The Topology ToolKit
http://topology-tool-kit.github.io/

Julien Tierny, Guillaume Favelier, Joshua Levine, Charles Gueunet, Michael Michaux
Topology in visualization

- Scalar data
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]
Topology in visualization

• Scalar data
  – $f : \mathcal{M} \rightarrow \mathbb{R}$

• Topological methods
  – Robust, multi-scale
  – Raw data to features
  – Many application successes
Challenges
Challenges

• For end users
  – Standard IO, generic inputs
  – GUI, scripting languages
Challenges

• For end users
  – Standard IO, generic inputs
  – GUI, scripting languages

• For developers
  – Unified data structure
  – Optional software dependencies
Challenges

• For end users
  – Standard IO, generic inputs
  – GUI, scripting languages

• For developers
  – Unified data structure
  – Optional software dependencies

• For researchers
  – Federating framework for reproducibility
Related work
High dimensions

• Point cloud data
  – Topology inference
    • No input scalar data
High dimensions

• Point cloud data
  – Topology inference
    • No input scalar data
  – Persistence homology
    • [Mor10, Nan13, ATV14]
    • [MBG14, BKR14, BD17]
  – Mapper [SMC08]
High dimensions

• Point cloud data
  – Topology inference
    • No input scalar data
  – Persistence homology
    • [Mor10, Nan13, ATV14]
    • [MBG14, BKR14, BD17]
  – Mapper [SMC08]

• Different settings / applications
Low dimensions

• Scientific visualization
Low dimensions

• Scientific visualization
  – Contour trees & Reeb graphs
    • [Dil07, Tie09, Dor12]
  – Morse-Smale complexes
    • [Sou11, SN15]
Low dimensions

• Scientific visualization
  – Contour trees & Reeb graphs
    • [Dil07, Tie09, Dor12]
  – Morse-Smale complexes
    • [Sou11, SN15]

• Technical limitations
  – No standard IO, no generic rep.
  – No end user feature
Contributions
Contributions

• An algorithm
  – PL compliant discrete gradient
Contributions

• An algorithm
  – PL compliant discrete gradient

• A data structure
  – Efficient 2D/3D triangulation traversals
Contributions

• An algorithm
  – PL compliant discrete gradient

• A data structure
  – Efficient 2D/3D triangulation traversals

• A software architecture
  – Easy development and distribution
Contributions

• An algorithm
  – PL compliant discrete gradient

• A data structure
  – Efficient 2D/3D triangulation traversals

• A software architecture
  – Easy development and distribution

• A software collection
  – Reference algorithms (ParaView, Python, VTK/C++, C++)
Background
Piecewise linear setting

- Input PL scalar data
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]
Piecewise linear setting

- Input PL scalar data

\[ f : \mathcal{M} \rightarrow \mathbb{R} \]
Piecewise linear setting

- Input PL scalar data
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]
- Topological abstractions
Piecewise linear setting

- Input PL scalar data
  \[ f : M \rightarrow \mathbb{R} \]
- Topological abstractions
  - Critical points
The Topology ToolKit

http://topology-tool-kit.github.io/

- Input PL scalar data
- Topological abstractions
  - Critical points

Piecewise linear setting
Piecewise linear setting

- Input PL scalar data
  - $f : \mathcal{M} \rightarrow \mathbb{R}$

- Topological abstractions
  - Critical points
Piecewise linear setting

• Input PL scalar data
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]

• Topological abstractions
  – Critical points
Piecewise linear setting

• Input PL scalar data
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]
• Topological abstractions
  – Critical points
  – Persistence diagrams
Piecewise linear setting

- Input PL scalar data
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]
- Topological abstractions
  - Critical points
  - Persistence diagrams
  - Persistence curves
Piecewise linear setting

- Input PL scalar data
  - $f : \mathcal{M} \rightarrow \mathbb{R}$
- Topological abstractions
  - Critical points
  - Persistence diagrams
  - Persistence curves
Piecewise linear setting

- Input PL scalar data
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]
- Topological abstractions
  - Critical points
  - Persistence diagrams
  - Persistence curves
Piecewise linear setting

- Input PL scalar data
  - \( f : \mathcal{M} \rightarrow \mathbb{R} \)

- Topological abstractions
  - Critical points
  - Persistence diagrams
  - Persistence curves
  - Reeb graphs
Piecewise linear setting

