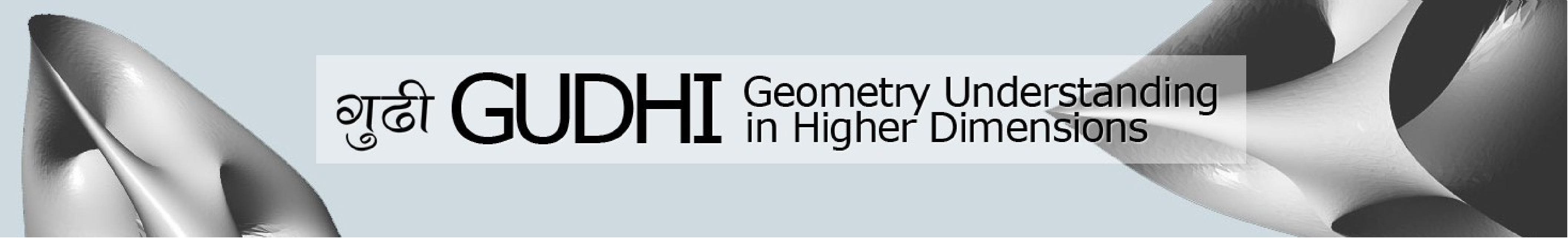


GUDHI library

Jean-Daniel Boissonnat, Claire Brécheteau, Mickaël Buchet, Mathieu Carrière, Frédéric Chazal, Paweł Dłotko, Marc Glisse, François Godi, Clément Jamin, Siargey Kachanovich, Miroslav Kramar, Bertrand Michel, Clément Maria, Steve Oudot, Owen Rouillé, Vincent Rouvreau and David Salinas

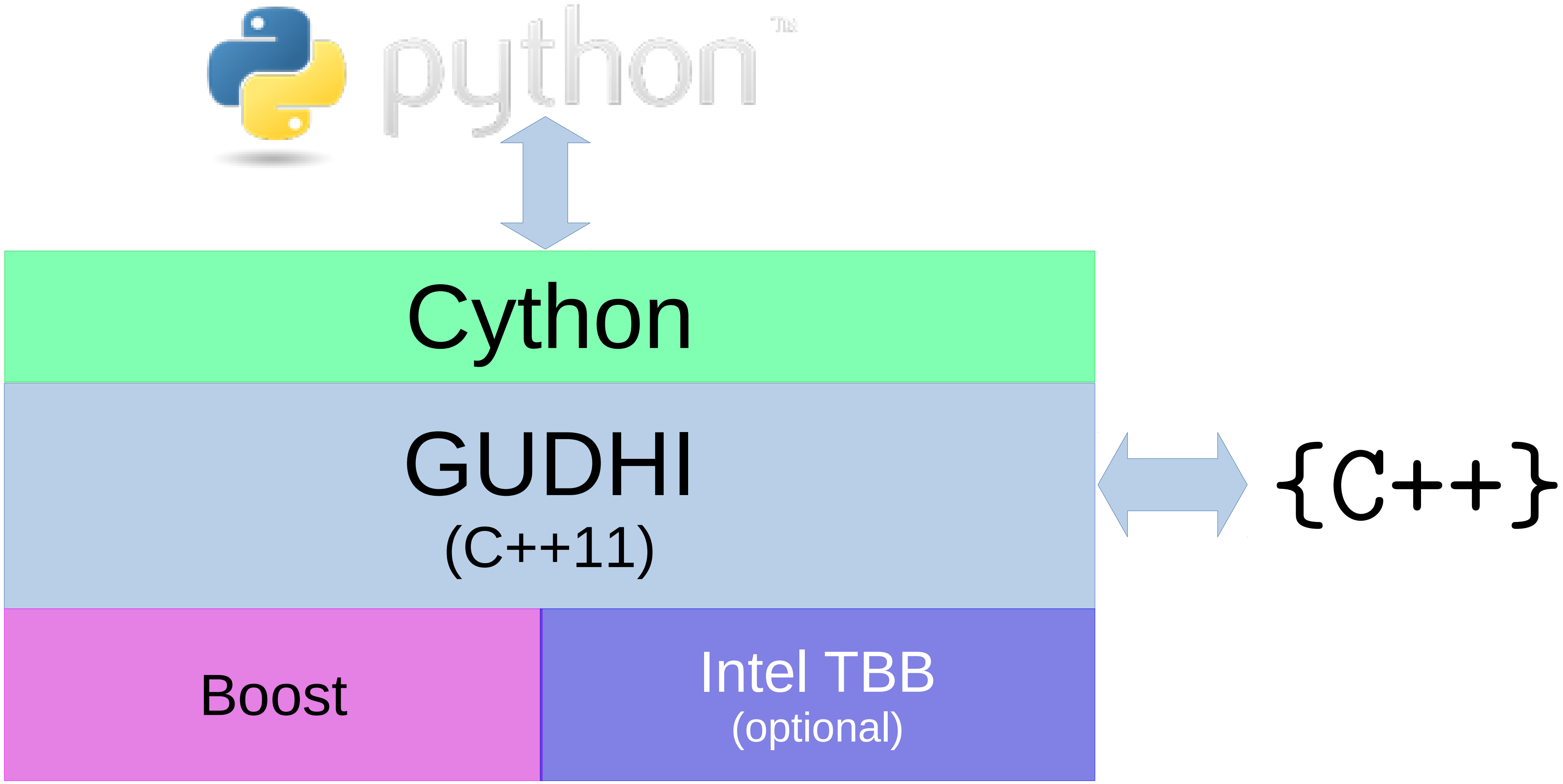
DataShape, Inria Saclay and Sophia-Antipolis

गुढी GUDHI Geometry Understanding in Higher Dimensions

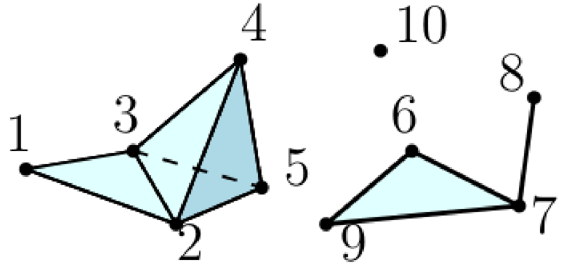
GUDHI is a five years project supported by a Grant of the European Research Council and hosted by INRIA

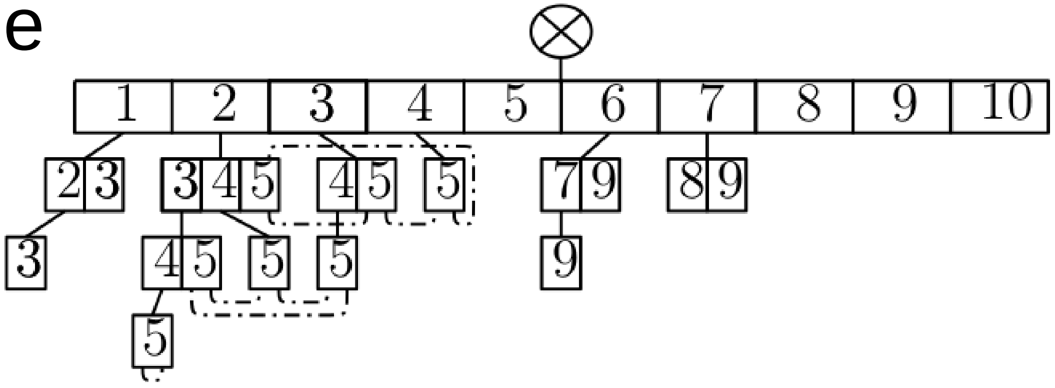
- develop and understand geometrical data structures
- develop associated statistical, geometric and topological functions

Installation



Data structure – Simplex tree

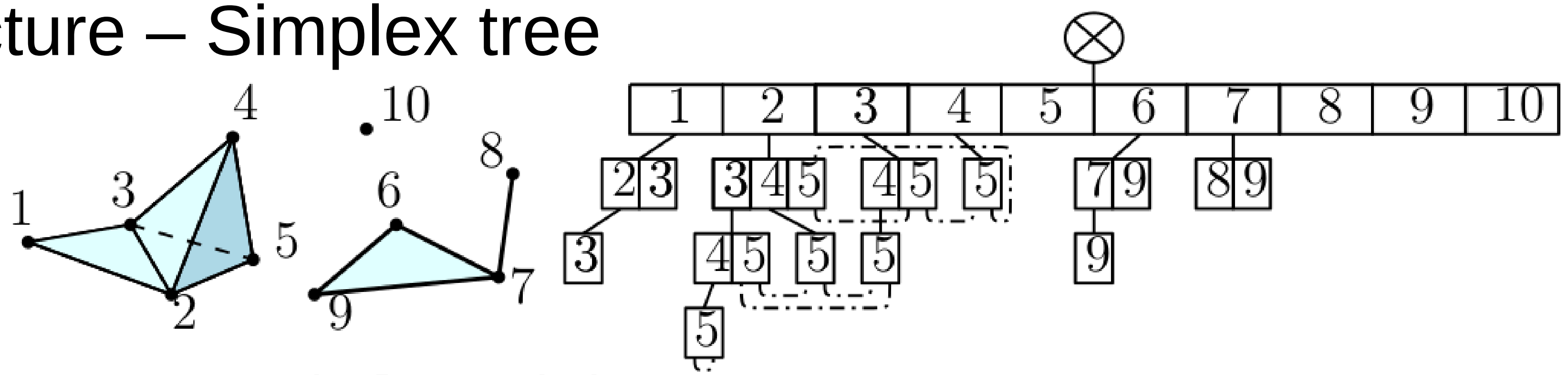




```
import gudhi
st = gudhi.SimplexTree()

st.insert([1, 2, 3], 0.4)
st.insert([2, 3, 4, 5], 1.)
st.insert([6, 7, 9], 0.8)
st.insert([7, 8], 0.6)
st.insert([10], 2.2)
print(st.num_vertices())
print(st.num_simplices())
print(st.dimension())
```

Jean-Daniel Boissonnat and Clément Maria. **The simplex tree:** an efficient data structure for general simplicial complexes. Algorithmica, pages 1–22, 2014

Data structure – Simplex tree

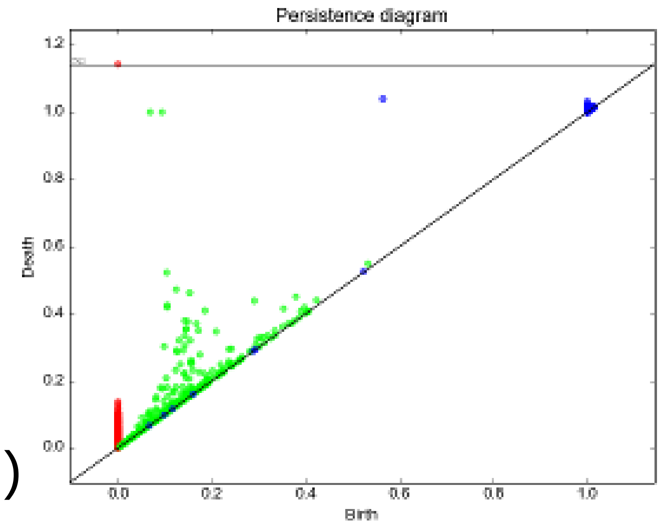


```
for skel in st.get_skeleton(2):
    print(skel)
for coface in st.get_cofaces(simplex = [1,2],
                             codimension = 1):
    print(coface)
for star in st.get_star(simplex = [1,2]):
    print(star)
st.remove_maximal_simplex([1, 2, 3])
st.assign_filtration([1, 2], st.filtration([1, 2]))
```

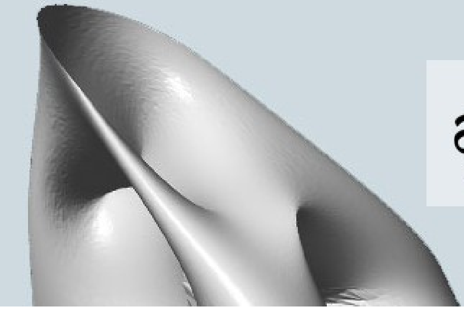
Maximum number of simplices to compute persistence is about 4 billions of simplices (std::uint32_t – can be up/downgraded)

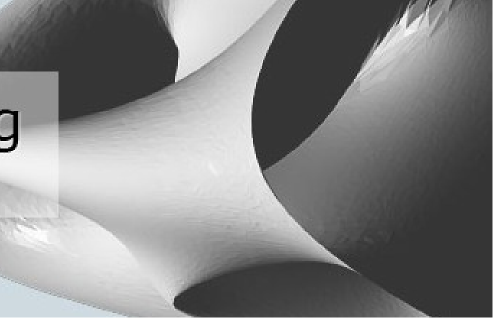
Toolbox – Persistence cohomology

```
st.initialize_filtration()
diag = st.persistence()
print(st.betti_numbers())

# barcode
plt = gudhi.plot_persistence_barcode(diag)
plt.show()

# persistence diagram
plt = gudhi.plot_persistence_diagram(diag)
plt.show()
```



Jean-Daniel Boissonnat, Tamal K. Dey, and Clément Maria. The compressed annotation matrix: An efficient data structure for computing persistent cohomology. In ESA, pages 695–706, 2013.



