

TTK Installation Instructions

- TTK comes with Paraview, it can be downloaded here:
 - <https://www.paraview.org/>
- For those who want to follow along on your laptop during my presentation, you can find the same examples at:
 - <https://topology-tool-kit.github.io/examples/index.html>
- And while there are some direct links to individual datasets, if you want to download _all demos_ you would clone this repo (~1GB):
 - <https://github.com/topology-tool-kit/ttk-data>



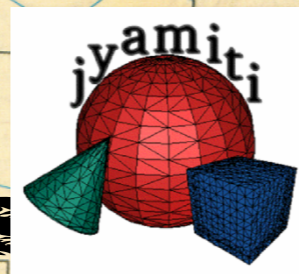
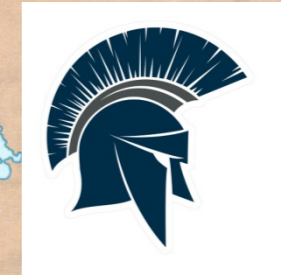
An Introduction to the Topology ToolKit (TTK)

Joshua A. Levine and Julien Tierny

University of Arizona

Topological Data Visualizaton Workshop, May 2022

About Josh



THE UNIVERSITY OF ARIZONA



Where the Parks Are

North, South, East, Midwest, West—each section of the United States has at least one national park to call its own. You don't have to visit the parks to enjoy their splendors, for this book is filled with their sights and their sounds. But if a trip is in the offing, this map and the chart on the two following pages will serve as your guide.

University of Arizona is in Tucson, AZ USA



Tucson

WEST UNIVERSITY

Computer Science Dept



Arizona State Museum

2nd Street Parking Garage

The University of Arizona

University of Arizona Libraries - Main Library

Flandrau Science Center and Planetarium
Stargazing spot with hands-on displays

Wyant College of Optical Sciences

Co



WEST UNIVERSITY

2nd Street Parking Garage

University of Arizona Libraries - Main Library

Tucson is in the Sonoran *Desert*
Average High Temperature in June/July: ~40°C
Annual rainfall: 270 mm / 10.61 inches

Center and Planetarium
Stargazing spot with hands-on displays

Wyant College of Optical Sciences

University of Arizona is in Tucson, AZ USA



University of Arizona is in Tucson, AZ USA



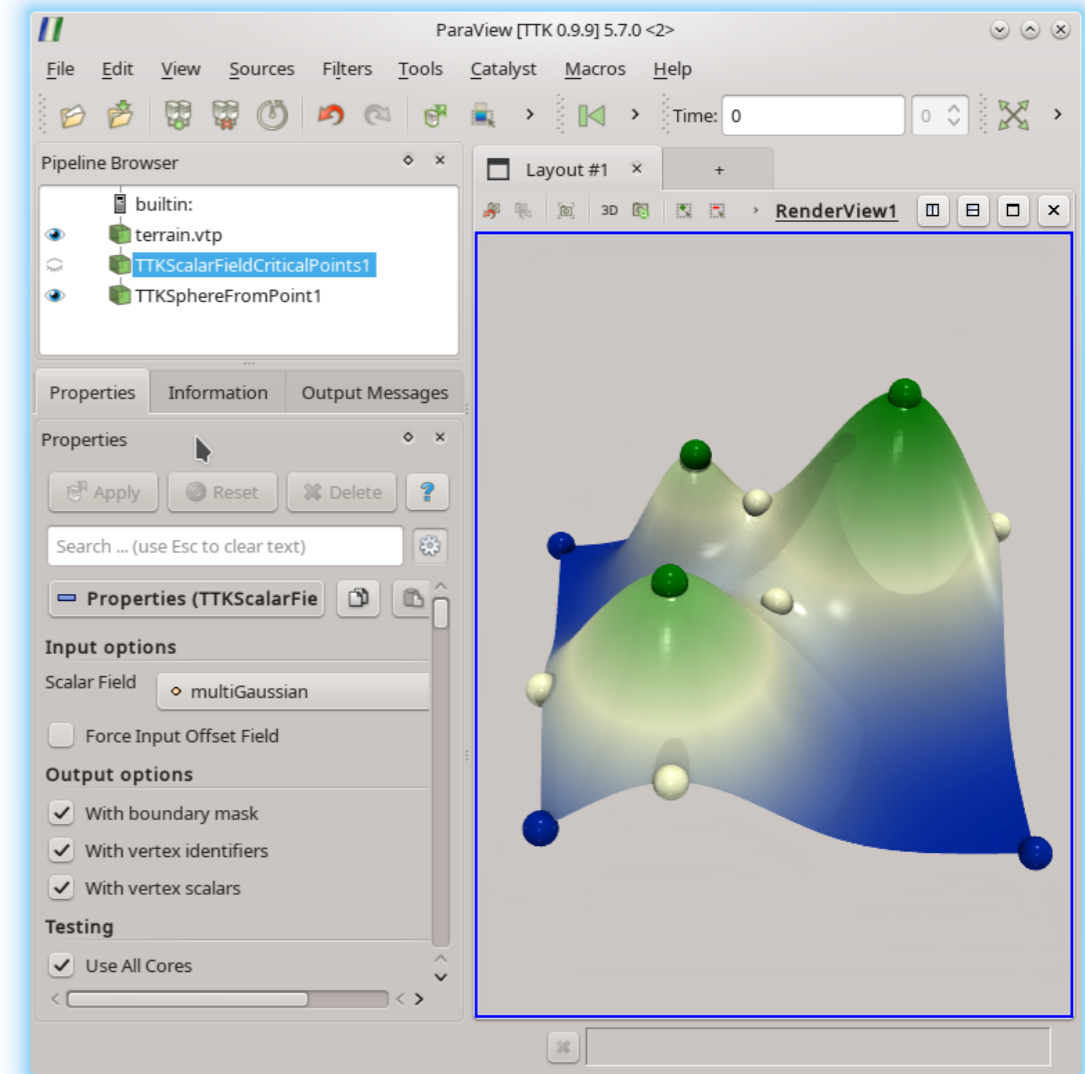
Collaborators and Credits

- Lead developer: Julien Tierny (Sorbonne / CNRS)
 - Public release in IEEE VIS 2017 involved Guillaume Favelier (Sorbonne), Charles Gueunet (Kitware), Michael Michaux (Sorbonne), and Josh
 - Dozens of international contributors to TTK since, mailing list: ttk-users@googlegroups.com



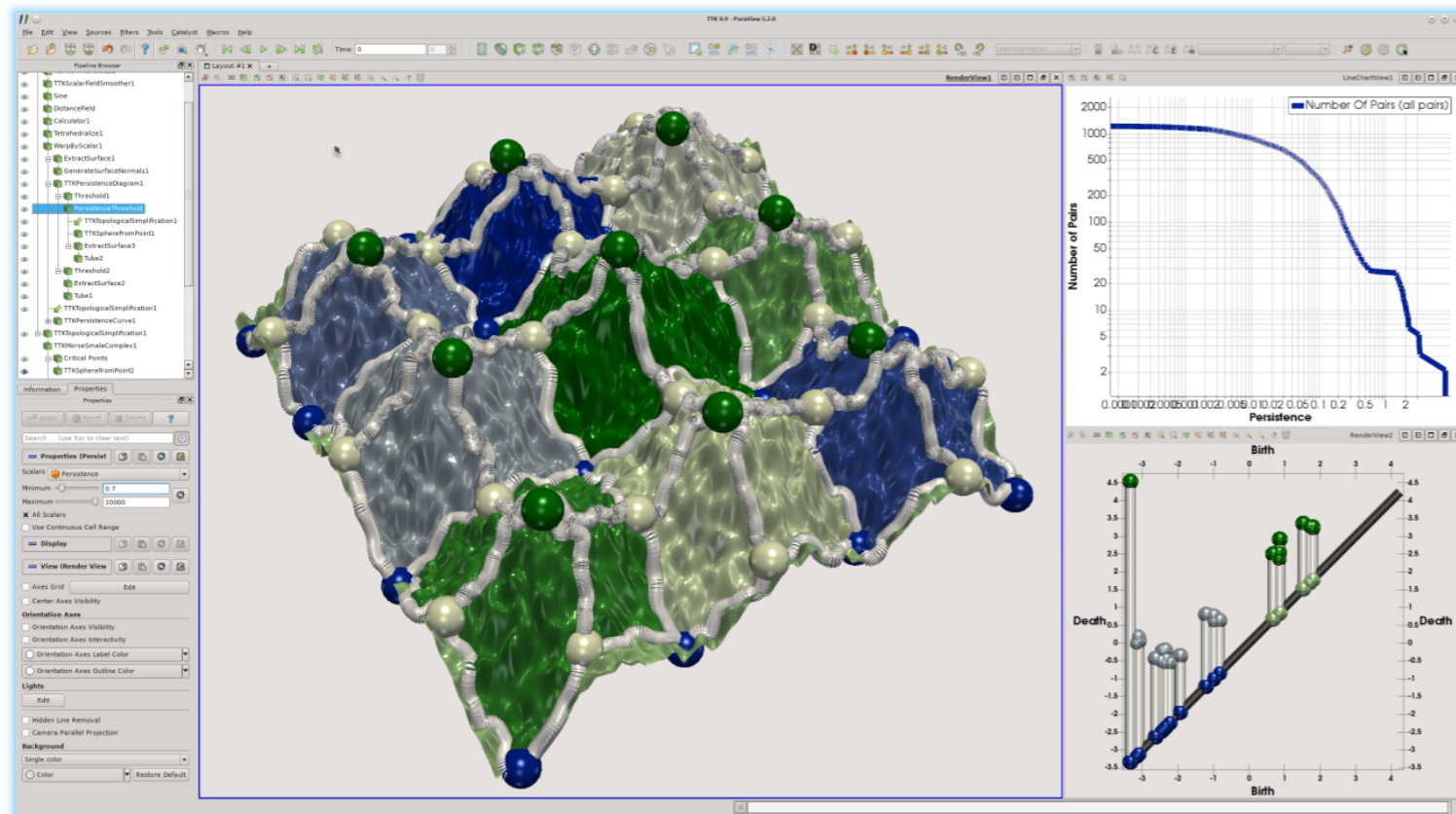
TTK is Built On Top of ParaView

- <https://www.paraview.org/>
- Provides many built-in features:
 - Rich IO support
 - Modern rendering
 - Advanced user interface
 - “Visual” programming
 - Python scripting

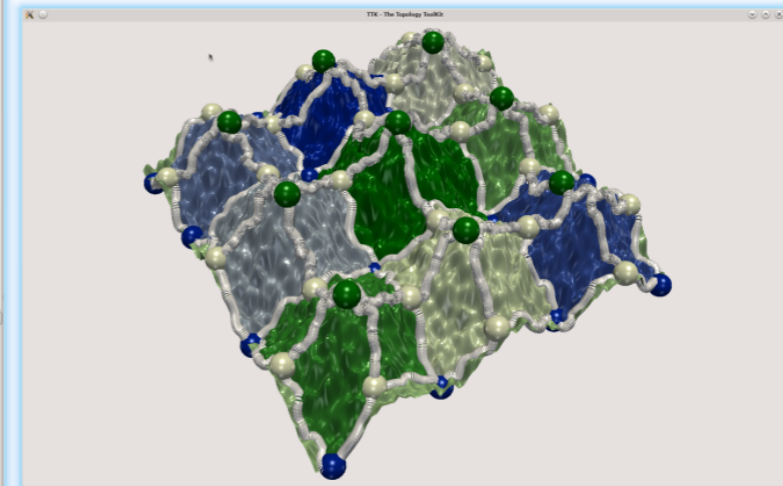
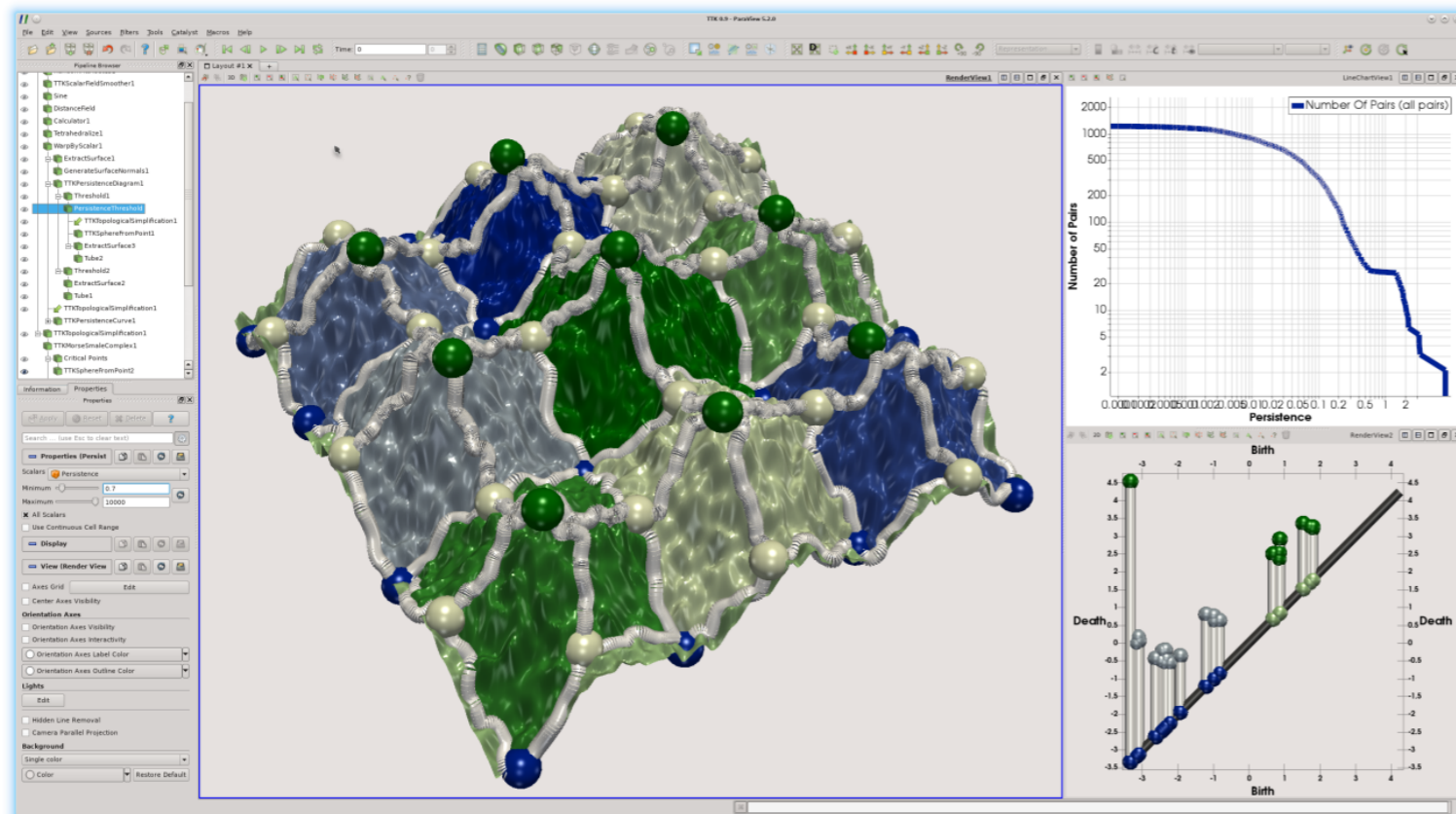


 **ParaView**

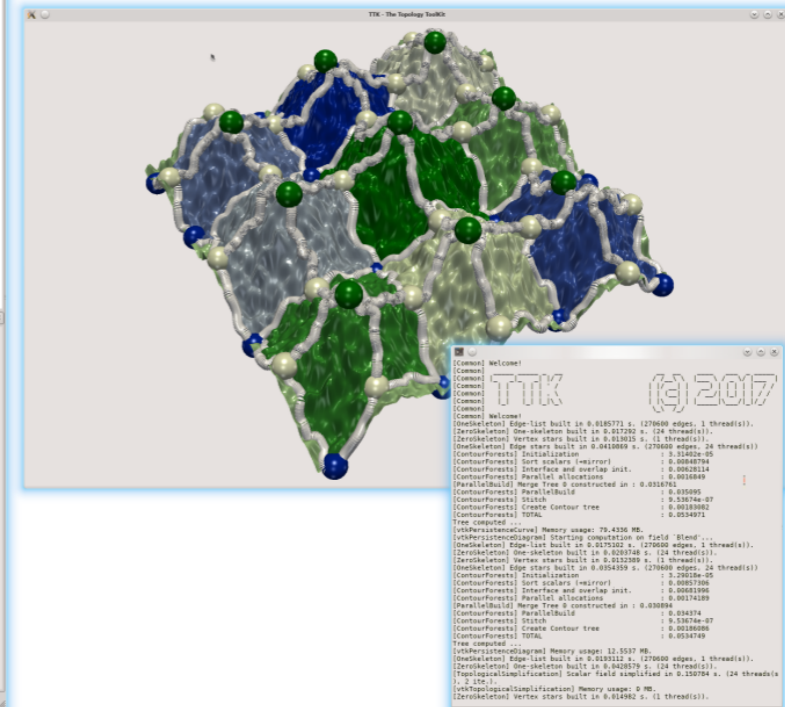
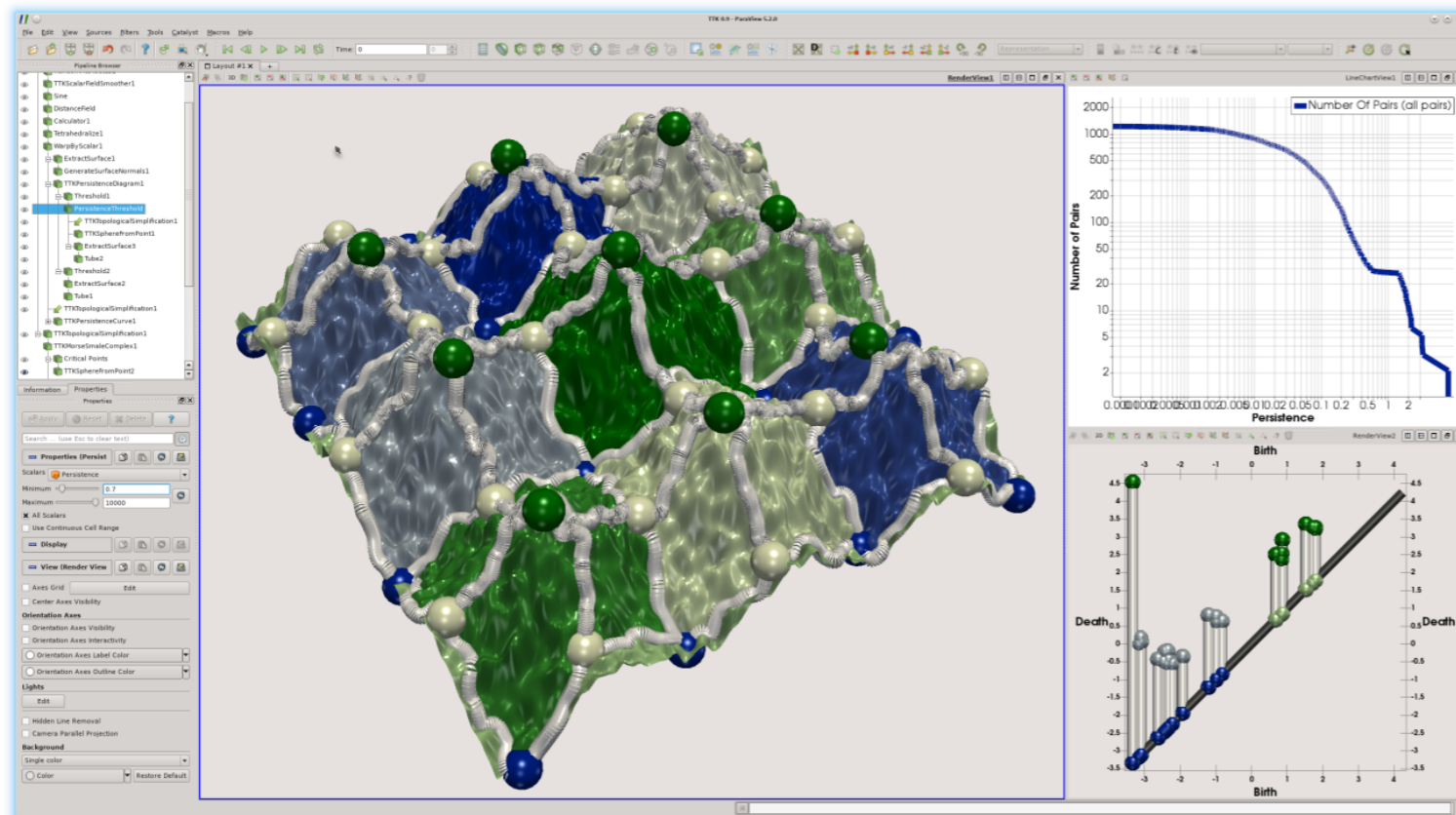
User experience



