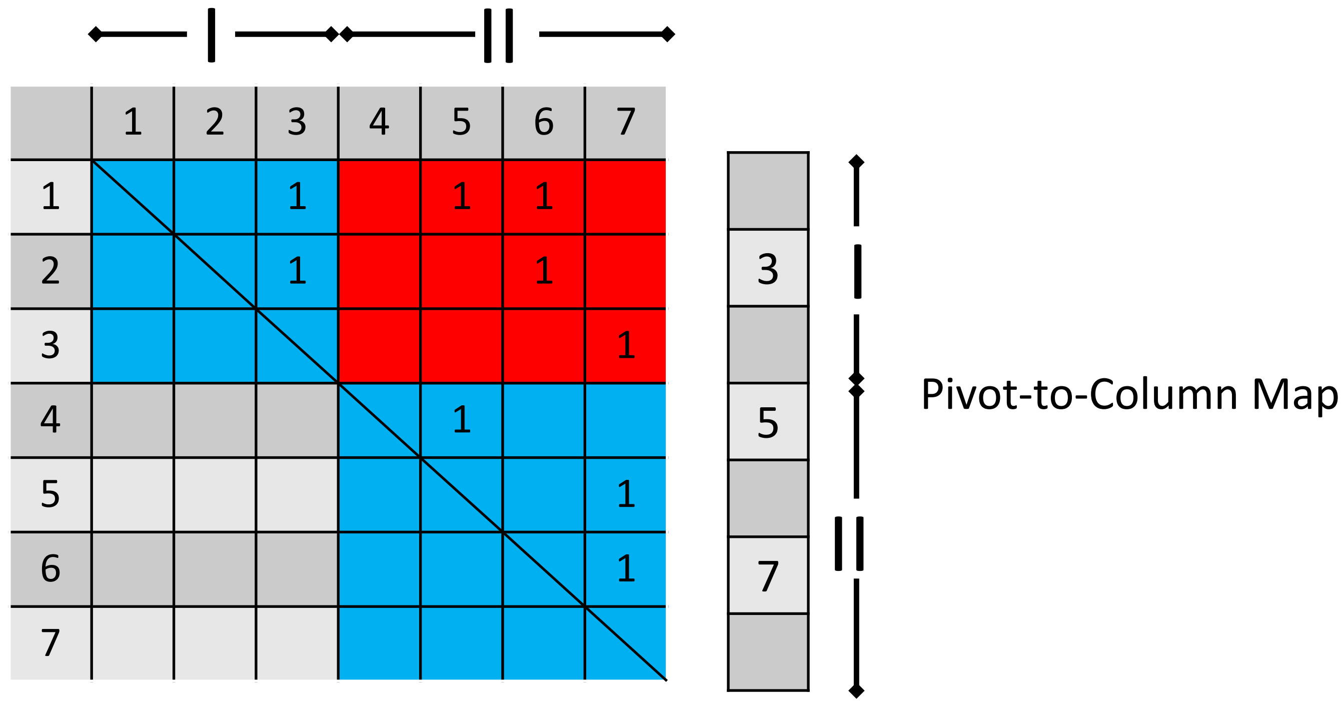
	Step 1	Step 2	Step 3
I	Load cols 1-3	Reduce to idx 1	idle
II	Load cols 4-7	Reduce to idx 4	Reduce to idx 1

Inefficient Distributed Matrix Reduction



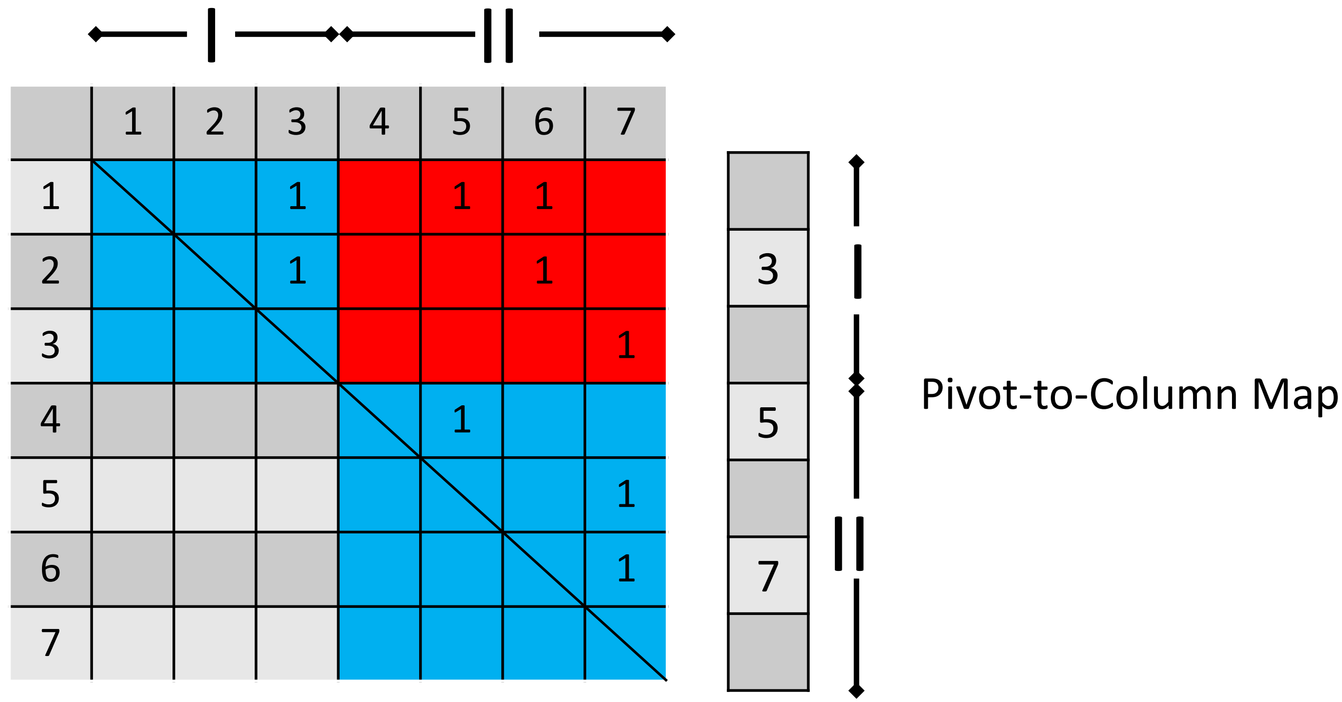
	Step 1	Step 2	Step 3
I	Load cols 1-3	Reduce to idx 1	idle
II	Load cols 4-7	Reduce to idx 4	Reduce to idx 1

Efficient Distributed Matrix Reduction



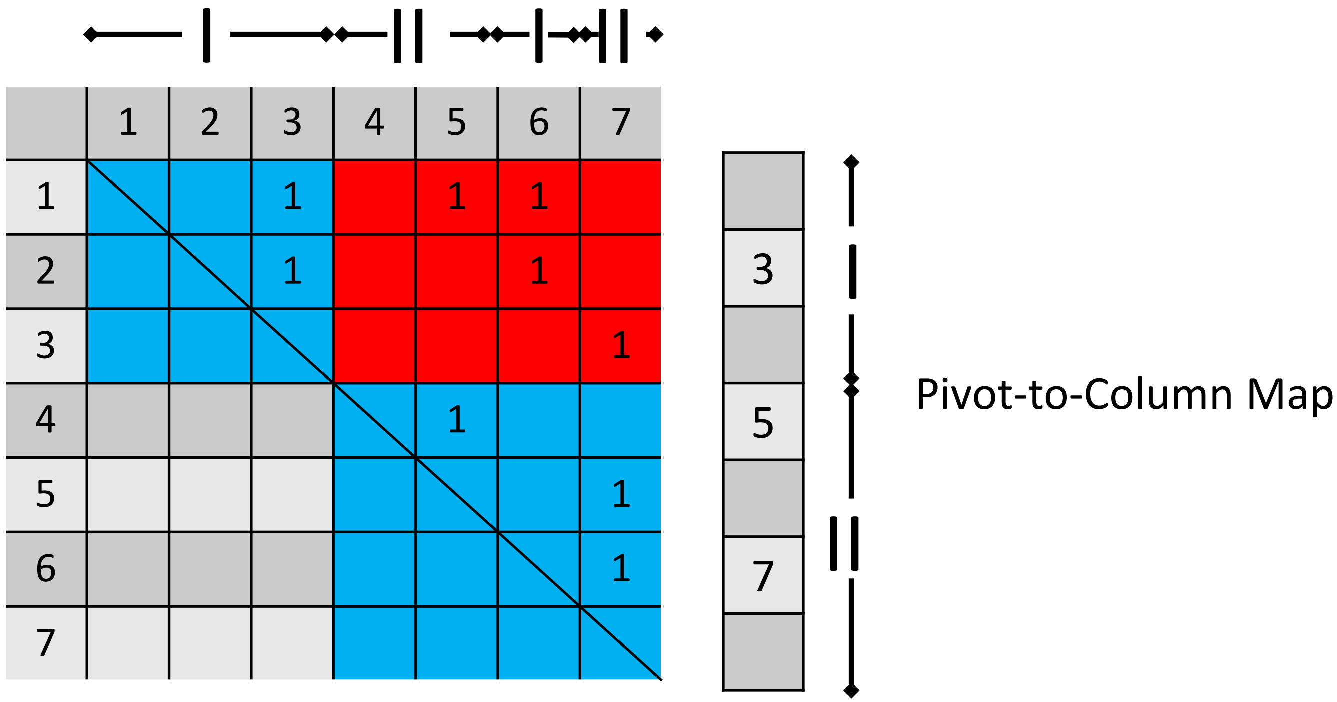
	Step 1	**Step 2**
I	Load cols 1-3	Reduce to idx 1
II	Load cols 4-7	Reduce to idx 4

Efficient Distributed Matrix Reduction



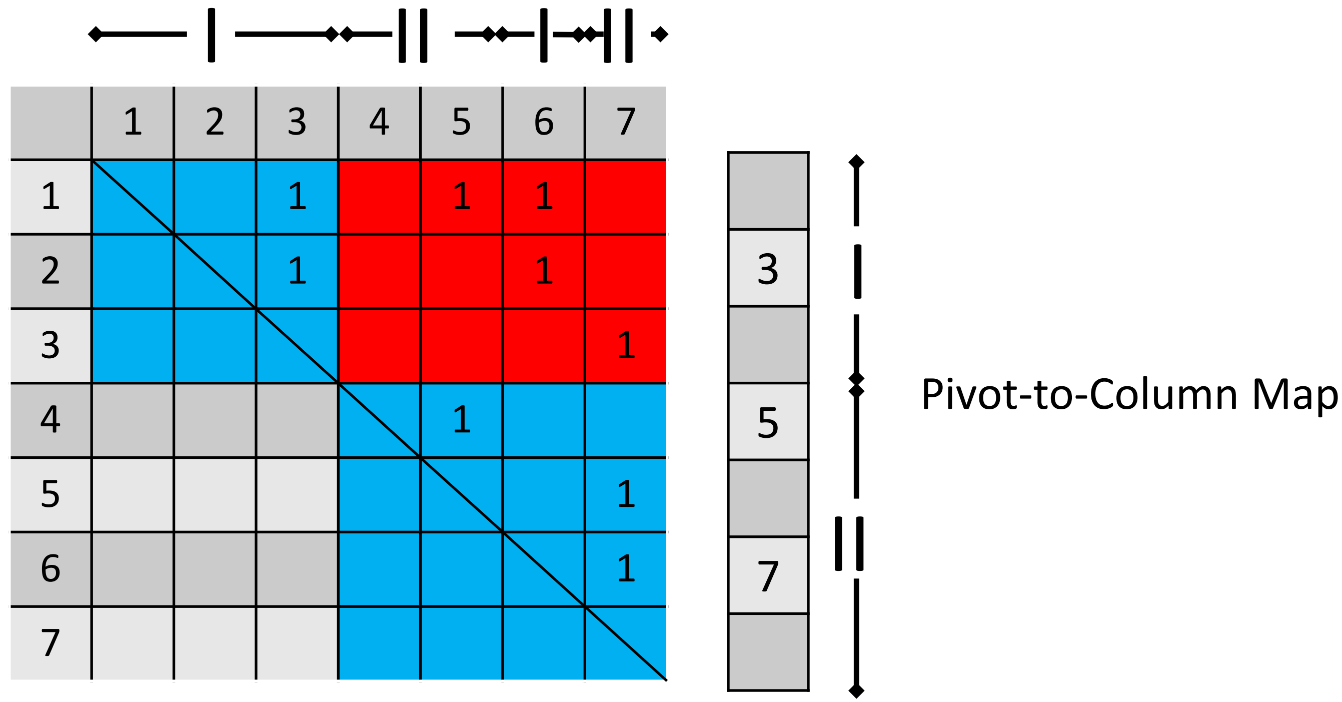
	Step 1	Step 2	Step 3
I	Load cols 1-3	Reduce to idx 1	Recv. unreduced cols from II
II	Load cols 4-7	Reduce to idx 4	Send. unreduced cols to I

Efficient Distributed Matrix Reduction



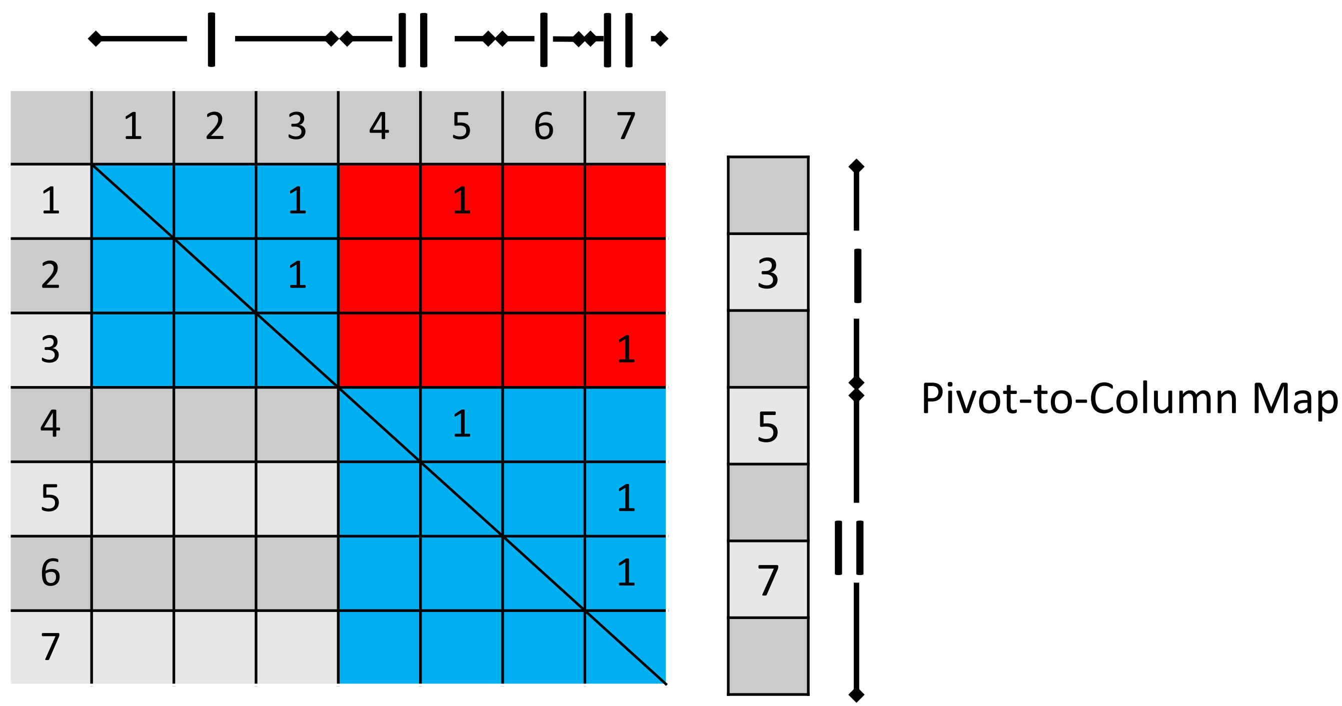
	Step 1	Step 2	Step 3
I	Load cols 1-3	Reduce to idx 1	Recv. unreduced cols from II
II	Load cols 4-7	Reduce to idx 4	Send. unreduced cols to I