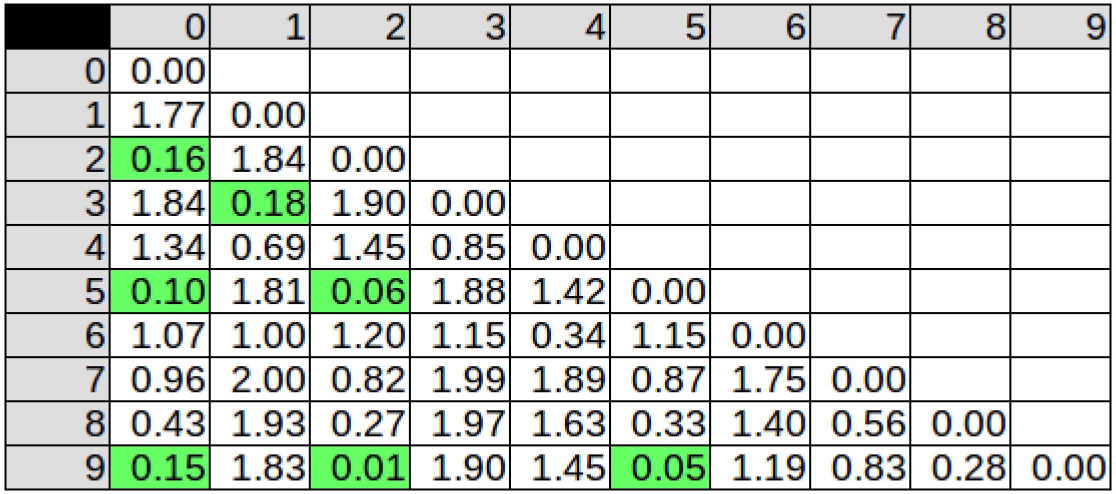


Geometric filtered complex creation – Rips complex

	0	1	2	3	4	5	6	7	8	9
0	0.00									
1	1.77	0.00								
2	0.16	1.84	0.00							
3	1.84	0.18	1.90	0.00						
4	1.34	0.69	1.45	0.85	0.00					
5	0.10	1.81	0.06	1.88	1.42	0.00				
6	1.07	1.00	1.20	1.15	0.34	1.15	0.00			
7	0.96	2.00	0.82	1.99	1.89	0.87	1.75	0.00		
8	0.43	1.93	0.27	1.97	1.63	0.33	1.40	0.56	0.00	
9	0.15	1.83	0.01	1.90	1.45	0.05	1.19	0.83	0.28	0.00

```
import gudhi
rips = gudhi.RipsComplex(distance_matrix = distance_matrix,
                         threshold = 0.2)
# could be also from a csv file
```

Geometric filtered complex creation – Rips complex



	0	1	2	3	4	5	6	7	8	9
0	0.00									
1	1.77	0.00								
2	0.16	1.84	0.00							
3	1.84	0.18	1.90	0.00						
4	1.34	0.69	1.45	0.85	0.00					
5	0.10	1.81	0.06	1.88	1.42	0.00				
6	1.07	1.00	1.20	1.15	0.34	1.15	0.00			
7	0.96	2.00	0.82	1.99	1.89	0.87	1.75	0.00		
8	0.43	1.93	0.27	1.97	1.63	0.33	1.40	0.56	0.00	
9	0.15	1.83	0.01	1.90	1.45	0.05	1.19	0.83	0.28	0.00

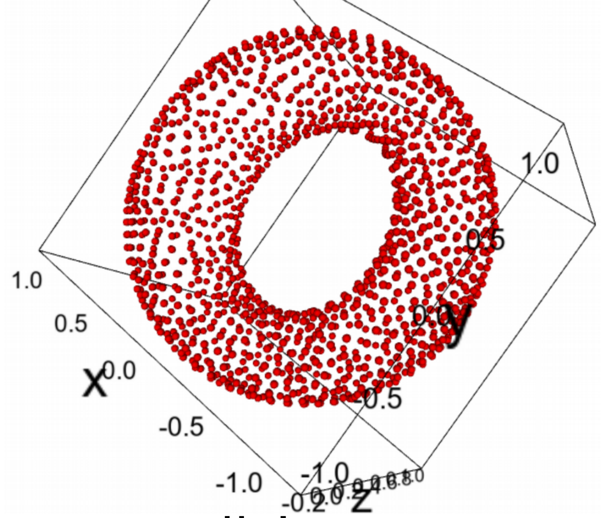
```
import gudhi
rips = gudhi.RipsComplex(distance_matrix = distance_matrix,
                         threshold = 0.2)

# could be also from a csv file
rips = gudhi.RipsComplex(csv_file = ‘distance_matrix.csv’,
                         threshold = 0.2)
```

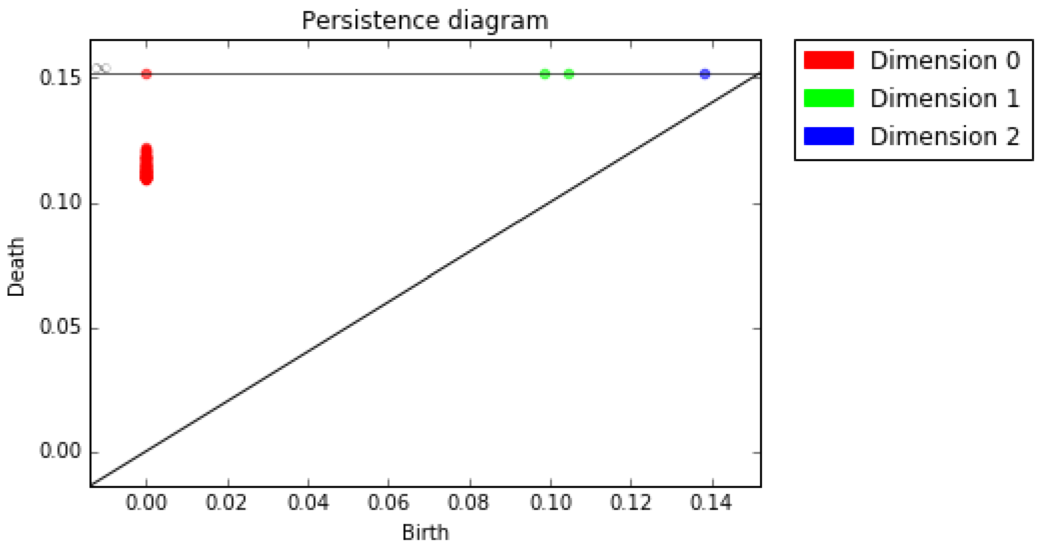
Geometric filtered complex creation – Rips complex

	0	1	2	3	4	5	6	7	8	9
0	0.00									
1	1.77	0.00								
2	0.16	1.84	0.00							
3	1.84	0.18	1.90	0.00						
4	1.34	0.69	1.45	0.85	0.00					
5	0.10	1.81	0.06	1.88	1.42	0.00				
6	1.07	1.00	1.20	1.15	0.34	1.15	0.00			
7	0.96	2.00	0.82	1.99	1.89	0.87	1.75	0.00		
8	0.43	1.93	0.27	1.97	1.63	0.33	1.40	0.56	0.00	
9	0.15	1.83	0.01	1.90	1.45	0.05	1.19	0.83	0.28	0.00

```
import gudhi
rips = gudhi.RipsComplex(distance_matrix = distance_matrix,
                         threshold = 0.2)

# expands the graph until dimension 2
st = rips.create_simplex_tree(max_dimension = 2)
# insertion of [0, 2, 5] and [2, 5, 9]
```

Geometric filtered complex creation – Rips complex

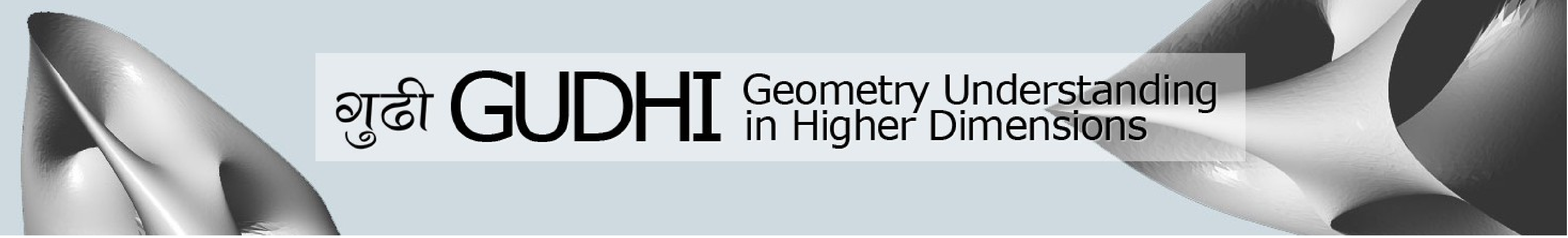




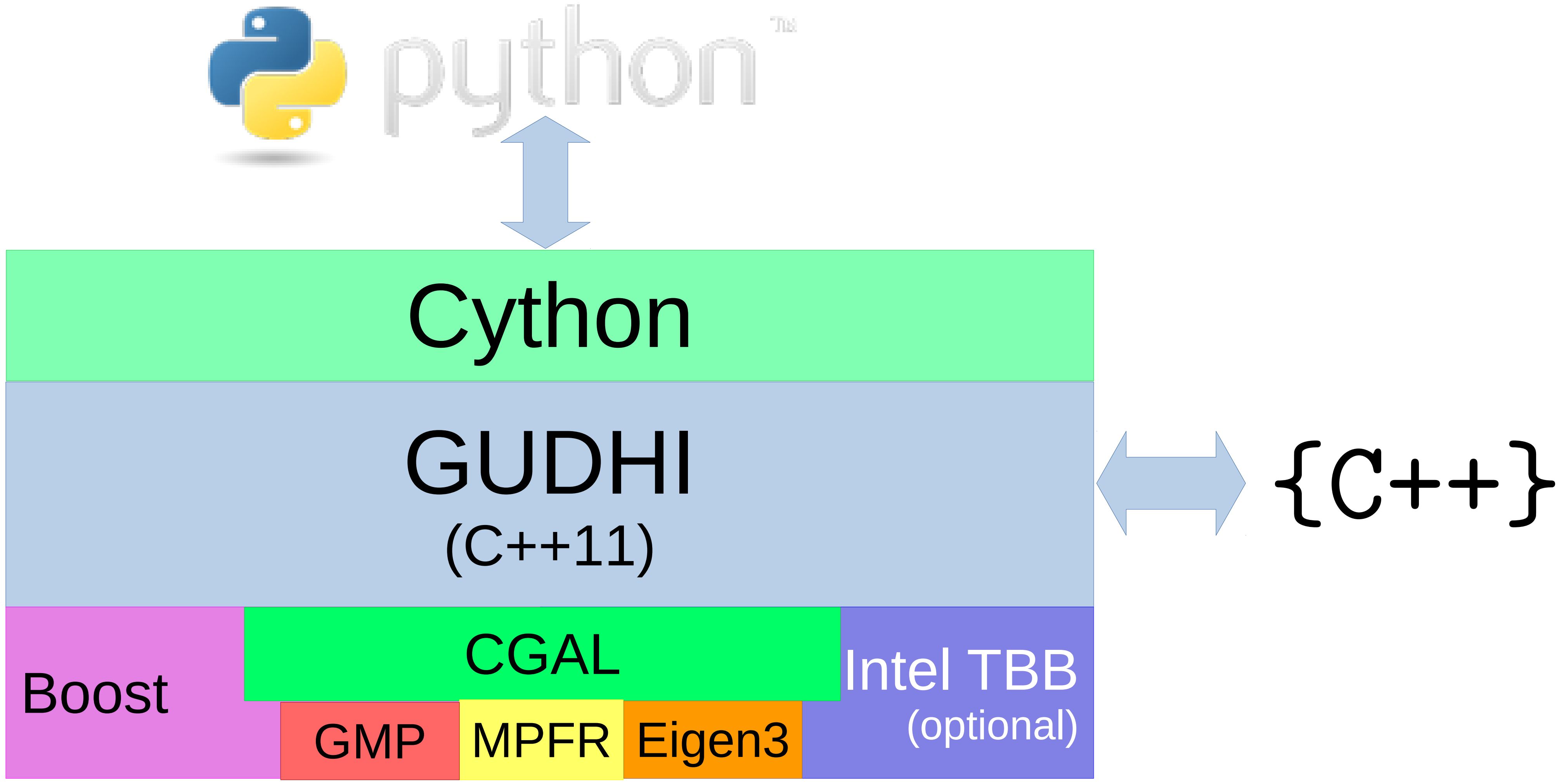
```
import gudhi
rips = gudhi.RipsComplex(points = points)

# could be also from an OFF file
rips = gudhi.RipsComplex(off_file = ‘off_file.off’)

# expands the graph until dimension 2
st = rips.create_simplex_tree(max_dimension = 2)
```

Geometric filtered complex creation – Alpha complex

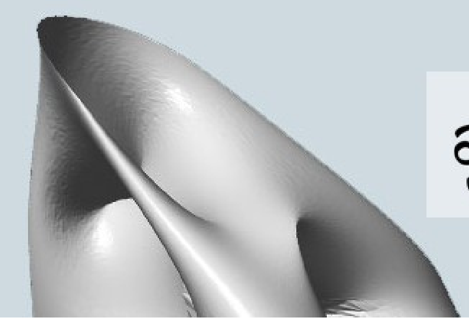
```
import gudhi
alpha = gudhi.AlphaComplex(points = points)

# could be also from an OFF file
alpha = gudhi.AlphaComplex(off_file = ‘off_file.off’)

# max_alpha_square default value is +infinity
# This means a Delaunay complex
st = alpha.create_simplex_tree()

# simplicial complex will be pruned above max_alpha_square
# filtration value
st = alpha.create_simplex_tree(max_alpha_square=0.2)
```