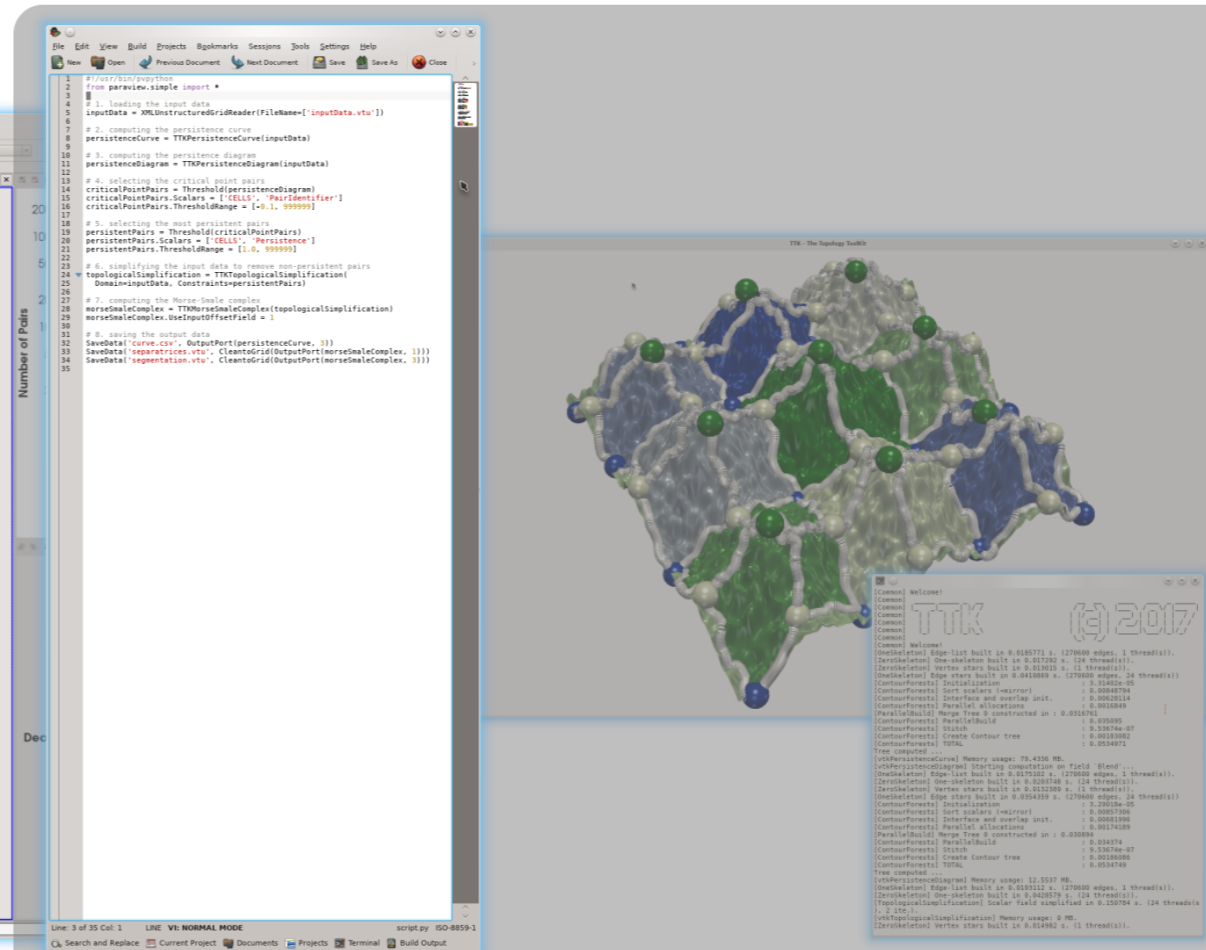
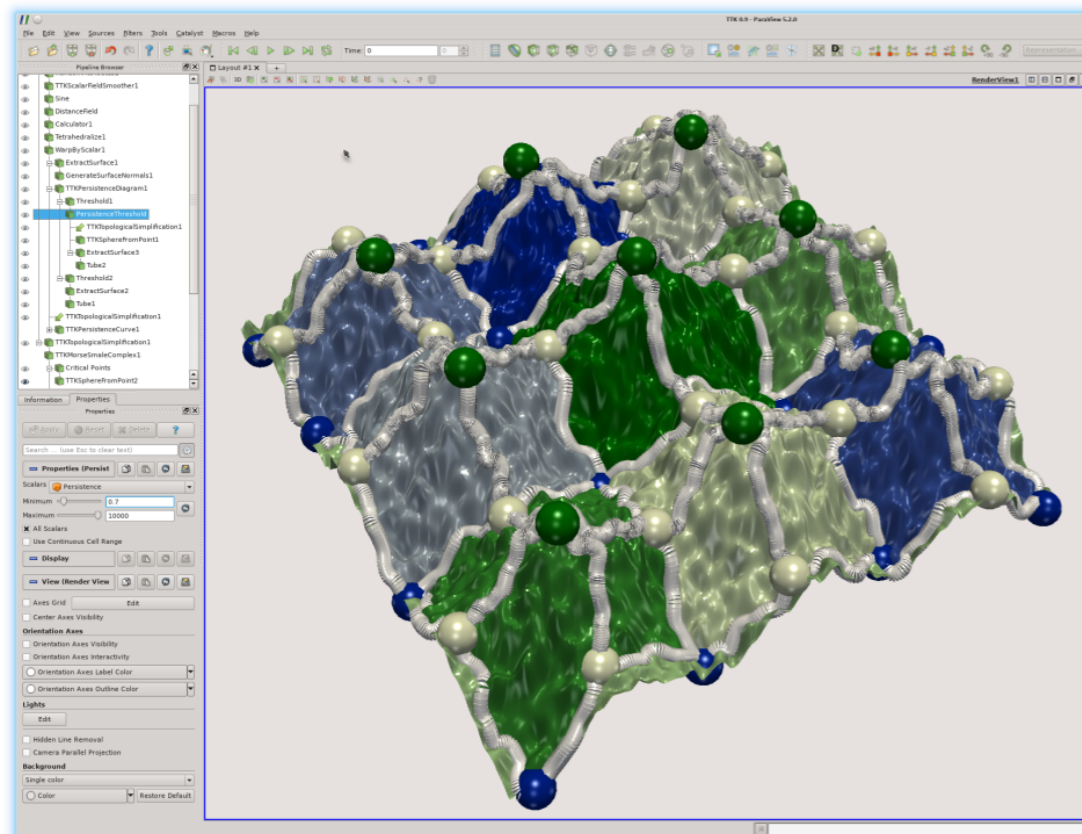
User experience



User experience

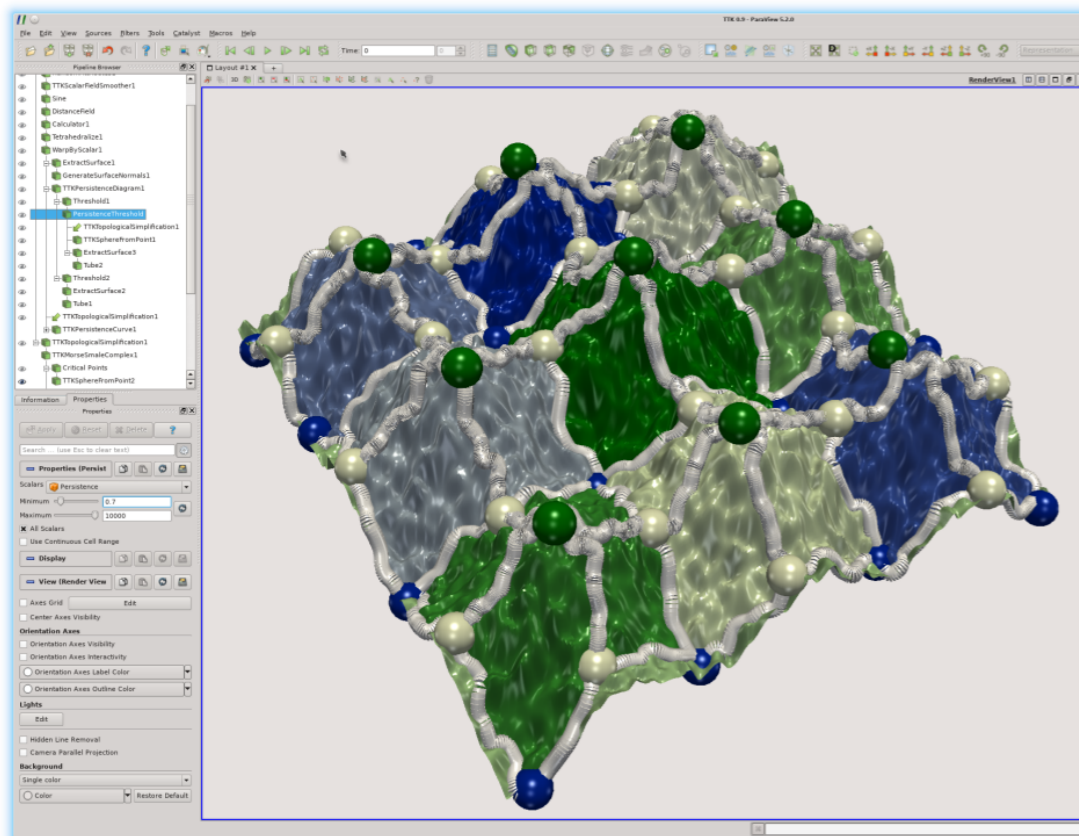


User experience



Python
(21 lines)

User experience



```
1 #!/usr/bin/python
2 from paraview.simple import *
3
4 # 1. loading the input data
5 inputData = XMLUnstructuredGridReader(FileNames=['inputData.vtu'])
6
7 # 2. computing the persistence curve
8 persistenceCurve = TTKPersistenceCurve(inputData)
9
10 # 3. computing the persistence diagram
11 persistenceDiagram = TTKPersistenceDiagram(inputData)
12
13 # 4. selecting the critical point pairs
14 criticalPointPairs = Threshold(persistenceDiagram)
15 criticalPointPairs.Scalars = ['CELLS: "PairIdentifier"']
16 criticalPointPairs.ThresholdRange = [0.1, 0.99999]
17
18 # 5. selecting the most persistent pairs
19 persistentPairs = Threshold(criticalPointPairs)
20 persistentPairs.Scalars = ['CELLS: "Persistence"']
21 persistentPairs.ThresholdRange = [1.0, 999999]
22
23 # 6. simplifying the input data to remove non-persistent pairs
24 topologicalSimplification = TTKTopologicalSimplification(
25     Domain=inputData, Constraints=persistentPairs)
26
27 # 7. computing the Morse-Smale complex
28 morseSmaleComplex = TTKMorseSmaleComplex(topologicalSimplification)
29 morseSmaleComplex.UseInputOffsetField = 1
30
31 # 8. saving the output data
32 SaveData('curve.vtu', OutputPort(persistenceCurve, 3))
33 SaveData('separatrices.vtu', CleanGrid(OutputPort(morseSmaleComplex, 1)))
34 SaveData('separation.vtu', CleanGrid(OutputPort(morseSmaleComplex, 3)))
35
```

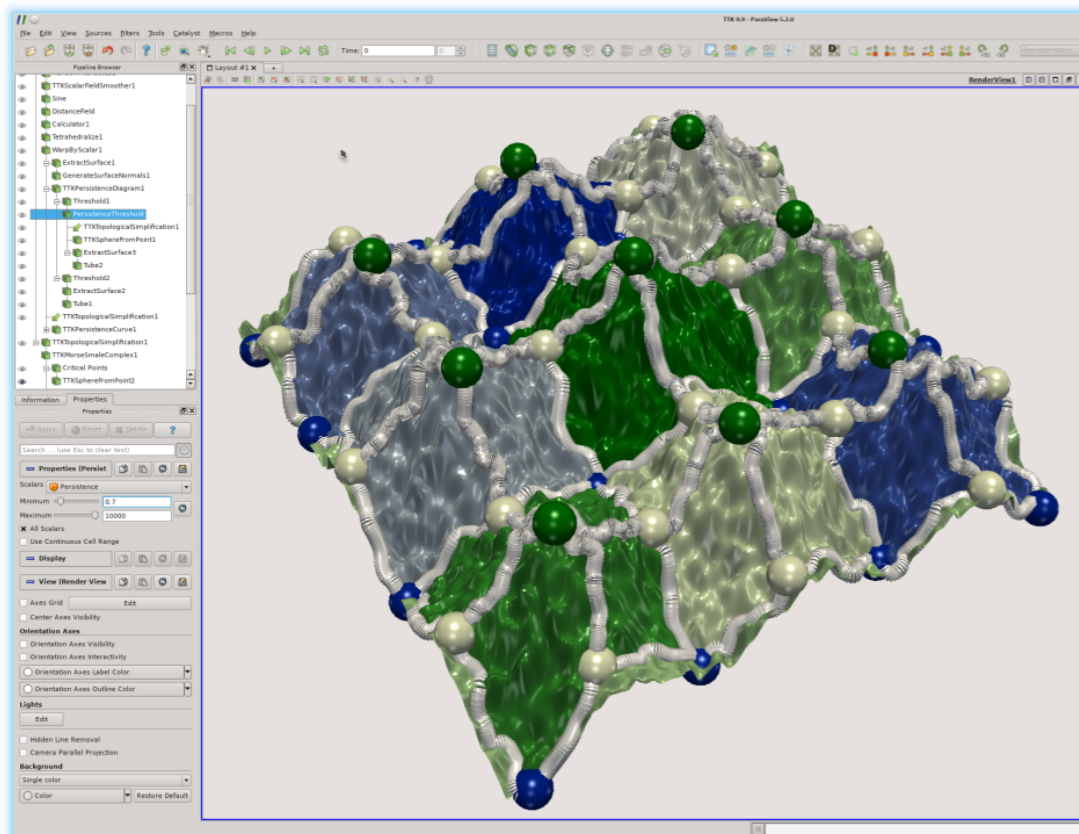
Python
(21 lines)

```
1 #include <vtkMorseSmaleComplex.h>
2 #include <vtkPersistenceCurve.h>
3 #include <vtkPersistenceDiagram.h>
4 #include <vtkTableWriter.h>
5 #include <vtkThreshold.h>
6 #include <vtkTopologicalSimplification.h>
7 #include <vtkXMLUnstructuredGridReader.h>
8 #include <vtkXMLUnstructuredGridWriter.h>
9
10 int main(int argc, char **argv){
11
12     // 1. loading the input data
13     vtkSmartPointer<XMLUnstructuredGridReader> reader =
14     vtkSmartPointer<XMLUnstructuredGridReader>::New();
15     reader->SetFileName(argv[1]);
16
17     // 2. computing the persistence curve
18     vtkSmartPointer<PersistenceCurve> curve =
19     vtkSmartPointer<PersistenceCurve>::New();
20     curve->SetInputConnection(reader->GetOutputPort(1));
21
22     // 3. computing the persistence diagram
23     vtkSmartPointer<PersistenceDiagram> diagram =
24     vtkSmartPointer<PersistenceDiagram>::New();
25     diagram->SetInputConnection(reader->GetOutputPort(1));
26
27     // 4. selecting the critical point pairs
28     vtkSmartPointer<Threshold> criticalPairs =
29     vtkSmartPointer<Threshold>::New();
30     criticalPairs->SetInputConnection(diagram->GetOutputPort(1));
31     criticalPairs->SetInputArrayToProcess(0, 0,
32     vtkObject::FIELD_ASSOCIATION_CELLS, "PairIdentifier");
33     criticalPairs->ThresholdBetween(0.1, 0.99999);
34
35     // 5. selecting the most persistent pairs
36     vtkSmartPointer<Threshold> persistentPairs =
37     vtkSmartPointer<Threshold>::New();
38     persistentPairs->SetInputConnection(criticalPairs->GetOutputPort(1));
39     persistentPairs->SetInputArrayToProcess(0, 0,
40     vtkObject::FIELD_ASSOCIATION_CELLS, "Persistence");
41     persistentPairs->ThresholdBetween(1.0, 999999);
42
43     // 6. simplifying the input data to remove non-persistent pairs
44     vtkSmartPointer<TopologicalSimplification> topologicalSimplification =
45     vtkSmartPointer<TopologicalSimplification>::New();
46     topologicalSimplification->SetInputConnection(0, reader->GetOutputPort(1));
47     topologicalSimplification->SetInputConnection(1,
48     persistentPairs->GetOutputPort(1));
49
50     // 7. computing the Morse-Smale complex
51     vtkSmartPointer<MorseSmaleComplex> morseSmaleComplex =
52     vtkSmartPointer<MorseSmaleComplex>::New();
53     morseSmaleComplex->SetInputConnection(1,
54     topologicalSimplification->GetOutputPort(1));
55     morseSmaleComplex->SetUseInputOffsetScalarField(true);
56
57     // 8. saving the output data
58     vtkSmartPointer<TableWriter> curveWriter =
59     vtkSmartPointer<TableWriter>::New();
60     curveWriter->SetInputConnection(curve->GetOutputPort(1));
61     curveWriter->SetFileName("curve.vtu");
62     curveWriter->Write();
63
64     vtkSmartPointer<XMLUnstructuredGridWriter> separator =
65     vtkSmartPointer<XMLUnstructuredGridWriter>::New();
66     separator->SetInputConnection(morseSmaleComplex->GetOutputPort(1));
67     separator->SetFileName("separatrices.vtu");
68     separator->Write();
69
70     vtkSmartPointer<XMLUnstructuredGridWriter> separator =
71     vtkSmartPointer<XMLUnstructuredGridWriter>::New();
72     separator->SetInputConnection(morseSmaleComplex->GetOutputPort(3));
73     separator->SetFileName("separation.vtu");
74     separator->Write();
75
76     return 0;
77 }
```

VTK/C++
(37 lines)



User experience



```
1 #!/usr/bin/python
2 from paraview.simple import *
3
4 # 1. loading the input data
5 inputData = XMLUnstructuredGridReader(FileNames=['inputData.vtu'])
6
7 # 2. computing the persistence curve
8 persistenceCurve = TTKPersistenceCurve(inputData)
9 persistenceDiagram = TTKPersistenceDiagram(inputData)
10
11 # 3. computing the persistence diagram
12 criticalPointPairs = Threshold(persistenceDiagram)
13 criticalPointPairs.Scalars = ['CELLS', 'PairIdentifier']
14 criticalPointPairs.ThresholdRange = [-0.1, 999999]
15
16 # 4. selecting the critical point pairs
17 persistentPairs = Threshold(criticalPointPairs)
18 persistentPairs.Scalars = ['CELLS', 'Persistence']
19 persistentPairs.ThresholdRange = [1.0, 999999]
20
21 # 5. selecting the most persistent pairs
22 topologicalSimplification = TTKTopologicalSimplification(
23     Domain=inputData, Constraints=persistentPairs)
24
25 # 6. simplifying the input data to remove non-persistent pairs
26 morseSmaleComplex = TTKMorseSmaleComplex(topologicalSimplification)
27 morseSmaleComplex.UseInputOffsetField = 1
28
29 # 7. computing the Morse-Smale complex
30 saveData('curve.csv', OutputPort(persistenceCurve, 3))
31 saveData('separatrices.vtu', CleanGrid(OutputPort(morseSmaleComplex, 1)))
32 saveData('separation.vtu', CleanGrid(OutputPort(morseSmaleComplex, 3)))
33
34 # 8. saving the output data
35
```

Python
(21 lines)

```
1 #include <vtkMorseSmaleComplex.h>
2 #include <vtkPersistenceCurve.h>
3 #include <vtkTableWriter.h>
4 #include <vtkThreshold.h>
5 #include <vtkXMLUnstructuredGridReader.h>
6 #include <vtkXMLUnstructuredGridWriter.h>
7
8 int main(int argc, char**argv)
9 {
10     // 1. loading the input data
11     vtkXMLUnstructuredGridReader* reader =
12         vtkXMLUnstructuredGridReader::New();
13     reader->SetFileName(argv[1]);
14
15     // 2. computing the persistence curve
16     vtkPersistenceCurve* curve =
17         vtkPersistenceCurve::New();
18     curve->SetInputConnection(reader->GetOutputPort(1));
19
20     // 3. computing the persistence diagram
21     vtkPersistenceDiagram* diagram =
22         vtkPersistenceDiagram::New();
23     diagram->SetInputConnection(reader->GetOutputPort(1));
24
25     // 4. selecting the critical point pairs
26     vtkThreshold* threshold =
27         vtkThreshold::New();
28     threshold->SetInputConnection(diagram->GetOutputPort(1));
29     threshold->SetInputArrayToProcess(0, 0,
30         vtkDataObject::FIELD_ASSOCIATION_CELLS, 'PairIdentifier');
31     threshold->SetThresholdRange(-0.1, 999999);
32
33     // 5. selecting the most persistent pairs
34     vtkThreshold* threshold2 =
35         vtkThreshold::New();
36     threshold2->SetInputConnection(threshold->GetOutputPort(1));
37     threshold2->SetInputArrayToProcess(0, 0,
38         vtkDataObject::FIELD_ASSOCIATION_CELLS, 'Persistence');
39     threshold2->SetThresholdRange(1.0, 999999);
40
41     // 6. simplifying the input data to remove non-persistent pairs
42     vtkTopologicalSimplification* topologicalSimplification =
43         vtkTopologicalSimplification::New();
44     topologicalSimplification->SetInputConnection(
45         threshold2->GetOutputPort(1));
46     topologicalSimplification->SetInputConnection(0,
47         reader->GetOutputPort(1));
48     topologicalSimplification->SetOutputPort(1);
49
50     // 7. computing the Morse-Smale complex
51     vtkMorseSmaleComplex* morseSmaleComplex =
52         vtkMorseSmaleComplex::New();
53     morseSmaleComplex->SetInputConnection(
54         topologicalSimplification->GetOutputPort(1));
55     morseSmaleComplex->SetUseInputOffsetField(true);
56
57     // 8. saving the output data
58     vtkTableWriter* curveWriter =
59         vtkTableWriter::New();
60     curveWriter->SetInputConnection(curve->GetOutputPort(1));
61     curveWriter->SetFileName('curve.csv');
62     curveWriter->Write();
63
64     vtkXMLUnstructuredGridWriter* separator =
65         vtkXMLUnstructuredGridWriter::New();
66     separator->SetInputConnection(morseSmaleComplex->GetOutputPort(1));
67     separator->SetFileName('separatrices.vtu');
68     separator->Write();
69
70     vtkXMLUnstructuredGridWriter* separation =
71         vtkXMLUnstructuredGridWriter::New();
72     separation->SetInputConnection(morseSmaleComplex->GetOutputPort(3));
73     separation->SetFileName('separation.vtu');
74     separation->Write();
75
76     return 0;
77 }
```