Efficient Distributed Matrix Reduction



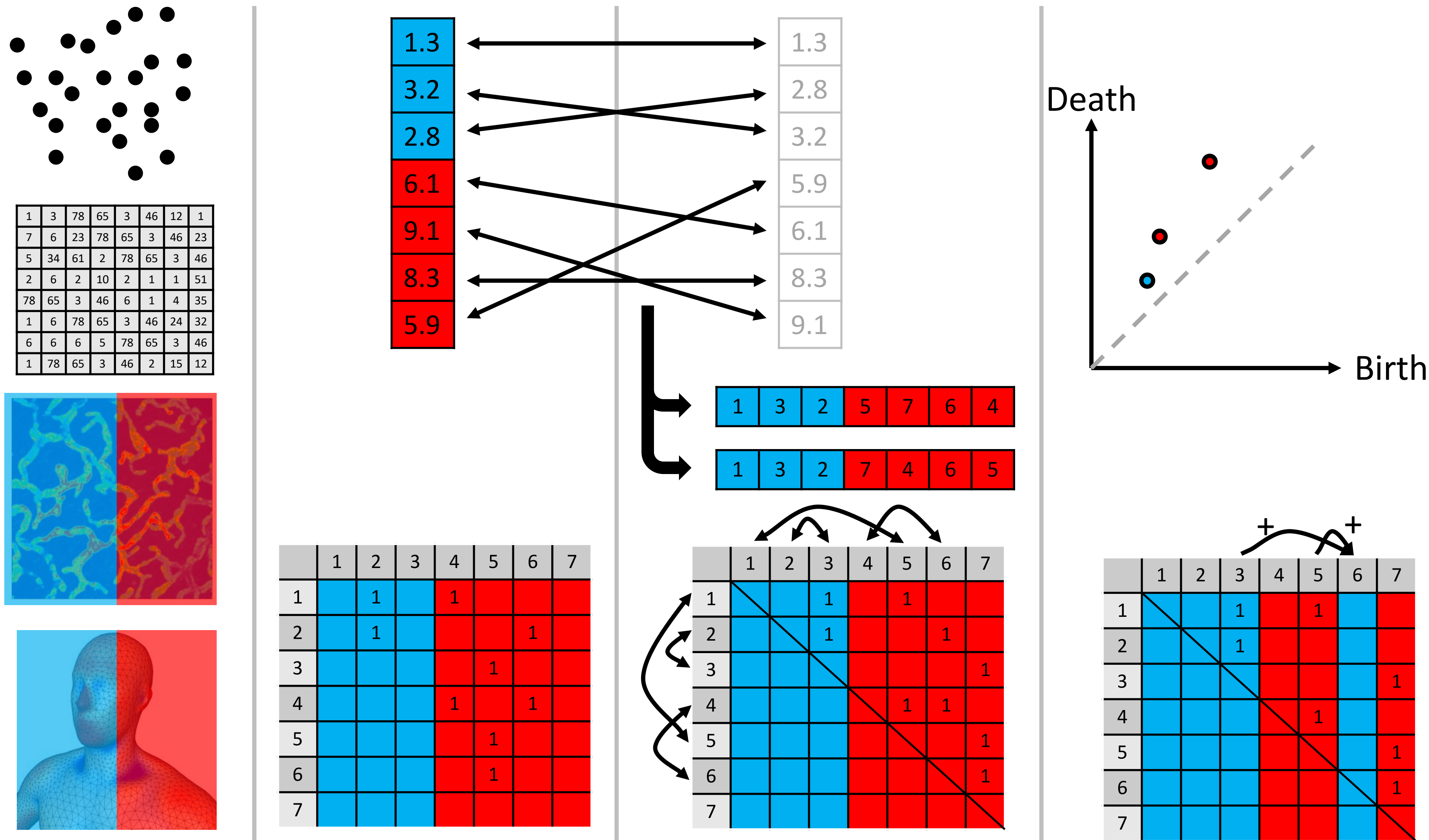
	Step 1	Step 2	Step 3	Step 4
I	Load cols 1-3	Reduce to idx 1	Recv. unreduced cols from II	Reduce to idx 1
II	Load cols 4-7	Reduce to idx 4	Send. unreduced cols to I	idle

Efficient Distributed Matrix Reduction



	Step 1	Step 2	Step 3	Step 4
I	Load cols 1-3	Reduce to idx 1	Recv. unreduced cols from II	Reduce to idx 1
II	Load cols 4-7	Reduce to idx 4	Send. unreduced cols to I	idle

Distributed Computational Pipeline



Input data

Boundary matrix
+ weights

Filtration matrix
+ permutation maps

Reduced matrix
+ diagram

Dipha summary

1. Identical computational cost as *Spectral Sequence* algorithm

Dipha summary

1. Identical computational cost as *Spectral Sequence* algorithm

2. Twist, fast column addition, and dualization are applicable

Dipha summary

1. Identical computational cost as *Spectral Sequence* algorithm
2. Twist, fast column addition, and dualization are applicable
3. N nodes → O(N) messages per node (→ latency insensitive)

Dipha summary

1. Identical computational cost as *Spectral Sequence* algorithm
2. Twist, fast column addition, and dualization are applicable
3. N nodes → O(N) messages per node (→ latency insensitive)
4. Only unreduced columns with high persistence get sent often
   (→ no communication bottleneck when using Gbit network)

Dipha summary

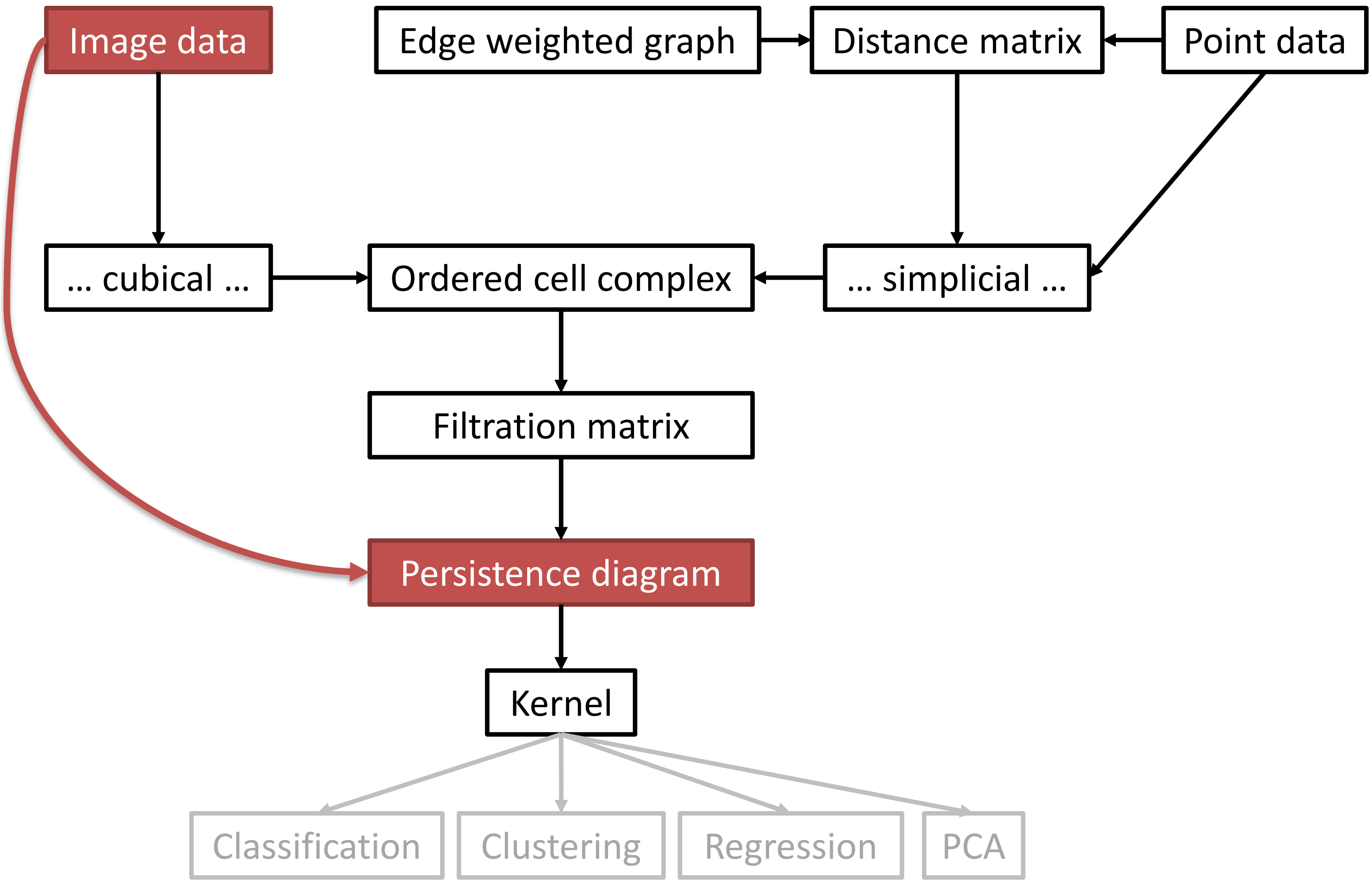
1. Identical computational cost as *Spectral Sequence* algorithm
2. Twist, fast column addition, and dualization are applicable
3. N nodes → O(N) messages per node (→ latency insensitive)
4. Only unreduced columns with high persistence get sent often
   (→ no communication bottleneck when using Gbit network)
5. Morse: 1024^3 noise image (→ 8 billion columns) with 256 nodes:
   → 13 min, 3.2 GB Memory peak, 10 GB peak traffic per node pair
   (single machine would need ~640 GB RAM)

Dipha summary

1. Identical computational cost as *Spectral Sequence* algorithm
2. Twist, fast column addition, and dualization are applicable
3. N nodes → O(N) messages per node (→ latency insensitive)
4. Only unreduced columns with high persistence get sent often
   (→ no communication bottleneck when using Gbit network)
5. Morse: 1024^3 noise image (→ 8 billion columns) with 256 nodes:
   → 13 min, 3.2 GB Memory peak, 10 GB peak traffic per node pair
   (single machine would need ~640 GB RAM)
6. Rips: 3000 points, dimension: 2 (→ 5 billion columns), with 380 nodes:
   → 20 min, 4 GB Memory peak

Part 3: Discrete Morse theory

Joint work with: D. Guenther, I. Hotz, H. Wagner

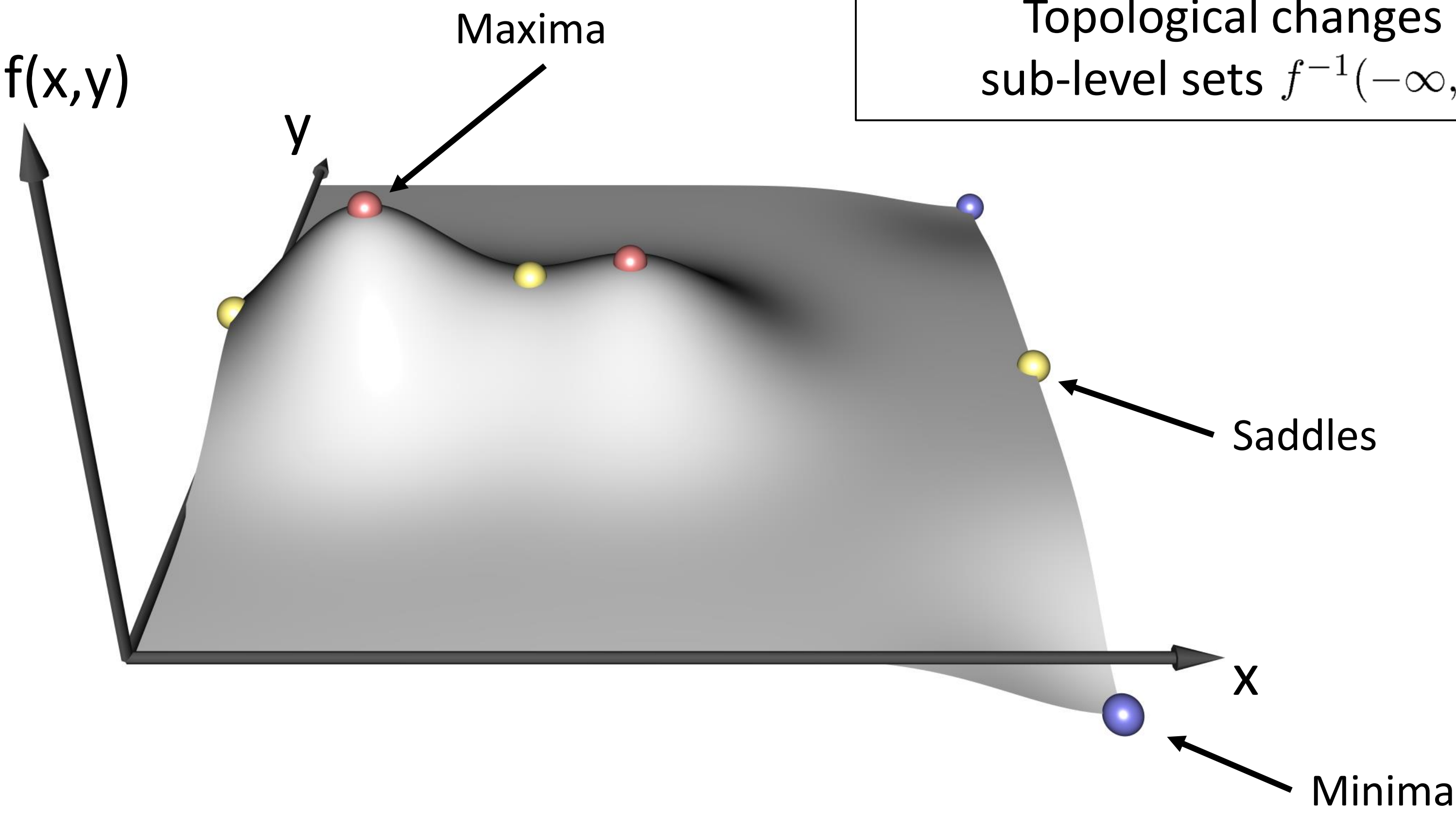


What is … Morse theory?