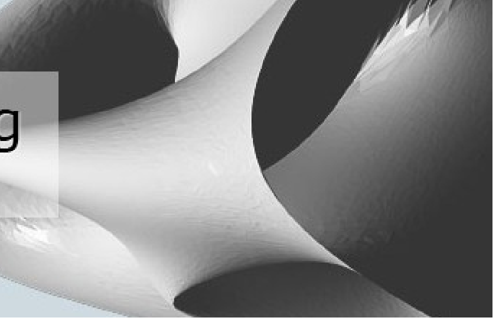


Installation



```
cmake -DCGAL_DIR=/your/path/to/CGAL ..
```

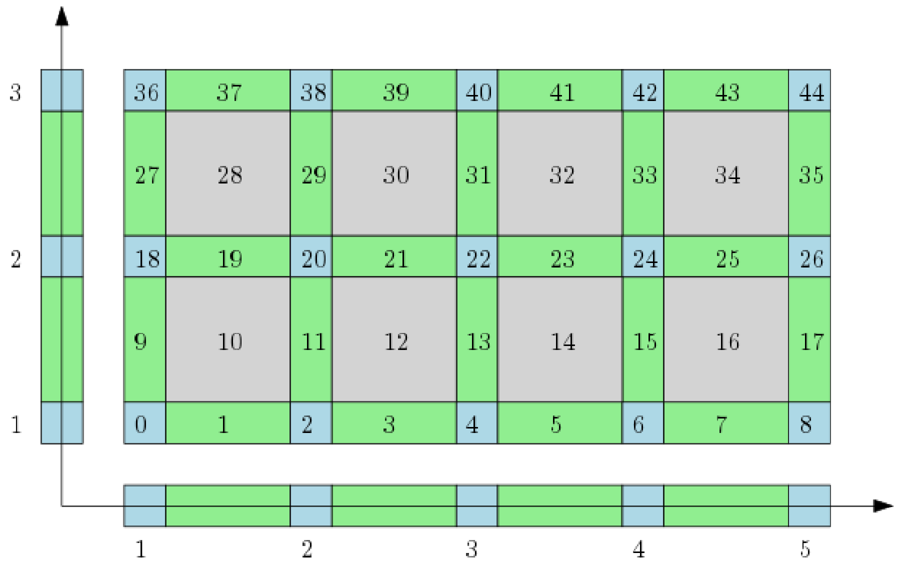




Geometric filtered complex creation – Witness complex

```
from gudhi import EuclideanWitnessComplex as EWC

witnesses = gudhi.read_off(off_file='off_file.off')
landmarks = gudhi.pick_n_random_points(points=witnesses,
                                       nb_points=100)

ewc = EWC(witnesses=witnesses,
          landmarks=landmarks)
simplex_tree = ewc.create_simplex_tree(max_alpha_square=0.2,
                                       limit_dimension=3)
```

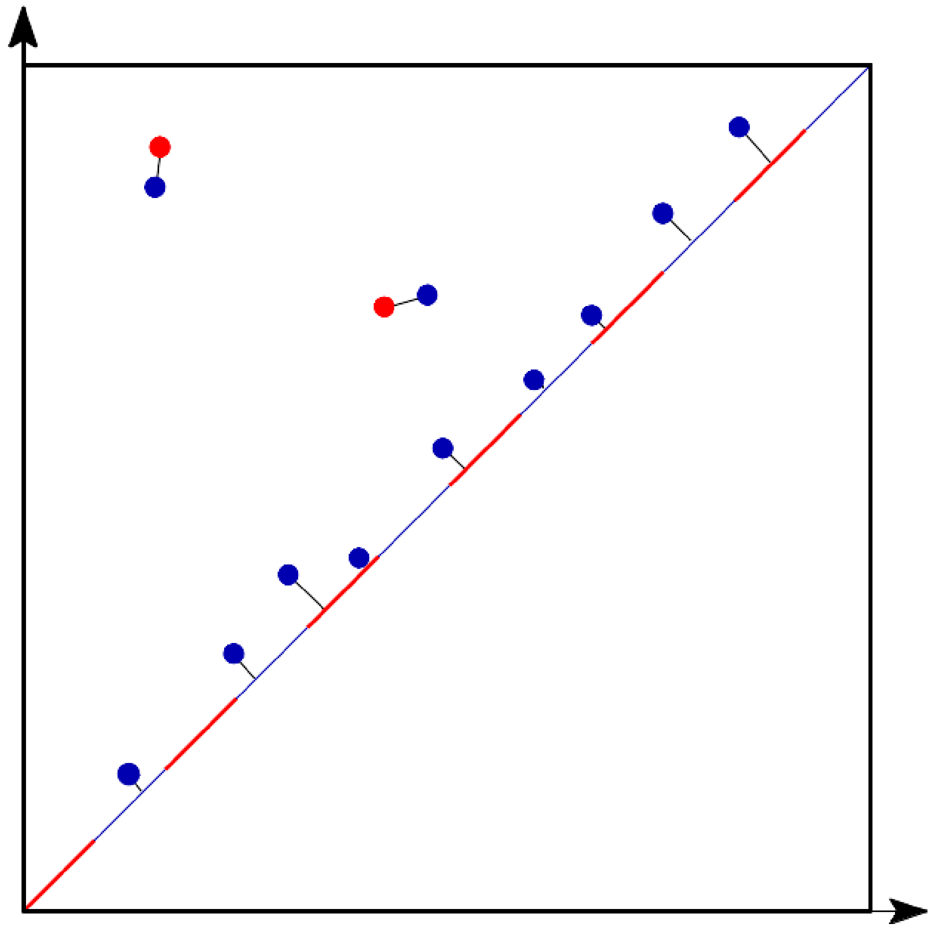
Data structure – Cubical complex

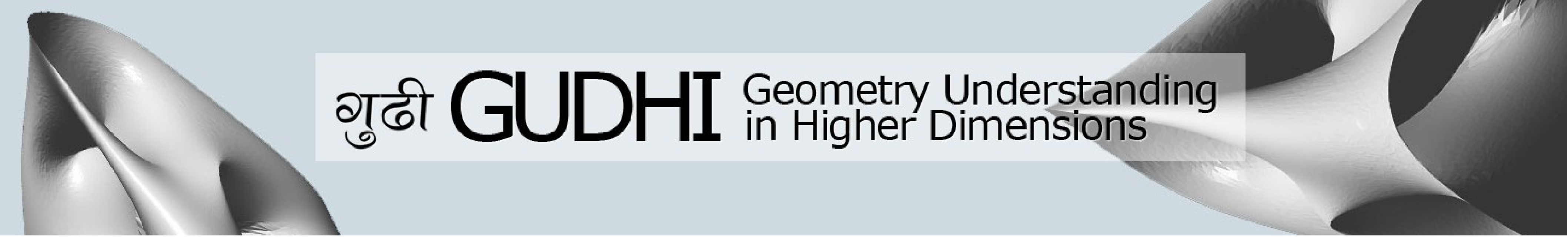


```
from gudhi import PeriodicCubicalComplex as pcc
cc = pcc(dimensions=[3],
         top_dimensional_cells= [0, 1, 0],
         periodic_dimensions=[True])
diag = cc.persistence()
cc. betti_numbers()
```

Hubert Wagner, Chao Chen, and Erald Vucini. Efficient Computation of Persistent Homology for **Cubical Data**, pages 91–106. Mathematics and Visualization. Springer Berlin Heidelberg, 2012.

Toolbox – Bottleneck distance



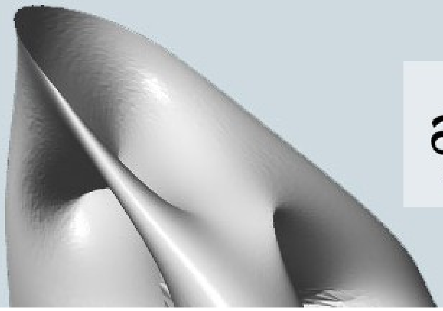


Geometric filtered complex creation – Tangential complex

```
# For manifold reconstruction
```

Data structure – Skeleton blockers

```
# Efficient for edge contraction
```

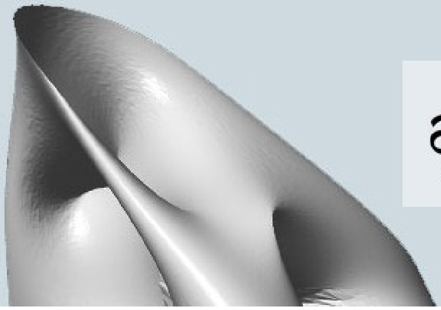


What will arrive in the next GUDHI version ?

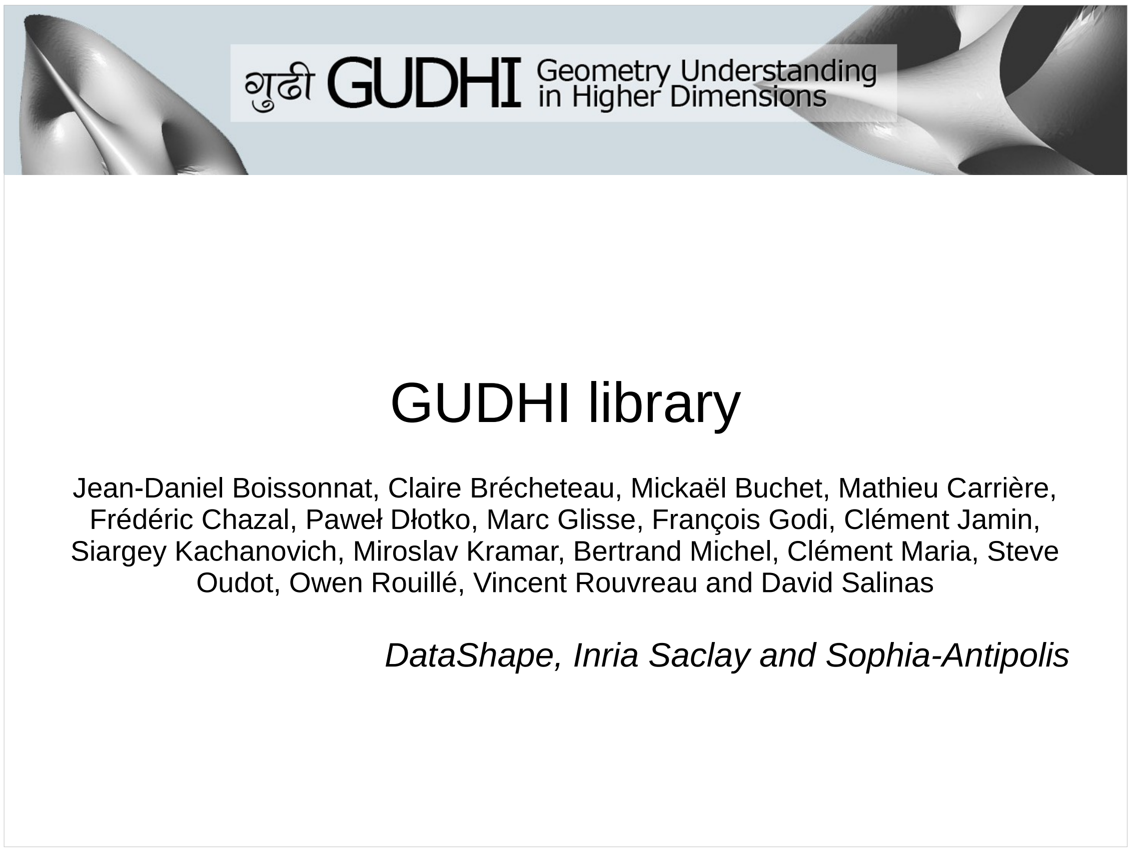
```
# Persistence representation
# Cover complexes
# Weighted and periodic alpha shapes in 3d
# Toplex map
```

Our website:
http://gudhi.gforge.inria.fr

If you want to join the GUDHI users community:
gudhi-users@lists.gforge.inria.fr



Thank you !


गुढी GUDHI Geometry Understanding in Higher Dimensions

GUDHI library

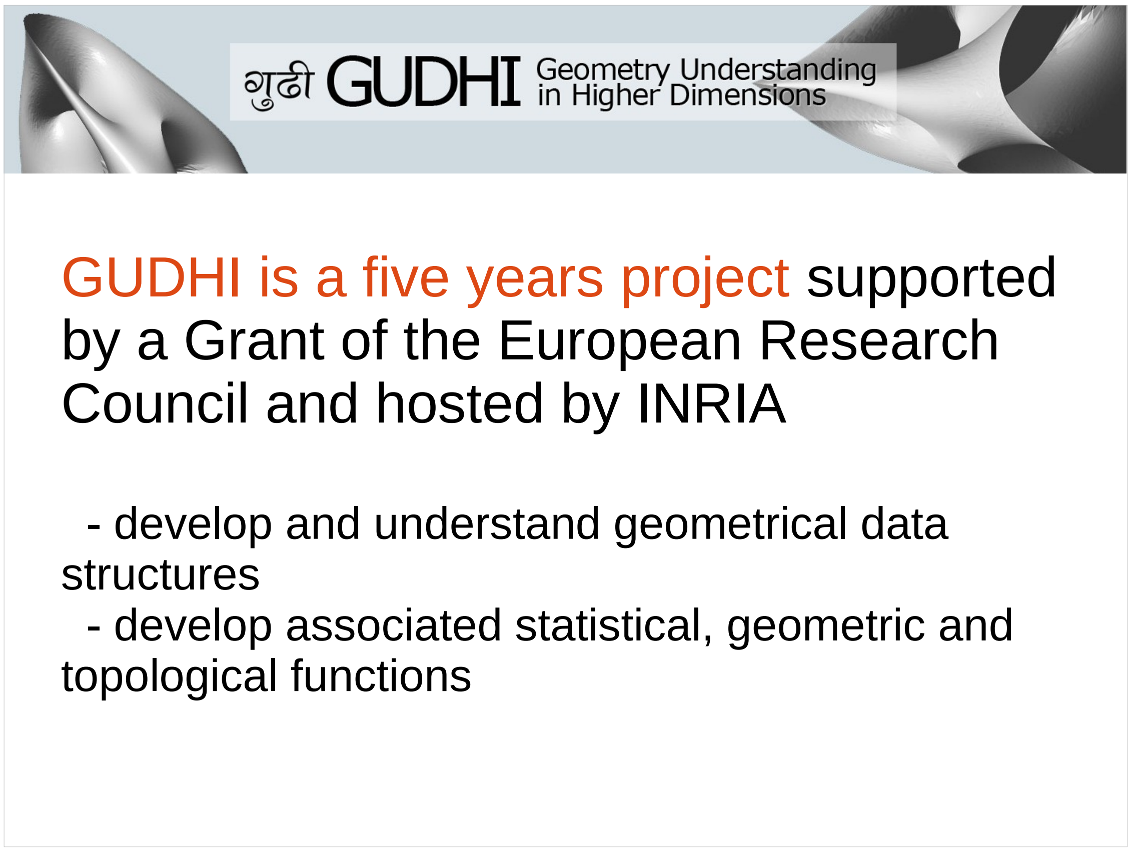
Jean-Daniel Boissonnat, Claire Brécheteau, Mickaël Buchet, Mathieu Carrière, Frédéric Chazal, Paweł Dłotko, Marc Glisse, François Godi, Clément Jamin, Siargey Kachanovich, Miroslav Kramar, Bertrand Michel, Clément Maria, Steve Oudot, Owen Rouillé, Vincent Rouvreau and David Salinas

DataShape, Inria Saclay and Sophia-Antipolis

Hello everybody, my name is Vincent Rouvreau and I am a software engineer at Inria in a research team called DataShape.