VTK/C++
(37 lines)

```
1 #include <vtkMorseSmaleComplex.h>
2 #include <vtkPersistenceCurve.h>
3 #include <vtkTableWriter.h>
4 #include <vtkThreshold.h>
5 #include <vtkXMLUnstructuredGridReader.h>
6 #include <vtkXMLUnstructuredGridWriter.h>
7
8 int main(int argc, char**argv)
9 {
10     // 1. loading the input data
11     vtkXMLUnstructuredGridReader* reader =
12         vtkXMLUnstructuredGridReader::New();
13     reader->SetFileName(argv[1]);
14
15     // 2. computing the persistence curve
16     vtkPersistenceCurve* curve =
17         vtkPersistenceCurve::New();
18     curve->SetInputConnection(reader->GetOutputPort(1));
19     curve->SetInputArrayToProcess(0, 0,
20         vtkDataObject::FIELD_ASSOCIATION_CELLS, 'PairIdentifier');
21     curve->SetInputArrayToProcess(1, 0,
22         vtkDataObject::FIELD_ASSOCIATION_CELLS, 'Persistence');
23     curve->SetInputOffsetField('offsets');
24     curve->SetOutputPort(1);
25     curve->Execute();
26
27     // 3. computing the persistence diagram
28     vtkPersistenceDiagram* diagram =
29         vtkPersistenceDiagram::New();
30     diagram->SetInputConnection(reader->GetOutputPort(1));
31     diagram->SetInputArrayToProcess(0, 0,
32         vtkDataObject::FIELD_ASSOCIATION_CELLS, 'PairIdentifier');
33     diagram->SetInputArrayToProcess(1, 0,
34         vtkDataObject::FIELD_ASSOCIATION_CELLS, 'Persistence');
35     diagram->SetInputOffsetField('offsets');
36     diagram->SetOutputPort(1);
37     diagram->Execute();
38
39     // 4. selecting the critical point pairs
40     vtkThreshold* threshold =
41         vtkThreshold::New();
42     threshold->SetInputConnection(diagram->GetOutputPort(1));
43     threshold->SetInputArrayToProcess(0, 0,
44         vtkDataObject::FIELD_ASSOCIATION_CELLS, 'PairIdentifier');
45     threshold->SetInputArrayToProcess(1, 0,
46         vtkDataObject::FIELD_ASSOCIATION_CELLS, 'Persistence');
47     threshold->SetThresholdRange(-0.1, 999999);
48
49     // 5. selecting the most persistent pairs
50     vtkThreshold* threshold2 =
51         vtkThreshold::New();
52     threshold2->SetInputConnection(threshold->GetOutputPort(1));
53     threshold2->SetInputArrayToProcess(0, 0,
54         vtkDataObject::FIELD_ASSOCIATION_CELLS, 'Persistence');
55     threshold2->SetThresholdRange(1.0, 999999);
56
57     // 6. simplifying the input data to remove non-persistent pairs
58     vtkTopologicalSimplification* topologicalSimplification =
59         vtkTopologicalSimplification::New();
60     topologicalSimplification->SetInputConnection(
61         threshold2->GetOutputPort(1));
62     topologicalSimplification->SetInputConnection(0,
63         reader->GetOutputPort(1));
64     topologicalSimplification->SetOutputPort(1);
65
66     // 7. computing the Morse-Smale complex
67     vtkMorseSmaleComplex* morseSmaleComplex =
68         vtkMorseSmaleComplex::New();
69     morseSmaleComplex->SetInputConnection(
70         topologicalSimplification->GetOutputPort(1));
71     morseSmaleComplex->SetUseInputOffsetField(true);
72
73     // 8. saving the output data
74     vtkTableWriter* curveWriter =
75         vtkTableWriter::New();
76     curveWriter->SetInputConnection(curve->GetOutputPort(1));
77     curveWriter->SetFileName('curve.csv');
78     curveWriter->Write();
79
80     vtkXMLUnstructuredGridWriter* separator =
81         vtkXMLUnstructuredGridWriter::New();
82     separator->SetInputConnection(morseSmaleComplex->GetOutputPort(1));
83     separator->SetFileName('separatrices.vtu');
84     separator->Write();
85
86     vtkXMLUnstructuredGridWriter* separation =
87         vtkXMLUnstructuredGridWriter::New();
88     separation->SetInputConnection(morseSmaleComplex->GetOutputPort(3));
89     separation->SetFileName('separation.vtu');
90     separation->Write();
91
92     return 0;
93 }
```

Pure C++
(101 lines)

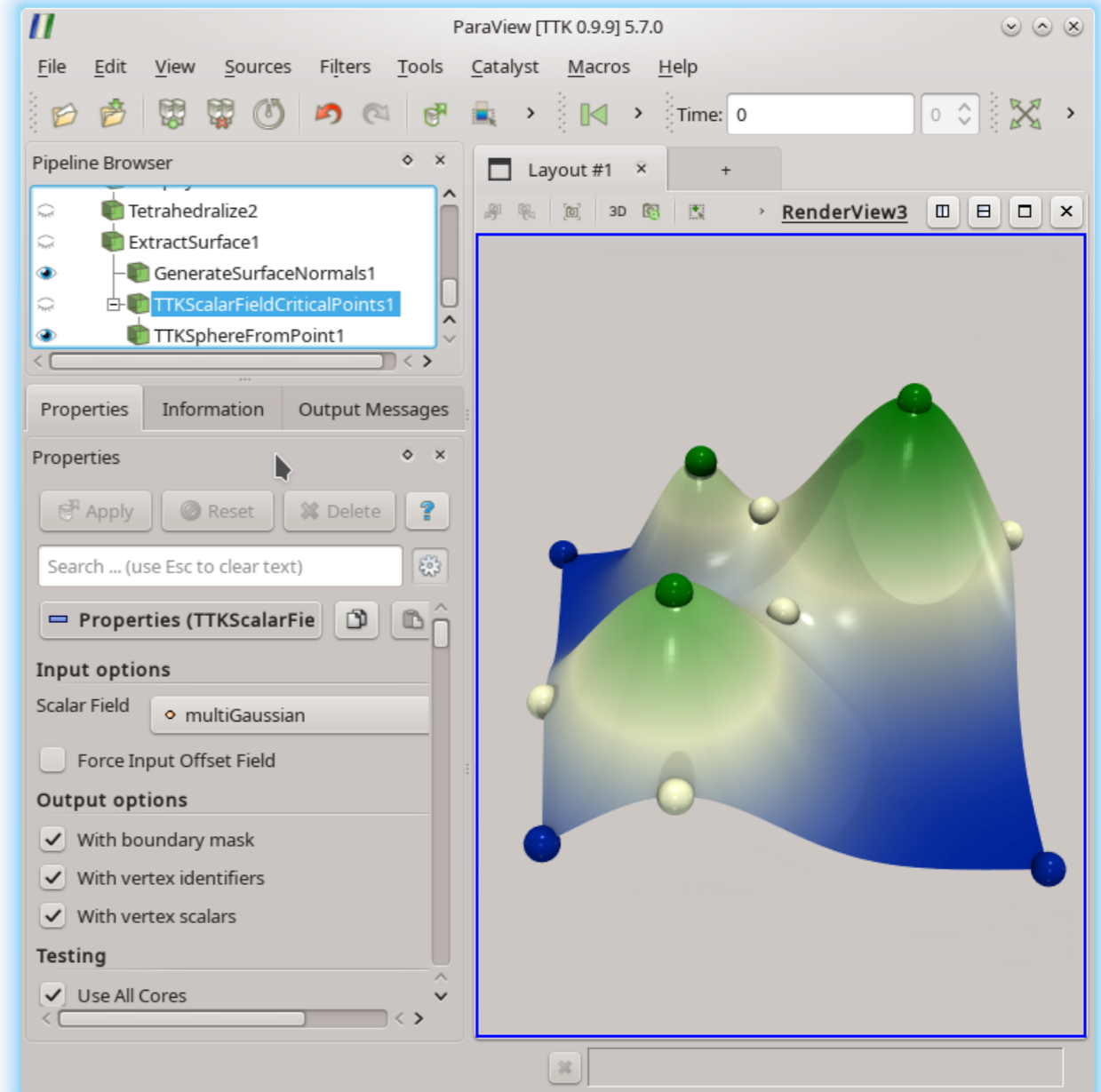


Part 2: A Tour of Topological Features of Scalar Data in TTK

<https://topology-tool-kit.github.io/>

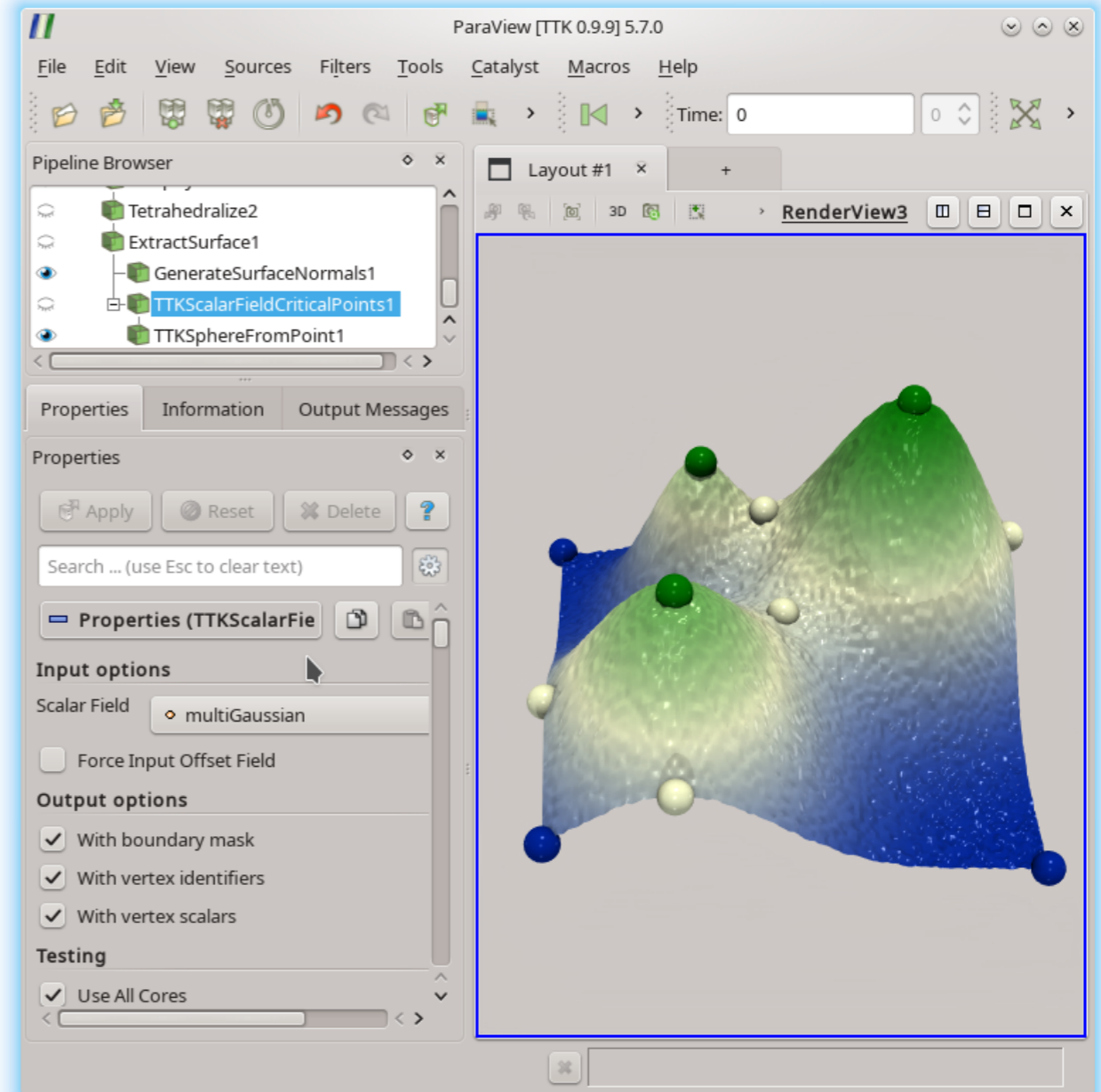
Critical points

- Module
 - **ScalarFieldCriticalPoints**
 - Extract the singularities and their index
- Algorithm
 - Banchoff, J. Diff. Geom. 1967
 - Linear time, trivially parallel
- Output
 - Points with index and vertex IDs



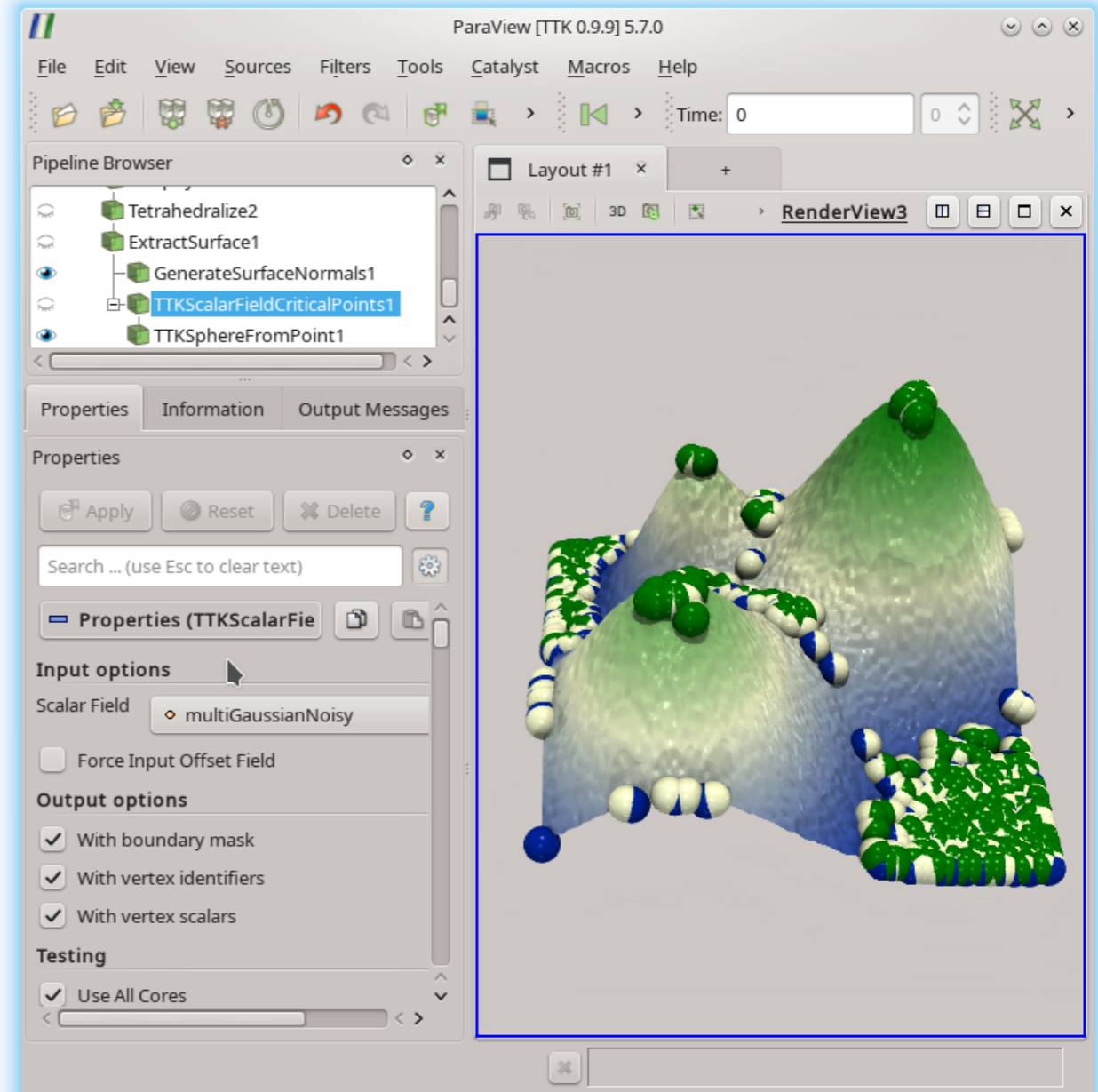
Critical points

- Module
 - **ScalarFieldCriticalPoints**
 - Extract the singularities and their index
- Algorithm
 - Banchoff, J. Diff. Geom. 1967
 - Linear time, trivially parallel
- Output
 - Points with index and vertex IDs



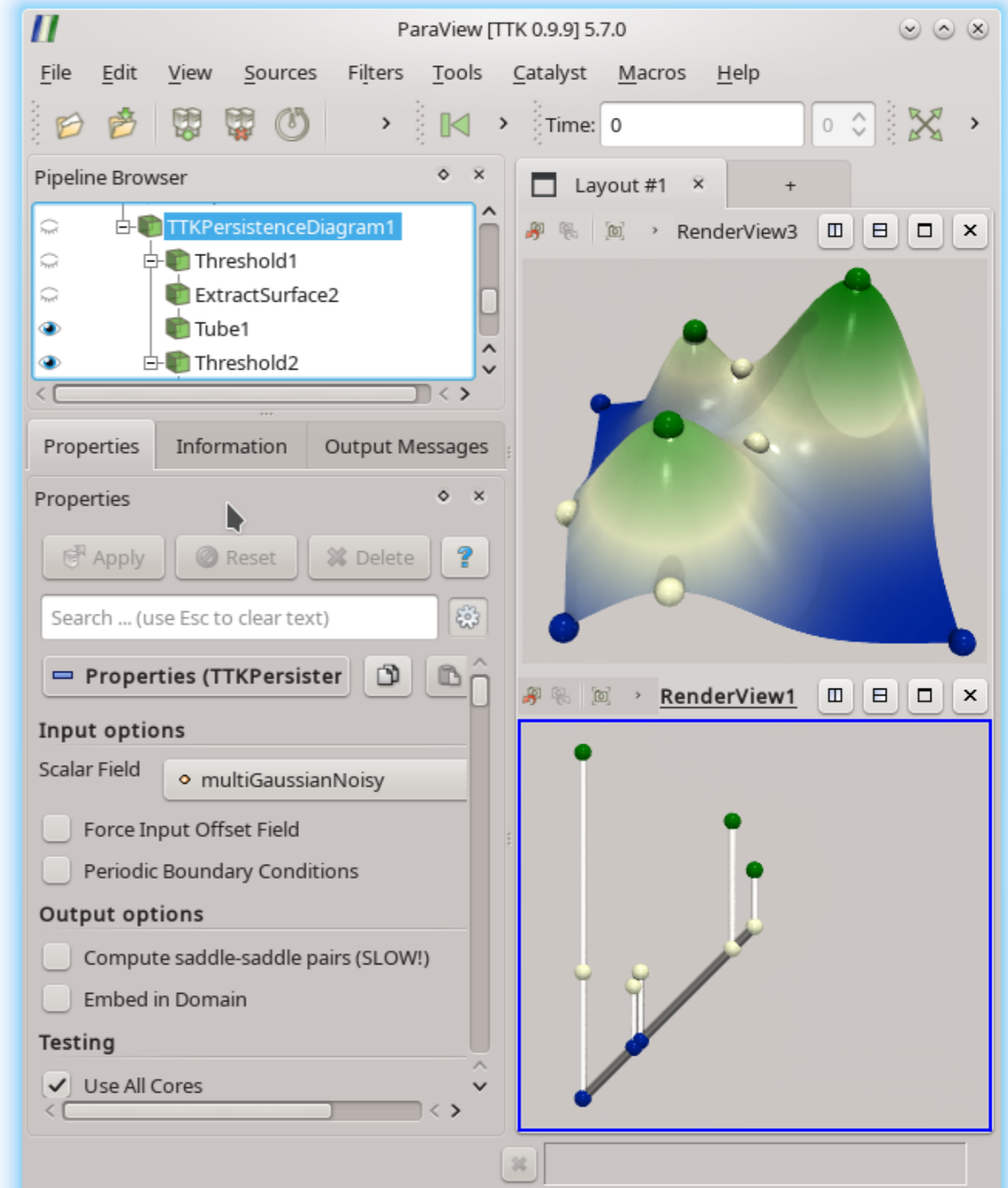
Critical points

- Module
 - **ScalarFieldCriticalPoints**
 - Extract the singularities and their index
- Algorithm
 - Banchoff, J. Diff. Geom. 1967
 - Linear time, trivially parallel
- Output
 - Points with index and vertex IDs