- Input PL scalar data
  - $f : \mathcal{M} \rightarrow \mathbb{R}$
- Topological abstractions
  - Critical points
  - Persistence diagrams
  - Persistence curves
  - Reeb graphs
Discrete Morse theory

• Morse-Smale complex
Discrete Morse theory

- Morse-Smale complex
  - Challenging PL computation
Discrete Morse theory

• Morse-Smale complex
  – Challenging PL computation

• Discrete gradient field
  – Loop free V-paths
  • \( \{ \sigma_i^0 < \sigma_{i+1}^0 \} \)
Discrete Morse theory

- Morse-Smale complex
  - Challenging PL computation
- Discrete gradient field
  - Loop free V-paths
    - $\{\sigma_i^0 < \sigma_{i+1}^0\}$
Discrete Morse theory

- Morse-Smale complex
  - Challenging PL computation
- Discrete gradient field
  - Loop free V-paths
    \[ \{\sigma_i^0 < \sigma_{i+1}^0\} \ldots \{\sigma_i^n < \sigma_{i+1}^n\} \]
Discrete Morse theory

• Morse-Smale complex
  – Challenging PL computation

• Discrete gradient field
  – Loop free V-paths
    • \( \{\sigma^0_i < \sigma^0_{i+1}\} \ldots \{\sigma^n_i < \sigma^n_{i+1}\} \)
    • \( \sigma^j_i \neq \sigma^j_{i+1} < \sigma^j_{i+1} \)
Discrete Morse theory

- Morse-Smale complex
  - Challenging PL computation

- Discrete gradient field
  - Loop free V-paths
    - $\{\sigma_i^0 < \sigma_i^0 \} \ldots \{\sigma_i^n < \sigma_i^n \}$
    - $\sigma_i^j \neq \sigma_i^{j+1} < \sigma_i^{j+1}$
Discrete Morse theory

- Morse-Smale complex
  - Challenging PL computation

- Discrete gradient field
  - Loop free V-paths
    - $\{\sigma_i^0 < \sigma_{i+1}^0\} \ldots \{\sigma_i^n < \sigma_{i+1}^n\}$
    - $\sigma_i^j \neq \sigma_{i+1}^{j+1} < \sigma_i^{j+1}$
  - Critical points: unpaired simplices
  - Separatrices: V-paths

The Topology ToolKit
http://topology-tool-kit.github.io/
What for?
What for?
What for?
What for?
What for?
What for?
What for?
What for?
What for?
What for?
What for?
What for?
Unified topological analysis
Unification motivation
Unification motivation
Unification motivation
Unification motivation
Unification motivation
Unification motivation

- Consistency
- Maintenance
PL compliant discrete gradient

• Initial gradient algorithm [SN12]
  – For each i-simplex $\sigma_i$
PL compliant discrete gradient

- Initial gradient algorithm [SN12]
  - For each $i$-simplex $\sigma_i$
    - $C(\sigma_i)$: co-faces for which $\sigma_i$ is an $i$-maximizer
PL compliant discrete gradient

• Initial gradient algorithm [SN12]
  – For each i-simplex $\sigma_i$
    • $C(\sigma_i)$: co-faces for which $\sigma_i$ is an i-maximizer
    • $\{\sigma_i < \arg\min_{C(\sigma_i)} (\sigma_{i+1})\}$
PL compliant discrete gradient

• Initial gradient algorithm [SN12]
  – For each i-simplex $\sigma_i$
    • $C(\sigma_i)$: co-faces for which $\sigma_i$ is an i-maximizer
    • $\{\sigma_i < \arg\min_{C(\sigma_i)} (\sigma_{i+1})\}$

• PL matching property
  – For each PL critical point
  – Critical simplex in its star
PL compliant discrete gradient

• Initial gradient algorithm [SN12]
  – For each i-simplex $\sigma_i$
    • $C(\sigma_i)$: co-faces for which $\sigma_i$ is an i-maximizer
      • $\{\sigma_i < \arg\min_{C(\sigma_i)} (\sigma_{i+1})\}$
  • PL matching property
    – For each PL critical point
      – Critical simplex in its star

[Image of a diagram with labeled nodes and edges representing the PL compliant discrete gradient process]
PL compliant discrete gradient