In this team, the research focuses on the shape of the data, in other words the geometry or the topology of the data.

My job is to develop and to help researchers to develop a consistent and efficient library based on these researches.

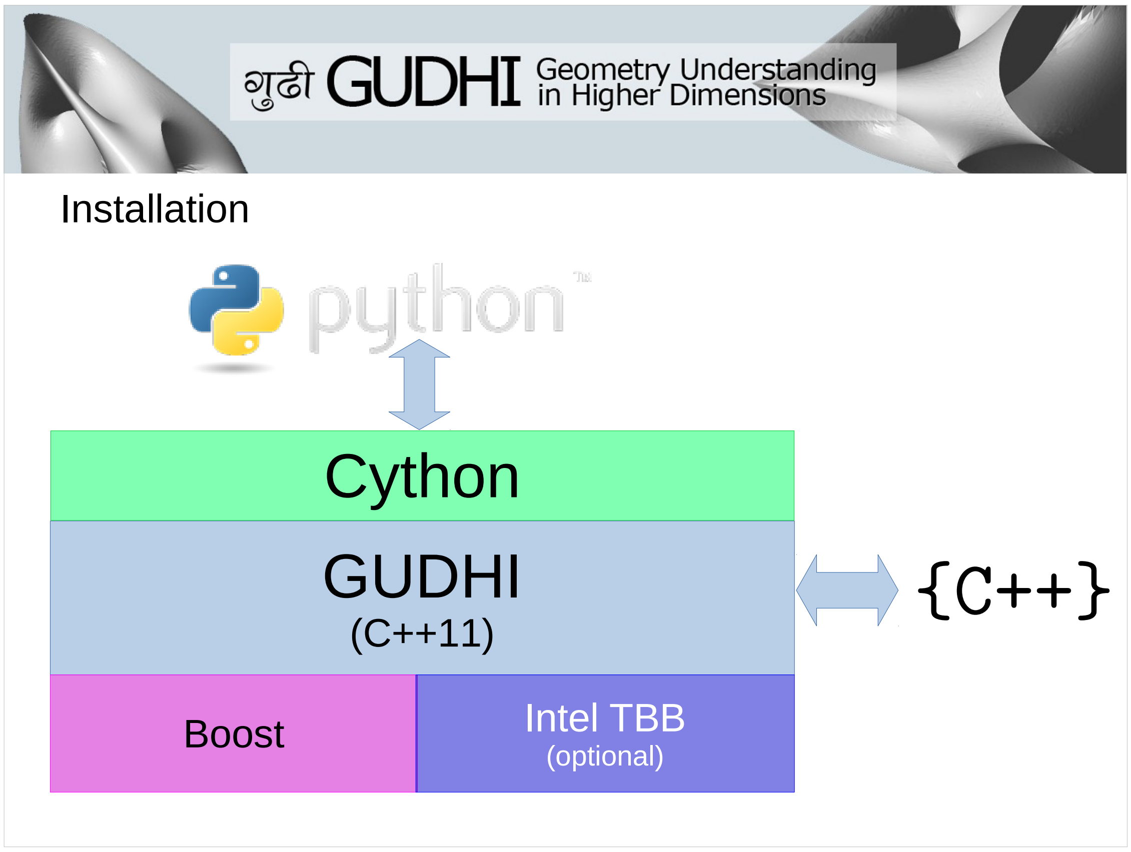

गुढी GUDHI Geometry Understanding in Higher Dimensions

GUDHI is a five years project supported by a Grant of the European Research Council and hosted by INRIA

- develop and understand geometrical data structures
- develop associated statistical, geometric and topological functions

The GUDHI library is developed as part of the GUDHI project supported by the European Research Council.

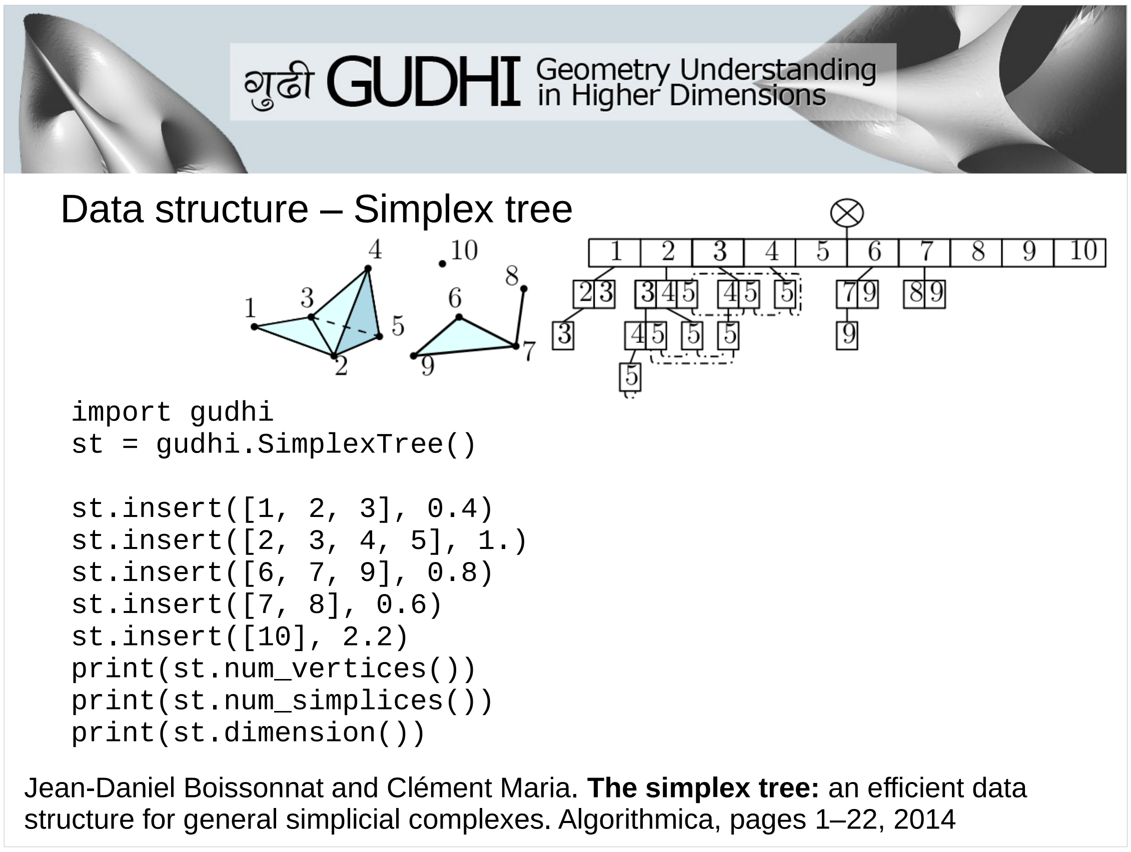
The central goal of this project is to build a software library for geometry understanding in dimensions higher than 3.



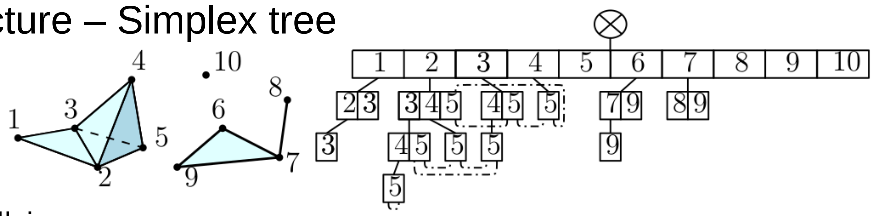
GUDHI is a C++ open source library. You just need to install Cmake and Boost to make it compile with a modern compiler that accepts C++11 syntaxes. Some few parts of the code have been parallelized with the Intel TBB library but it is optional.

And if you prefer Python, you will need to install Cython to compile it, numpy and matplotlib for persistence visualization tools. Pre compiled Python libraries are also available on Mac and Windows.

At GUDHI, we try to do what we call an “example driven documentation and development”. We want the users to be able to play with the examples in order to discover the library possibilities.


गुडी GUDHI Geometry Understanding in Higher Dimensions

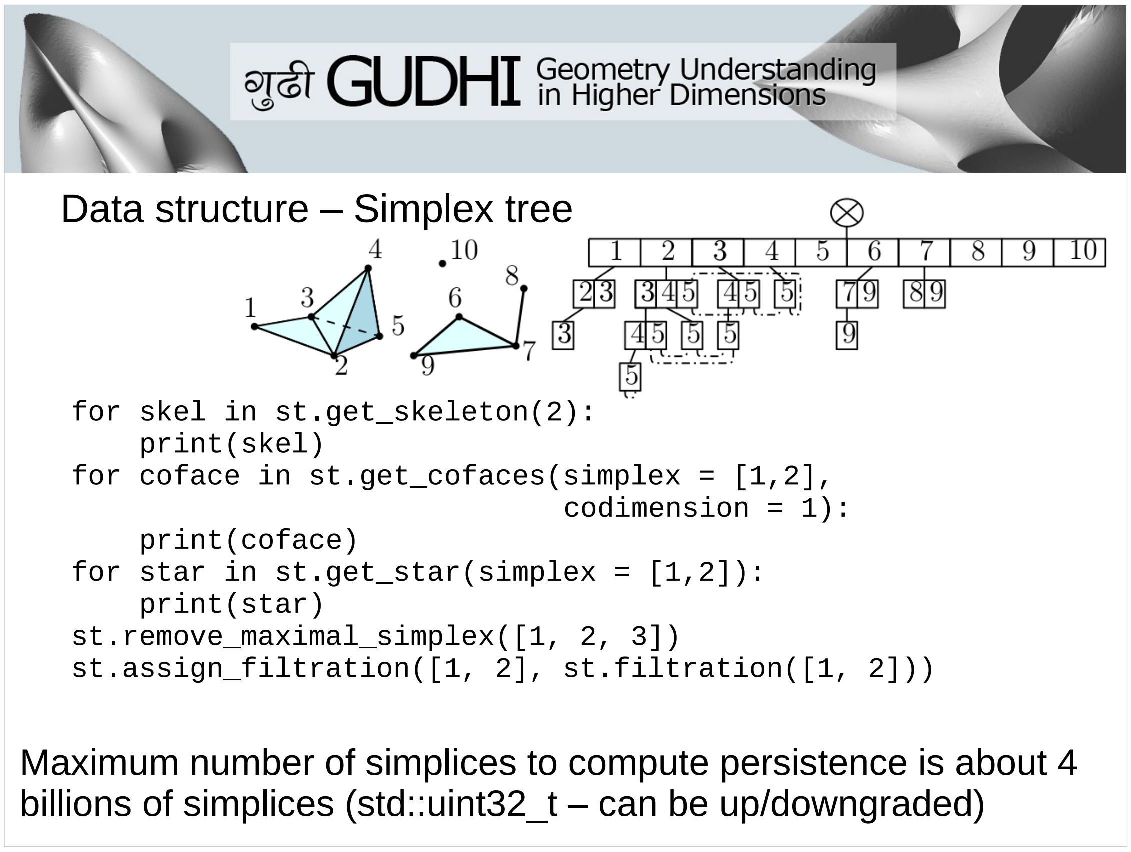
Data structure – Simplex tree



```
import gudhi
st = gudhi.SimplexTree()

st.insert([1, 2, 3], 0.4)
st.insert([2, 3, 4, 5], 1.)
st.insert([6, 7, 9], 0.8)
st.insert([7, 8], 0.6)
st.insert([10], 2.2)
print(st.num_vertices())
print(st.num_simplices())
print(st.dimension())
```

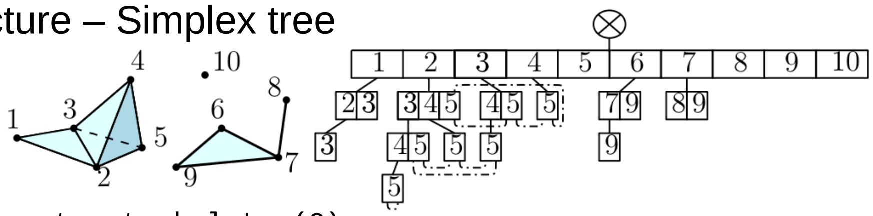
Jean-Daniel Boissonnat and Clément Maria. **The simplex tree:** an efficient data structure for general simplicial complexes. Algorithmica, pages 1–22, 2014

GUDHI is developped in a modular way. One of the main module is the simplex tree. It is a tree that is quite efficient for basic operations and to store simplicial complexes. If you want more design details, I will let you go through this publication. From a user point of view, you can insert or remove simplicial complexes (with or without filtration values). Have access to global information on your simplicial complex.