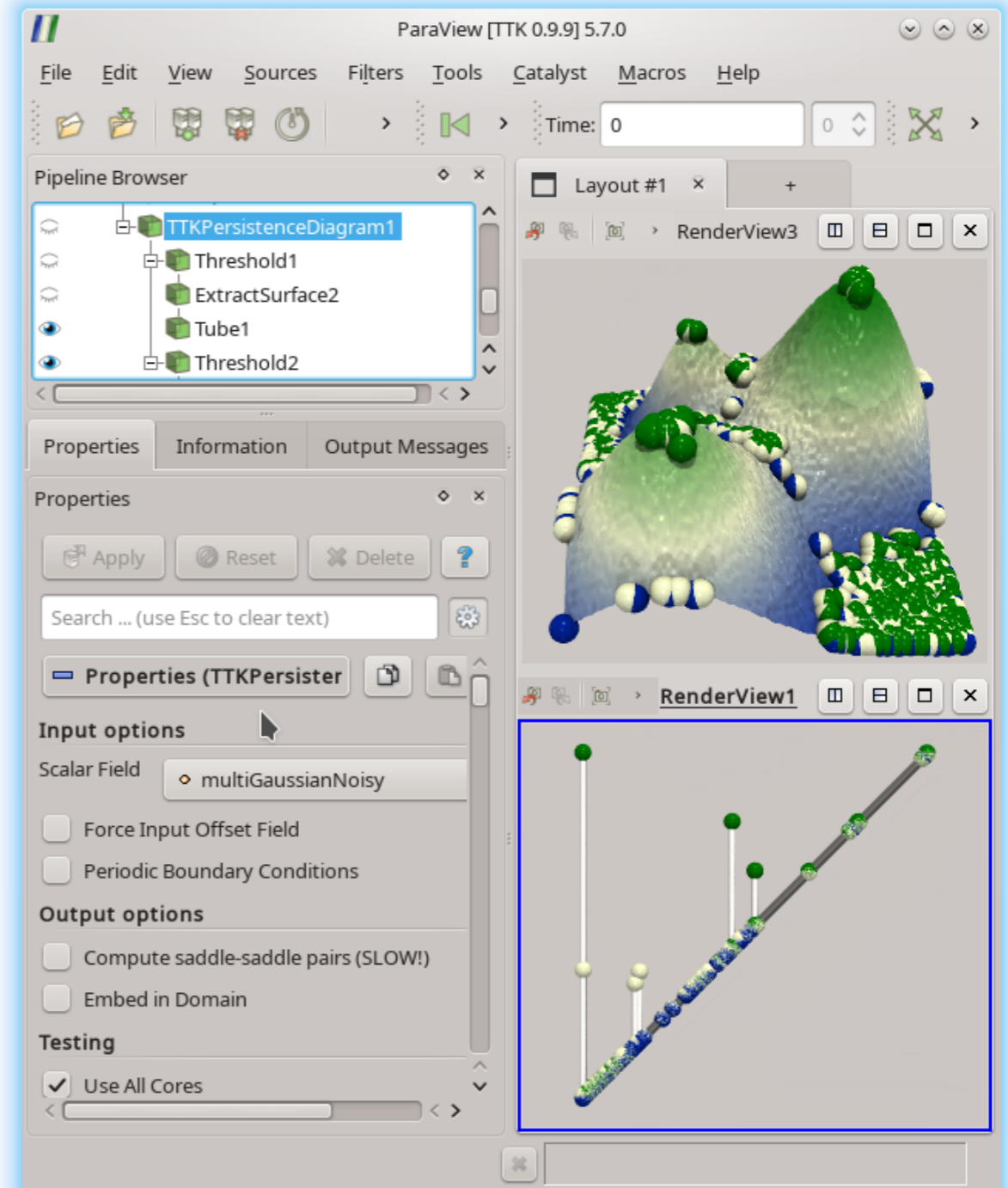
Persistence diagrams

- Module
 - PersistenceDiagram
 - PersistenceCurve
- Algorithms
 - Extremum/saddle pairs
 - Gueunet et al. IEEE TPDS 2019
 - Linearithmic time, efficient parallelization
 - Saddle/saddle pairs w/ saddle connectors
- Output
 - One curve per pair type



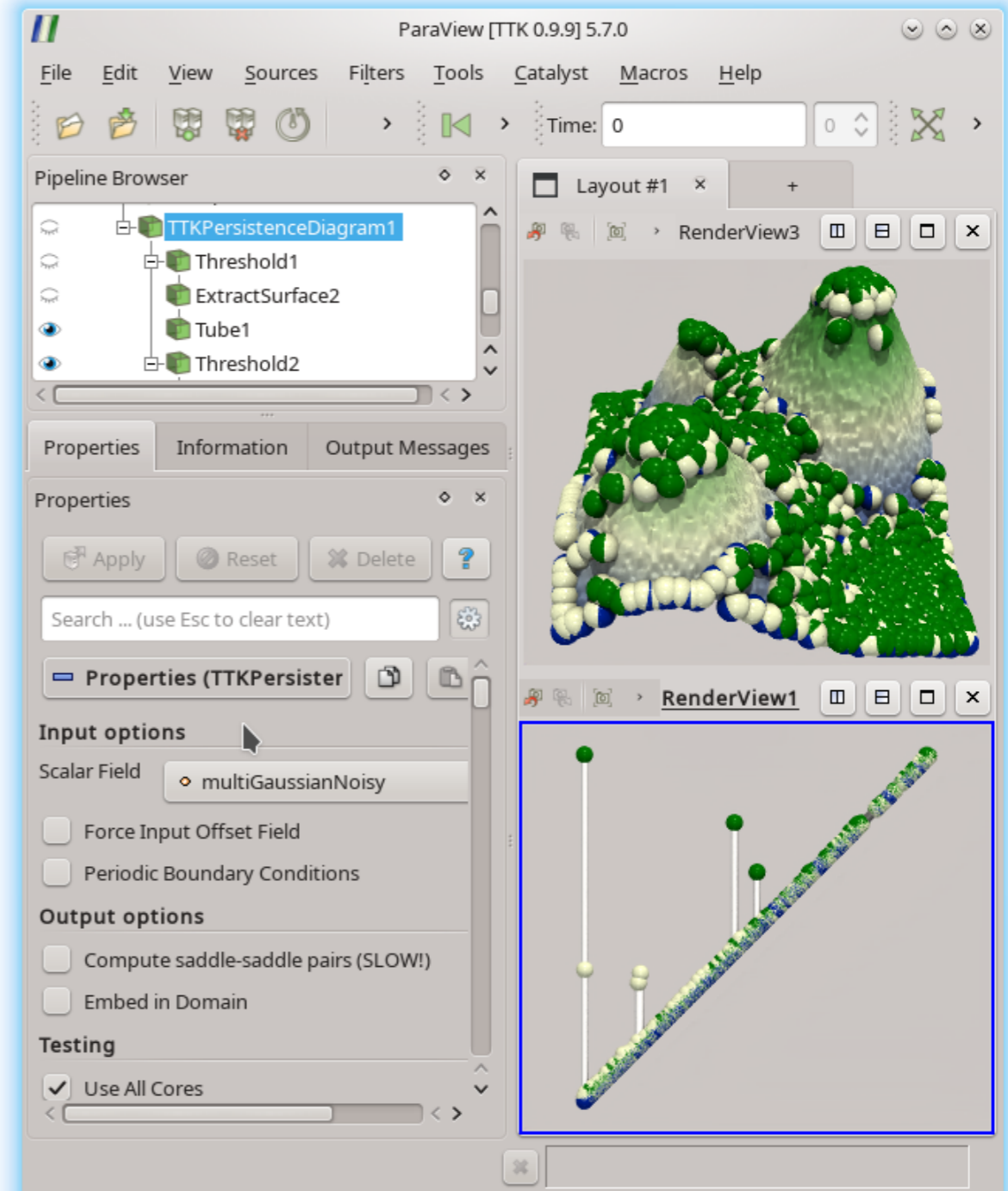
Persistence diagrams

- Module
 - PersistenceDiagram
 - PersistenceCurve
- Algorithms
 - Extremum/saddle pairs
 - Gueunet et al. IEEE TPDS 2019
 - Linearithmic time, efficient parallelization
 - Saddle/saddle pairs w/ saddle connectors
- Output
 - One curve per pair type



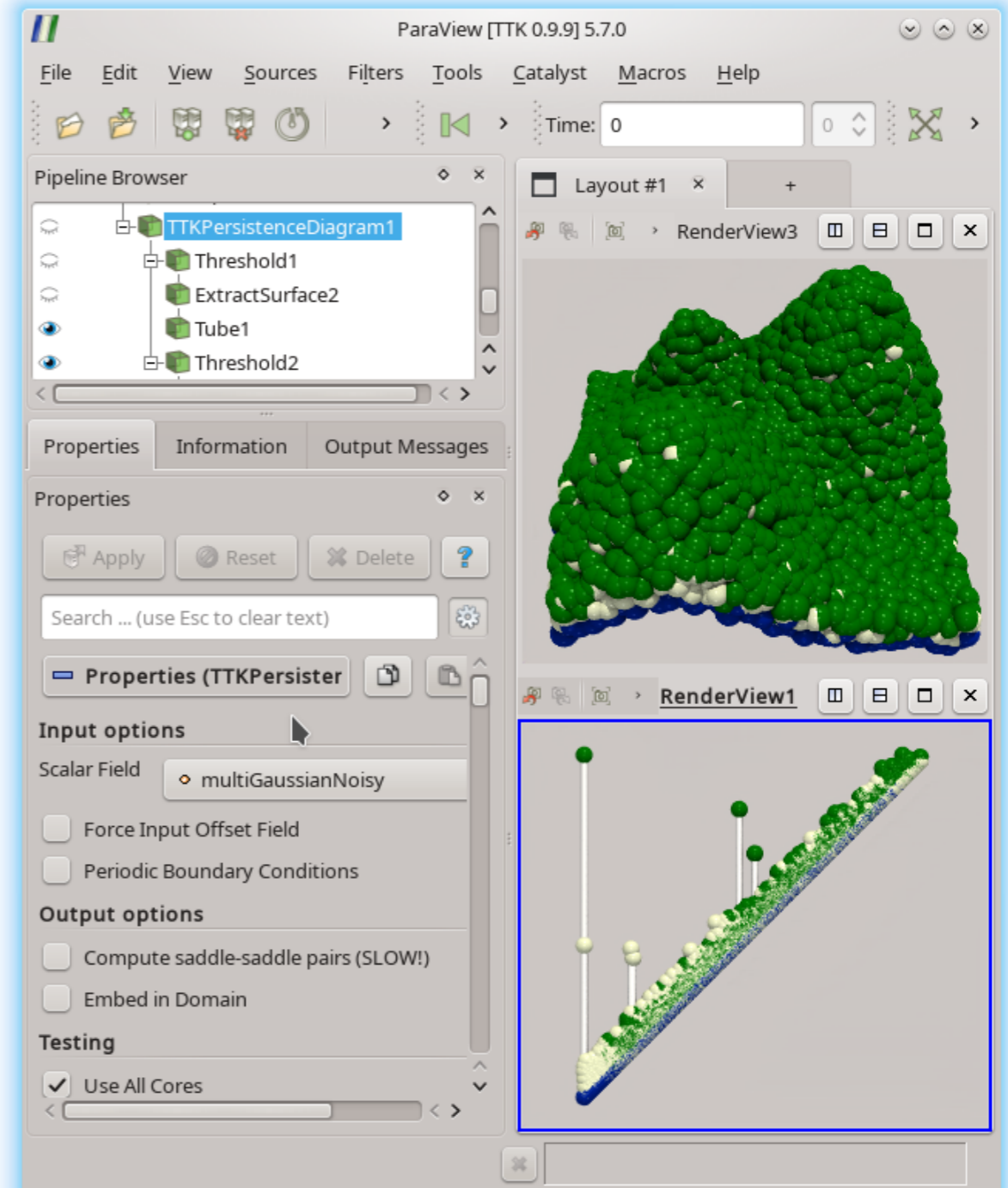
Persistence diagrams

- Module
 - PersistenceDiagram
 - PersistenceCurve
- Algorithms
 - Extremum/saddle pairs
 - Gueunet et al. IEEE TPDS 2019
 - Linearithmic time, efficient parallelization
 - Saddle/saddle pairs w/ saddle connectors
- Output
 - One curve per pair type



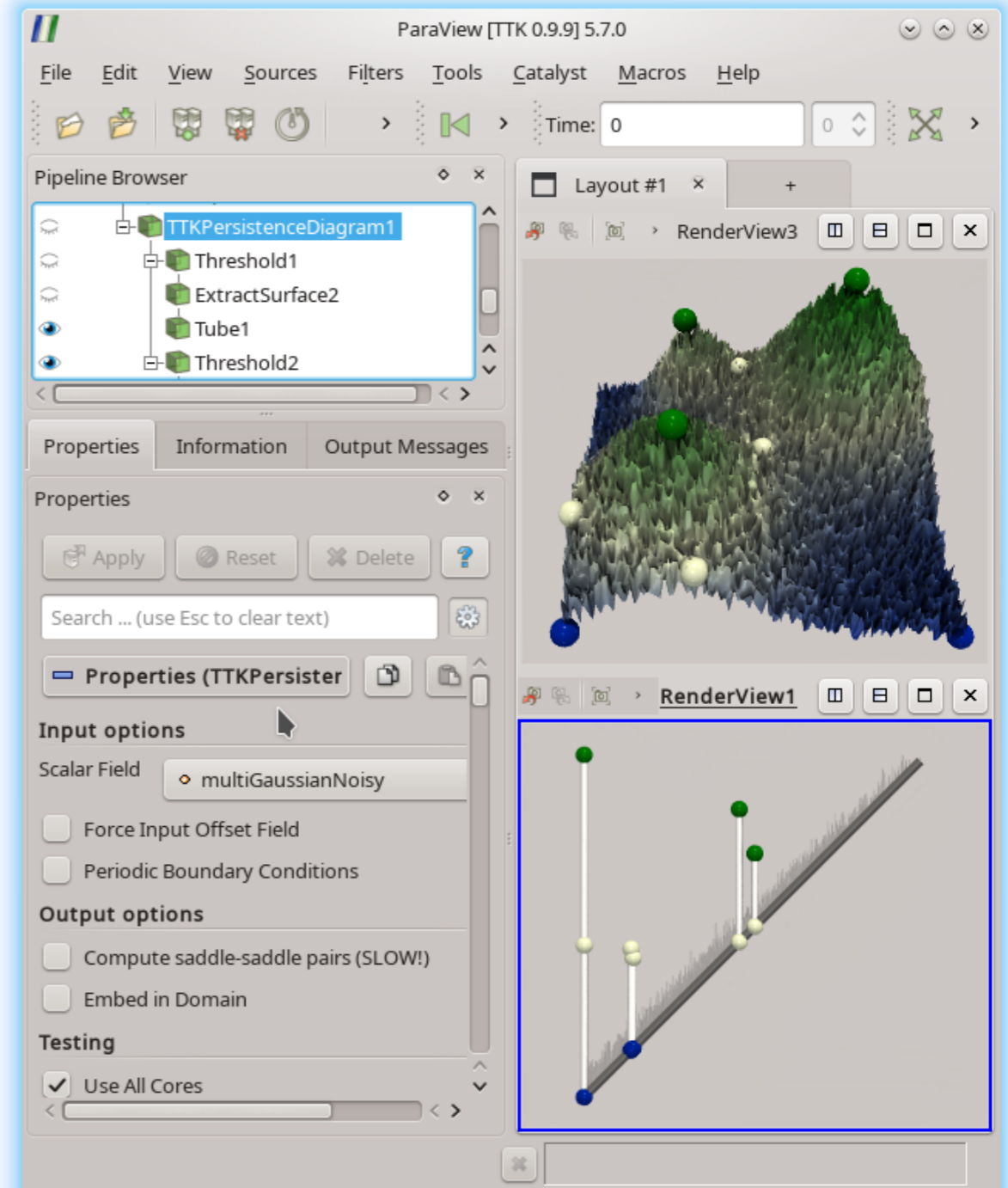
Persistence diagrams

- Module
 - PersistenceDiagram
 - PersistenceCurve
- Algorithms
 - Extremum/saddle pairs
 - Gueunet et al. IEEE TPDS 2019
 - Linearithmic time, efficient parallelization
 - Saddle/saddle pairs w/ saddle connectors
- Output
 - One curve per pair type



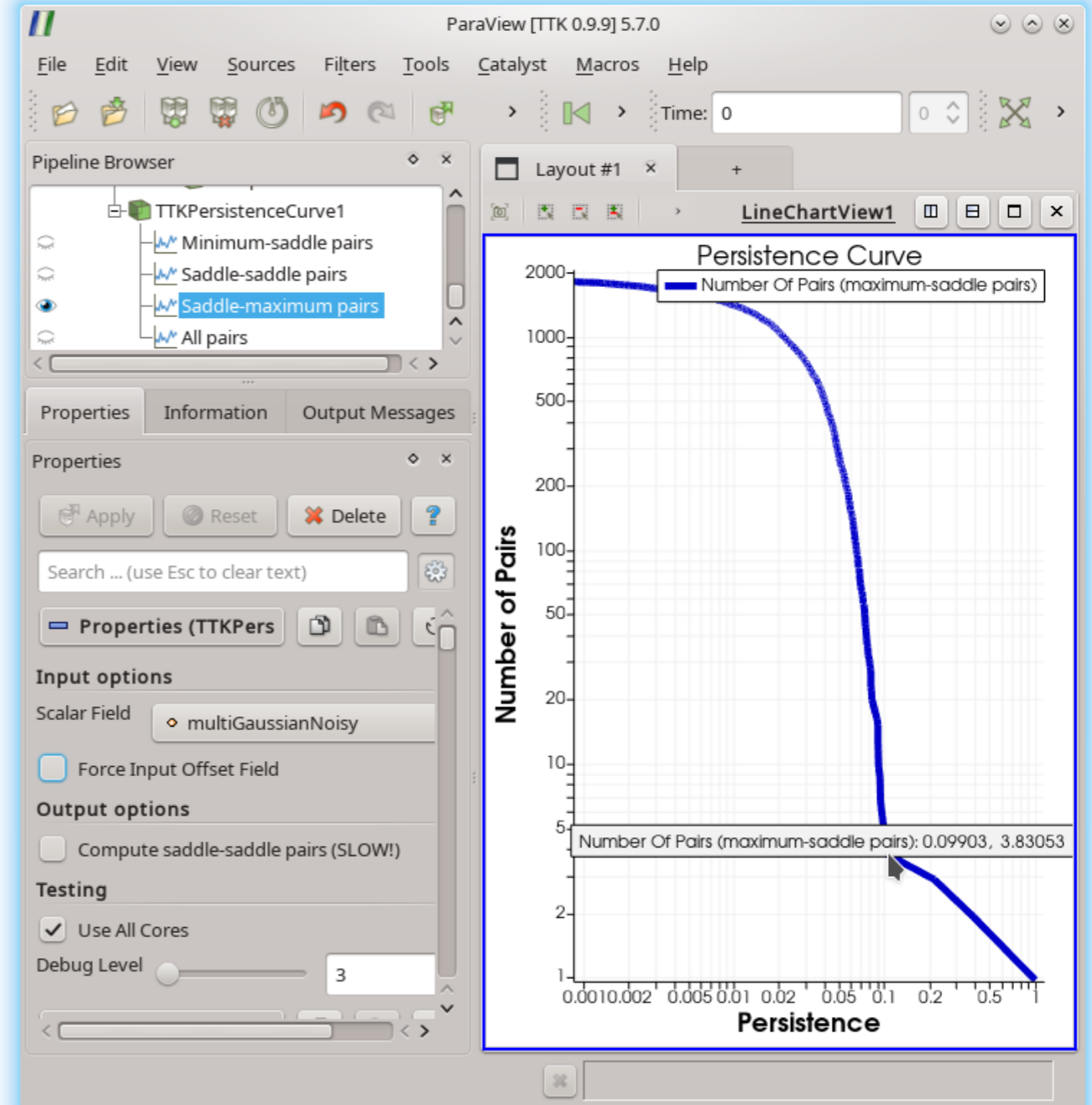
Persistence diagrams

- Module
 - PersistenceDiagram
 - PersistenceCurve
- Algorithms
 - Extremum/saddle pairs
 - Gueunet et al. IEEE TPDS 2019
 - Linearithmic time, efficient parallelization
 - Saddle/saddle pairs w/ saddle connectors
- Output
 - One curve per pair type



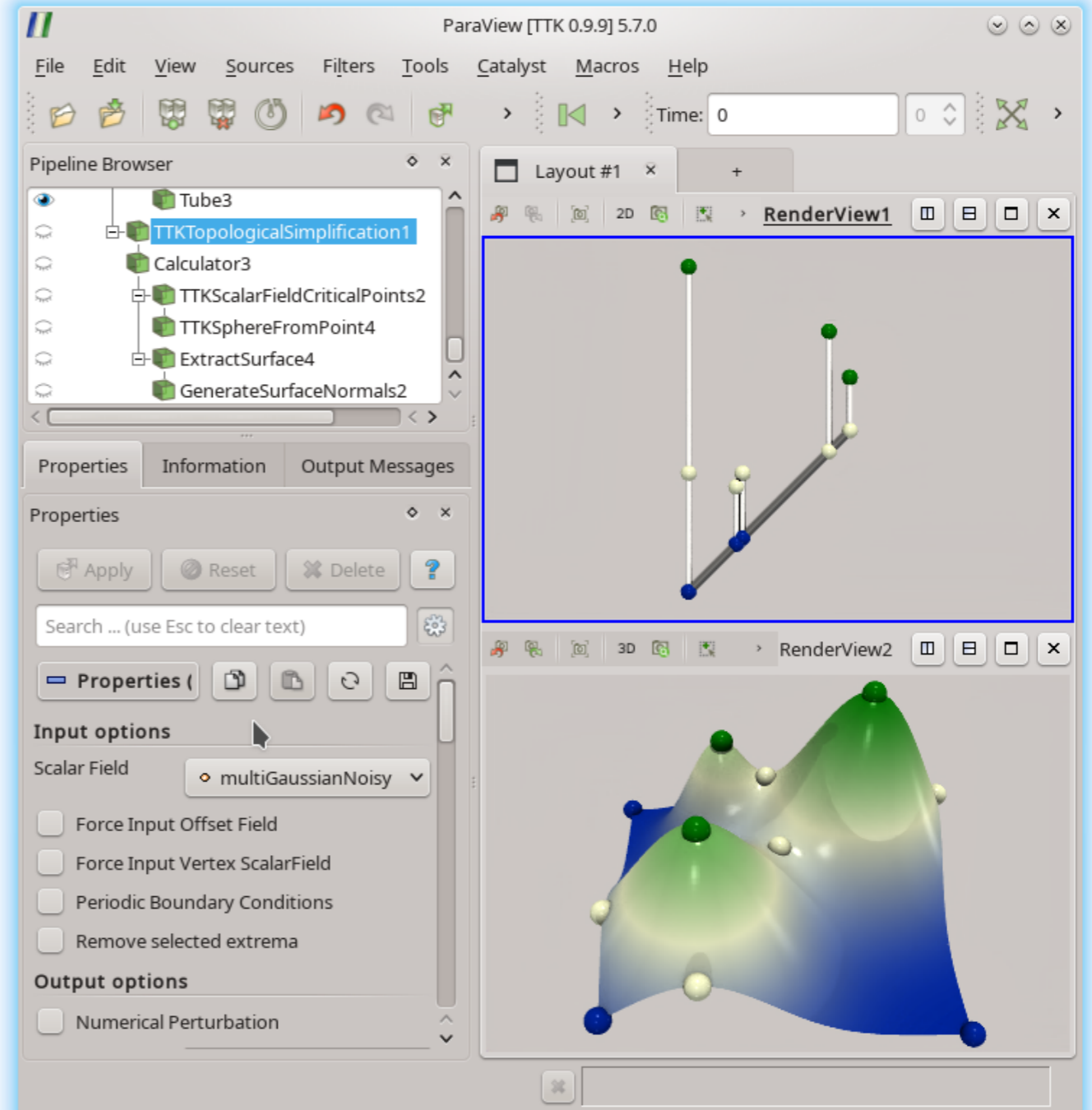
Persistence diagrams

- Module
 - PersistenceDiagram
 - PersistenceCurve
- Algorithms
 - Extremum/saddle pairs
 - Gueunet et al. IEEE TPDS 2019
 - Linearithmic time, efficient parallelization
 - Saddle/saddle pairs w/ saddle connectors
- Output
 - One curve per pair type