Critical points of $f : \Omega \to \mathbb{R}$

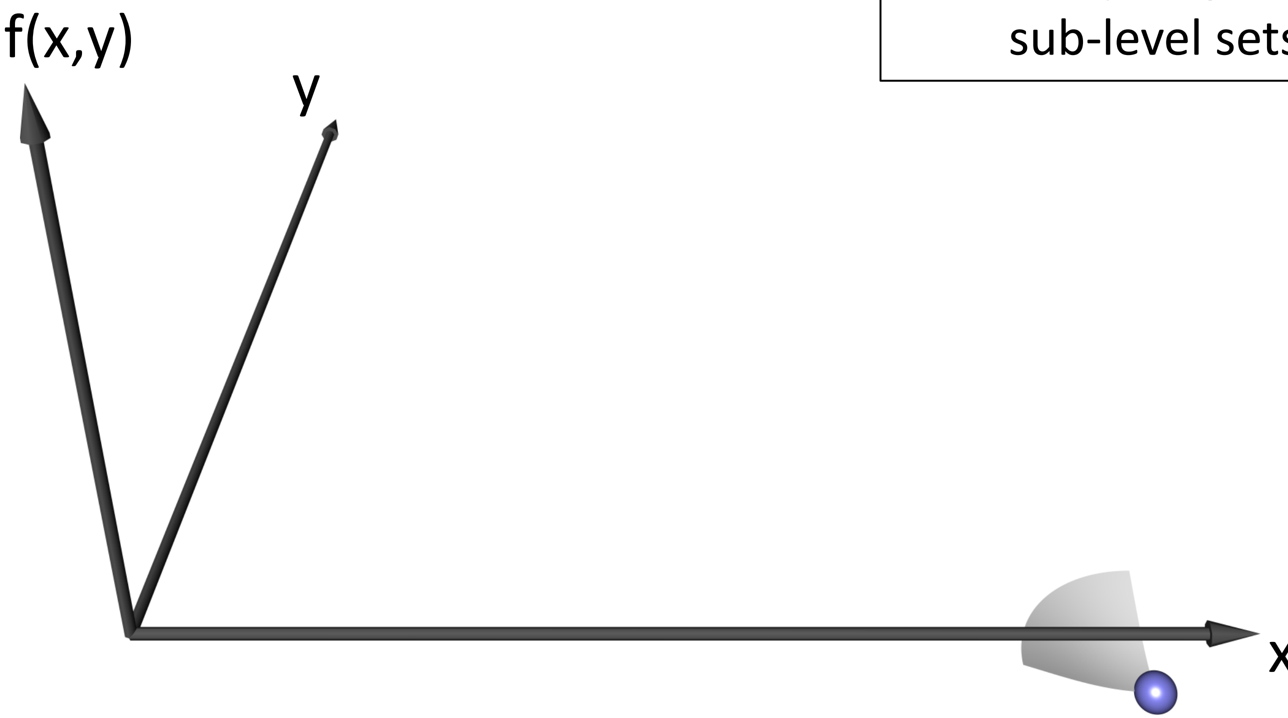
$\Leftrightarrow$

Topological changes of sub-level sets $f^{-1}(-\infty, t]$



What is … Morse theory?

Critical points of $f : \Omega \to \mathbb{R}$
$\Leftrightarrow$
Topological changes of
sub-level sets $f^{-1}(-\infty, t]$

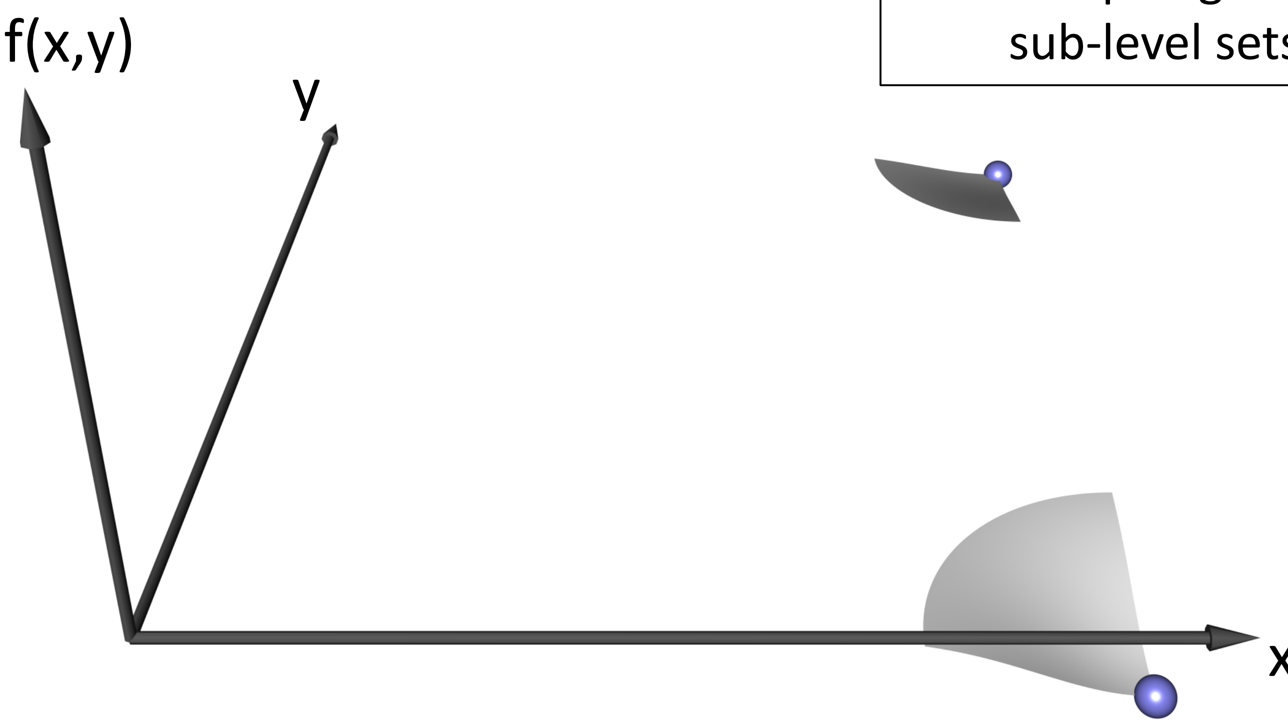


What is … Morse theory?

Critical points of $f : \Omega \to \mathbb{R}$

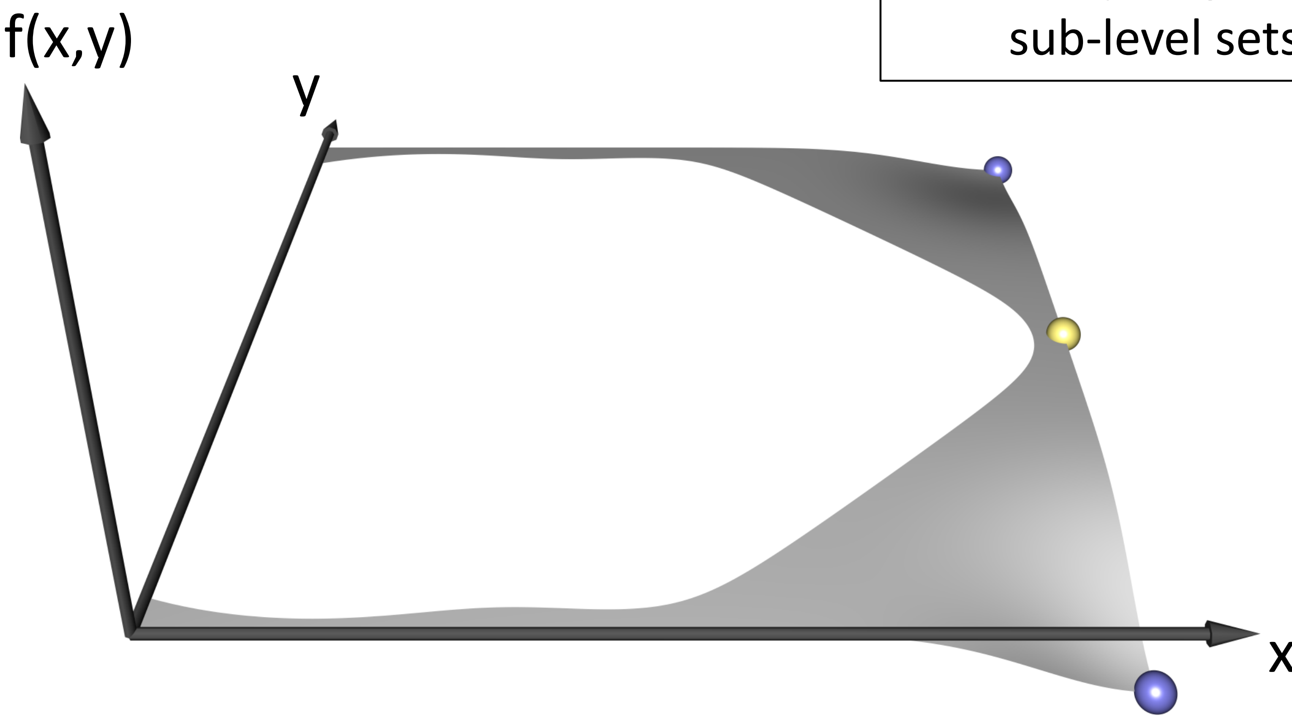
$\Leftrightarrow$

Topological changes of sub-level sets $f^{-1}(-\infty, t]$



What is … Morse theory?

Critical points of $f : \Omega \to \mathbb{R}$
$\Leftrightarrow$
Topological changes of
sub-level sets $f^{-1}(-\infty, t]$

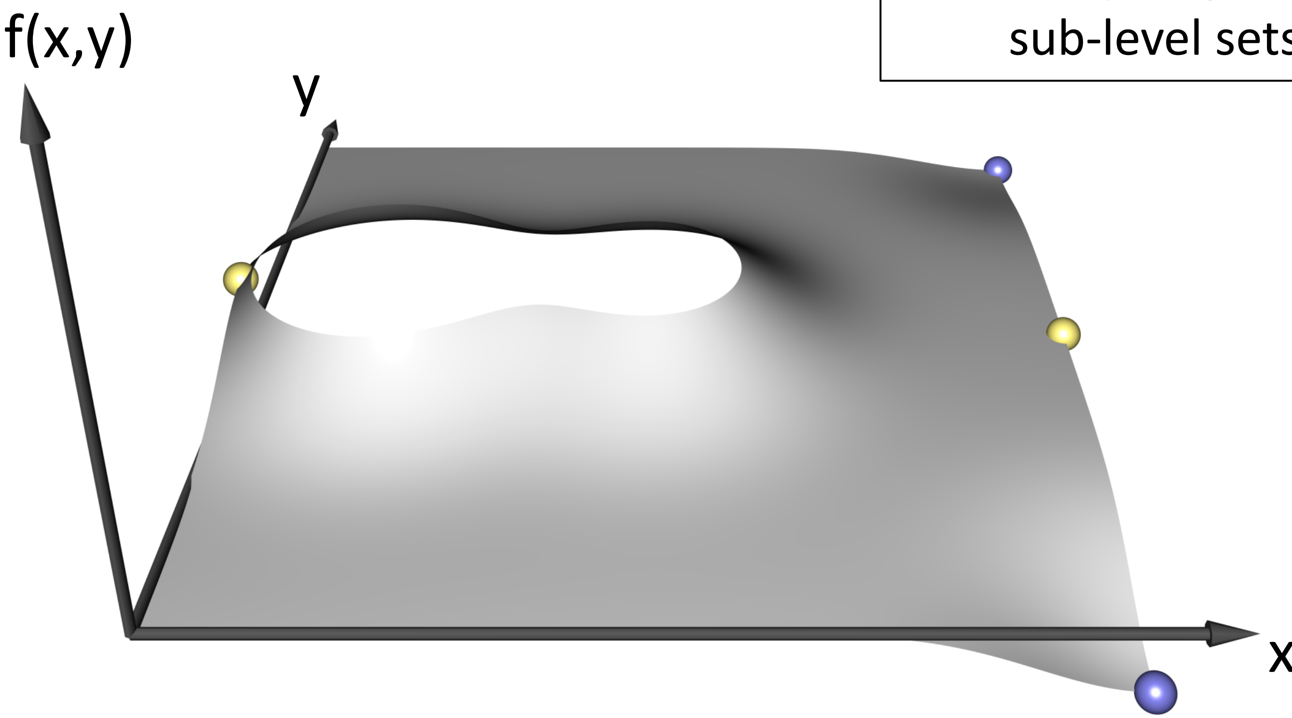


What is … Morse theory?

Critical points of $f : \Omega \to \mathbb{R}$

$\Leftrightarrow$

Topological changes of sub-level sets $f^{-1}(-\infty, t]$

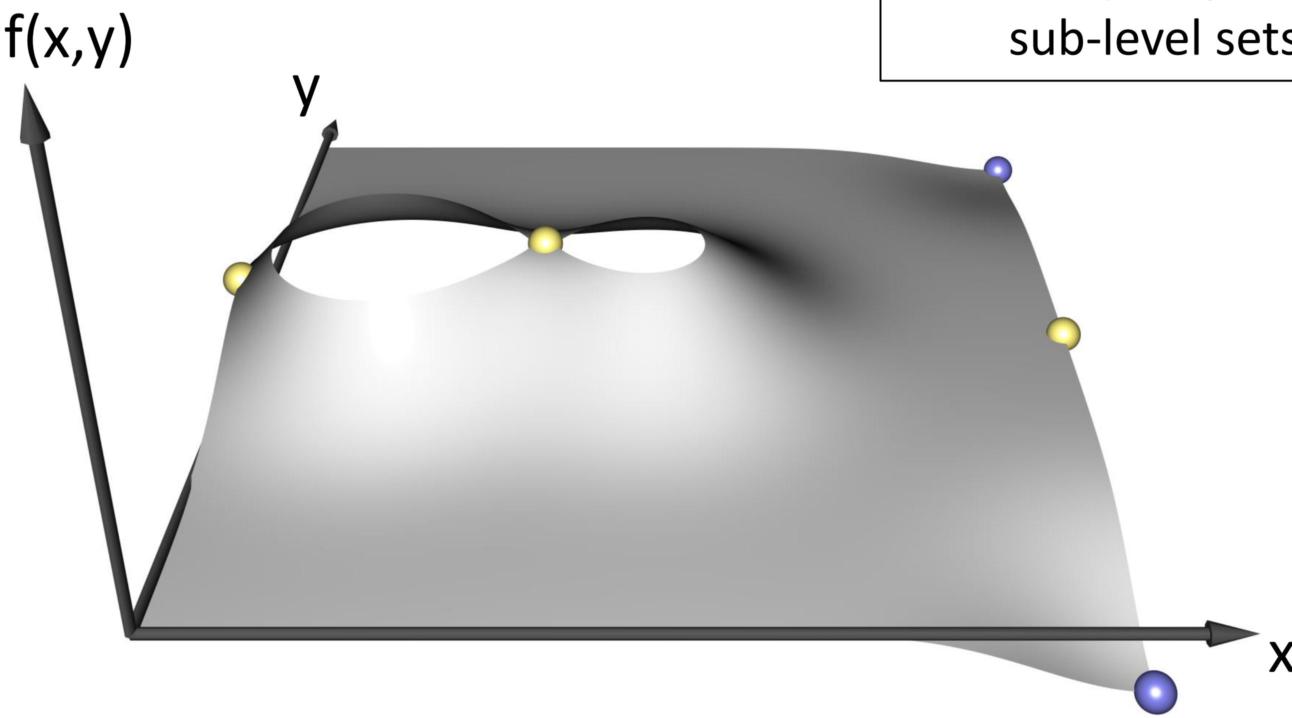


What is … Morse theory?