- Initial gradient algorithm [SN12]
  - For each $i$-simplex $\sigma_i$
    - $C(\sigma_i)$: co-faces for which $\sigma_i$ is an $i$-maximizer
    - $\left\{ \sigma_i < \arg \min_{C(\sigma_i)} (\sigma_{i+1}) \right\}$

- PL matching property
  - For each PL critical point
    - Critical simplex in its star

- Path reversal for non PL
PL compliant discrete gradient

- Initial gradient algorithm [SN12]
  - For each i-simplex $\sigma_i$
    - $C(\sigma_i)$: co-faces for which $\sigma_i$ is an i-maximizer
      - $\{\sigma_i < \arg\min_{C(\sigma_i)} (\sigma_{i+1})\}$
  
- PL matching property
  - For each PL critical point
    - Critical simplex in its star

- Path reversal for non PL
PL compliant discrete gradient
PL compliant discrete gradient
PL compliant discrete gradient

| Dataset     | $|DMT(f)|$ | $|PL(f)|$ | Alg. 1 | $PL(f)$ | S-M | S-S | $M^S(f)$ | $M^S'(f)$ |
|-------------|-----|-----|-------|-------|------|-----|-----|----------|----------|
| Dragon      | 1,118 | 318 | 0.016 | 0.018 | 0.004 | 0   |     | 0.074    | 0.072    |
| EthaneDiol  | 6,109 | 93  | 4.943 | 1.525 | 0.144 | 3.864 |     | 13.829   | 11.804   |

- Xeon CPU 2.6GHz
- 2x6 cores
PL compliant discrete gradient

| Dataset     | $|DMT(f)|$ | $|PL(f)|$ | Alg. 1 | $PL(f)$ | S-M  | S-S | $MS(f)$ | $MS'(f)$ |
|-------------|-----|-----|-------|--------|-------|------|-----|---------|---------|
| Dragon      | 1,118 | 318 | 0.016 | 0.018  | 0.004 | 0    |     | 0.074   | 0.072   |
| EthaneDiol  | 6,109 | 93  | 4.943 | 1.525  | 0.144 | 3.864|     | 13.829  | 11.804  |

- Xeon CPU 2.6GHz
- 2x6 cores
Unified simplification

• Pre-simplify the PL data only
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]
Unified simplification

• Pre-simplify the PL data only
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]

• Several algorithms
  – [AGH09, BLW12, EMP06, TP12]
  – [TP12] arbitrary feature selection
Unified simplification

- Pre-simplify the PL data only
  \[ f : \mathcal{M} \rightarrow \mathbb{R} \]

- Several algorithms
  - [AGH09, BLW12, EMP06, TP12]
  - [TP12] arbitrary feature selection

- Outcome
  - Consistently simplified abstractions
Cached triangulation
Traversal specifications

• Topological data analysis
  – « Just » mesh traversals
Traversals specifications

• Topological data analysis
  – « Just » mesh traversals
  – Boundary tests
  – Efficient accesses
    • Skeleton, face, co-face, link, star
Traversals specifications

- Topological data analysis
  - "Just" mesh traversals
  - Boundary tests
  - Efficient accesses
    - Skeleton, face, co-face, link, star

- For each $i$-simplex
  - Access its $k$-faces and $l$-co-faces
    - $0 \leq k \leq i \leq l \leq d$
Cached lookups

• Explicit mode
  – Pre-condition functions
Cached lookups

• Explicit mode
  – Pre-condition functions
    • Specify traversals
Cached lookups

- Explicit mode
  - Pre-condition functions
    - Specify traversals
    - Populate data structure
Cached lookups

• Explicit mode
  – Pre-condition functions
    • Specify traversals
    • Populate data structure
  – Traversal: constant time lookups
Cached lookups

• Explicit mode
  – Pre-condition functions
    • Specify traversals
    • Populate data structure
  – Traversal: constant time lookups

• Implicit mode
Cached lookups

• Explicit mode
  – Pre-condition functions
    • Specify traversals
    • Populate data structure
  – Traversal: constant time lookups

• Implicit mode
  – Compatible API 2D/3D
  – No memory overhead

Vertex map:
\[ V(i,j) = j \times w + i \]