गुडी GUDHI Geometry Understanding in Higher Dimensions

Data structure – Simplex tree



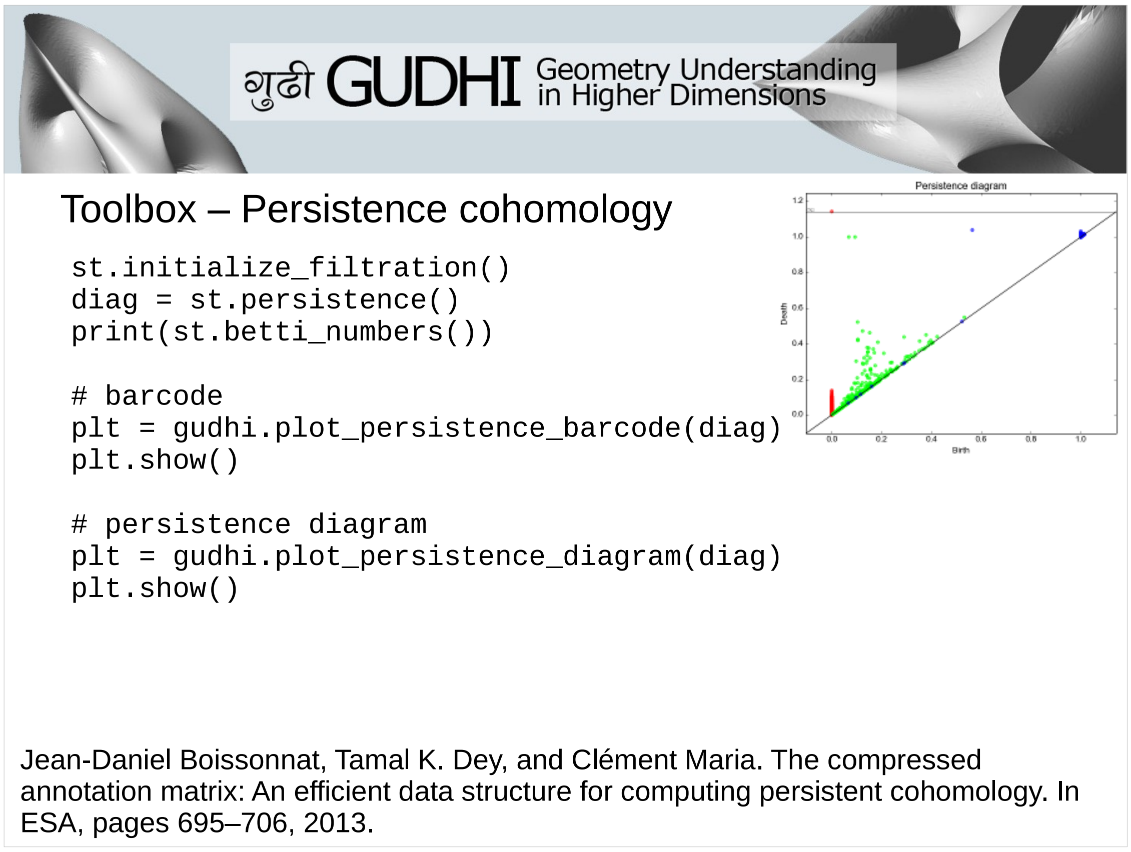
```
for skel in st.get_skeleton(2):
    print(skel)
for coface in st.get_cofaces(simplex = [1,2],
                             codimension = 1):
    print(coface)
for star in st.get_star(simplex = [1,2]):
    print(star)
st.remove_maximal_simplex([1, 2, 3])
st.assign_filtration([1, 2], st.filtration([1, 2]))
```

Maximum number of simplices to compute persistence is about 4 billions of simplices (std::uint32_t – can be up/downgraded)

You can also iterate efficiently on the simplicial complex skeleton, cofaces and stars. Update fonctions are also available if required.

In the Python version, the maximum number of simplices to compute persistence is about 4 billions of simplices, corresponding to the maximal value of the unsigned integer on 32 bits type in C++. This limit can be upgraded or downgraded in function of your needs through Simplex tree options in the C++ version.

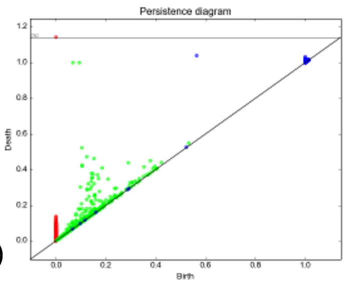
Just for you to have an idea, 4 billions of simplices for the Simplex tree is representing 225 GB of RAM. It is a compromise between memory usage and persistence computation performance.



गुढी GUDHI Geometry Understanding in Higher Dimensions

Toolbox – Persistence cohomology

```
st.initialize_filtration()
diag = st.persistence()
print(st.betti_numbers())

# barcode
plt = gudhi.plot_persistence_barcode(diag)
plt.show()

# persistence diagram
plt = gudhi.plot_persistence_diagram(diag)
plt.show()
```



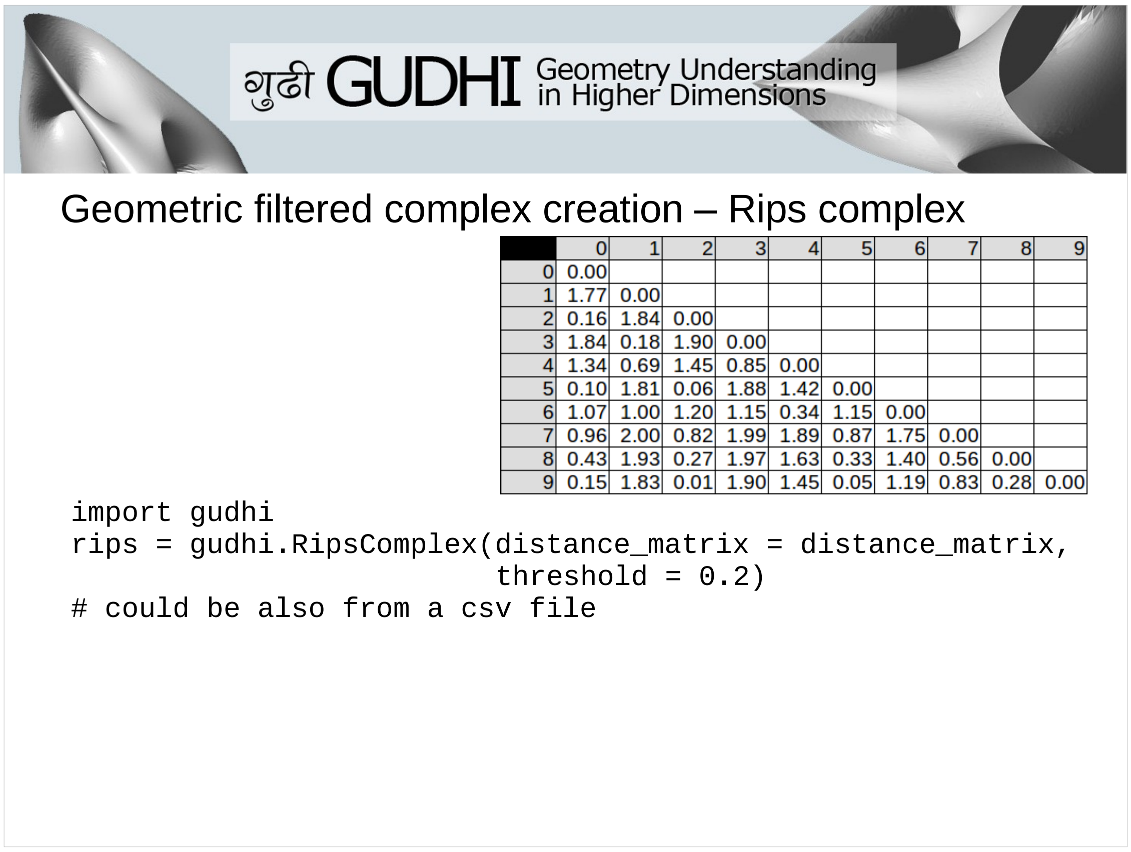
Jean-Daniel Boissonnat, Tamal K. Dey, and Clément Maria. The compressed annotation matrix: An efficient data structure for computing persistent cohomology. In ESA, pages 695–706, 2013.

Another main module of GUDHI is the persistence one.

I will let you again go through this paper if you want more details, but the main idea is to browse a list of simplices sorted by its filtration value in order to find global topological features like connected components, holes, cavities, and so on.

Thanks to this module, we are then able to find betti numbers, or to visualize persistence trough barcodes or diagrams.

As you can see, you are now able to build your own simplicial complexes, compute persistence and visualize it with only few lines of code.



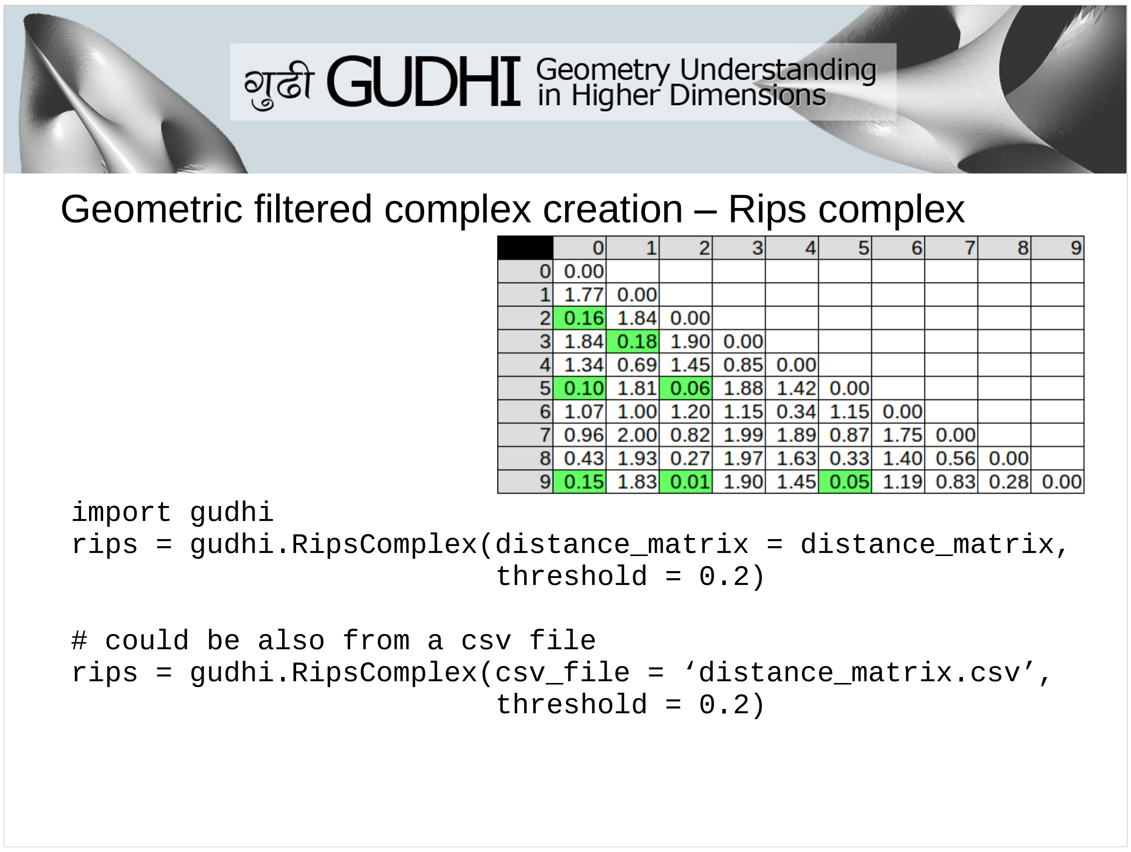
गुढी GUDHI Geometry Understanding in Higher Dimensions

Geometric filtered complex creation – Rips complex

	0	1	2	3	4	5	6	7	8	9
0	0.00									
1	1.77	0.00								
2	0.16	1.84	0.00							
3	1.84	0.18	1.90	0.00						
4	1.34	0.69	1.45	0.85	0.00					
5	0.10	1.81	0.06	1.88	1.42	0.00				
6	1.07	1.00	1.20	1.15	0.34	1.15	0.00			
7	0.96	2.00	0.82	1.99	1.89	0.87	1.75	0.00		
8	0.43	1.93	0.27	1.97	1.63	0.33	1.40	0.56	0.00	
9	0.15	1.83	0.01	1.90	1.45	0.05	1.19	0.83	0.28	0.00

```
import gudhi
rips = gudhi.RipsComplex(distance_matrix = distance_matrix,
                         threshold = 0.2)
# could be also from a csv file
```

As an input to the simplex tree, in GUDHI, there are also some geometric creation helpers. One of them is the Rips complex that can be constructed from a distance matrix for instance. Here is how it works.