Critical points of $f : \Omega \to \mathbb{R}$

$\Leftrightarrow$

Topological changes of sub-level sets $f^{-1}(-\infty, t]$

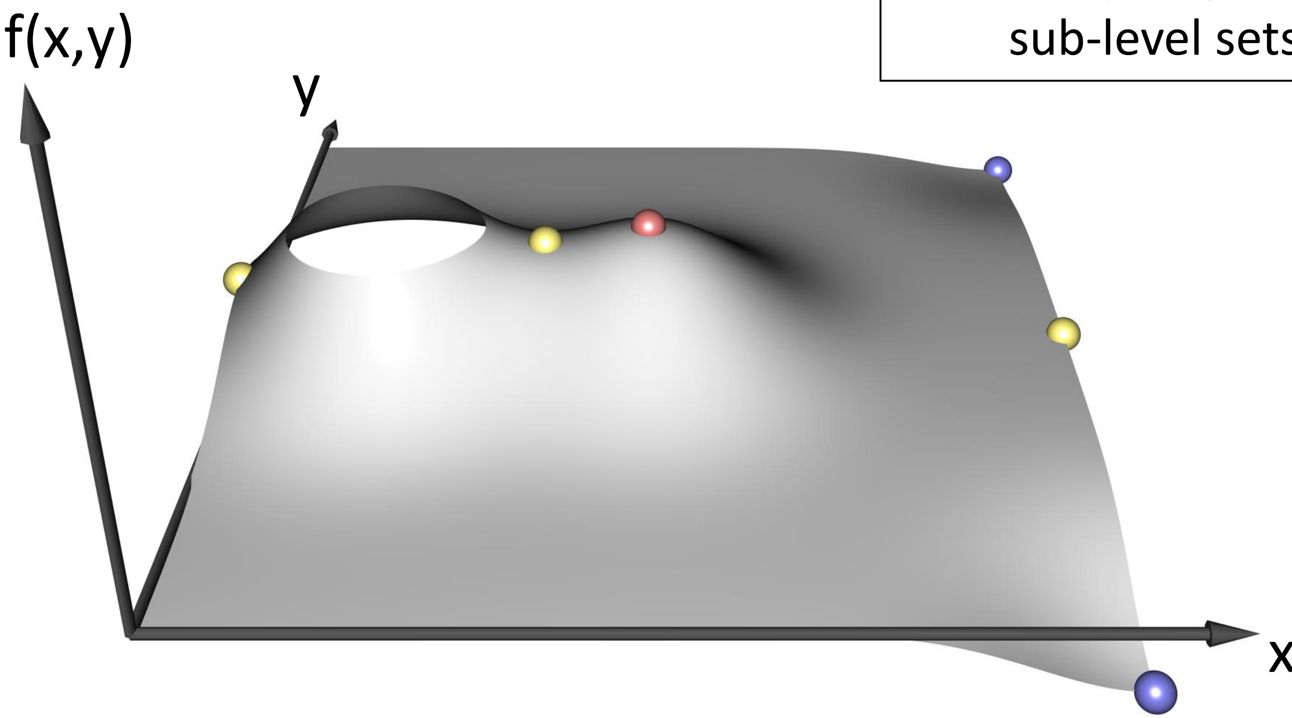


What is … Morse theory?

Critical points of $f : \Omega \to \mathbb{R}$

$\Leftrightarrow$

Topological changes of sub-level sets $f^{-1}(-\infty, t]$

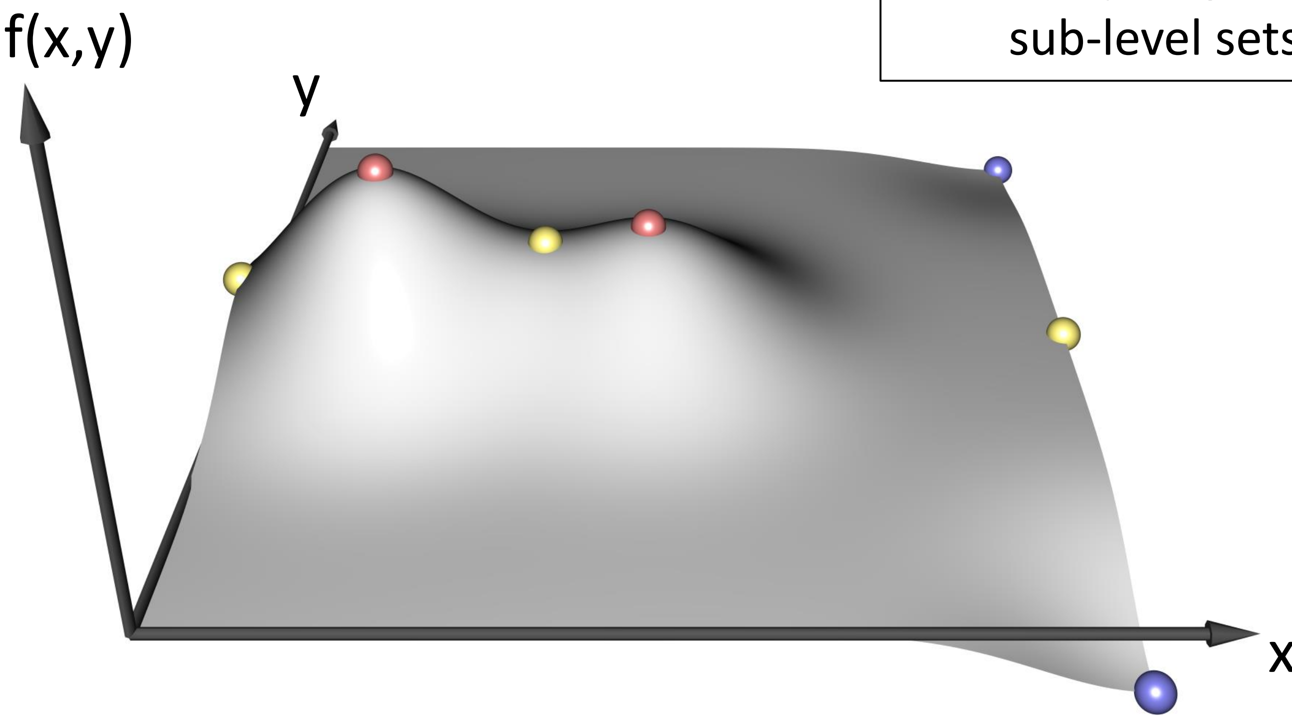


What is … Morse theory?

Critical points of $f : \Omega \to \mathbb{R}$

$\Leftrightarrow$

Topological changes of sub-level sets $f^{-1}(-\infty, t]$

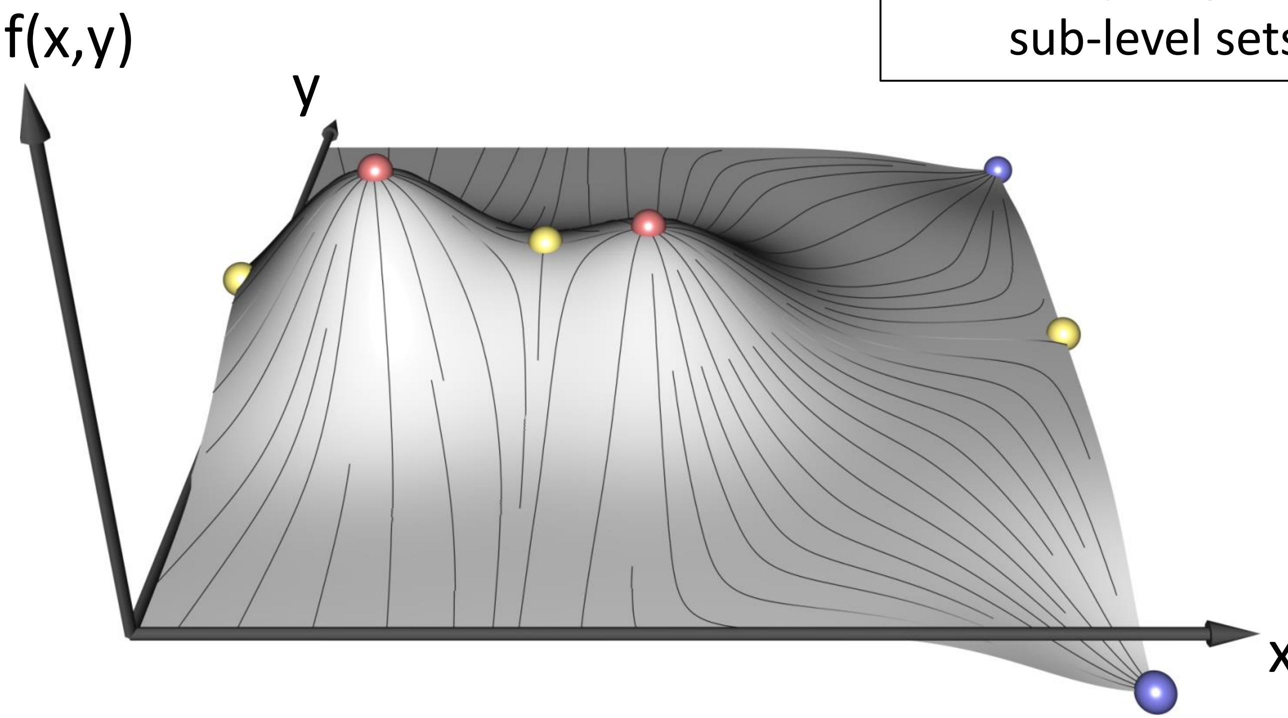


What is … Morse theory?

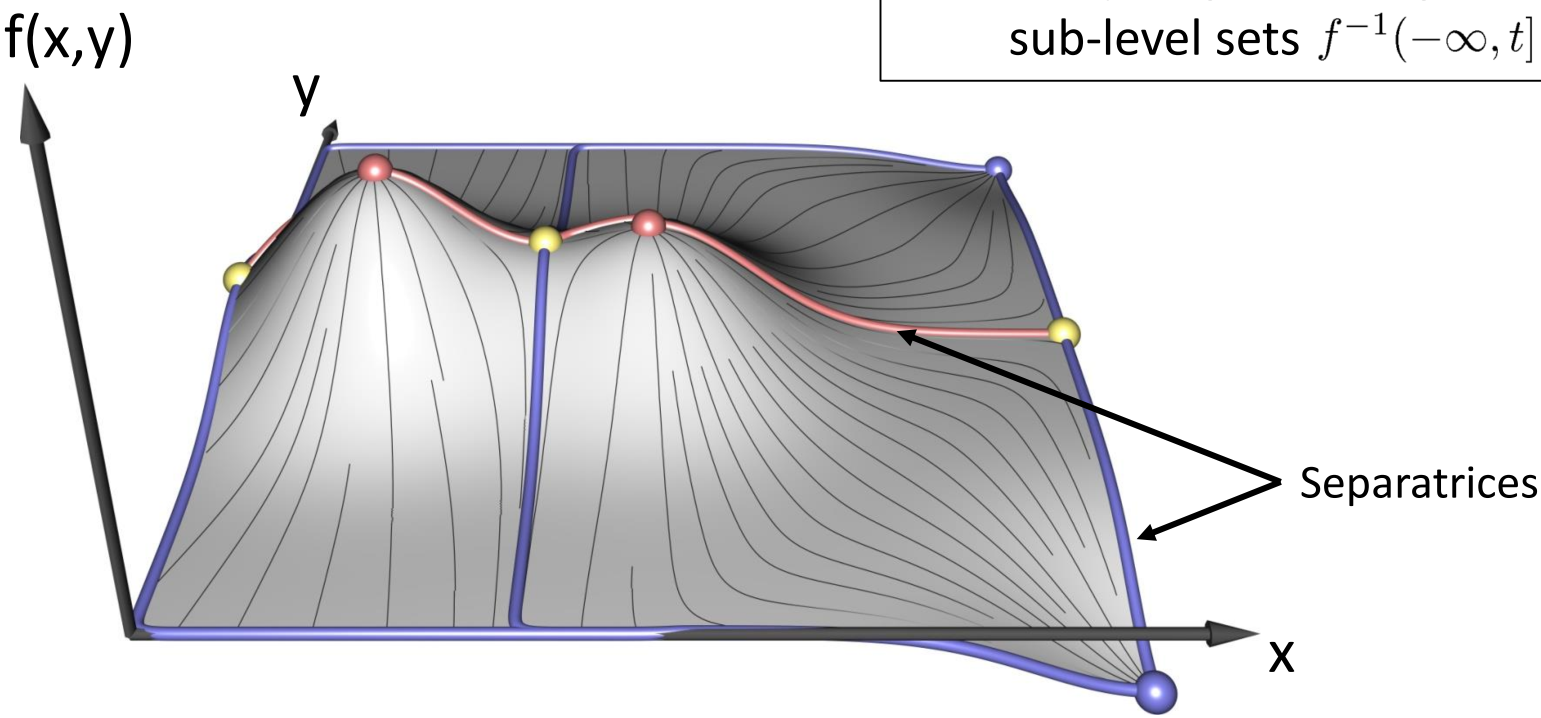
Critical points of $f : \Omega \to \mathbb{R}$

$\Leftrightarrow$

Topological changes of sub-level sets $f^{-1}(-\infty, t]$



What is … Morse theory?



Critical points + separatrices

$\Leftrightarrow$

Topological changes of sub-level sets $f^{-1}(-\infty, t]$

Computing persistent homology using Morse theory

d-cells

boundary maps

Cell complex

Morse-Smale complex

$$\begin{array}{ccccccccc} C: & C_3 & \xrightarrow{\partial_3} & C_2 & \xrightarrow{\partial_2} & C_1 & \xrightarrow{\partial_1} & C_0 \\ & \cup & & \cup & & \cup & & \cup \\ C_V: & C_3^V & \xrightarrow{\partial_3^V} & C_2^V & \xrightarrow{\partial_2^V} & C_1^V & \xrightarrow{\partial_1^V} & C_0^V \end{array}$$

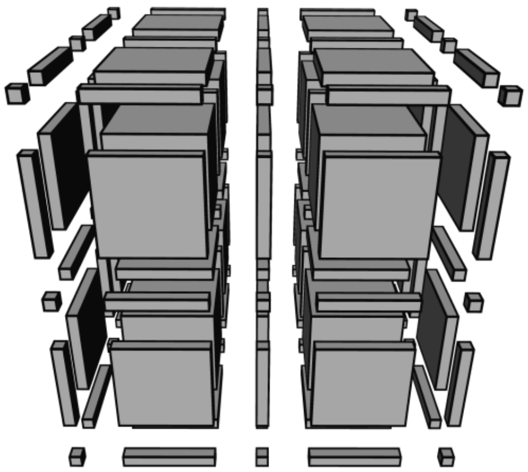
critical points of index d

separatrices

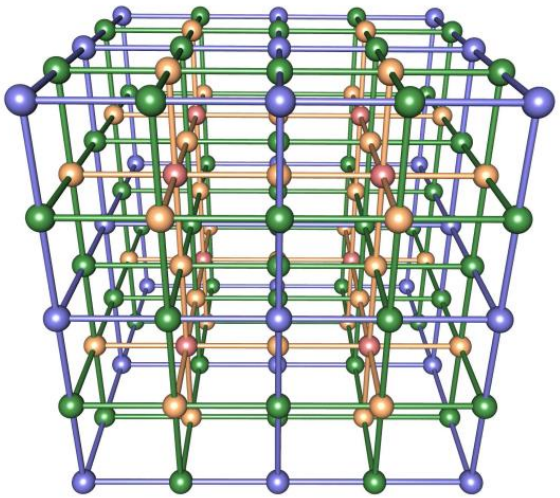
$$\boxed{Persistence(C, f) \quad = \quad Persistence(C_V, f)}$$