Edge map:
\[ E_H(i,j) = Q(i,j) \]
\[ E_V(i,j) = (w - 1) \times h + V(i,j) \]
\[ E_D(i,j) = (w - 1) \times h + (h - 1) \times w + Q(i,j) \]

Triangle map:
\[ T_L(i,j) = 2 \times Q(i,j) \]
\[ T_R(i,j) = 2 \times Q(i,j) + 1 \]
# Cached lookups

<table>
<thead>
<tr>
<th>Traversal Example</th>
<th>Precondition Time</th>
<th>Memory Footprint</th>
<th>Memory Overhead</th>
<th>e-Time</th>
<th>Cache Speedup</th>
<th>i-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary Vertices (0)</td>
<td>1.082</td>
<td>48</td>
<td>68 %</td>
<td>0.001</td>
<td>997</td>
<td>0.003</td>
</tr>
<tr>
<td>Boundary Edges (1)</td>
<td>1.568</td>
<td>101</td>
<td>142 %</td>
<td>0.008</td>
<td>200</td>
<td>0.027</td>
</tr>
<tr>
<td>Boundary Triangles (2)</td>
<td>1.099</td>
<td>50</td>
<td>70 %</td>
<td>0.011</td>
<td>100</td>
<td>0.038</td>
</tr>
<tr>
<td>1-skeleton (0)</td>
<td>0.200</td>
<td>23</td>
<td>32 %</td>
<td>0.025</td>
<td>9</td>
<td>0.107</td>
</tr>
<tr>
<td>Vertex Link (0)</td>
<td>1.391</td>
<td>105</td>
<td>148 %</td>
<td>0.035</td>
<td>41</td>
<td>0.135</td>
</tr>
<tr>
<td>Edge Link (1)</td>
<td>0.512</td>
<td>128</td>
<td>180 %</td>
<td>0.129</td>
<td>5</td>
<td>1.857</td>
</tr>
<tr>
<td>Triangle Link (2)</td>
<td>1.135</td>
<td>67</td>
<td>94 %</td>
<td>0.551</td>
<td>3</td>
<td>0.810</td>
</tr>
<tr>
<td>Edge FaceCoFace (1)</td>
<td>1.800</td>
<td>155</td>
<td>218 %</td>
<td>0.280</td>
<td>7</td>
<td>0.477</td>
</tr>
<tr>
<td>Triangle FaceCoFace (2)</td>
<td>1.310</td>
<td>119</td>
<td>168 %</td>
<td>0.447</td>
<td>4</td>
<td>0.451</td>
</tr>
</tbody>
</table>

- 64^3 grid, 1.25 Mtets, 71 MB (binary VTU)
### Cached lookups

<table>
<thead>
<tr>
<th>Traversal Example</th>
<th>Precondition Time</th>
<th>Memory Footprint</th>
<th>Memory Overhead</th>
<th>e-Time</th>
<th>Cache Speedup</th>
<th>i-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary Vertices (0)</td>
<td>1.082</td>
<td>48</td>
<td>68 %</td>
<td>0.001</td>
<td>997</td>
<td>0.003</td>
</tr>
<tr>
<td>Boundary Edges (1)</td>
<td>1.568</td>
<td>101</td>
<td>142 %</td>
<td>0.008</td>
<td>200</td>
<td>0.027</td>
</tr>
<tr>
<td>Boundary Triangles (2)</td>
<td>1.099</td>
<td>50</td>
<td>70 %</td>
<td>0.011</td>
<td>100</td>
<td>0.038</td>
</tr>
<tr>
<td>1-skeleton (0)</td>
<td>0.200</td>
<td>23</td>
<td>32 %</td>
<td>0.025</td>
<td>9</td>
<td>0.107</td>
</tr>
<tr>
<td>Vertex Link (0)</td>
<td>1.391</td>
<td>105</td>
<td>148 %</td>
<td>0.035</td>
<td>41</td>
<td>0.135</td>
</tr>
<tr>
<td>Edge Link (1)</td>
<td>0.512</td>
<td>128</td>
<td>180 %</td>
<td>0.129</td>
<td>5</td>
<td>1.857</td>
</tr>
<tr>
<td>Triangle Link (2)</td>
<td>1.135</td>
<td>67</td>
<td>94 %</td>
<td>0.551</td>
<td>3</td>
<td>0.810</td>
</tr>
<tr>
<td>Edge FaceCoFace (1)</td>
<td>1.800</td>
<td>155</td>
<td>218 %</td>
<td>0.280</td>
<td>7</td>
<td>0.477</td>
</tr>
<tr>
<td>Triangle FaceCoFace (2)</td>
<td>1.310</td>
<td>119</td>
<td>168 %</td>
<td>0.447</td>
<td>4</td>
<td>0.451</td>
</tr>
</tbody>
</table>