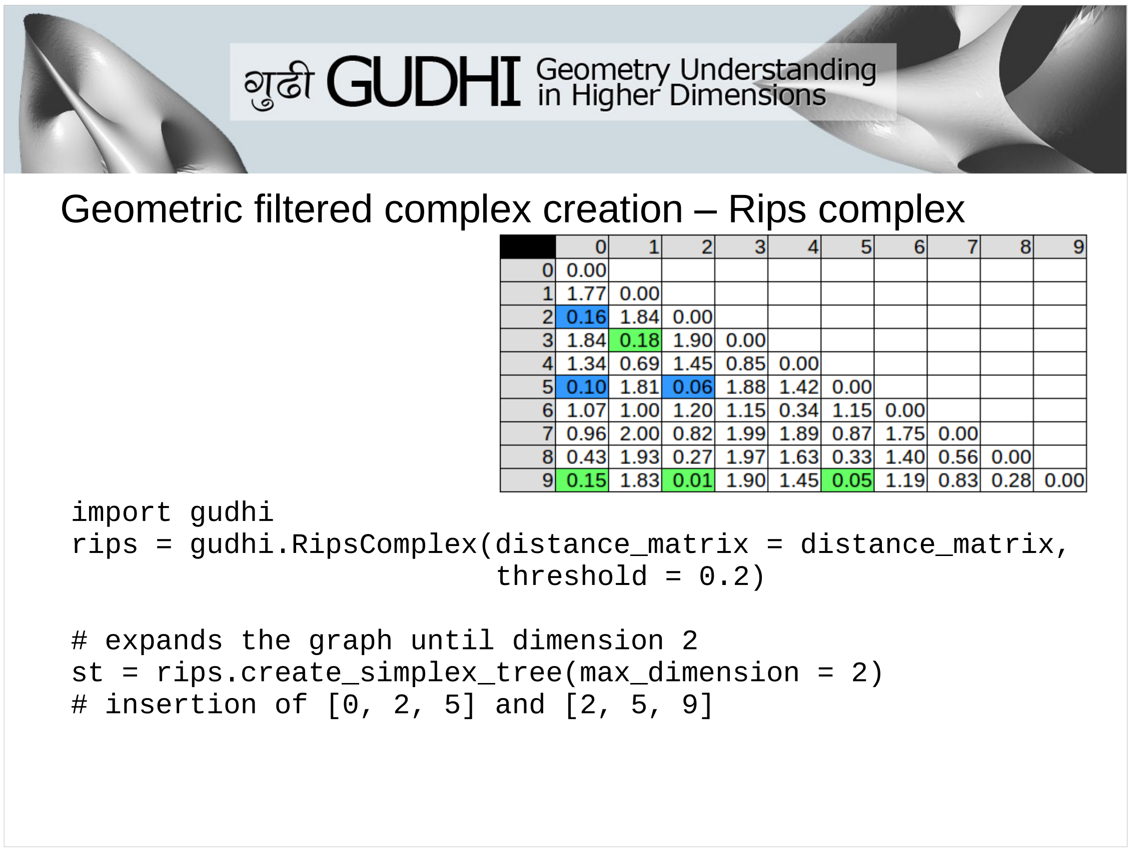
ग़ुढी GUDHI Geometry Understanding in Higher Dimensions

Geometric filtered complex creation – Rips complex

	0	1	2	3	4	5	6	7	8	9
0	0.00									
1	1.77	0.00								
2	0.16	1.84	0.00							
3	1.84	0.18	1.90	0.00						
4	1.34	0.69	1.45	0.85	0.00					
5	0.10	1.81	0.06	1.88	1.42	0.00				
6	1.07	1.00	1.20	1.15	0.34	1.15	0.00			
7	0.96	2.00	0.82	1.99	1.89	0.87	1.75	0.00		
8	0.43	1.93	0.27	1.97	1.63	0.33	1.40	0.56	0.00	
9	0.15	1.83	0.01	1.90	1.45	0.05	1.19	0.83	0.28	0.00

```
import gudhi
rips = gudhi.RipsComplex(distance_matrix = distance_matrix,
                         threshold = 0.2)

# could be also from a csv file
rips = gudhi.RipsComplex(csv_file = ‘distance_matrix.csv’,
                         threshold = 0.2)
```

With a given threshold value, the one-skeleton graph is constructed with the edges that are less or equal to the threshold value. Here, for instance, it means the green edges [0,2], [0,5], [0,9] and so on are inserted in the one-skeleton graph.



GUDHI Geometry Understanding in Higher Dimensions

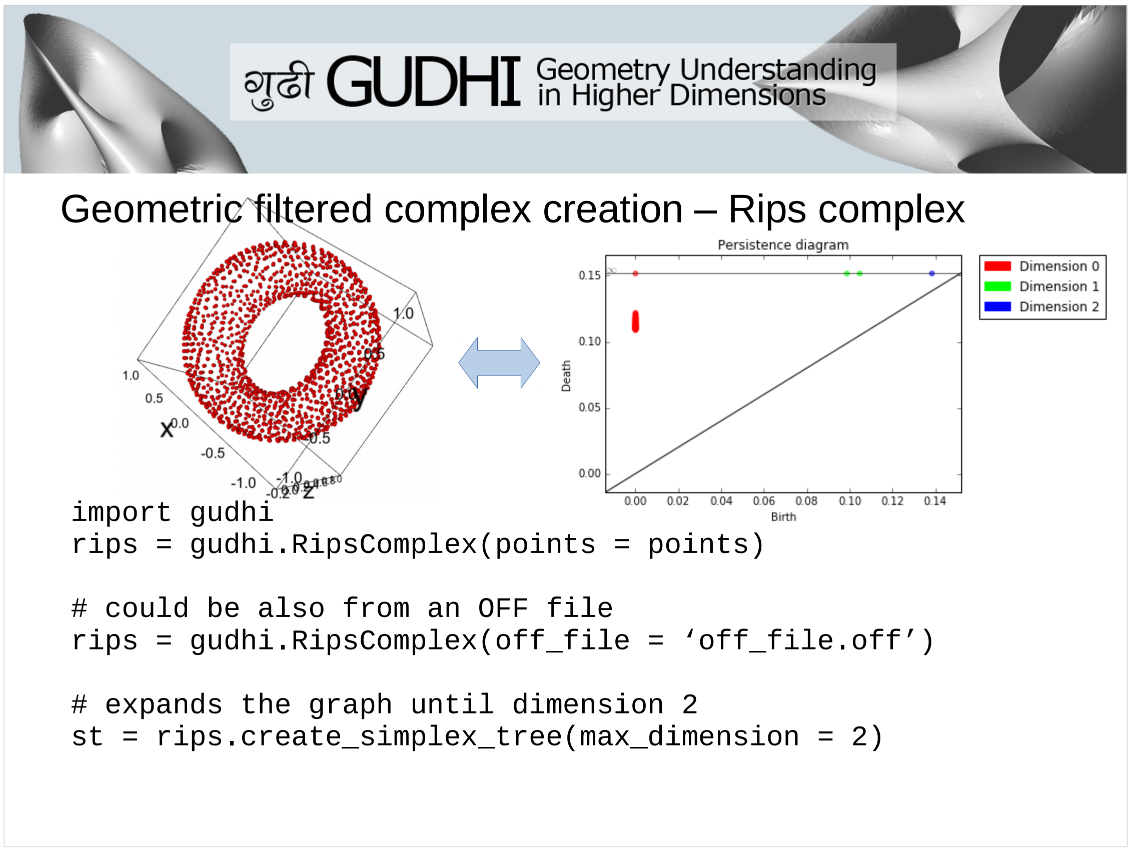
Geometric filtered complex creation – Rips complex

	0	1	2	3	4	5	6	7	8	9
0	0.00									
1	1.77	0.00								
2	0.16	1.84	0.00							
3	1.84	0.18	1.90	0.00						
4	1.34	0.69	1.45	0.85	0.00					
5	0.10	1.81	0.06	1.88	1.42	0.00				
6	1.07	1.00	1.20	1.15	0.34	1.15	0.00			
7	0.96	2.00	0.82	1.99	1.89	0.87	1.75	0.00		
8	0.43	1.93	0.27	1.97	1.63	0.33	1.40	0.56	0.00	
9	0.15	1.83	0.01	1.90	1.45	0.05	1.19	0.83	0.28	0.00

```
import gudhi
rips = gudhi.RipsComplex(distance_matrix = distance_matrix,
                         threshold = 0.2)

# expands the graph until dimension 2
st = rips.create_simplex_tree(max_dimension = 2)
# insertion of [0, 2, 5] and [2, 5, 9]
```

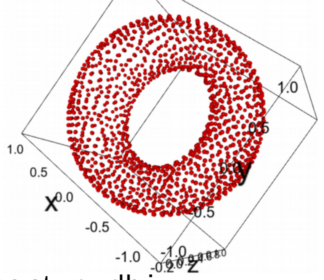
Then, when creating the simplex tree from the Rips complex, it will expand the graph until the given dimension. In this example, as edges [0, 2], [0, 5] and [2, 5] are there, the triangle [0, 2, 5] is expanded. The same for the triangle [2, 5, 9].

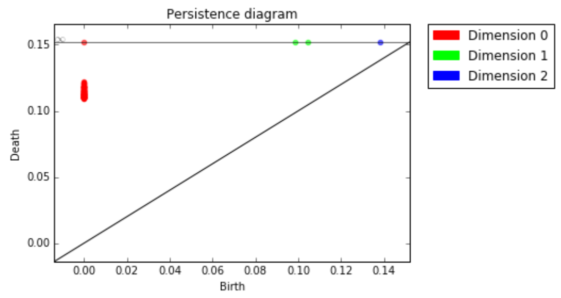
You can again see that it is quite easy to build a Vietoris-Rips simplicial complex with GUDHI. With only few lines of code, you can compute and display Rips persistence from a distance matrix.



गुढी GUDHI Geometry Understanding in Higher Dimensions

Geometric filtered complex creation – Rips complex



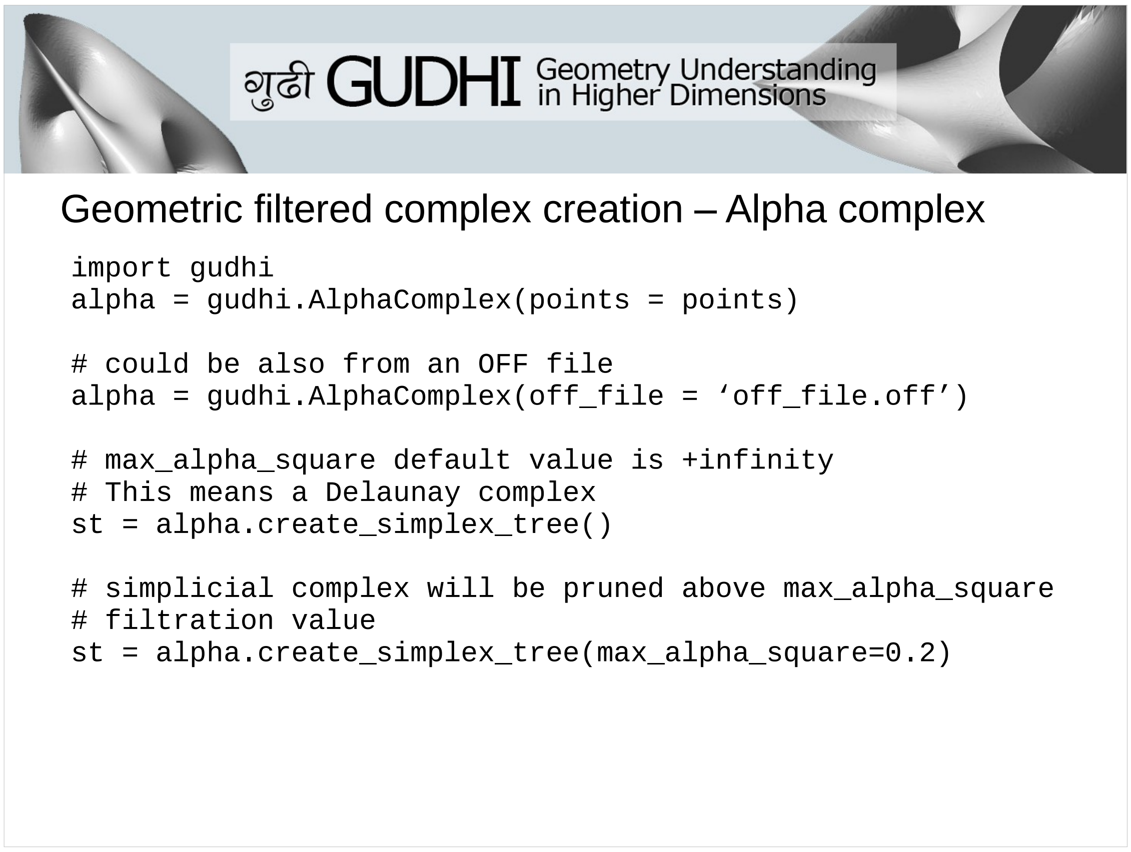


```
import gudhi
rips = gudhi.RipsComplex(points = points)

# could be also from an OFF file
rips = gudhi.RipsComplex(off_file = ‘off_file.off’)

# expands the graph until dimension 2
st = rips.create_simplex_tree(max_dimension = 2)
```

The Rips complex constructor is also available with a point cloud given as a user input, or with a point cloud from an OFF file. In this case, the Euclidean distance will be computed for each pair of points.

If you want to build it on top of your own customized distance function, it is still possible, but on the C++ level.