Discrete Morse theory intuition

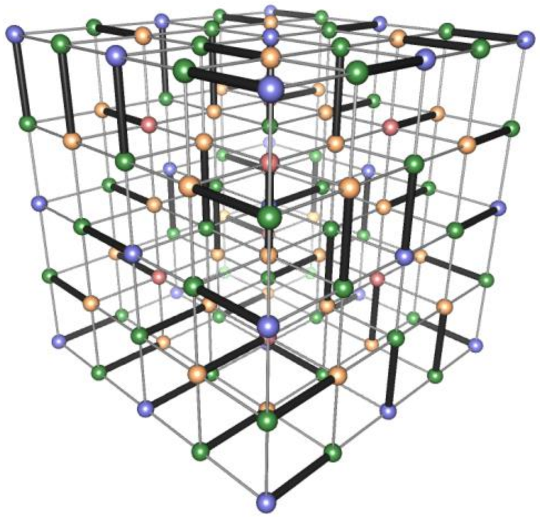
Combinatorial gradient field ⇔ *acyclic matching* of the cell graph



Cell complex



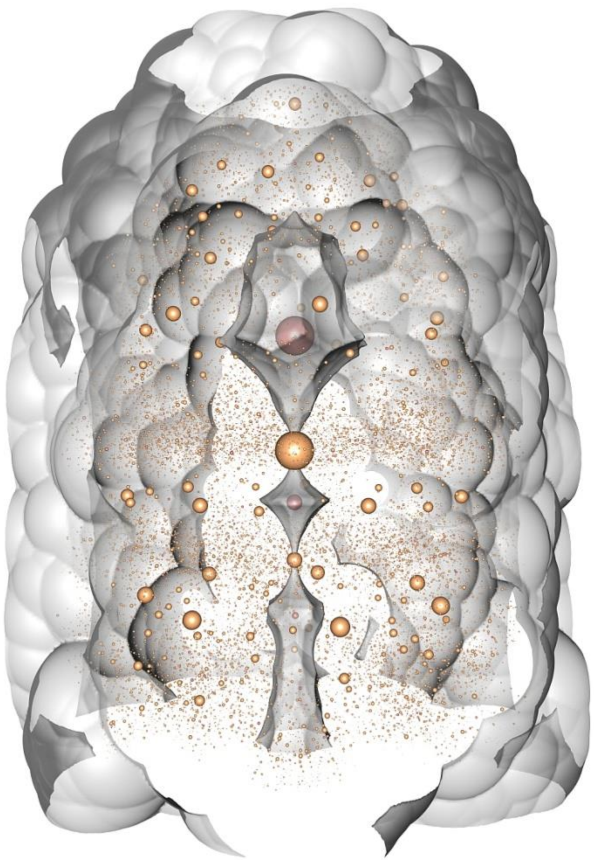
Cell graph



Acyclic matching

Critical points	–	unmatched nodes
Separatrices	–	alternating paths

Example



Data set size: 1120 x 1131 x 1552 (8 GB)

Total running time: ca. 5h

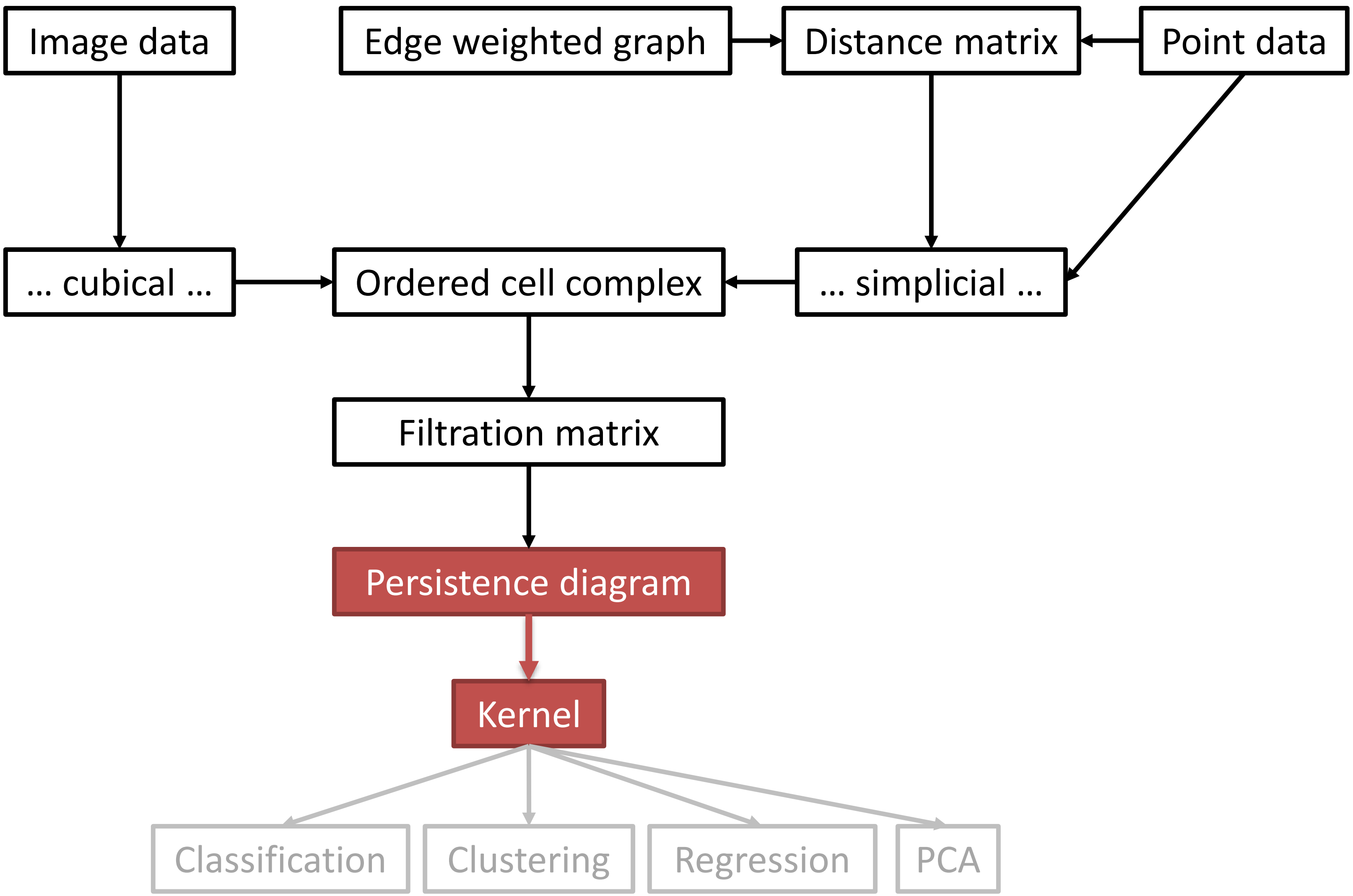
Memory requirement: 1.3 TB → 24 GB

Critical points scaled by persistence of a distance field of a molecular surface (data courtesy: D. Baum)

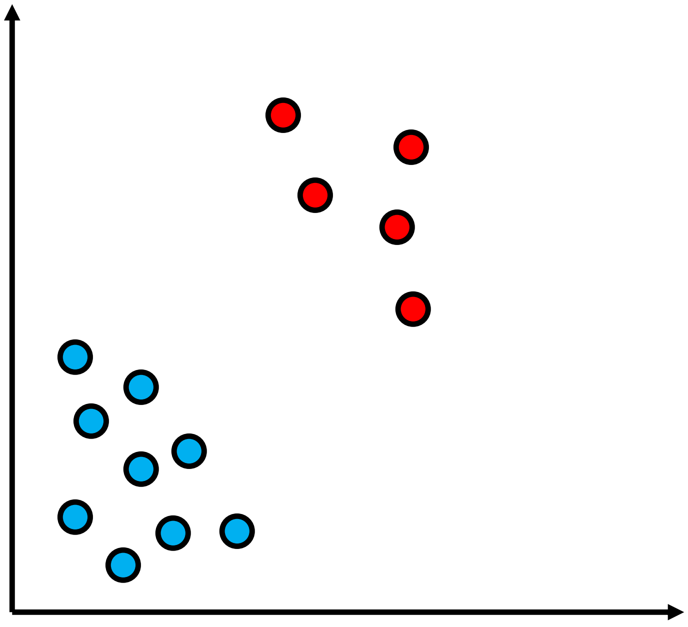
The full cell complex is never represented explicitly

Part 4: Persistence-Scale-Space kernel

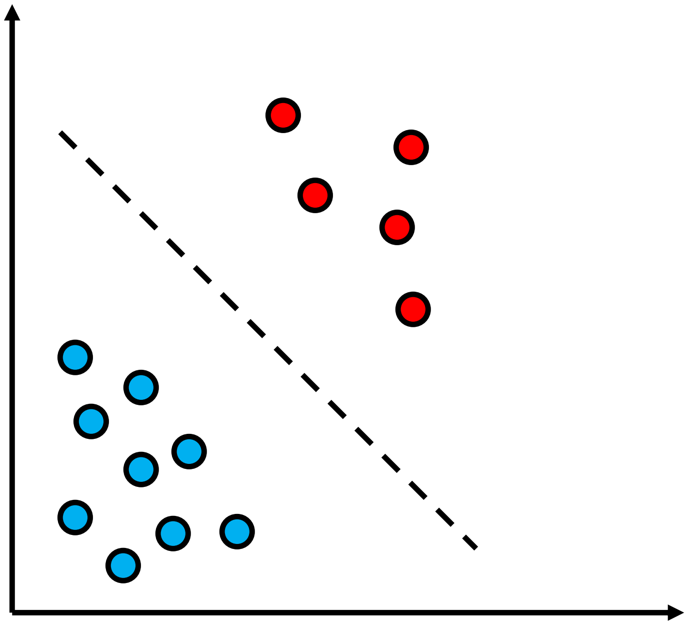
Joint work with: S. Huber, R. Kwitt, U. Bauer



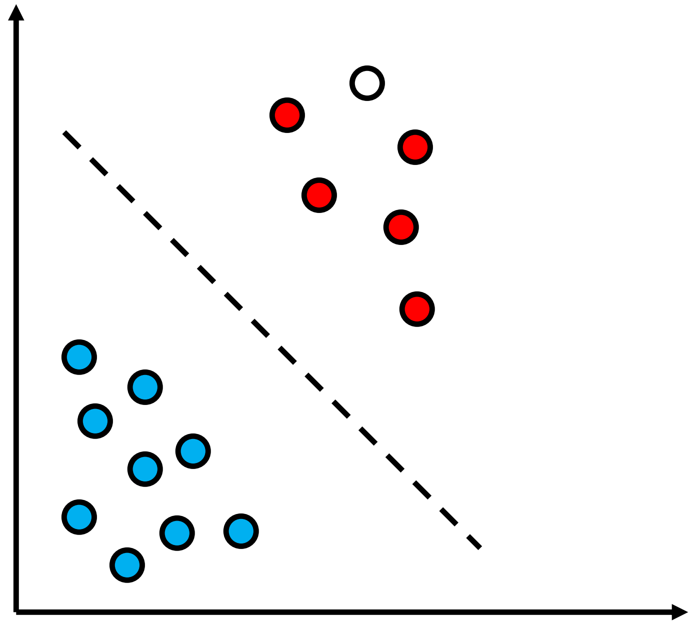
Topological Machine Learning



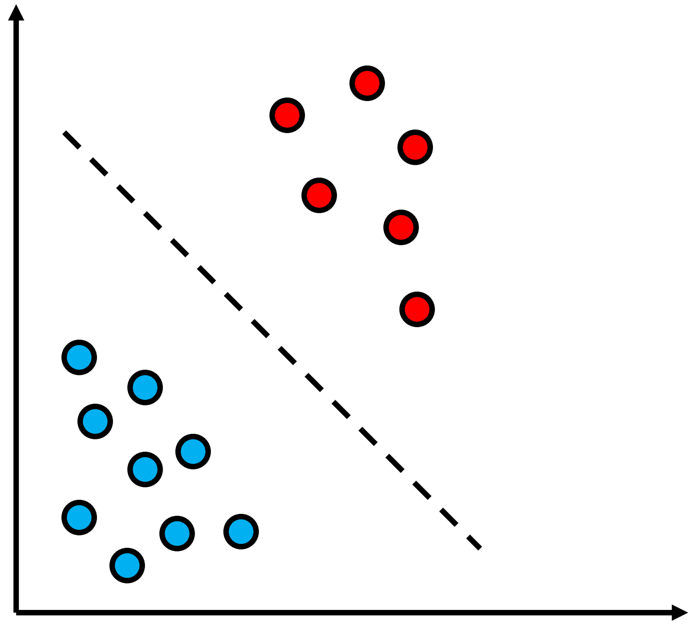
Topological Machine Learning



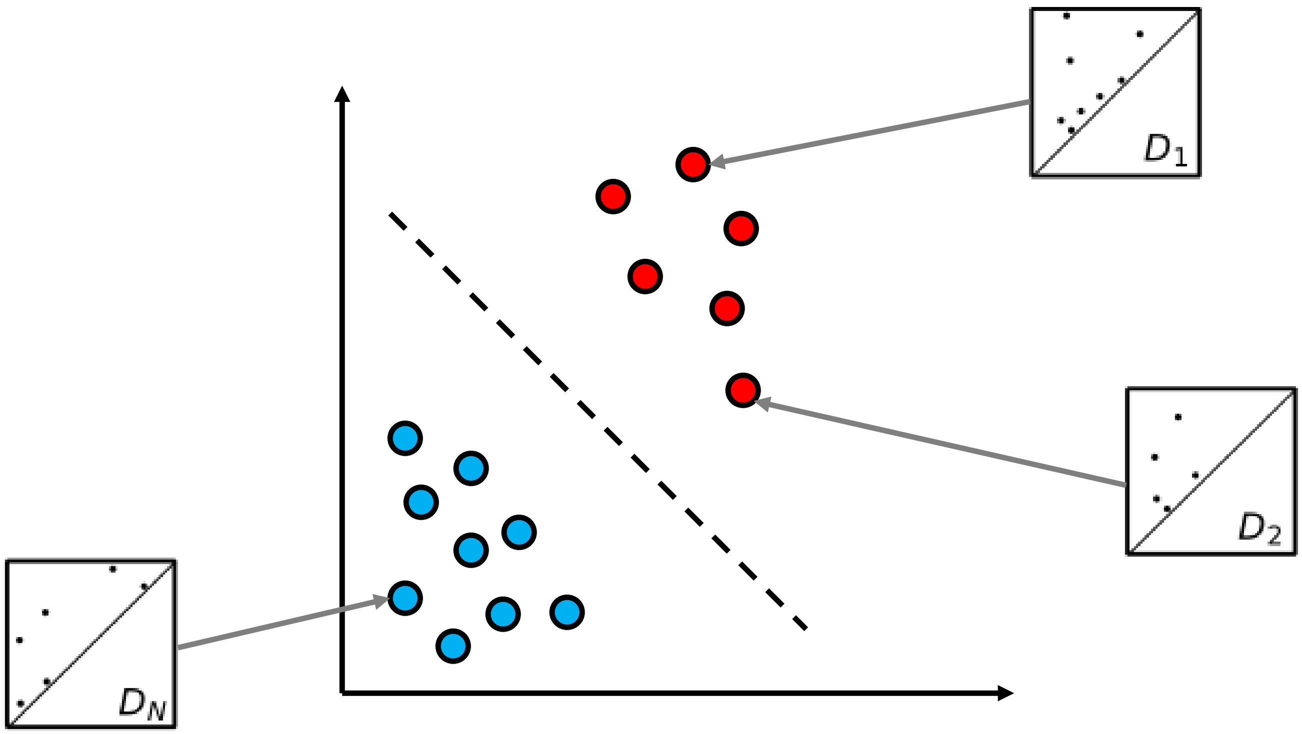
Topological Machine Learning



Topological Machine Learning



Topological Machine Learning



Kernels

Definition

Given a set $\mathcal{X}$, a function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if there exists a Hilbert space $\mathcal{H}$, called feature space, and a map $\Phi\colon \mathcal{X} \to \mathcal{H}$, called feature map, such that $k(x, y) = \langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}}$ for all $x, y \in \mathcal{X}$.