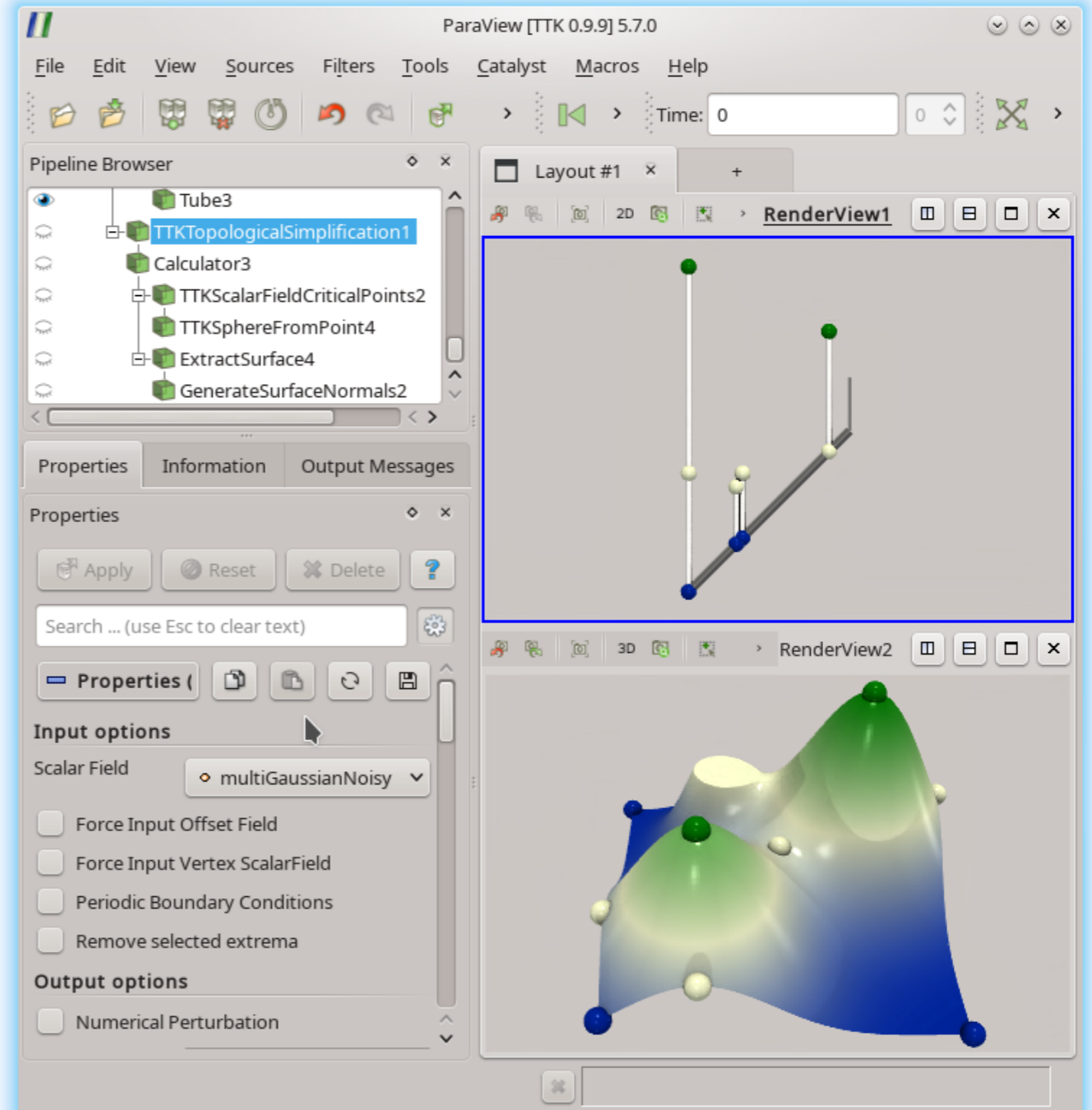
Topological data simplification

- Module
 - **TopologicalSimplification**
 - Extremum removal
 - Default multiscale mechanism
- Algorithm
 - Tierny & Pascucci IEEE TVCG 2012
 - Linearithmic time
- Output
 - “Flattened” data



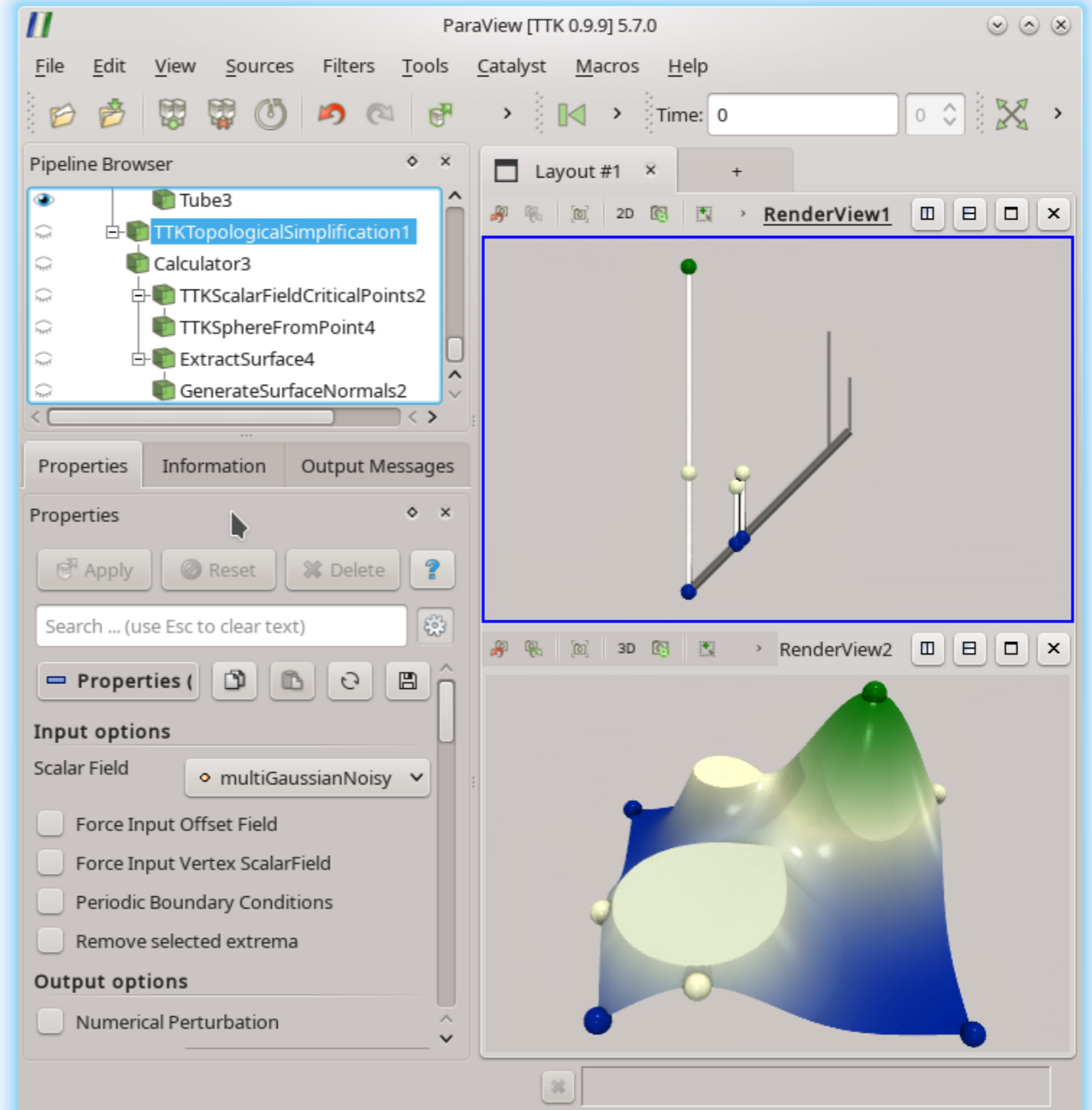
Topological data simplification

- Module
 - **TopologicalSimplification**
 - Extremum removal
 - Default multiscale mechanism
- Algorithm
 - Tierny & Pascucci IEEE TVCG 2012
 - Linearithmic time
- Output
 - “Flattened” data



Topological data simplification

- Module
 - **TopologicalSimplification**
 - Extremum removal
 - Default multiscale mechanism
- Algorithm
 - Tierny & Pascucci IEEE TVCG 2012
 - Linearithmic time
- Output
 - “Flattened” data



Experimental Flow Vis



$R = 32$



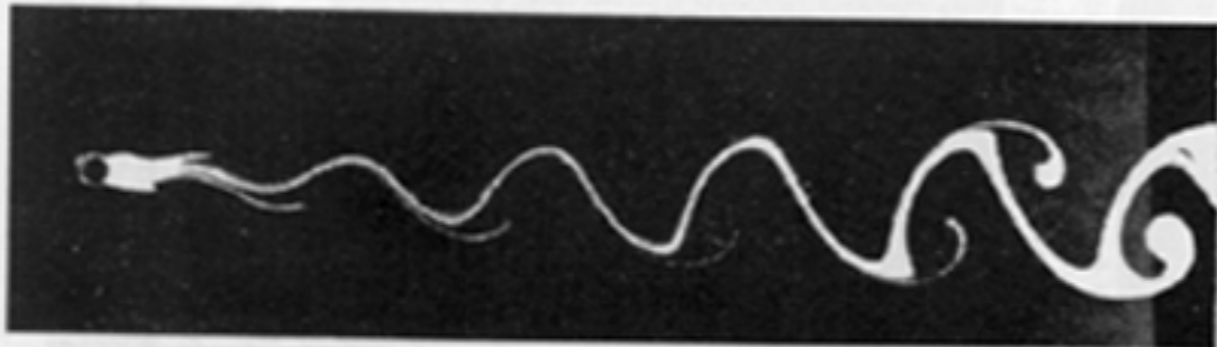
$R = 73$



$R = 55$



$R = 102$



$R = 65$



$R = 161$

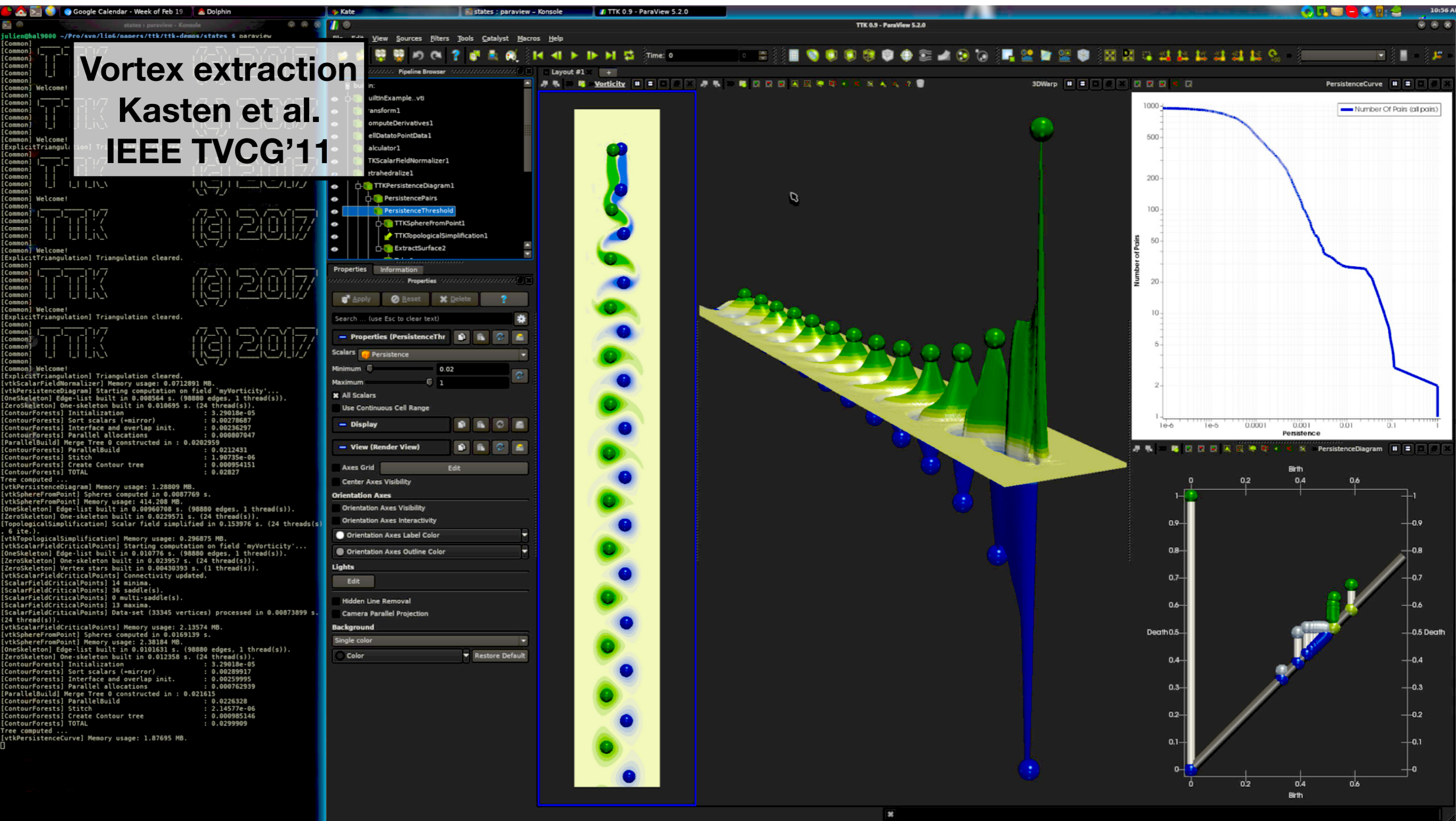
von Kármán vortex street, depending on Reynolds number

Guadalupe Island



<http://envsci.rutgers.edu/~lintner/teaching.html>

TDA Apps in Visualization

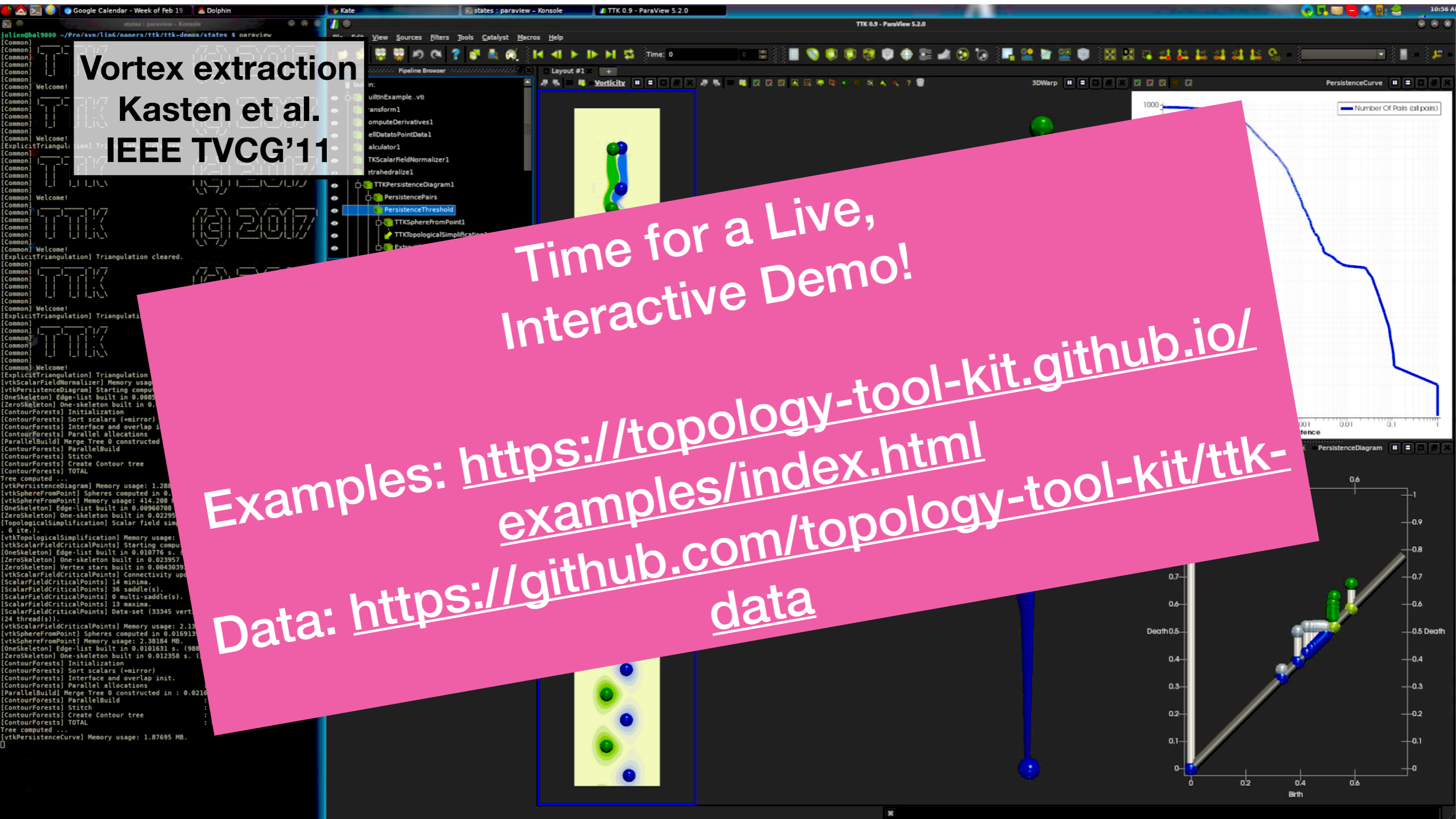


TDA Apps in Visualization

Vortex extraction
Kasten et al.
IEEE TVCG'11

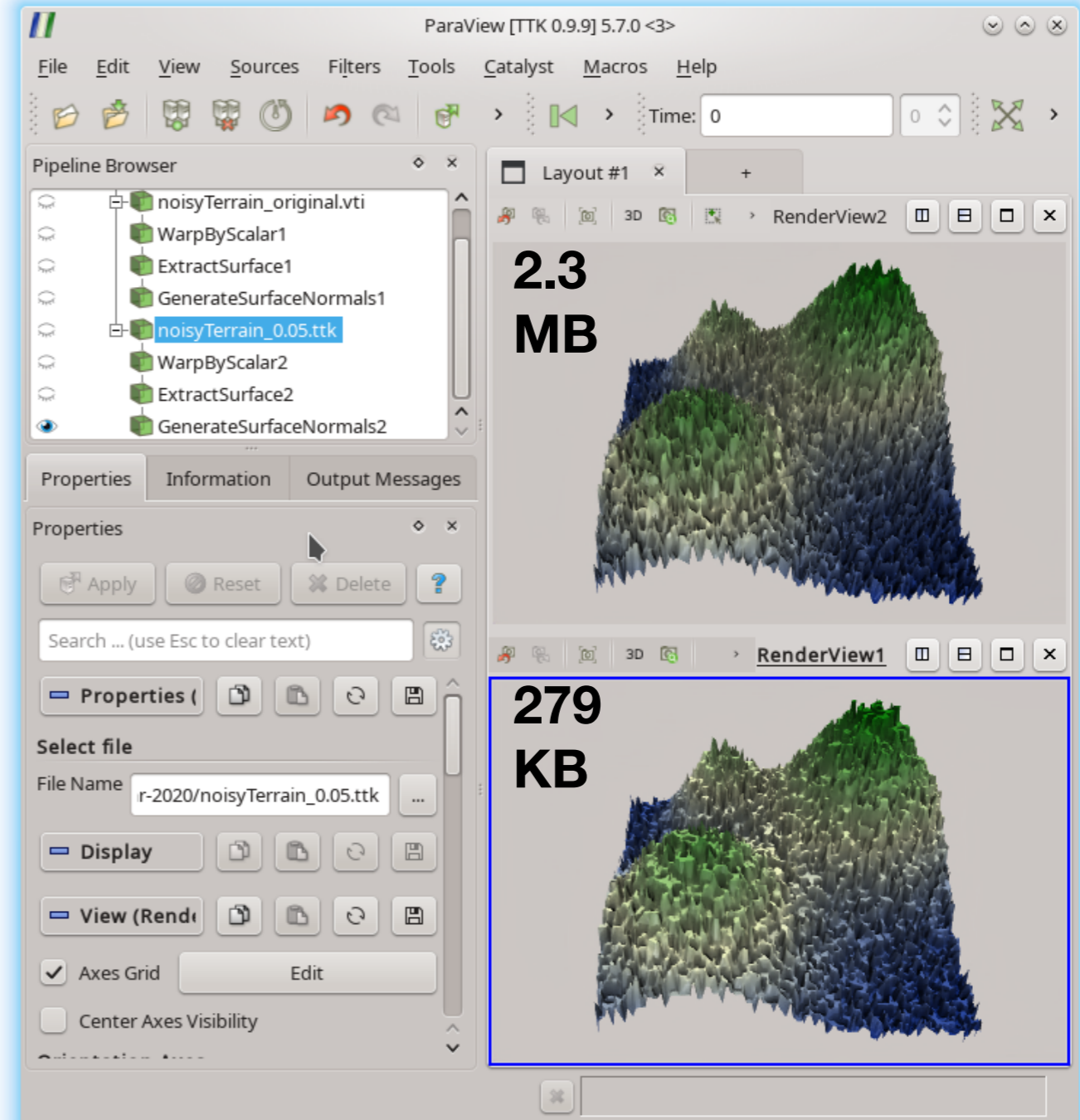
Time for a Live,
Interactive Demo!