[DEMO Rips complex]



गुढी GUDHI Geometry Understanding in Higher Dimensions

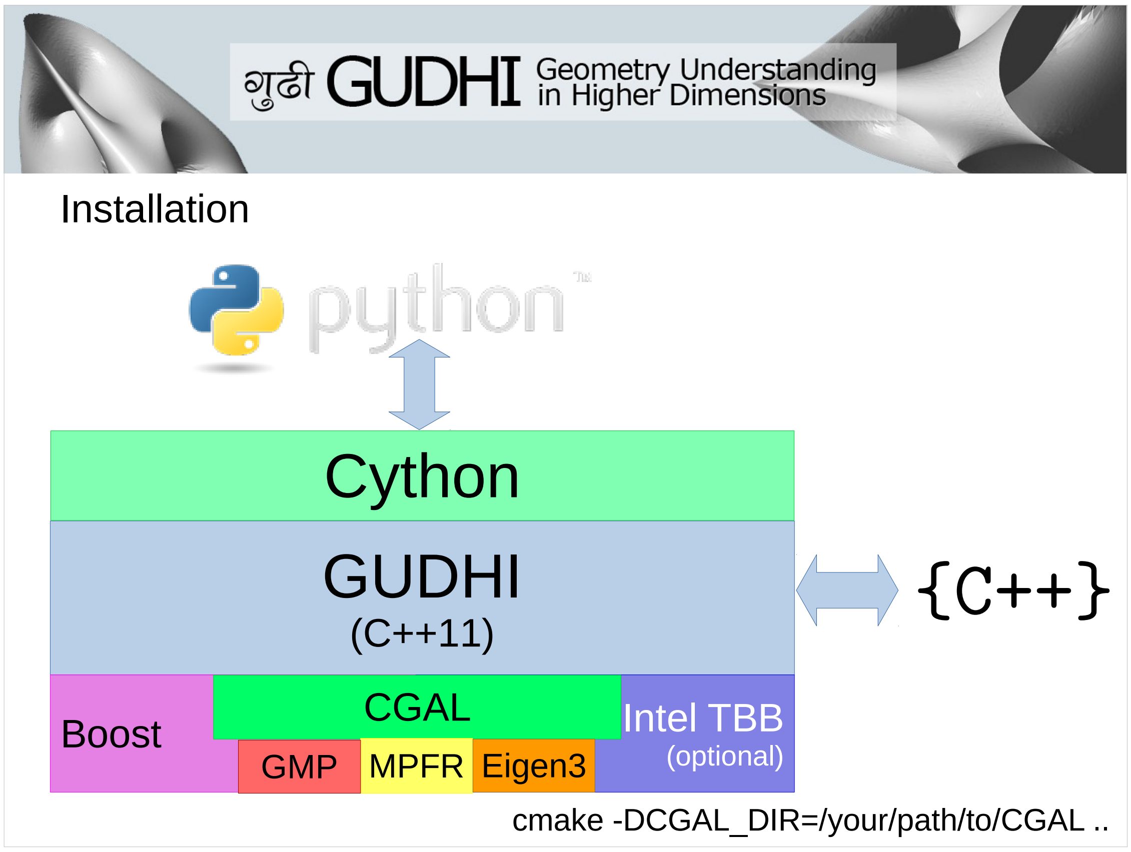
Geometric filtered complex creation – Alpha complex

```
import gudhi
alpha = gudhi.AlphaComplex(points = points)

# could be also from an OFF file
alpha = gudhi.AlphaComplex(off_file = ‘off_file.off’)

# max_alpha_square default value is +infinity
# This means a Delaunay complex
st = alpha.create_simplex_tree()

# simplicial complex will be pruned above max_alpha_square
# filtration value
st = alpha.create_simplex_tree(max_alpha_square=0.2)
```

If your data are in an Euclidean space, and in a quite low dimension (let’s say 2, 3, 4 or even 5), we advise you to use Alpha Complex as it uses less memory and draws nicest persistence diagram than Rips in these conditions.

Alpha complex is equivalent to the union of growing balls.

[DEMO Alpha complex]



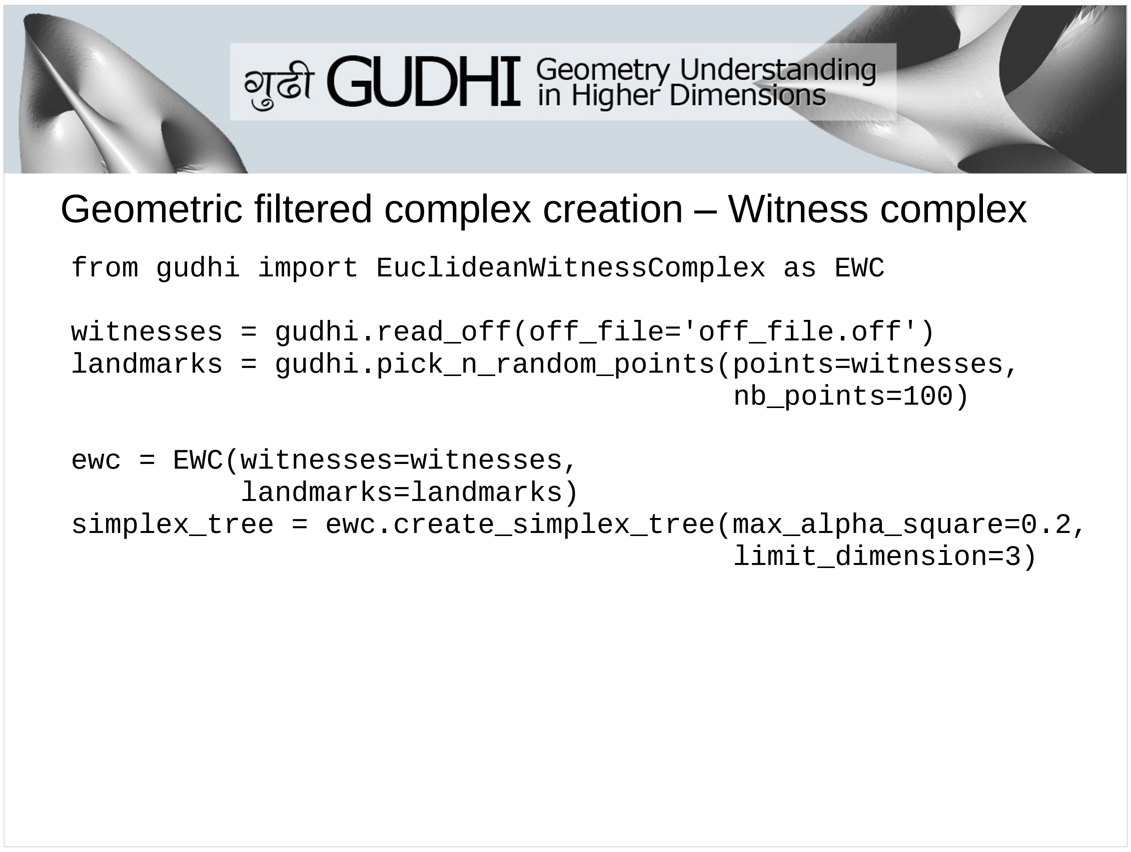
Installation

```
cmake -DCGAL_DIR=/your/path/to/CGAL ..
```

In order to compute such geometrical information, we went with the Computational Geometry Algorithms Library, called CGAL. The CGAL modules we are using in GUDHI requires at most GMP, MPFR and Eigen3 to be installed. For ease of installation, I advise you to install the header only version of CGAL, and then you just have to cmake -DCGAL_DIR=/your/path/to/CGAL to your GUDHI installation.

If you need to analyse Euclidean data in dimension 3, you can also find in the GUDHI C++ examples a specific version for 3d Alpha complexes that will be faster.

A periodic and a weighted version of this specific 3d case are also available.



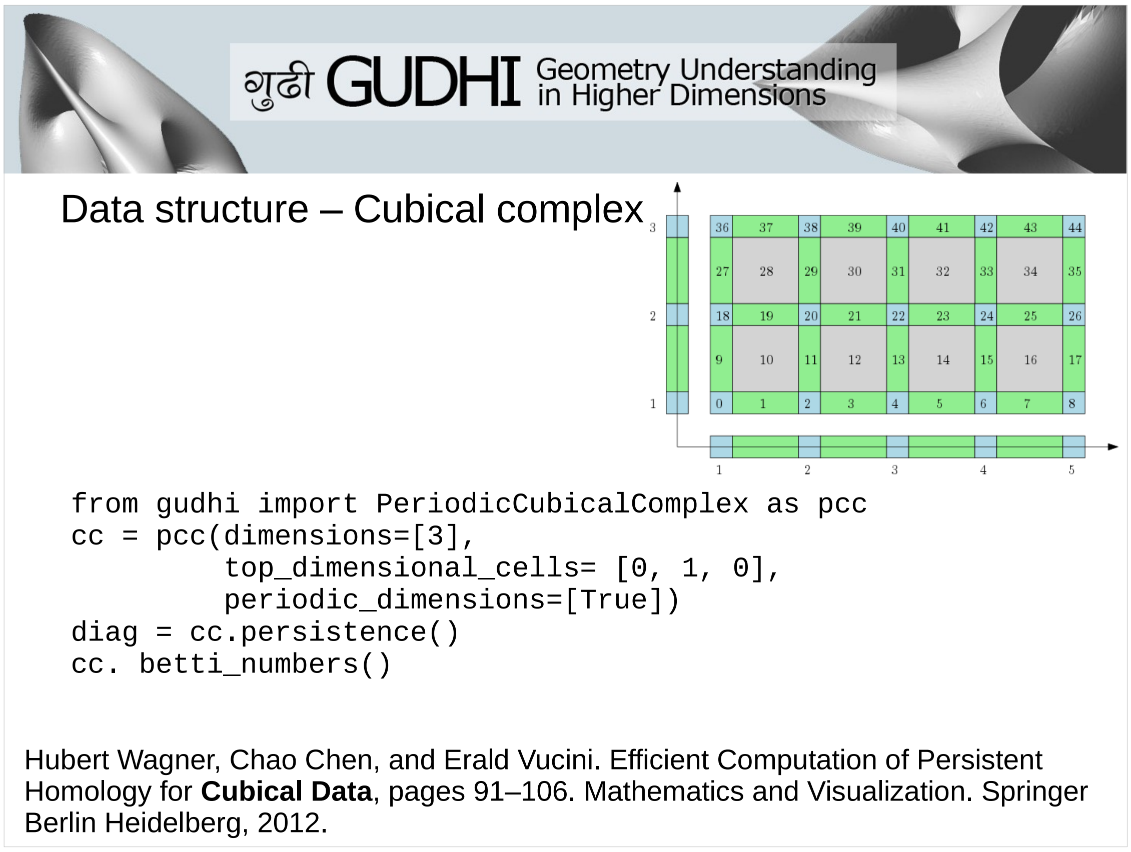
गुढी GUDHI Geometry Understanding in Higher Dimensions

Geometric filtered complex creation – Witness complex

```
from gudhi import EuclideanWitnessComplex as EWC

witnesses = gudhi.read_off(off_file='off_file.off')
landmarks = gudhi.pick_n_random_points(points=witnesses,
                                       nb_points=100)

ewc = EWC(witnesses=witnesses,
          landmarks=landmarks)
simplex_tree = ewc.create_simplex_tree(max_alpha_square=0.2,
                                       limit_dimension=3)
```

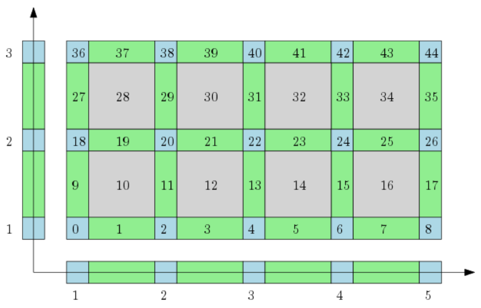
For larger datasets, we advise you to use Witness Complex as it will store less simplices than Rips and still efficient in higher dimensions.

It is a kind of compromise between the Rips that requires a lot of simplices and the Alpha complex that can take a lot of time to compute in high dimension.



GUDHI Geometry Understanding in Higher Dimensions

Data structure – Cubical complex



```
from gudhi import PeriodicCubicalComplex as pcc
cc = pcc(dimensions=[3],
         top_dimensional_cells= [0, 1, 0],
         periodic_dimensions=[True])
diag = cc.persistence()
cc. betti_numbers()
```

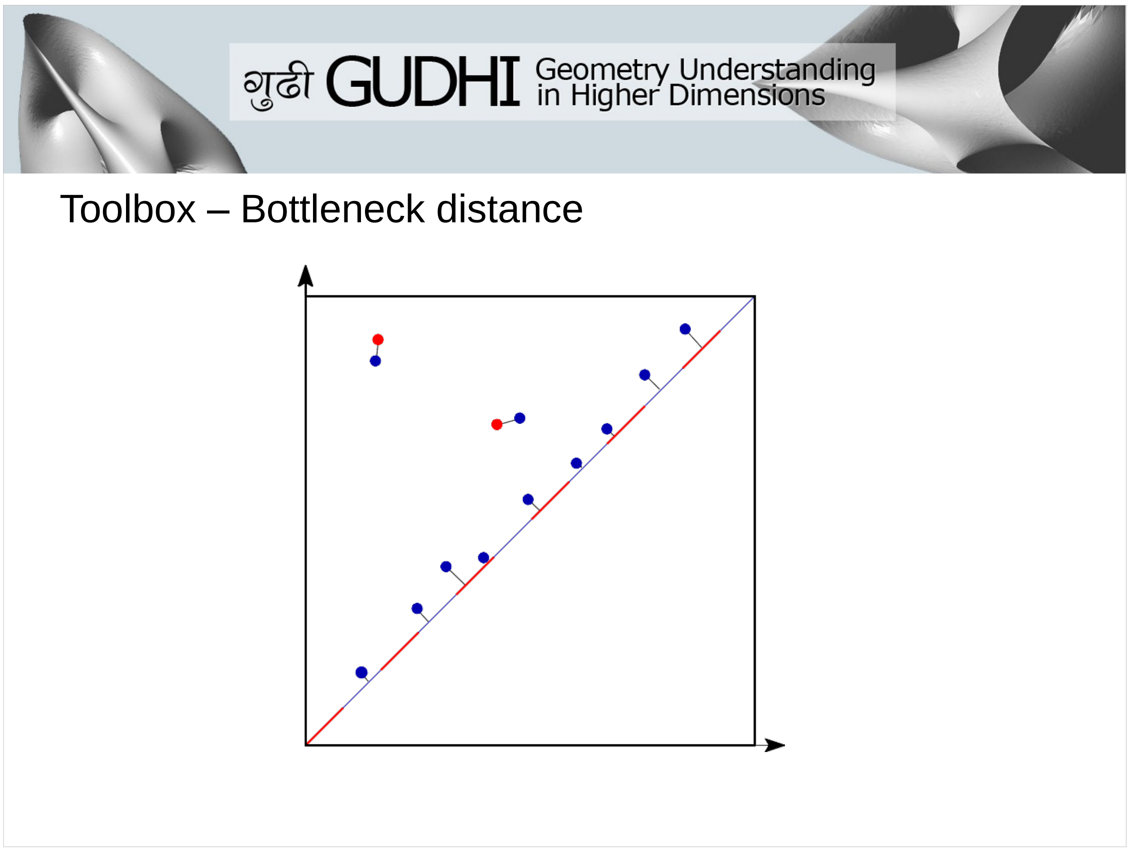
Hubert Wagner, Chao Chen, and Erald Vucini. Efficient Computation of Persistent Homology for **Cubical Data**, pages 91–106. Mathematics and Visualization. Springer Berlin Heidelberg, 2012.