(Equivalently: k is a kernel if it is symmetric and positive definite)

Kernels

Definition

Given a set $\mathcal{X}$, a function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if there exists a Hilbert space $\mathcal{H}$, called feature space, and a map $\Phi\colon \mathcal{X} \to \mathcal{H}$, called feature map, such that $k(x, y) = \langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}}$ for all $x, y \in \mathcal{X}$.

(Equivalently: k is a kernel if it is symmetric and positive definite)

Kernel induced pseudo-metric on $\mathcal{X}$:

$$d_k(x, y) = \sqrt{k(x, x) + k(y, y) - 2k(x, y)} = \|\Phi(x) - \Phi(y)\|_{\mathcal{H}}$$

Kernels

Definition

Given a set $\mathcal{X}$, a function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if there exists a Hilbert space $\mathcal{H}$, called feature space, and a map $\Phi\colon \mathcal{X} \to \mathcal{H}$, called feature map, such that $k(x, y) = \langle \Phi(x), \Phi(y) \rangle_{\mathcal{H}}$ for all $x, y \in \mathcal{X}$.

(Equivalently: k is a kernel if it is symmetric and positive definite)

Kernel induced pseudo-metric on $\mathcal{X}$:

$$d_k(x, y) = \sqrt{k(x, x) + k(y, y) - 2k(x, y)} = \|\Phi(x) - \Phi(y)\|_{\mathcal{H}}$$

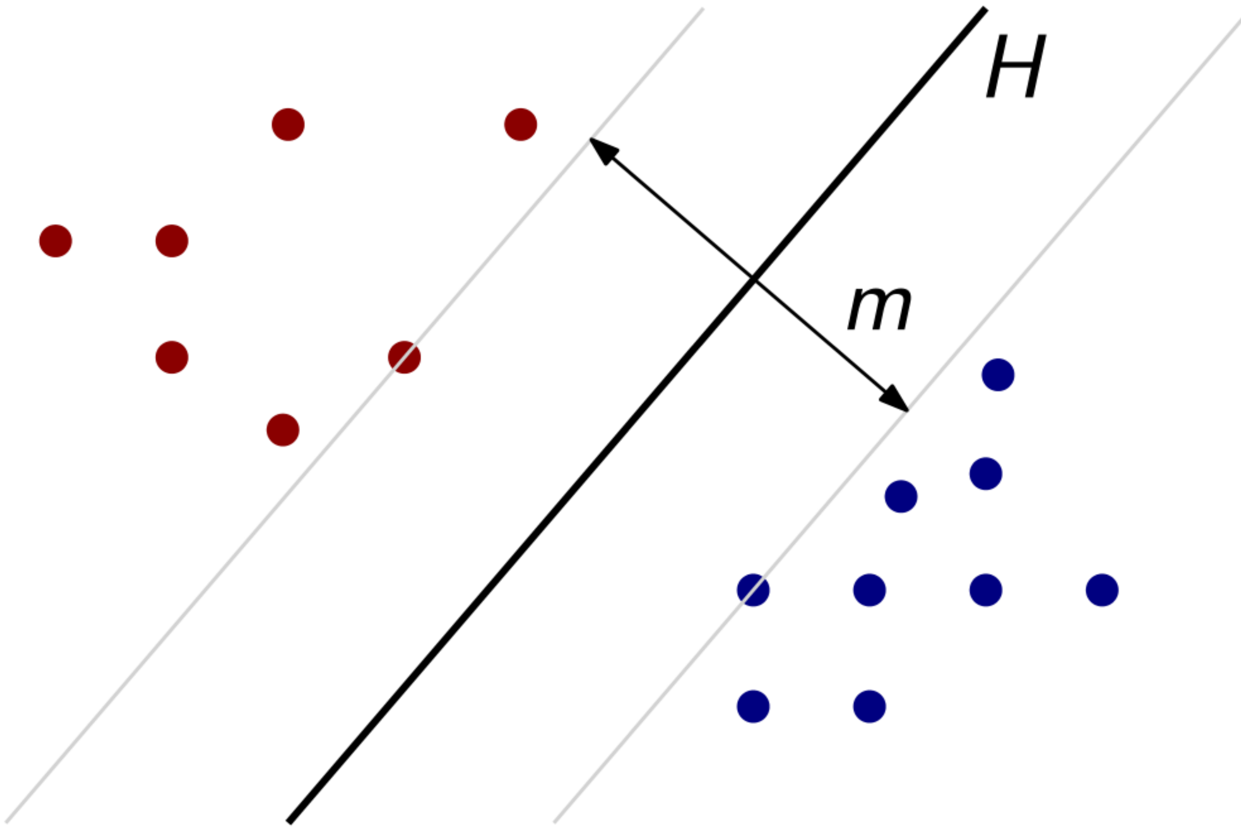
Definition

We call k stable w.r.t. a metric d on $\mathcal{X}$ if there is a constant $C > 0$ such that

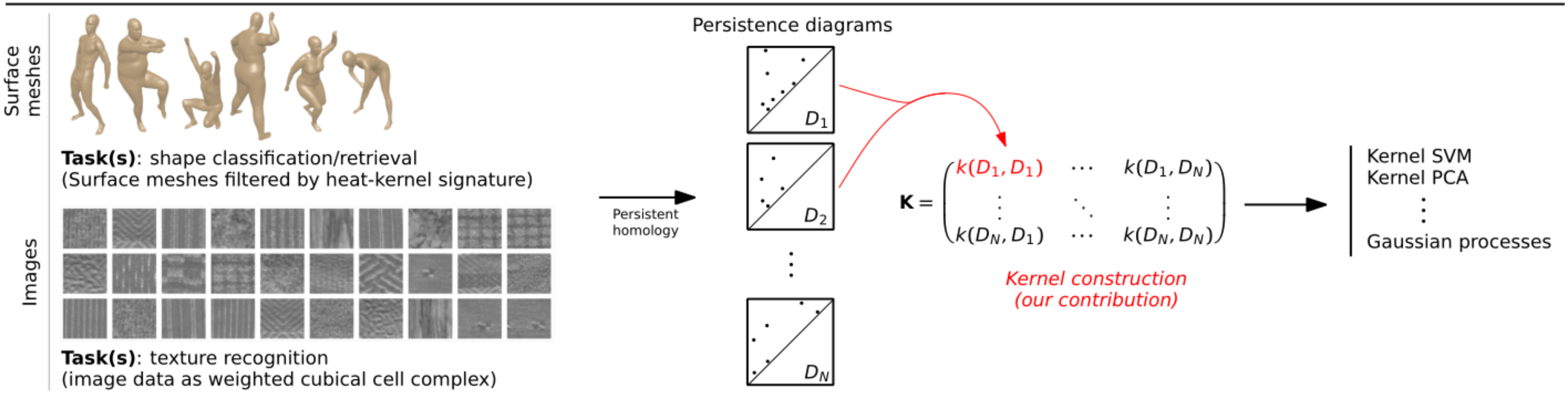
$$d_k(x, y) \leq C\, d(x, y)$$

(Equivalently: k is stable if Φ is Lipschitz-continuous)

Why Stability?



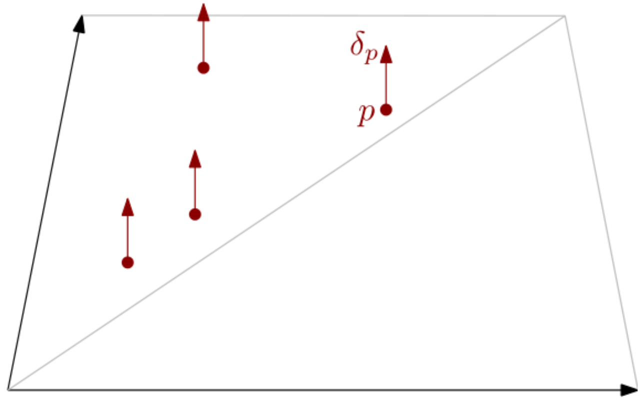
Basic idea


Surface meshes
Task(s): shape classification/retrieval
(Surface meshes filtered by heat-kernel signature)
Images
Task(s): texture recognition
(image data as weighted cubical cell complex)
Persistent homology
Persistence diagrams
D_1
D_2
D_N
$\mathbf{K} = \begin{pmatrix} k(D_1, D_1) & \cdots & k(D_1, D_N) \\ \vdots & \ddots & \vdots \\ k(D_N, D_1) & \cdots & k(D_N, D_N) \end{pmatrix}$
Kernel construction
(our contribution)
Kernel SVM
Kernel PCA
Gaussian processes

An Unstable Kernel

Persistence diagram ⇔ multi-set of points in $\Omega = \{(x_1, x_2) \in \mathbb{R}^2 : x_2 \geq x_1\}$

- Feature map: $\Phi : \mathcal{D} \to H^{-2}(\Omega) : \; D \mapsto \sum_{p \in D} \delta_p$



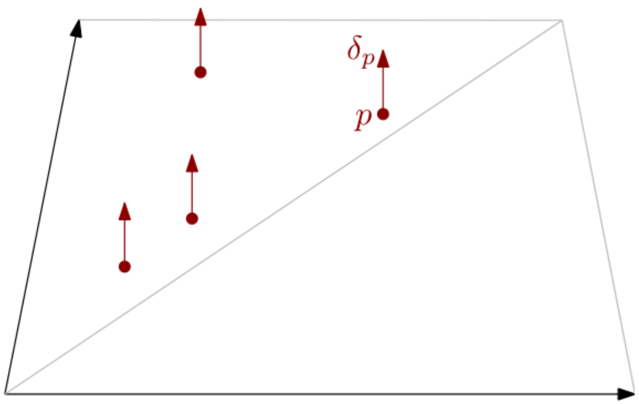
An Unstable Kernel

Persistence diagram ⇔ multi-set of points in $\Omega = \{(x_1, x_2) \in \mathbb{R}^2 : x_2 \geq x_1\}$

- Feature map: $\Phi : \mathcal{D} \to H^{-2}(\Omega) :\ D \mapsto \sum_{p \in D} \delta_p$

- Corresponding kernel is not Wasserstein stable

$$d_{W,p}(F,G) = \left(\inf_{\gamma} \sum_{x \in F} \|x - \gamma(x)\|_{\infty}^{p} \right)^{\frac{1}{p}}$$

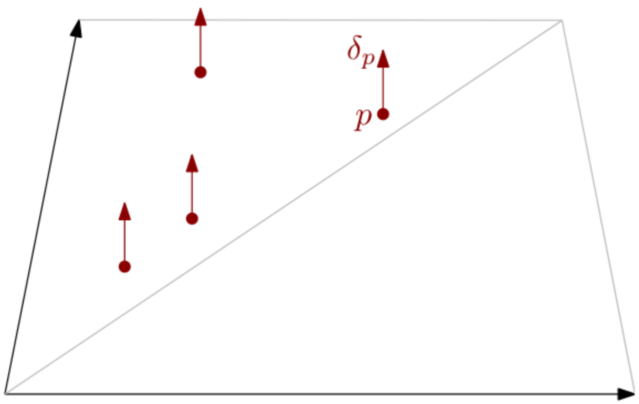


An Unstable Kernel

Persistence diagram ⇔ multi-set of points in $\Omega = \{(x_1, x_2) \in \mathbb{R}^2 : x_2 \geq x_1\}$

- Feature map: $\Phi : \mathcal{D} \to H^{-2}(\Omega) : D \mapsto \sum_{p \in D} \delta_p$
- Corresponding kernel is not Wasserstein stable

$$d_{W,p}(F,G) = \left(\inf_{\gamma} \sum_{x \in F} \|x - \gamma(x)\|_{\infty}^{p} \right)^{\frac{1}{p}}$$



- Use $\sum_{p \in D} \delta_p$ as initial condition for diffusion process
- Dirichlet boundary condition → stability

A Stable Multi-Scale Kernel

Definition

For a given persistence diagram D, we consider the solution $u\colon \Omega \times \mathbb{R}_{\geq 0} \to \mathbb{R}, (x,t) \mapsto u(x,t)$ of the partial differential equation

$$\Delta_x u = \partial_t u \qquad \text{in } \Omega \times \mathbb{R}_{>0}, \tag{1}$$

$$u = 0 \qquad \text{on } \partial\Omega \times \mathbb{R}_{\geq 0}, \tag{2}$$

$$u = \sum_{p \in D} \delta_p \qquad \text{on } \Omega \times \{0\}. \tag{3}$$

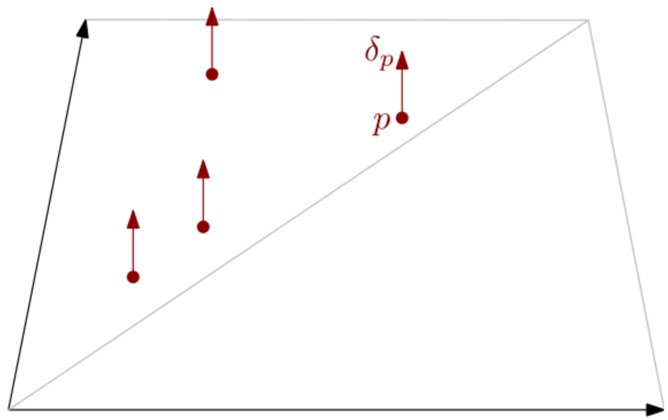
The feature map $\Phi_\sigma\colon \mathcal{D} \to L_2(\Omega)$ at scale $\sigma > 0$ is defined as

$$\Phi_\sigma(D) = u\big|_{t=\sigma} \tag{4}$$

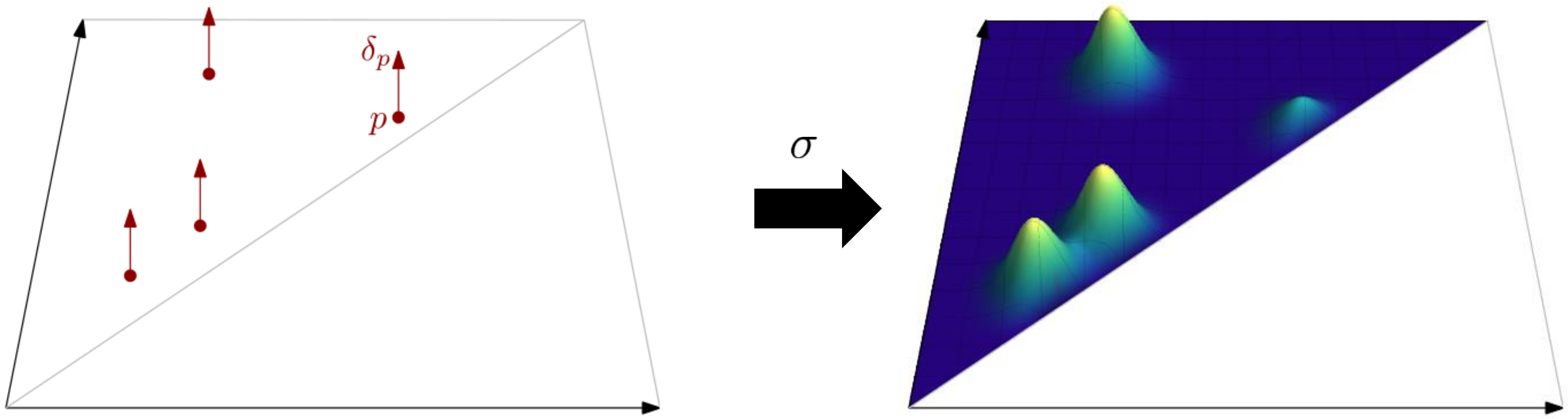
This map yields the persistence scale space kernel k_σ on $\mathcal{D}$ as

$$k_\sigma(F,G) = \langle \Phi_\sigma(F), \Phi_\sigma(G) \rangle_{L_2(\Omega)}. \tag{5}$$