Examples: <https://topology-tool-kit.github.io/examples/index.html>
Data: <https://github.com/topology-tool-kit/ttk-data>



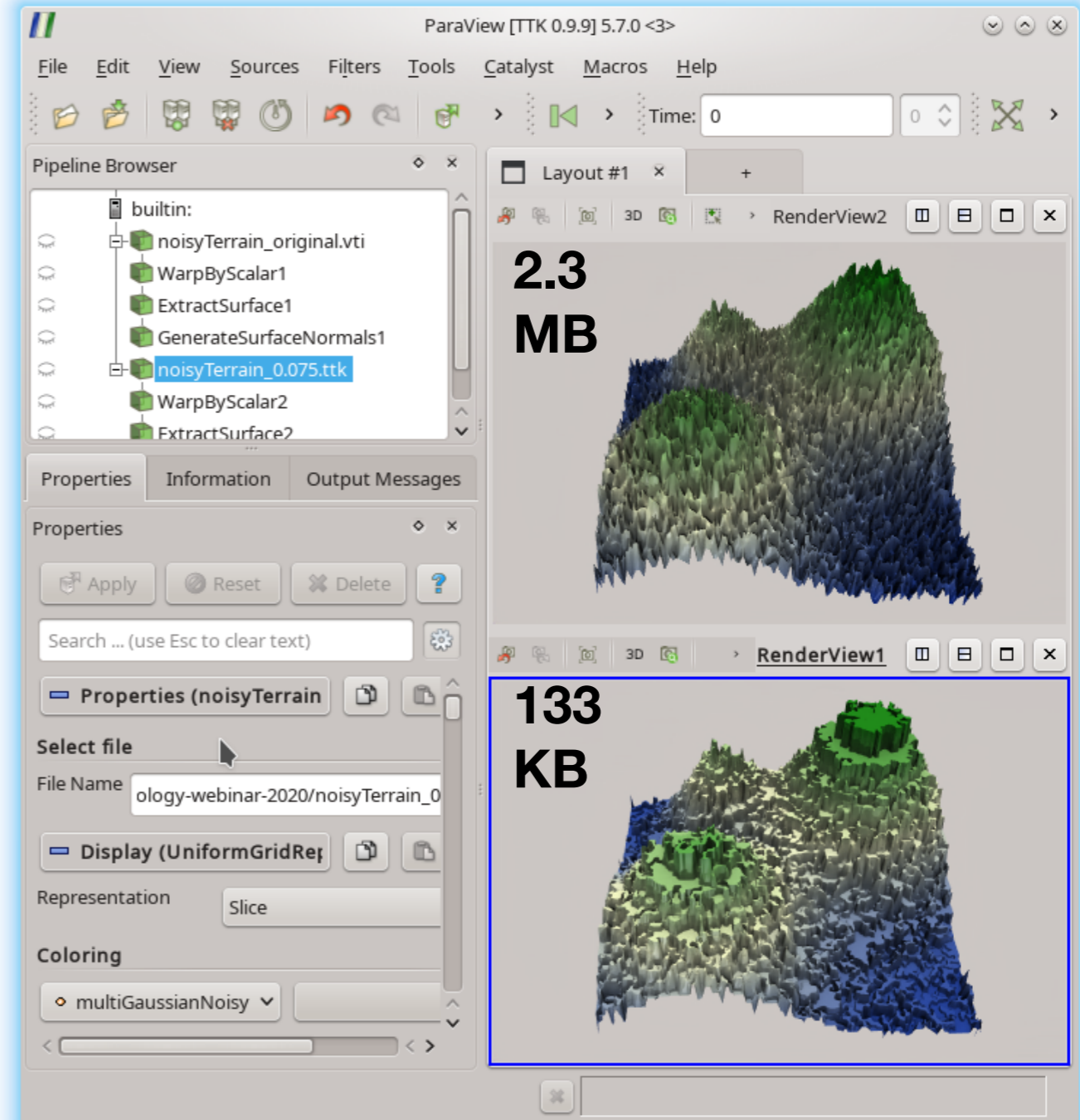
Topology aware compression

- Module
 - **TopologicalCompression** (IO)
 - Preserves the persistence diagram
 - Dimension 0 or (d-1)
- Algorithm
 - Soler et al. PacificViz 2018
 - Linearithmic time
- Output
 - Compressed image data
 - Geometry improvement with ZFP
 - Lindstrom IEEE TVCG 2014



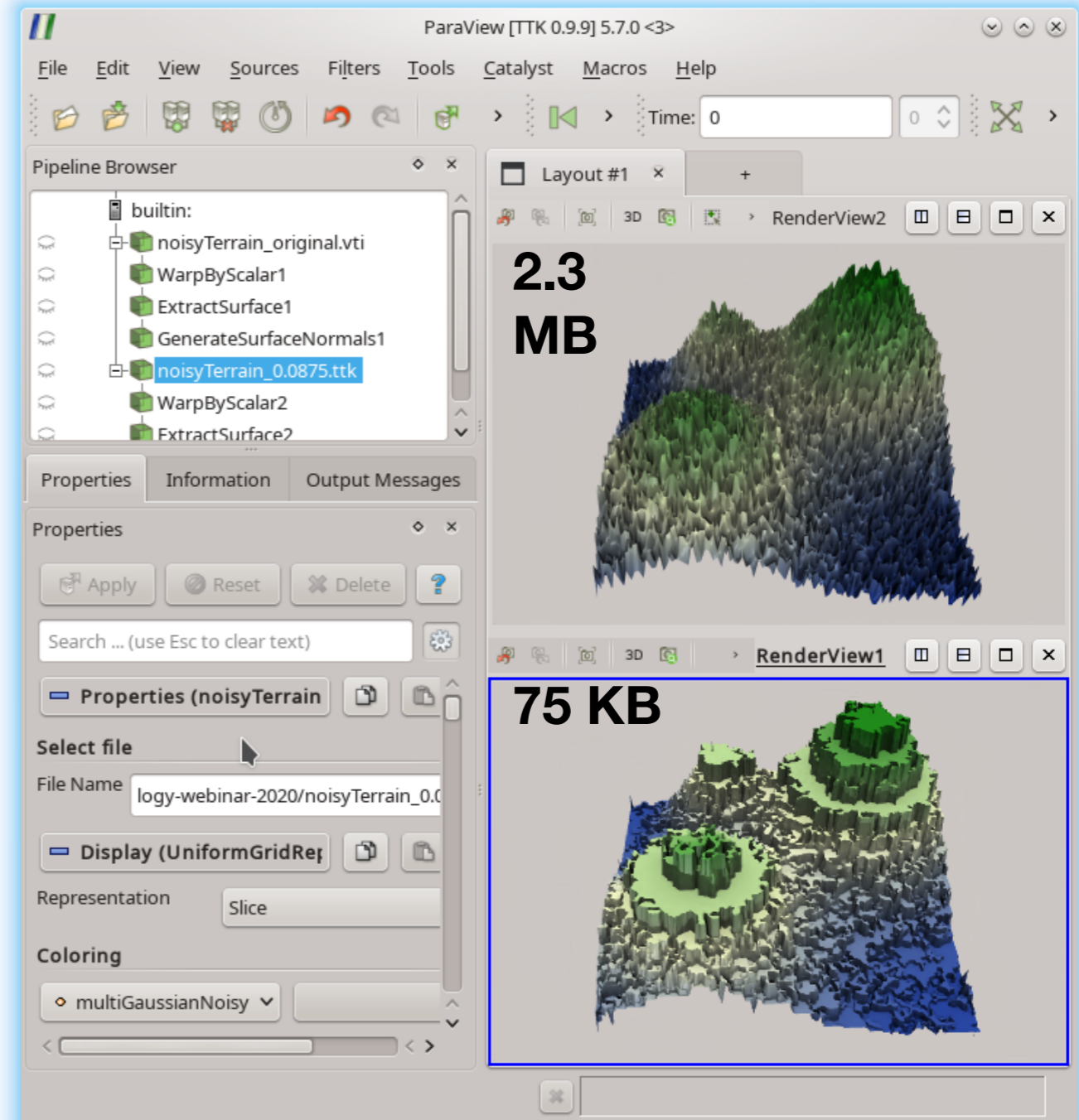
Topology aware compression

- Module
 - **TopologicalCompression** (IO)
 - Preserves the persistence diagram
 - Dimension 0 or (d-1)
- Algorithm
 - Soler et al. PacificViz 2018
 - Linearithmic time
- Output
 - Compressed image data
 - Geometry improvement with ZFP
 - Lindstrom IEEE TVCG 2014



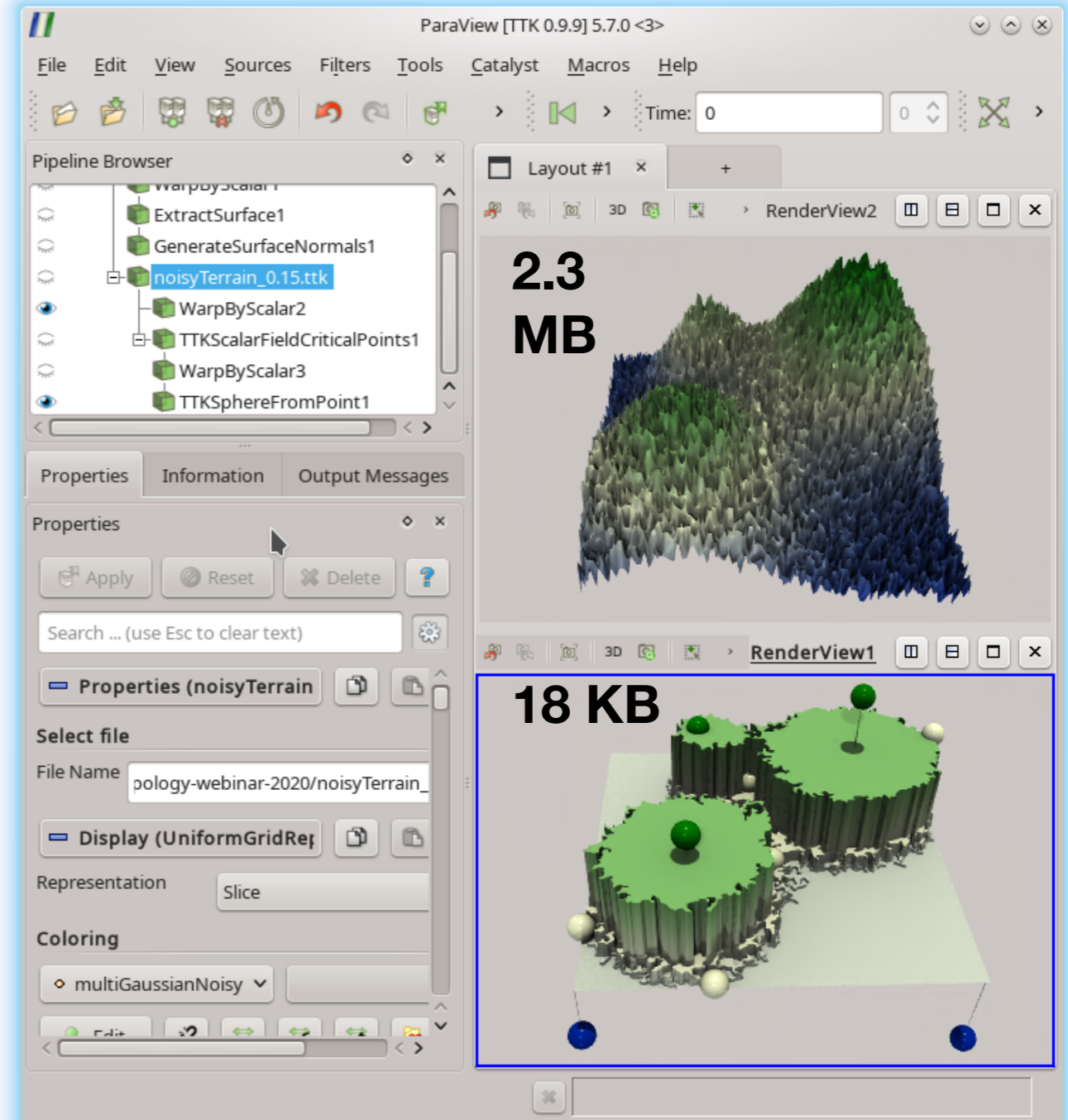
Topology aware compression

- Module
 - **TopologicalCompression** (IO)
 - Preserves the persistence diagram
 - Dimension 0 or (d-1)
- Algorithm
 - Soler et al. PacificViz 2018
 - Linearithmic time
- Output
 - Compressed image data
 - Geometry improvement with ZFP
 - Lindstrom IEEE TVCG 2014



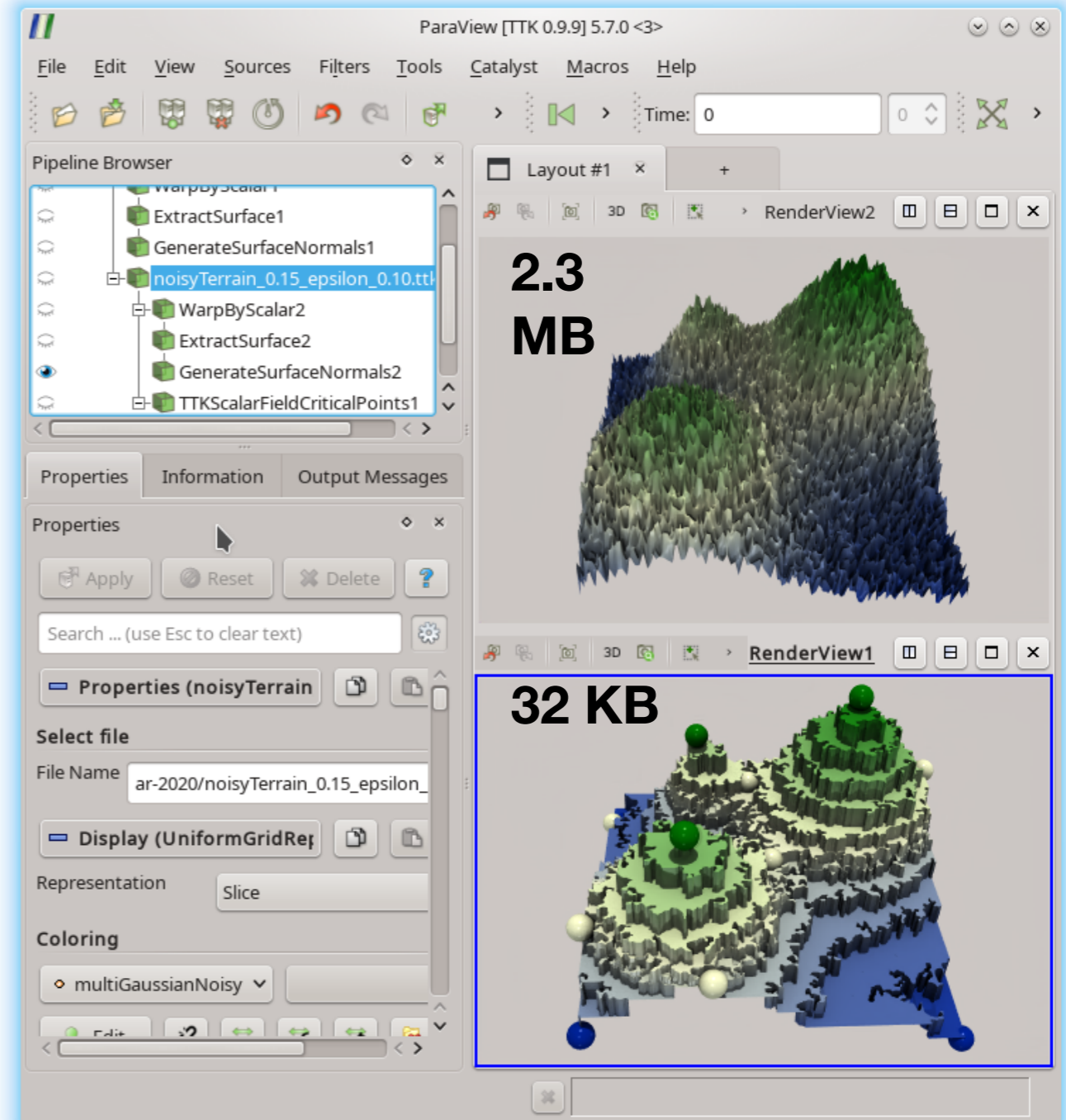
Topology aware compression

- Module
 - **TopologicalCompression** (IO)
 - Preserves the persistence diagram
 - Dimension 0 or (d-1)
- Algorithm
 - Soler et al. PacificViz 2018
 - Linearithmic time
- Output
 - Compressed image data
 - Geometry improvement with ZFP
 - Lindstrom IEEE TVCG 2014



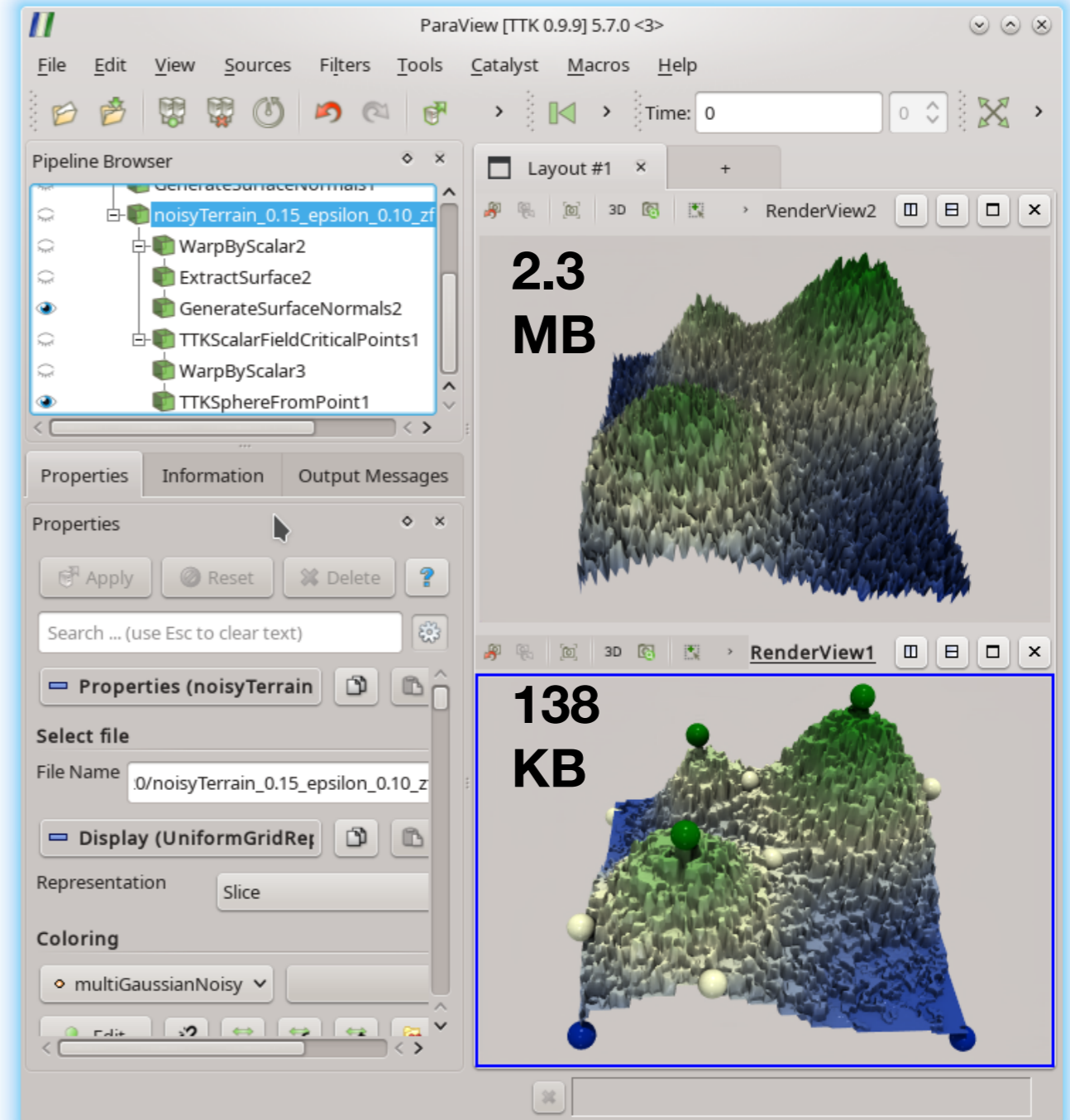
Topology aware compression

- Module
 - **TopologicalCompression** (IO)
 - Preserves the persistence diagram
 - Dimension 0 or (d-1)
- Algorithm
 - Soler et al. PacificViz 2018
 - Linearithmic time
- Output
 - Compressed image data
 - Geometry improvement with ZFP
 - Lindstrom IEEE TVCG 2014