Another data structure that is quite efficient to compute persistence on top of it when your data are on a grid is the Cubical complex.

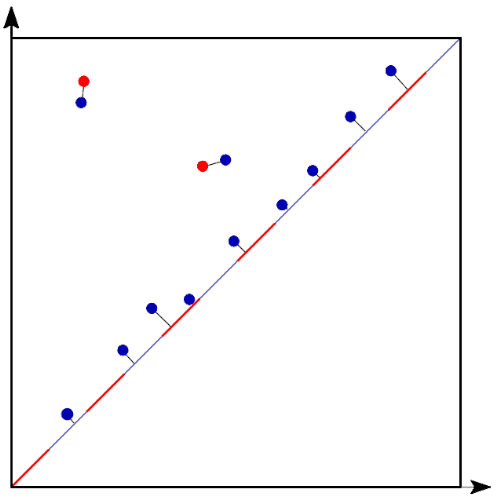
Here we just store the filtration value at a given grid cell.

A periodic version of this module is also available.

[DEMO Cubical complex]

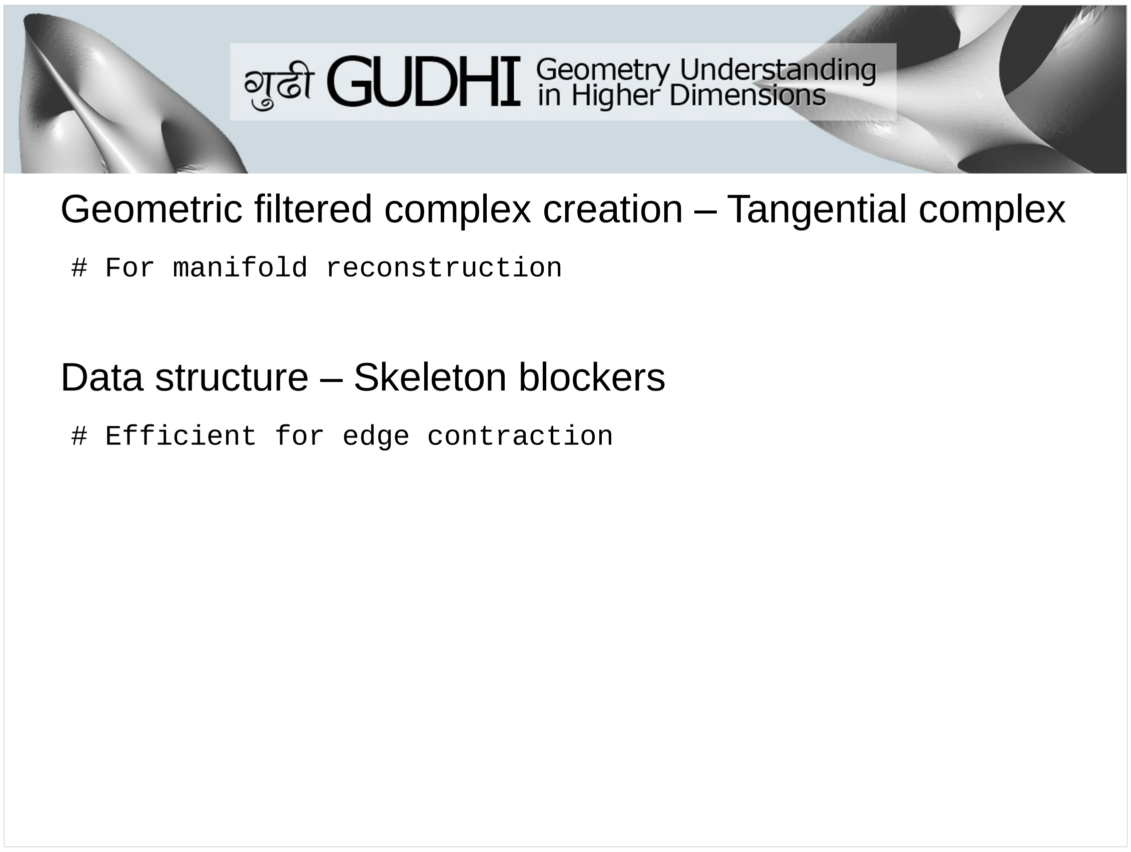


Toolbox – Bottleneck distance



On top of persistence module, there is also a bottleneck distance module that is quite efficient. It uses geometrical properties in order to compute bottleneck distance between 2 persistence diagrams.

[Bottleneck distance DEMO]


Geometric filtered complex creation – Tangential complex

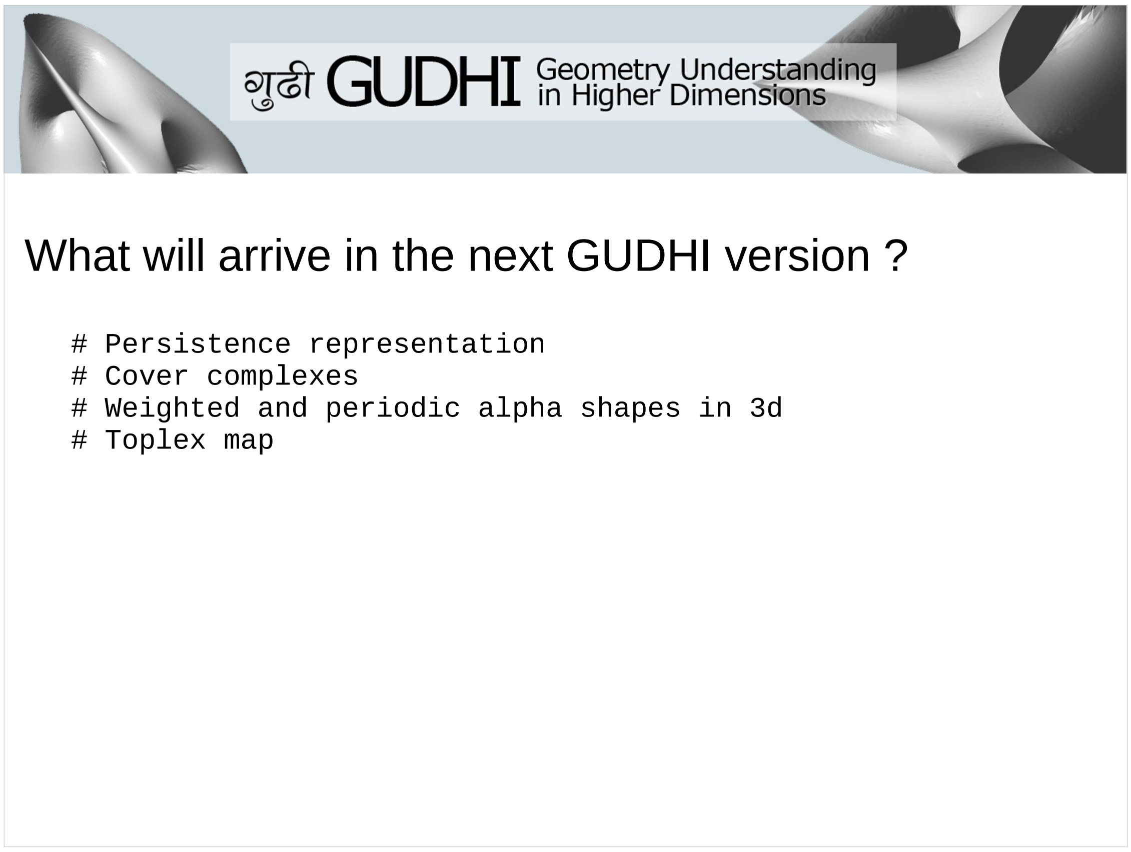
```
# For manifold reconstruction
```

Data structure – Skeleton blockers

```
# Efficient for edge contraction
```

If you are interested in manifold reconstructions, a tangential complex module is also available in GUDHI.

There is also a skeleton blockers data structure that is quite efficient for simplicial complex edge contractions.



गुढी GUDHI Geometry Understanding in Higher Dimensions

What will arrive in the next GUDHI version ?

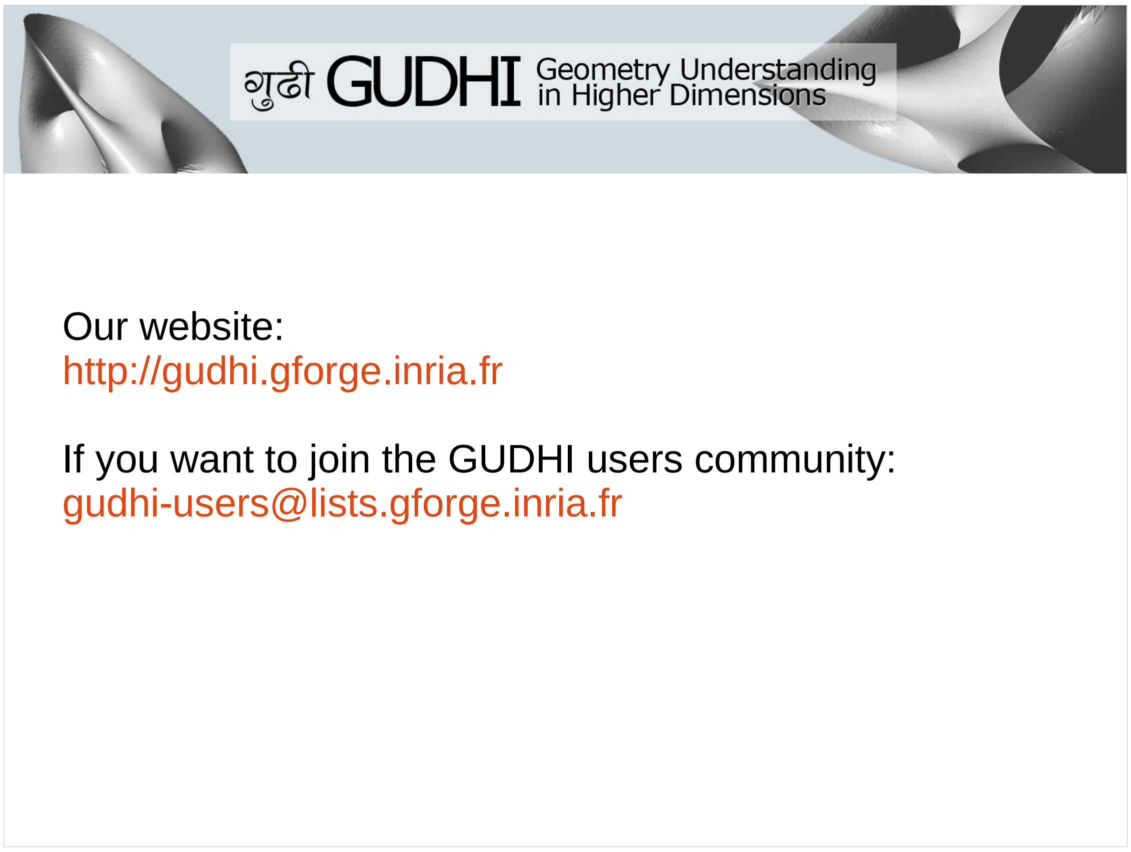
```
# Persistence representation
# Cover complexes
# Weighted and periodic alpha shapes in 3d
# Toplex map
```

In the next GUDHI 2.1.0 version, there will be a new persistence representation module if you are interested to build heat maps or landscapes on top of your persistence diagrams.

There will be also nerve and graph induced complexes. Quite interesting when you can cover your data.

Also an example of weighted and periodic alpha shapes in 3d will be available.

And if the development is ready before the version freeze, there will be a new data structure that stores only maximal simplices in order to reduce memory impact compared to the simplex tree, specifically in higher dimensions.



गुडी GUDHI Geometry Understanding in Higher Dimensions

Our website:
http://gudhi.gforge.inria.fr

If you want to join the GUDHI users community:
gudhi-users@lists.gforge.inria.fr

Do not hesitate to keep in touch on our website and to check our examples and utilities gallery.

If you want to join the GUDHI users community, you are welcome, and do not hesitate to join or contact us on our mailing list.

Thank you !

Thank you for your attention, if you have any question.