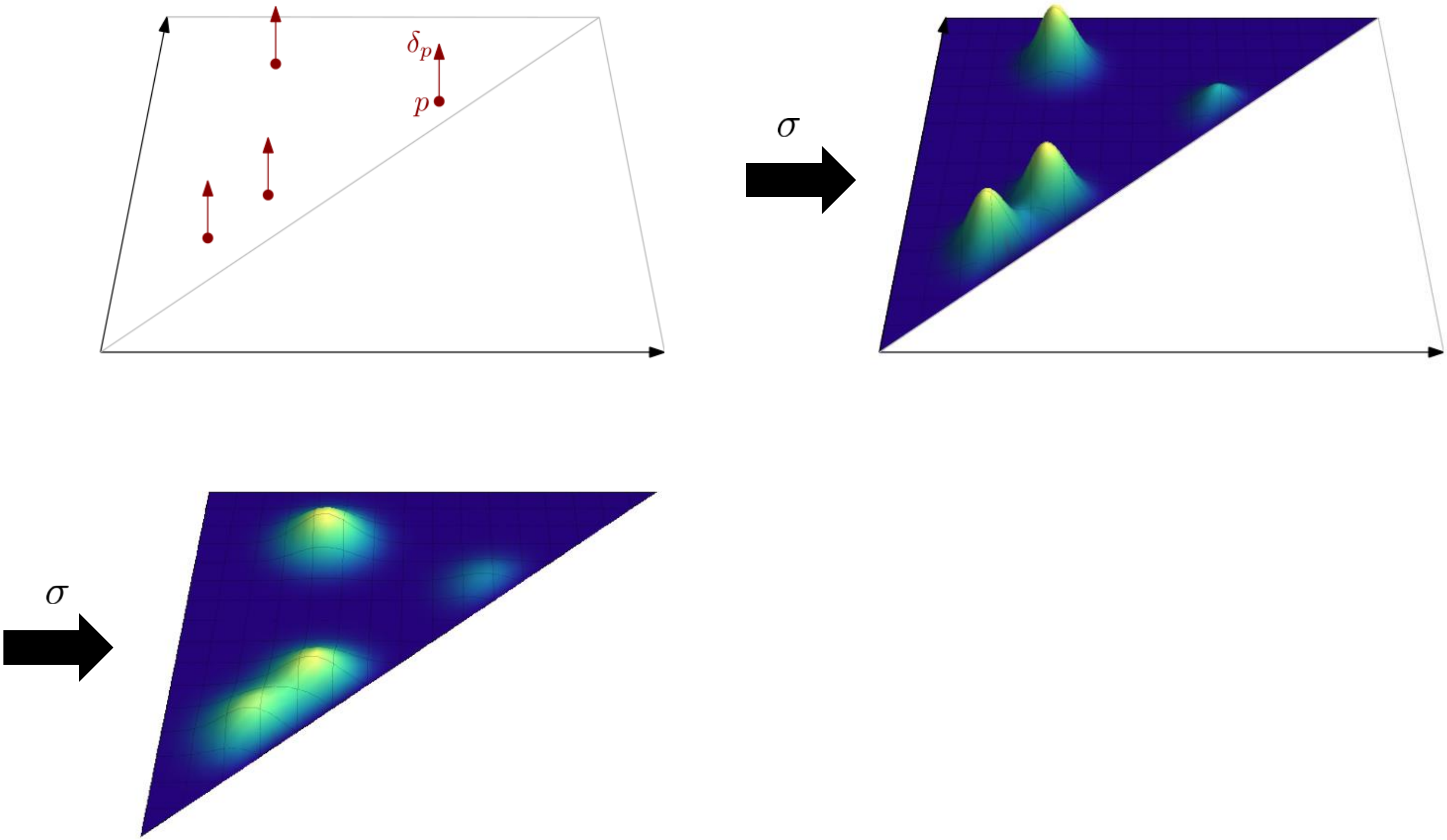
Example



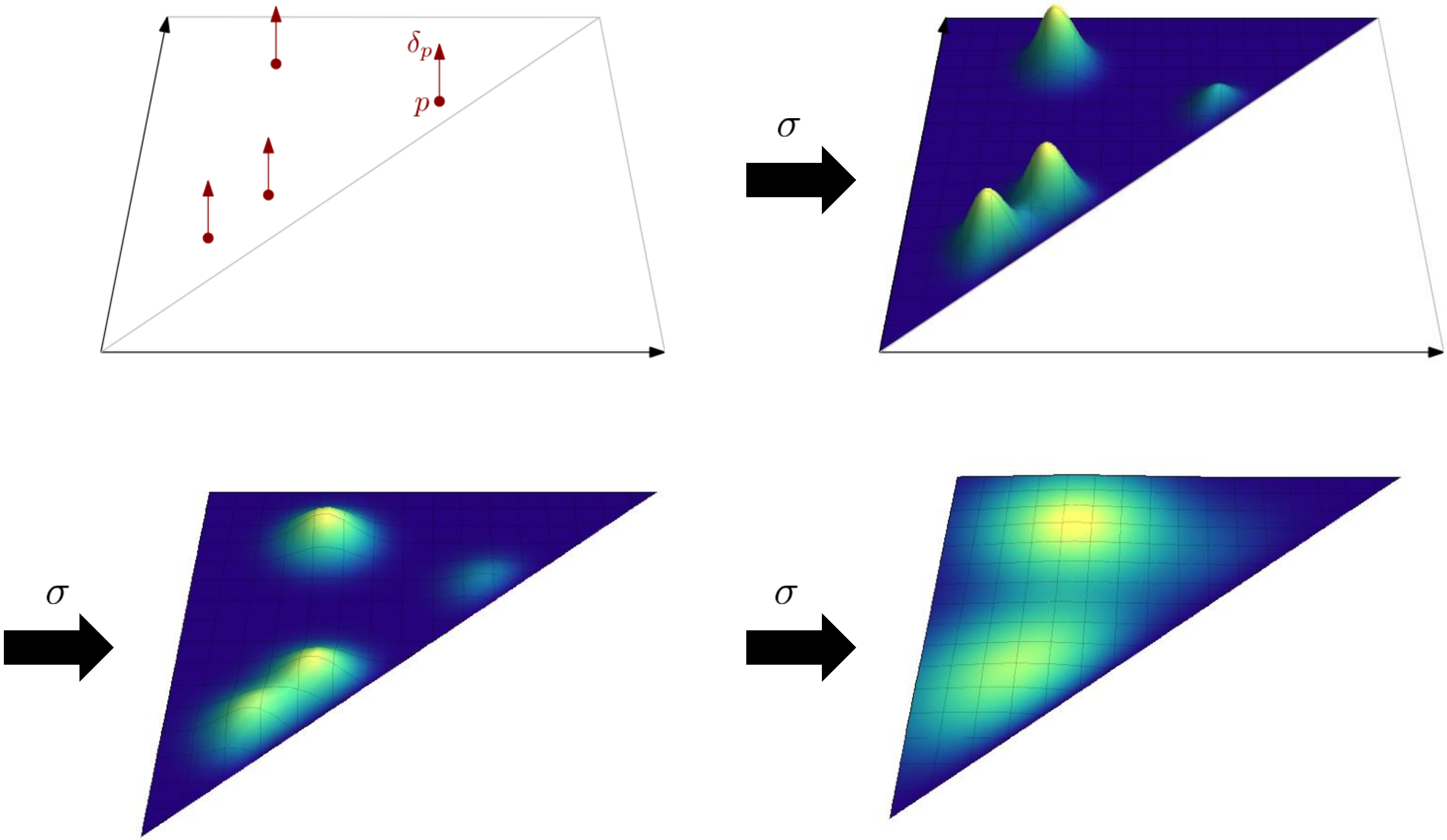
Example



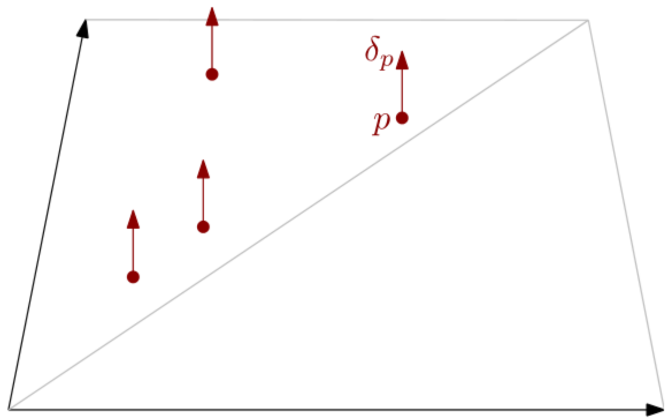
Example



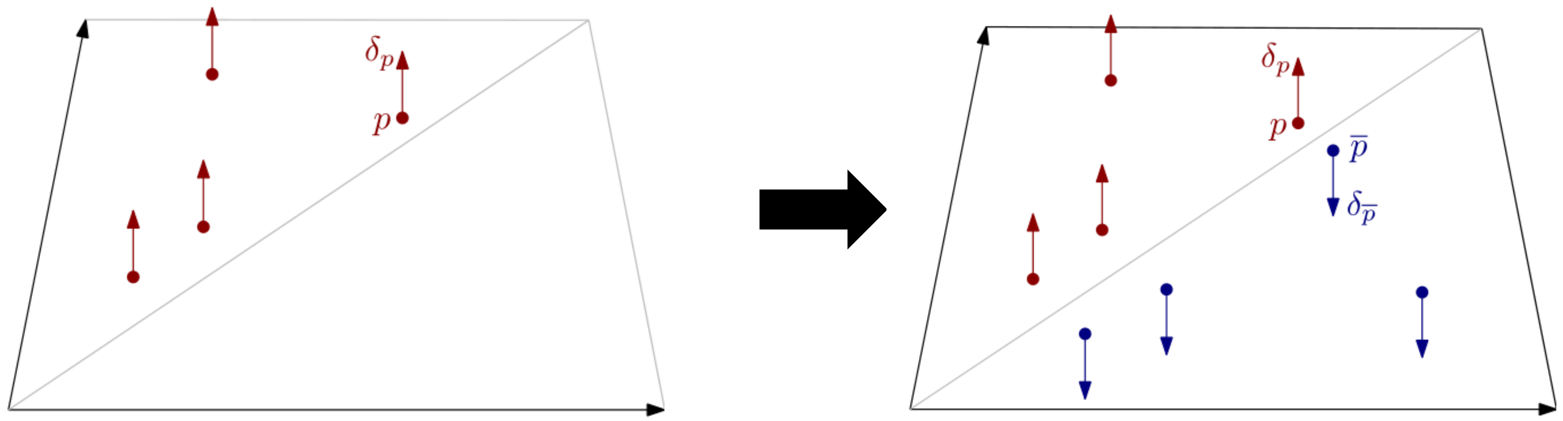
Example



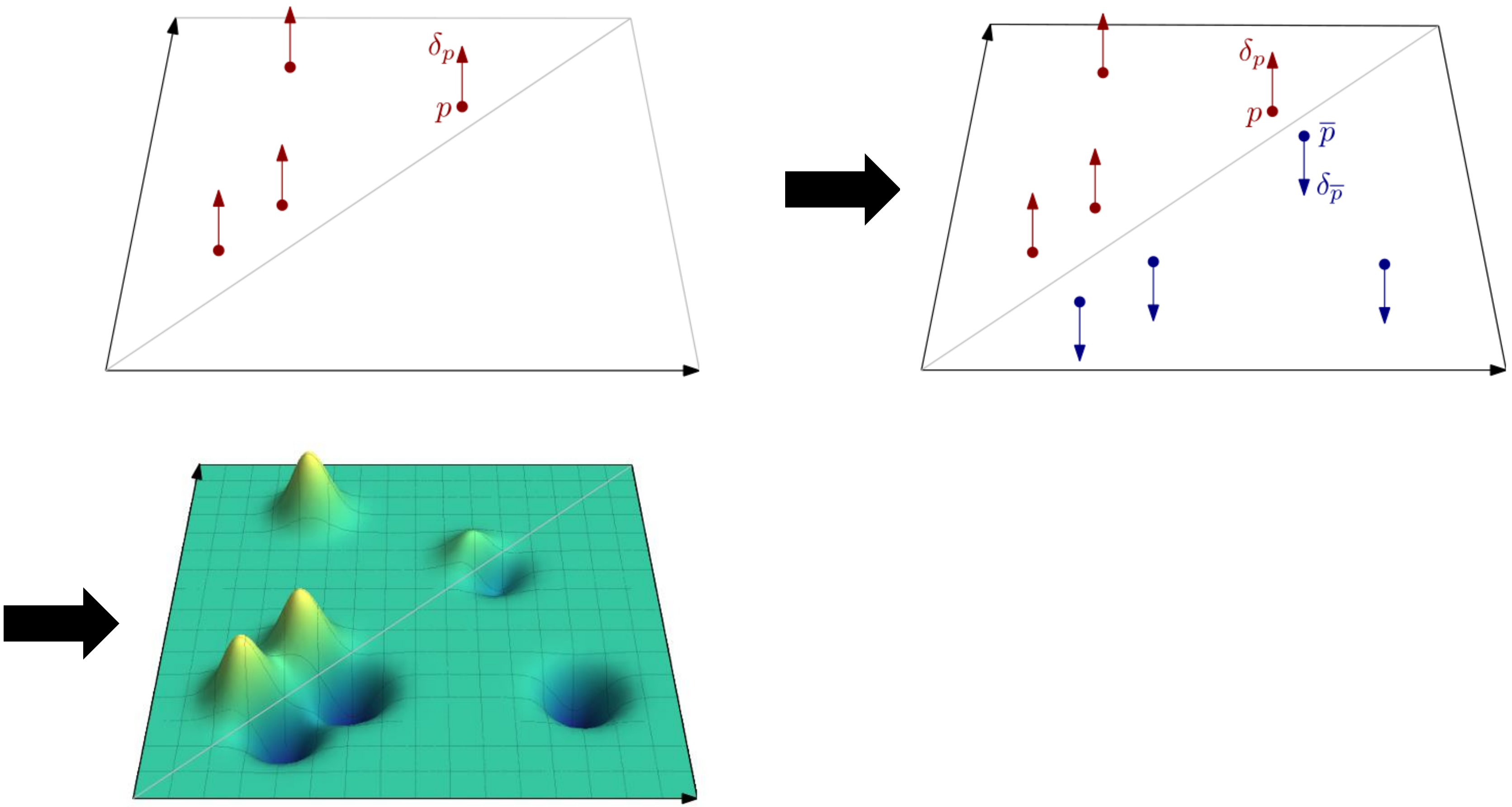
Analytic Solution



Analytic Solution

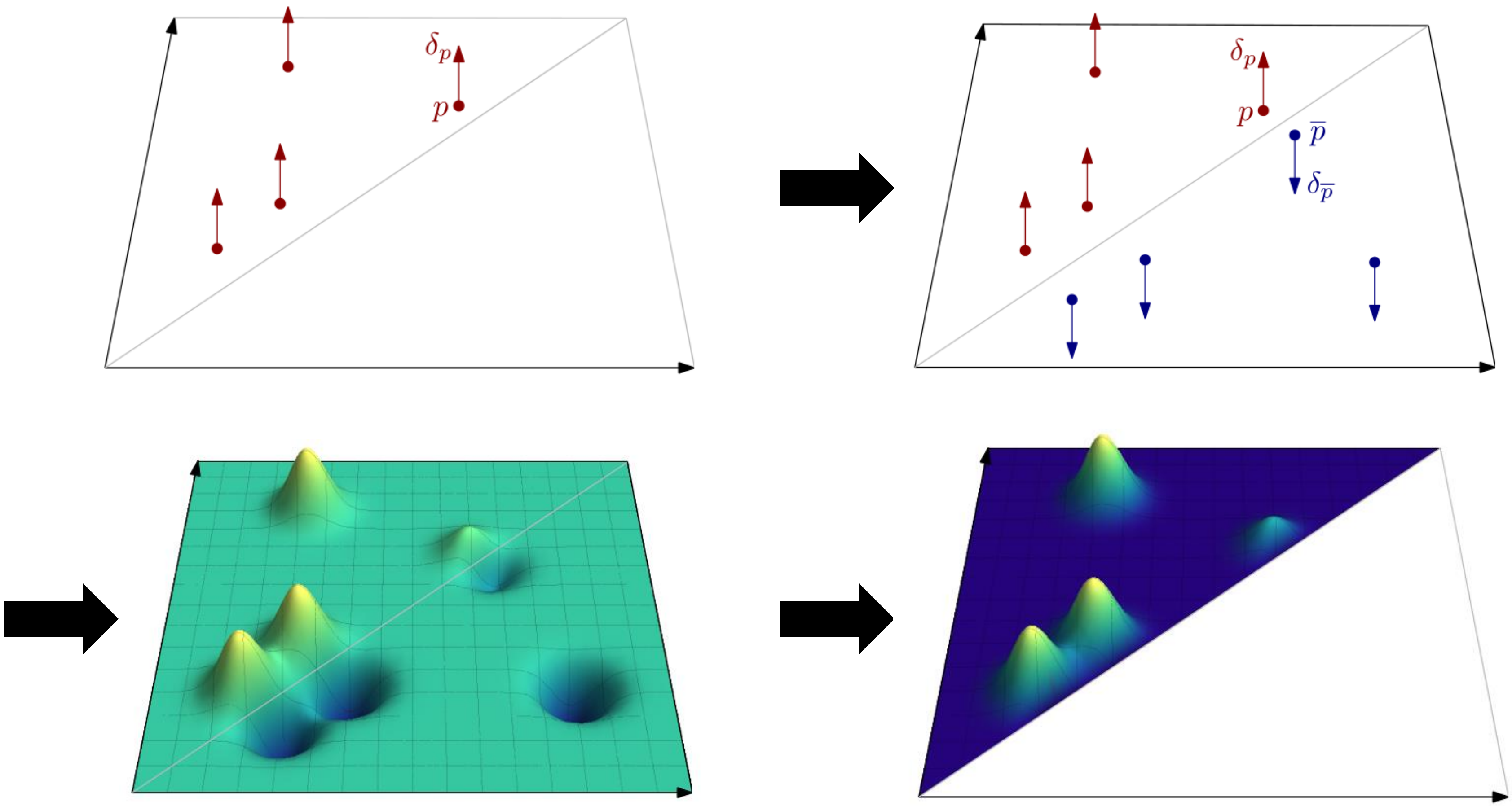


Analytic Solution



$$u(x,t) = \frac{1}{4\pi t} \sum_{p \in D} e^{-\frac{\|x-p\|^2}{4t}} - e^{-\frac{\|x-\overline{p}\|^2}{4t}}$$

Analytic Solution



$$u(x,t) = \frac{1}{4\pi t}\sum_{p\in D} e^{-\frac{\|x-p\|^2}{4t}} - e^{-\frac{\|x-\overline{p}\|^2}{4t}}$$

Kernel Evaluation

$$\Phi_\sigma(D)\colon \Omega \to \mathbb{R}, \quad x \mapsto \frac{1}{4\pi\sigma} \sum_{p \in D} e^{-\frac{\|x-p\|^2}{4\sigma}} - e^{-\frac{\|x-\overline{p}\|^2}{4\sigma}} \tag{7}$$

$$\begin{aligned}
k_\sigma(F,G) &= \int_\Omega \Phi_\sigma(F)\,\Phi_\sigma(G)\,dx \\
&= \frac{1}{2}\int_{\mathbb{R}^2} \Phi_\sigma(F)\,\Phi_\sigma(G)\,dx \\
&= \frac{1}{2}\frac{1}{(4\pi\sigma)^2} \sum_{\substack{p \in F \\ q \in G}} \int_{\mathbb{R}^2} \left(e^{-\frac{\|x-p\|^2}{4\sigma}} - e^{-\frac{\|x-\overline{p}\|^2}{4\sigma}} \right) \cdot \left(e^{-\frac{\|x-q\|^2}{4\sigma}} - e^{-\frac{\|x-\overline{q}\|^2}{4\sigma}} \right) \, dx \\
&= \frac{1}{8\pi\sigma} \sum_{\substack{p \in F \\ q \in G}} e^{-\frac{\|p-q\|^2}{8\sigma}} - e^{-\frac{\|p-\overline{q}\|^2}{8\sigma}}.
\end{aligned}$$

1-Wasserstein Stability

Theorem

The kernel k_σ is 1-Wasserstein stable.

Proof:

$$
\begin{aligned}
&\|\Phi_\sigma(F) - \Phi_\sigma(G)\|_{L_2(\Omega)} \\
&\leq \left\| \sum_{u \in F} (N_u - N_{\overline{u}}) - (N_{\gamma(u)} - N_{\overline{\gamma(u)}}) \right\|_{L_2(\mathbb{R}^2)} \\
&\leq 2 \sum_{u \in F} \|N_u - N_{\gamma(u)}\|_{L_2(\mathbb{R}^2)} \\
&\leq \frac{1}{\sigma\sqrt{8\pi}} \sum_{u \in F} \|u - \gamma(u)\|_2 \\
&\leq \frac{1}{2\sigma\sqrt{\pi}} \sum_{u \in F} \|u - \gamma(u)\|_\infty \\
&= \frac{1}{2\sigma\sqrt{\pi}} d_{W,1}(F, G)
\end{aligned}
$$

p-Wasserstein Stability