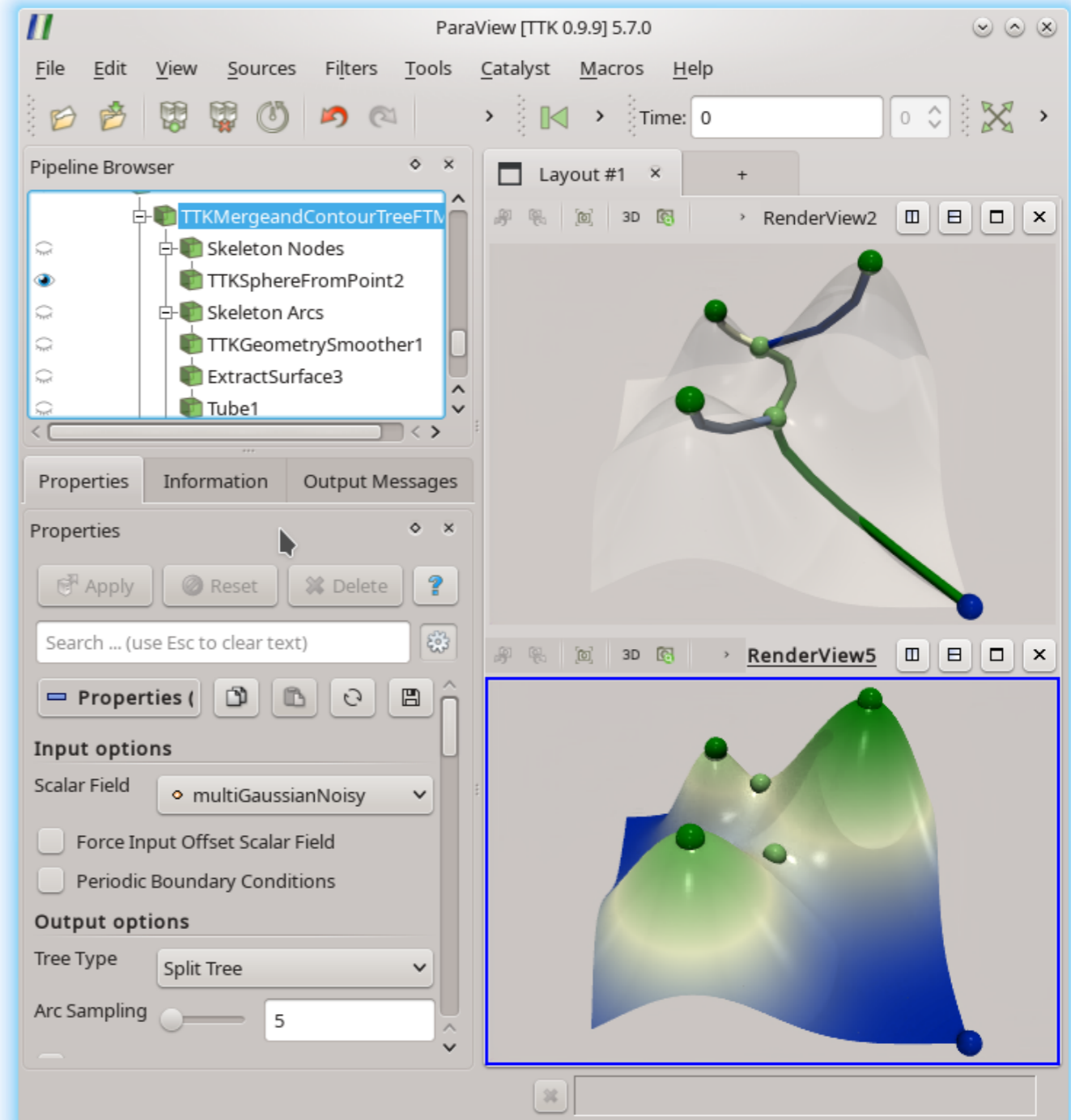
Topology aware compression

- Module
 - **TopologicalCompression** (IO)
 - Preserves the persistence diagram
 - Dimension 0 or (d-1)
- Algorithm
 - Soler et al. PacificViz 2018
 - Linearithmic time
- Output
 - Compressed image data
 - Geometry improvement with ZFP
 - Lindstrom IEEE TVCG 2014



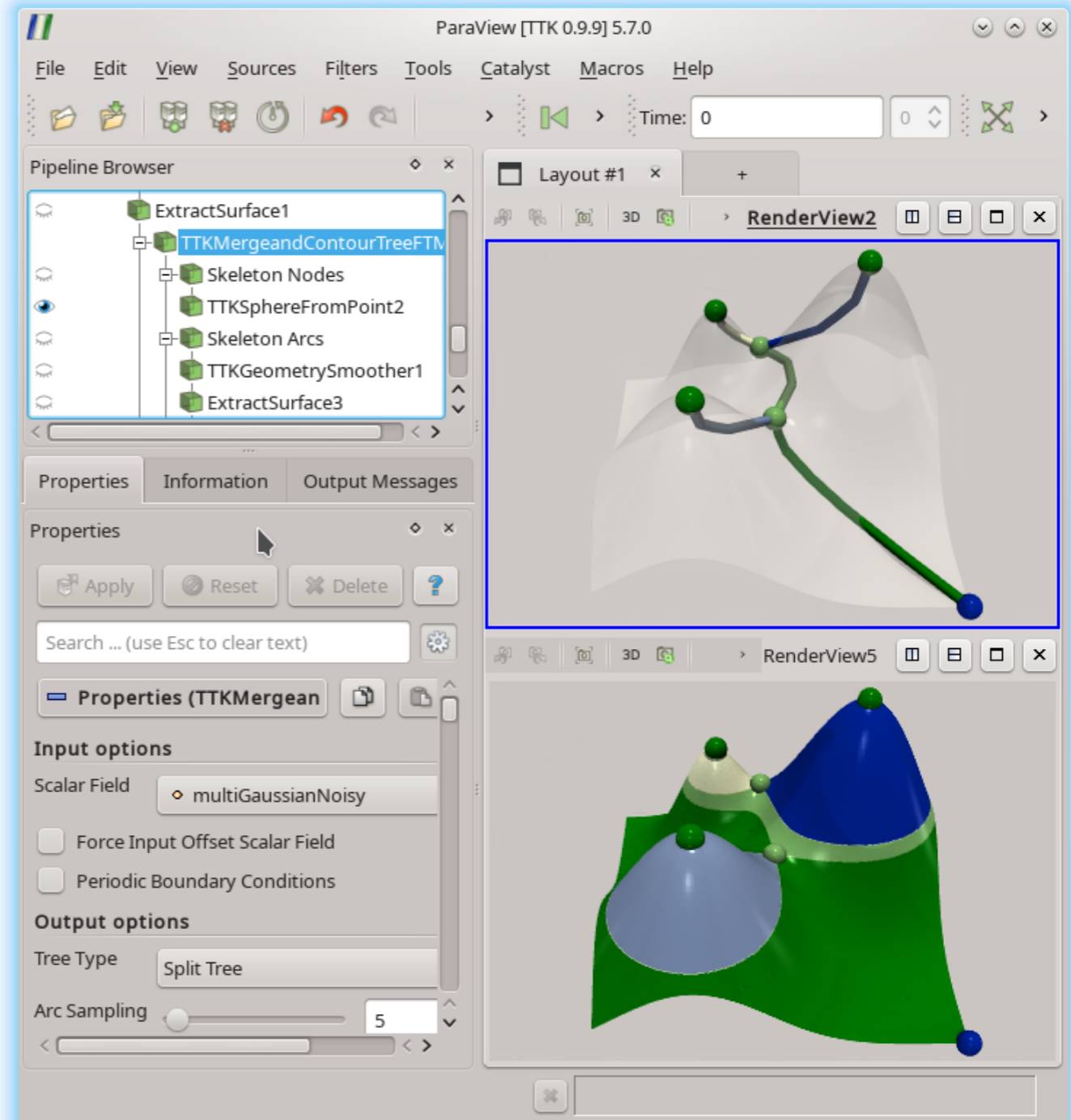
Merge & contour trees

- Module
 - **FTMTree**
 - Skeleton extraction
 - Level-set based segmentation
- Algorithm
 - Gueunet et al. IEEE TPDS 2019
 - Linearithmic time, efficient parallelization
- Output
 - Nodes of the trees
 - Arcs
 - Data segmentation



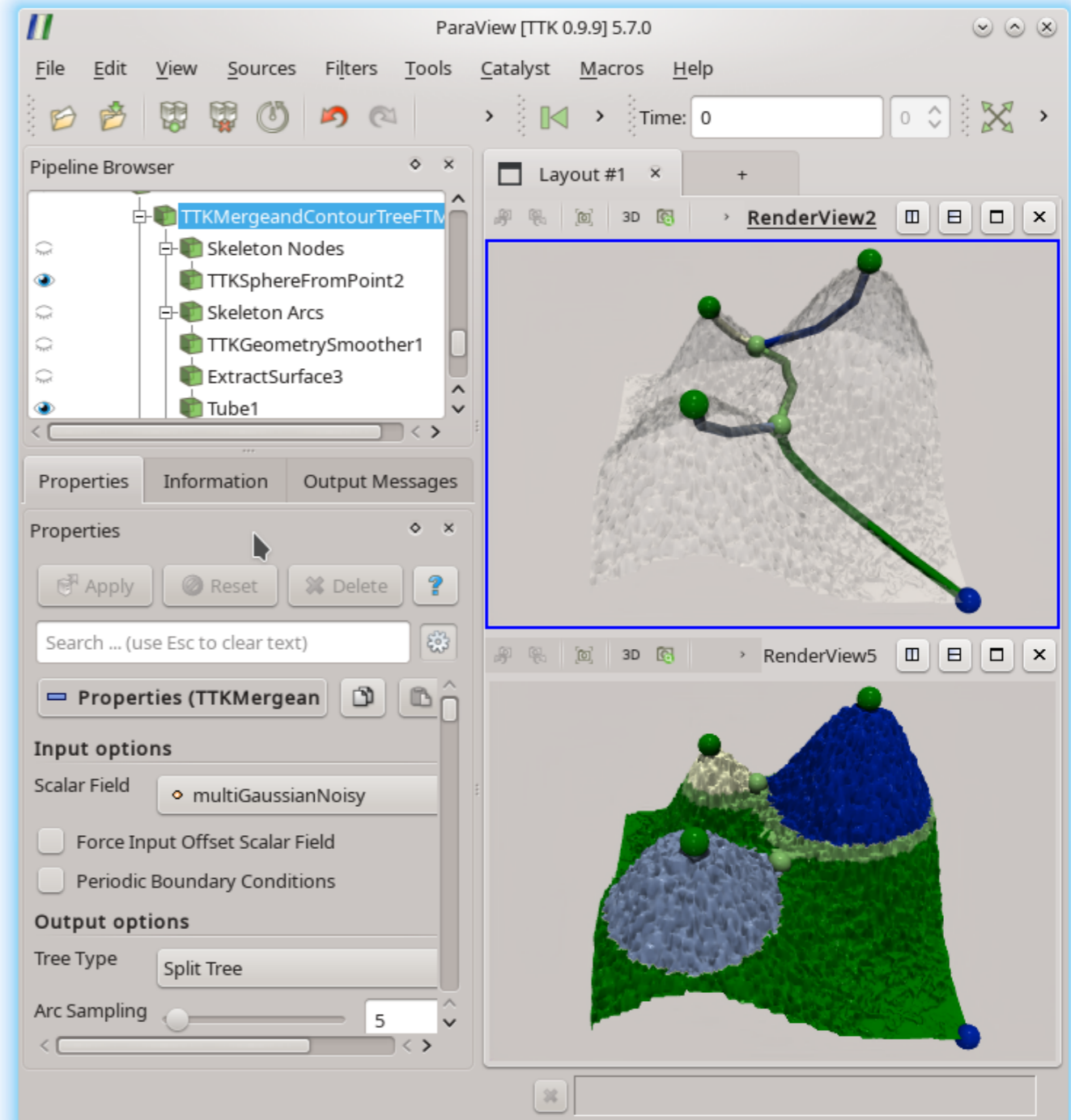
Merge & contour trees

- Module
 - **FTMTree**
 - Skeleton extraction
 - Level-set based segmentation
- Algorithm
 - Gueunet et al. IEEE TPDS 2019
 - Linearithmic time, efficient parallelization
- Output
 - Nodes of the trees
 - Arcs
 - Data segmentation

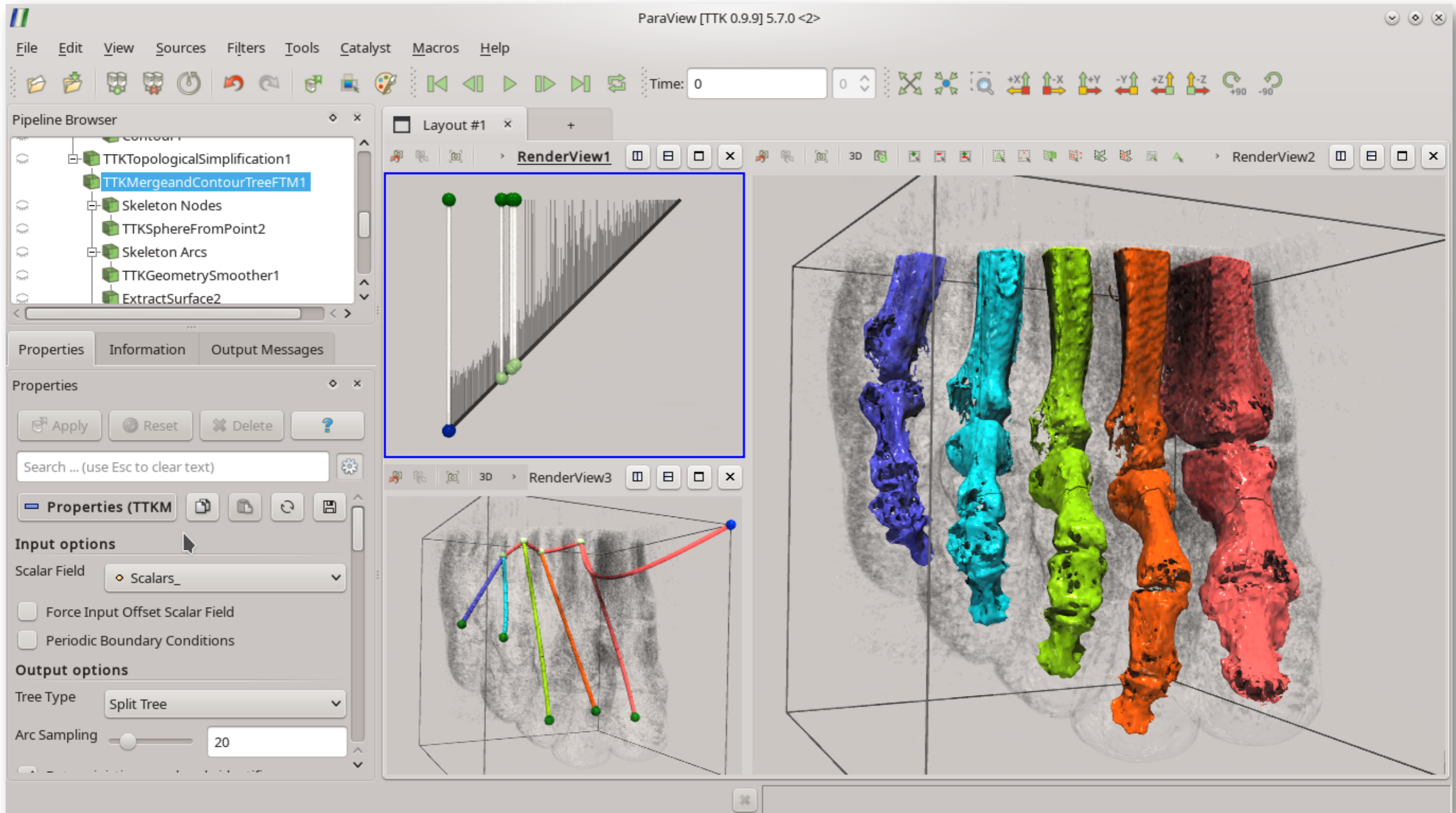


Merge & contour trees

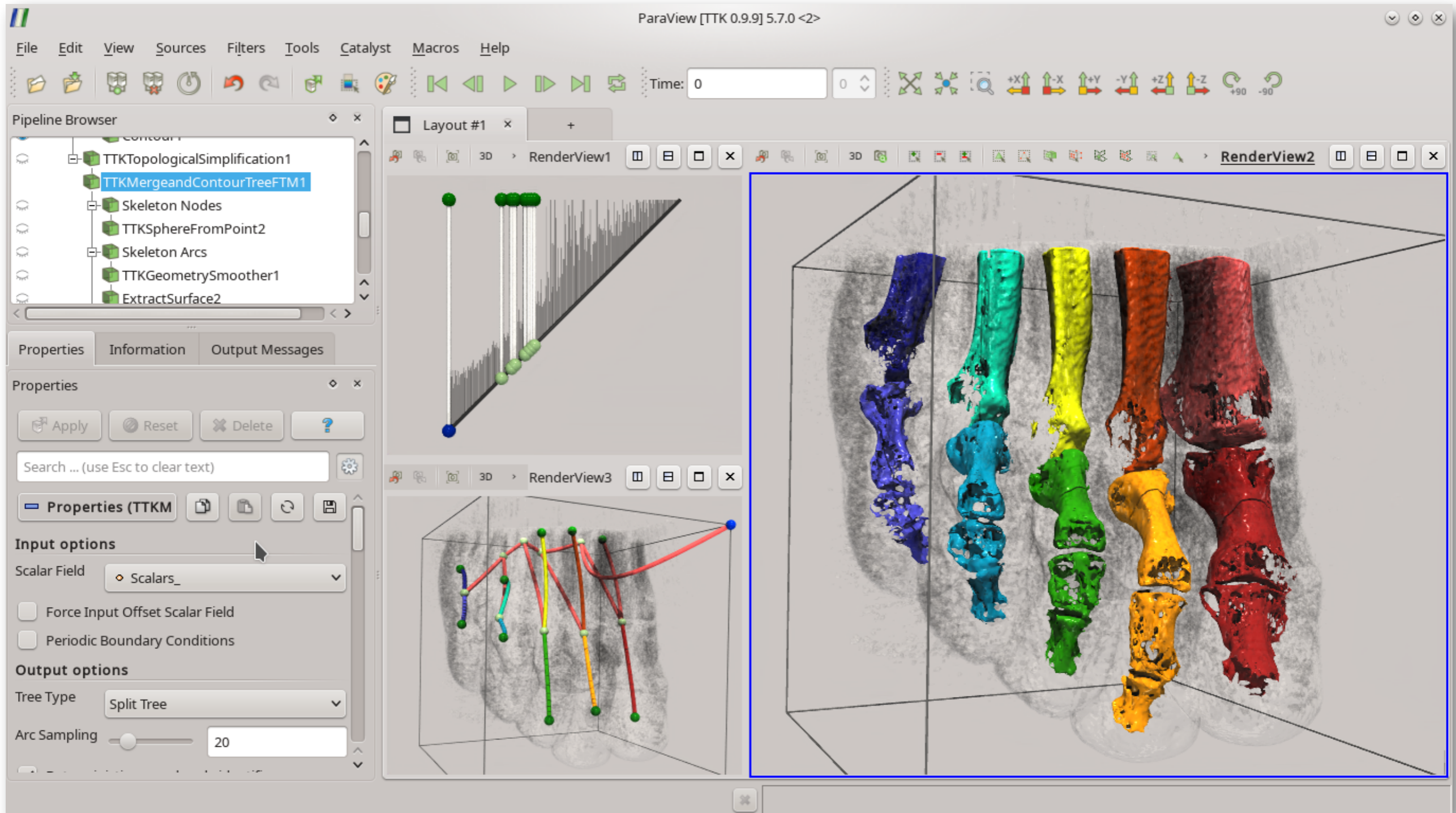
- Module
 - **FTMTree**
 - Skeleton extraction
 - Level-set based segmentation
- Algorithm
 - Gueunet et al. IEEE TPDS 2019
 - Linearithmic time, efficient parallelization
- Output
 - Nodes of the trees
 - Arcs
 - Data segmentation