- $64^3$ grid, 1.25 Mtets, 71 MB (binary VTU)
### Cached lookups

<table>
<thead>
<tr>
<th>Traversal Example</th>
<th>Precondition Time</th>
<th>Memory Footprint</th>
<th>Memory Overhead</th>
<th>e-Time</th>
<th>Cache Speedup</th>
<th>i-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary Vertices (0)</td>
<td>1.082</td>
<td>48</td>
<td>68 %</td>
<td>0.001</td>
<td>997</td>
<td>0.003</td>
</tr>
<tr>
<td>Boundary Edges (1)</td>
<td>1.568</td>
<td>101</td>
<td>142 %</td>
<td>0.008</td>
<td>200</td>
<td>0.027</td>
</tr>
<tr>
<td>Boundary Triangles (2)</td>
<td>1.099</td>
<td>50</td>
<td>70 %</td>
<td>0.011</td>
<td>100</td>
<td>0.038</td>
</tr>
<tr>
<td>1-skeleton (0)</td>
<td>0.200</td>
<td>23</td>
<td>32 %</td>
<td>0.025</td>
<td>9</td>
<td>0.107</td>
</tr>
<tr>
<td>Vertex Link (0)</td>
<td>1.391</td>
<td>105</td>
<td>148 %</td>
<td>0.035</td>
<td>41</td>
<td>0.135</td>
</tr>
<tr>
<td>Edge Link (1)</td>
<td>0.512</td>
<td>128</td>
<td>180 %</td>
<td>0.129</td>
<td>5</td>
<td>1.857</td>
</tr>
<tr>
<td>Triangle Link (2)</td>
<td>1.135</td>
<td>67</td>
<td>94 %</td>
<td>0.551</td>
<td>3</td>
<td>0.810</td>
</tr>
<tr>
<td>Edge FaceCoFace (1)</td>
<td>1.800</td>
<td>155</td>
<td>218 %</td>
<td>0.280</td>
<td>7</td>
<td>0.477</td>
</tr>
<tr>
<td>Triangle FaceCoFace (2)</td>
<td>1.310</td>
<td>119</td>
<td>168 %</td>
<td>0.447</td>
<td>4</td>
<td>0.451</td>
</tr>
</tbody>
</table>

- $64^3$ grid, 1.25 Mtets, 71 MB (binary VTU)
## Cached lookups

<table>
<thead>
<tr>
<th>Traversal Example</th>
<th>Precondition Time</th>
<th>Memory Footprint</th>
<th>Memory Overhead</th>
<th>e-Time</th>
<th>Cache Speedup</th>
<th>i-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boundary Vertices (0)</td>
<td>1.082</td>
<td>48</td>
<td>68 %</td>
<td>0.001</td>
<td>997</td>
<td>0.003</td>
</tr>
<tr>
<td>Boundary Edges (1)</td>
<td>1.568</td>
<td>101</td>
<td>142 %</td>
<td>0.008</td>
<td>200</td>
<td>0.027</td>
</tr>
<tr>
<td>Boundary Triangles (2)</td>
<td>1.099</td>
<td>50</td>
<td>70 %</td>
<td>0.011</td>
<td>100</td>
<td>0.038</td>
</tr>
<tr>
<td>1-skeleton (0)</td>
<td>0.200</td>
<td>23</td>
<td>32 %</td>
<td>0.025</td>
<td>9</td>
<td>0.107</td>
</tr>
<tr>
<td>Vertex Link (0)</td>
<td>1.391</td>
<td>105</td>
<td>148 %</td>
<td>0.035</td>
<td>41</td>
<td>0.135</td>
</tr>
<tr>
<td>Edge Link (1)</td>
<td>0.512</td>
<td>128</td>
<td>180 %</td>
<td>0.129</td>
<td>5</td>
<td>1.857</td>
</tr>
<tr>
<td>Triangle Link (2)</td>
<td>1.135</td>
<td>67</td>
<td>94 %</td>
<td>0.551</td>
<td>3</td>
<td>0.810</td>
</tr>
<tr>
<td>Edge FaceCoFace (1)</td>
<td>1.800</td>
<td>155</td>
<td>218 %</td>
<td>0.280</td>
<td>7</td>
<td>0.477</td>
</tr>
<tr>
<td>Triangle FaceCoFace (2)</td>
<td>1.310</td>
<td>119</td>
<td>168 %</td>
<td>0.447</td>
<td>4</td>
<td>0.451</td>
</tr>
</tbody>
</table>