- A kernel k is **additive** if $k(E \cup F, G) = k(E, G) + k(F, G)$
- A kernel k is **non-trivial** if there exists F, G s.t. $k(F, G) \neq 0$

Theorem

A non-trivial additive kernel k on $\mathcal{D}$ is not stable w.r.t. $d_{W,p}$ for any $1 < p \leq \infty$.

Proof: Compare rates of growth:

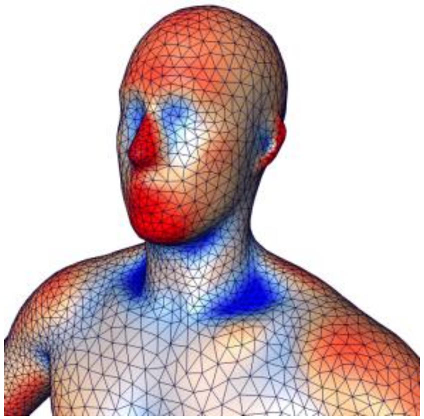
$$d_{k_\sigma}\left(\bigcup_{i=1}^{n} F, \emptyset\right) = n\sqrt{k(F,F)}$$

$$d_{W,p}\left(\bigcup_{i=1}^{n} F, \emptyset\right) = d_{W,p}(F, \emptyset) \cdot \begin{cases} \sqrt[p]{n} & \text{if } p < \infty, \\ 1 & \text{if } p = \infty \end{cases}$$

Numerical Experiments – Shapes

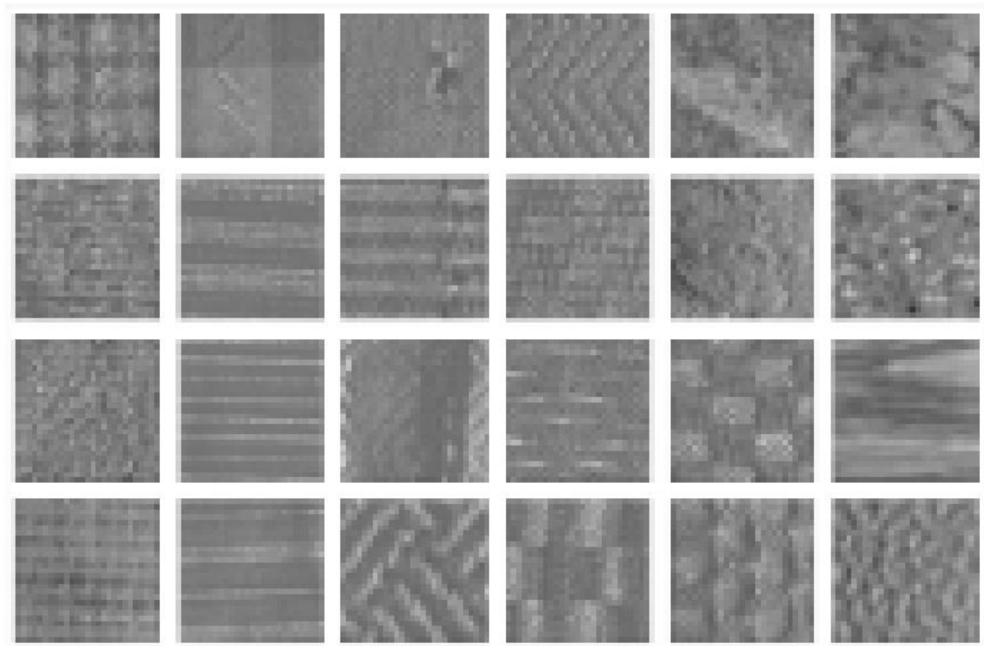
SHREC 2014 real: 400 meshes from 40 humans in 10 different poses
Filtration: Heat Kernel Signature

HKS t_i	k^L	k_σ	Δ
t_1	45.2 ± 5.8	48.8 ± 4.9	+3.5
t_2	31.0 ± 4.8	46.5 ± 5.3	+15.5
t_3	30.0 ± 7.3	37.8 ± 8.2	+7.8
t_4	41.2 ± 2.2	50.2 ± 5.4	+9.0
t_5	46.2 ± 5.8	62.5 ± 2.0	+16.2
t_6	33.2 ± 4.1	58.0 ± 4.0	+24.7
t_7	31.0 ± 5.7	$\mathbf{62.7} \pm 4.6$	+31.7
t_8	$\mathbf{51.7} \pm 2.9$	57.5 ± 4.2	+5.8
t_9	36.0 ± 5.3	41.2 ± 4.9	+5.2
t_{10}	2.8 ± 0.6	27.8 ± 5.8	+25.0

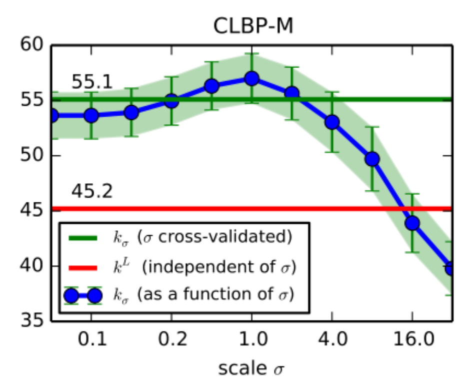


Numerical Experiments – Textures

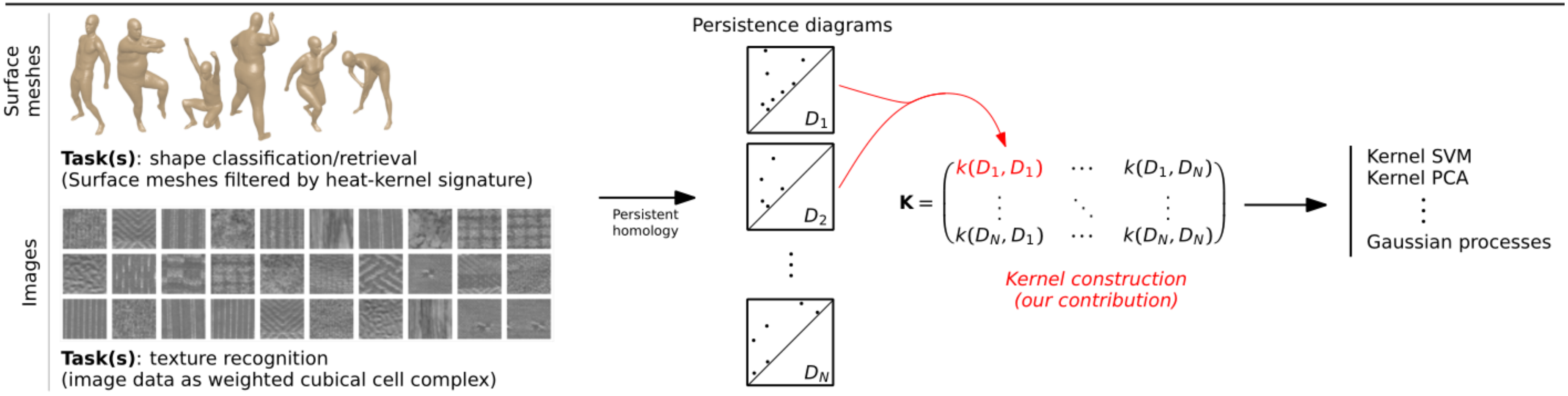
Outex-TC-00000: 240 textures (24 classes, 10 samples each)
Filtration: Complete-Local-Binary-Pattern filter



CLBP Operator	k^L	k_σ	Δ
CLBP-S	58.0 ± 2.3	$\mathbf{69.2 \pm 2.7}$	+11.2
CLBP-M	45.2 ± 2.5	$\mathbf{55.1 \pm 2.5}$	+9.9
CLBP-S (SVM-χ^2)	76.1 ± 2.2		
CLBP-M (SVM-χ^2)	76.7 ± 1.8		



Part 4: Conclusion



$$k_\sigma(F, G) = \frac{1}{8\pi\sigma} \sum_{\substack{p \in F \\ q \in G}} e^{-\frac{\|p-q\|^2}{8\sigma}} - e^{-\frac{\|p-\bar{q}\|^2}{8\sigma}}.$$

- Exact evaluation in O(n^2)
- Approximate evaluation in O(n)

Summary

- Part 1: Phat – fast matrix reduction algorithms: phat@github
- Part 2: Dipha – distributed computation: dipha@github
- Part 3: Discrete Morse theory – large image data
- Part 4: A kernel for topological machine learning