Application to biomedical imaging

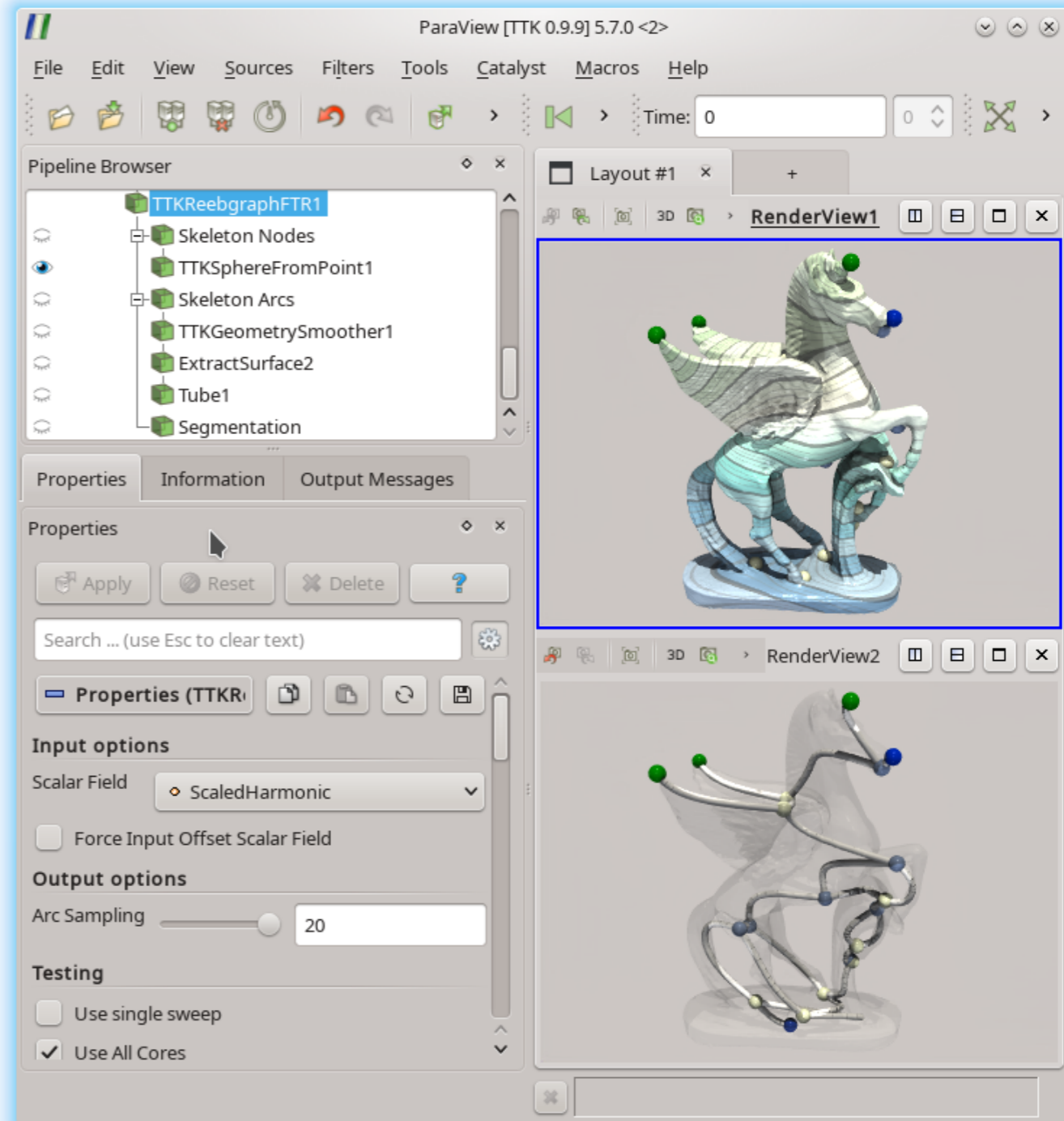


Application to biomedical imaging



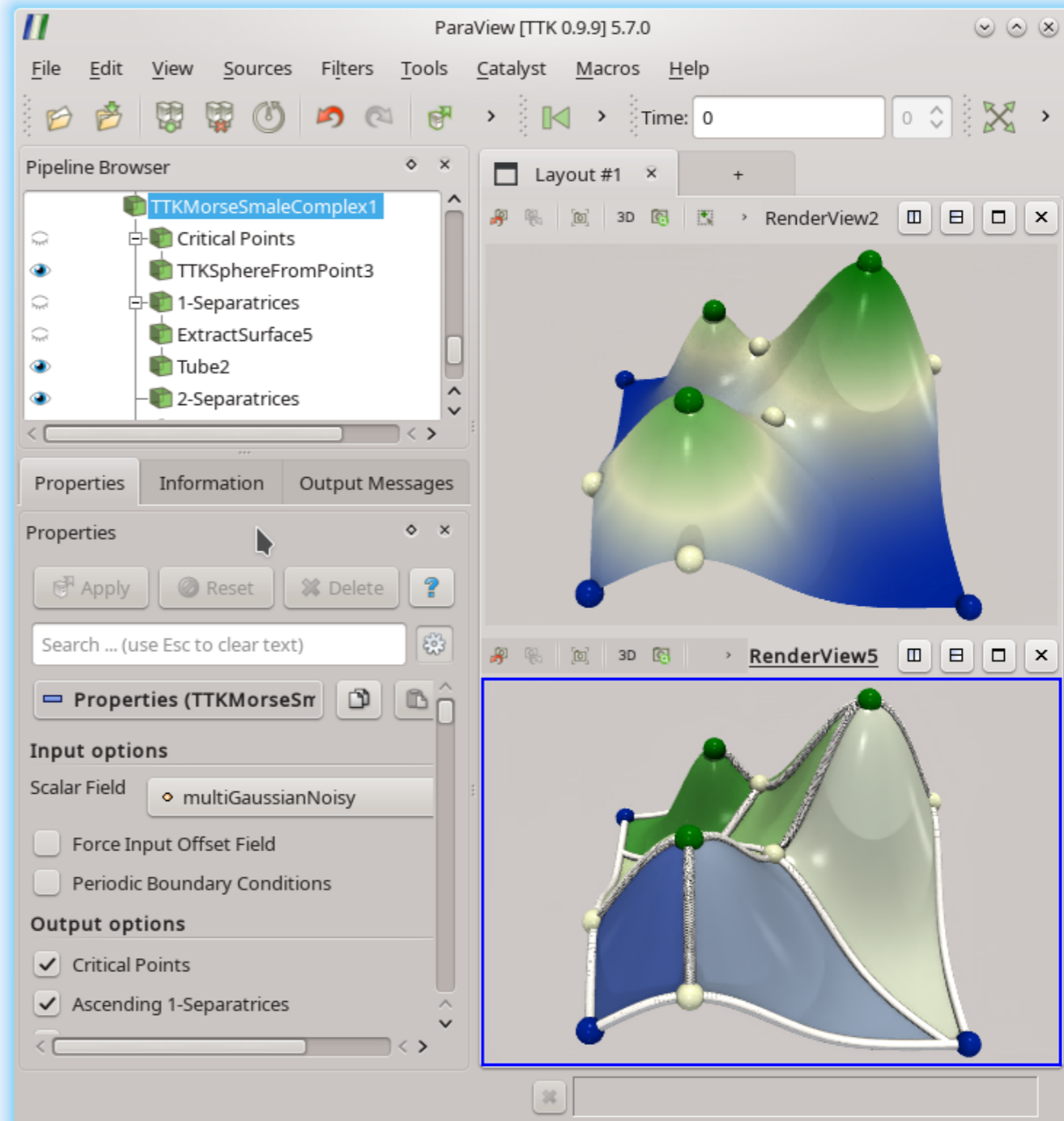
Reeb graphs

- Module
 - **FTMGraph**
 - Skeleton extraction
 - Level-set based segmentation
 - For non simply-connected domains
- Algorithm
 - Gueunet et al. EGPGV 2019
 - Linearithmic time
 - Fast Parallelization of Parsa SoCG 2012
- Output
 - Nodes of the trees, Arcs, Data segmentation



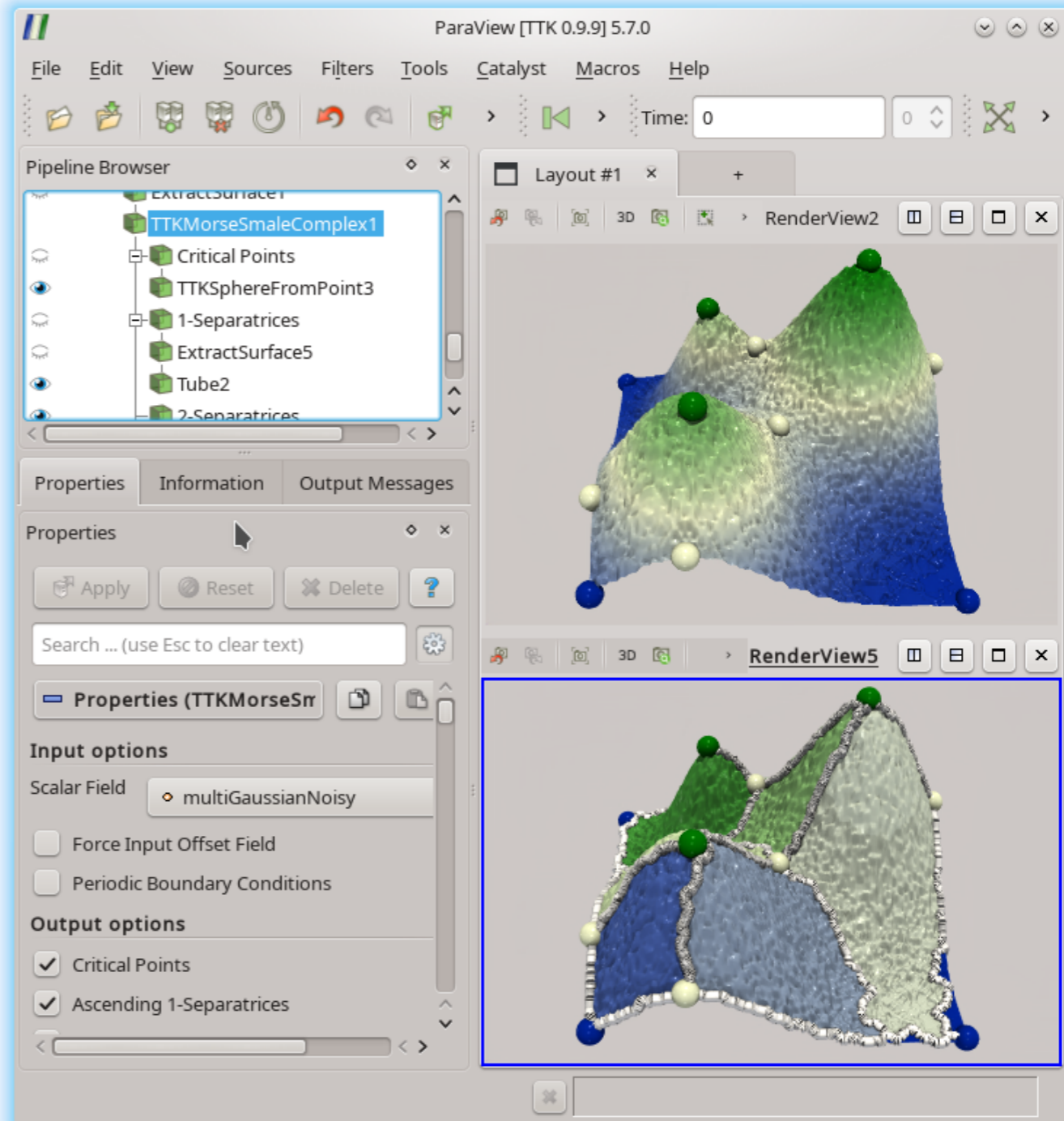
Morse-Smale complexes

- Module
 - **MorseSmaleComplex**
 - Extremal curves, separating surfaces
- Algorithms
 - Discrete Morse Theory, Forman SLC 2002
 - Complex extraction via v-path collection (BFS)
 - Quadratic time (worst case)
 - Discrete gradient
 - Tierny et al. IEEE TVCG 2017
- Output
 - Critical points, separatrices, segmentation

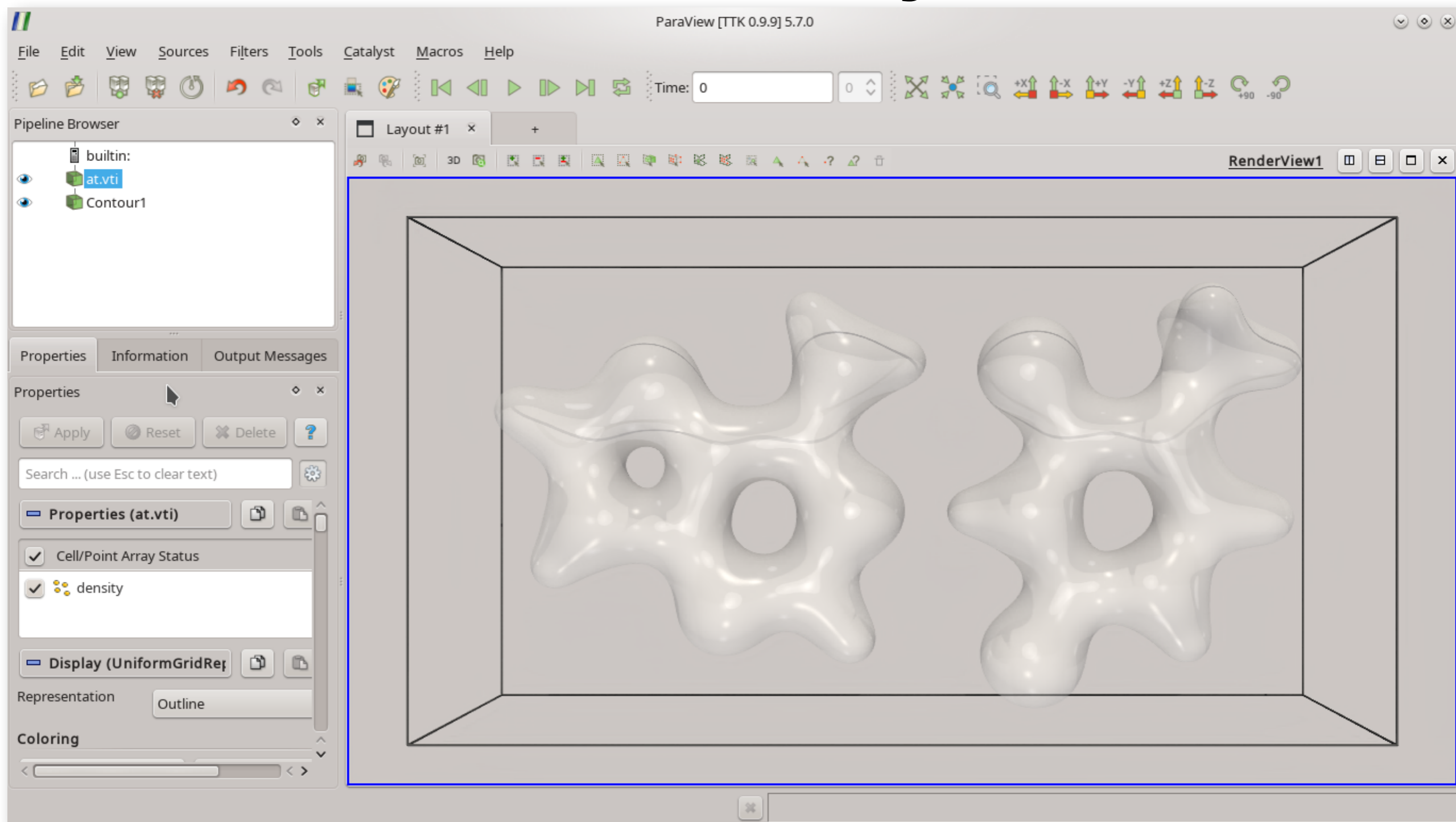


Morse-Smale complexes

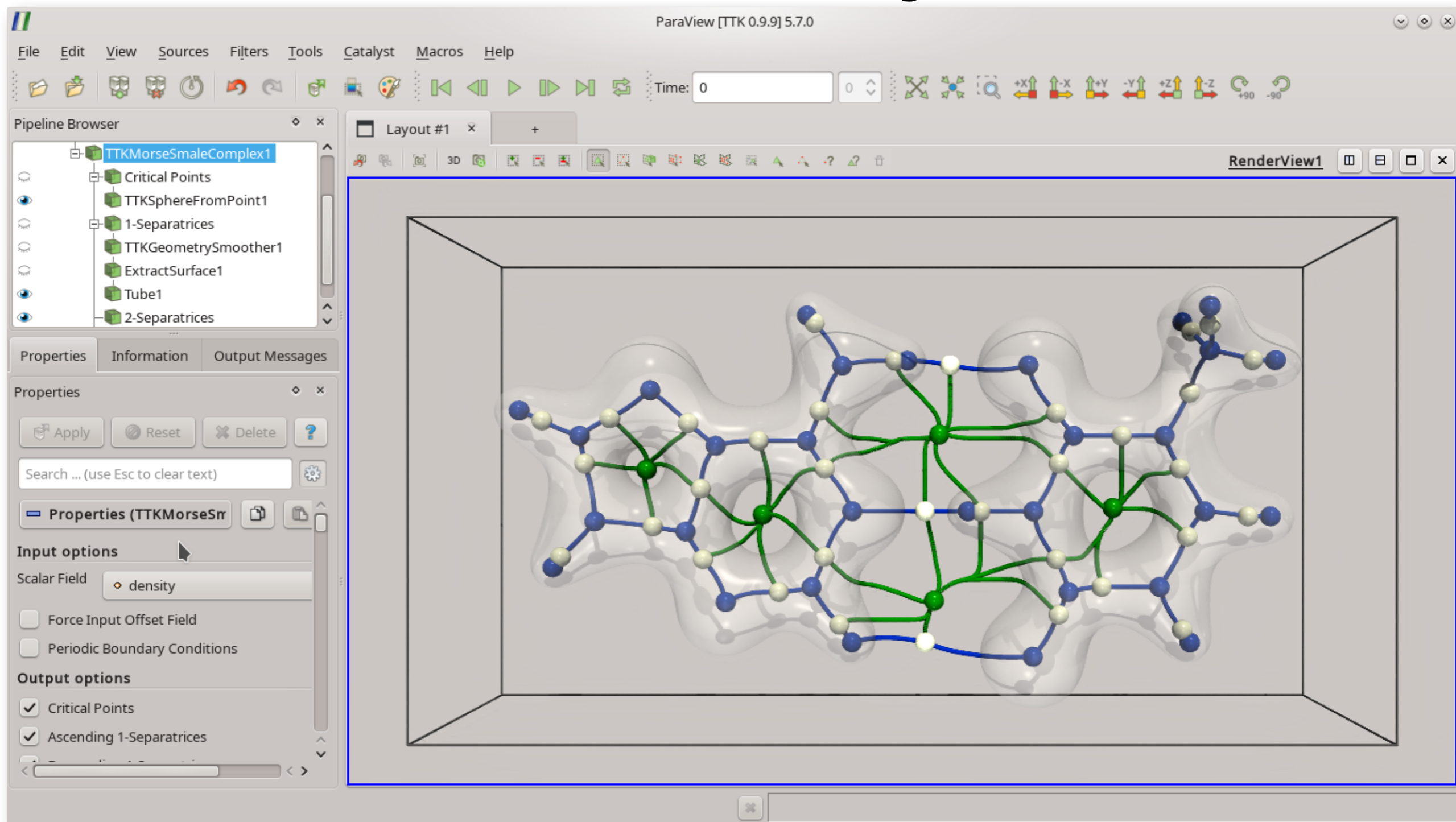
- Module
 - **MorseSmaleComplex**
- Extremal curves, separating surfaces
- Algorithms
 - Discrete Morse Theory, Forman SLC 2002
 - Complex extraction via v-path collection (BFS)
 - Quadratic time (worst case)
 - Discrete gradient
 - Tierny et al. IEEE TVCG 2017
- Output
 - Critical points, separatrices, segmentation



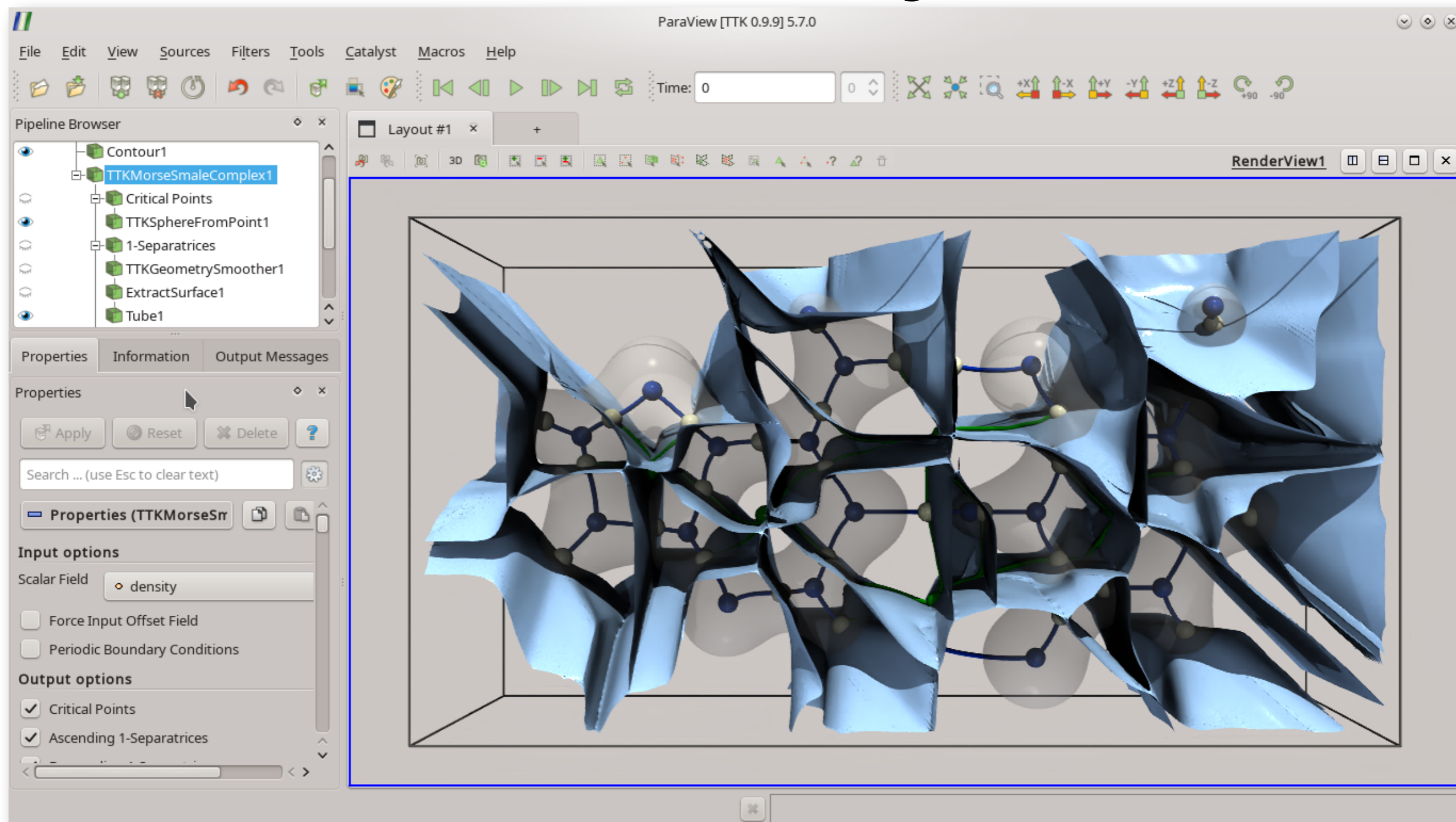
Application to quantum chemistry



Application to quantum chemistry



Application to quantum chemistry



More Demos!

Examples: <https://topology-tool-kit.github.io/examples/index.html>

Data: <https://github.com/topology-tool-kit/ttk-data>