- $64^3$ grid, 1.25 Mtets, 71 MB (binary VTU)
Software architecture
Design motivations

• End users
  – Tight VTK integration
Design motivations

• End users
  – Tight VTK integration
    • Easily accessible
    • Rich IO
Design motivations

• End users
  – Tight VTK integration
    • Easily accessible
    • Rich IO
    • Advanced GUI (ParaView)
    • Python binding
Design motivations

• Developers
  – Ease of integration into pre-existing systems
Design motivations

• Developers
  – Ease of integration into pre-existing systems
  – Isolated computation layer
    • Highly templated, dependence-free functors (STL)
Design motivations

• Developers
  – Ease of integration into pre-existing systems
  – Isolated computation layer
    • Highly templated, dependence-free functors (STL)
    • Fully independent from VTK (*backdoor* pointers)
Design motivations

• Developers
  – Ease of integration into pre-existing systems
  – Isolated computation layer
    • Highly templated, dependence-free functors (STL)
    • Fully independent from VTK (*backdoor* pointers)

• Researchers
  – Easily extensible modular architecture
Design motivations

• Developers
  – Ease of integration into pre-existing systems
  – Isolated computation layer
    • Highly templated, dependence-free functors (STL)
    • Fully independent from VTK (*backdoor* pointers)

• Researchers
  – Easily extensible modular architecture
    • Automated module management
    • No IO, rendering or interaction work, *just topology*
Architecture

Base code functors
  ttk::MorseSmaleComplex
  ttk::PersistenceDiagram
  ttk::ReebSpace
  ttk::...

The Topology ToolKit
http://topology-tool-kit.github.io/
Architecture

Base code functors
- ttk::MorseSmaleComplex
- ttk::PersistenceDiagram
- ttk::ReebSpace
- ttk:: ...

Pure C++ Access
Architecture

VTK wrappers

Base code functors

```cpp
ttk::MorseSmaleComplex

ttk::PersistenceDiagram

ttk::ReebSpace

ttk::...
```

Pure C++ Access
Architecture

VTK wrappers
Base code functors

```
ttk::MorseSmaleComplex
```

```
ttk::PersistenceDiagram
```

```
ttk::ReebSpace
```

```
ttk::...
```

Pure C++ Access  VTK/C++ Access
Architecture

ParaView plugins
VTK wrappers
Base code functors

- ttk::MorseSmaleComplex
- ttk::PersistenceDiagram
- ttk::ReebSpace
- ttk:: ...

MorseSmaleComplex
PersistenceDiagram
ReebSpace
...

Pure C++ Access  VTK/C++ Access
Architecture
Architecture
Architecture

ParaView plugins

VTK wrappers

Base code functors

- ttk::MorseSmaleComplex
- ttk::PersistenceDiagram
- ttk::ReebSpace
- ttk::...

VTK/C++ Access

Pure C++ Access

Python Access

MorseSmaleComplex
PersistenceDiagram
ReebSpace
...

Command-line

- morseSmaleComplex
- persistenceDiagram
- reebSpace
- ...

VTK-based GUI

- morseSmaleComplexGui
- persistenceDiagramGui
- reebSpaceGui
- ...

Standalone programs
• Module management scripts
• Module management scripts
Software collection
Software collection
Software collection
Software collection
Software collection
Software collection
Software collection
User experience
User experience
User experience
User experience
User experience
User experience
Limitations
Limitations

• PL matching property
  – Only for interior critical points
Limitations

• PL matching property
  – Only for interior critical points

• Data pre-simplification [TP12]
  – Only (0, 1) and (d-1, d) pairs
Limitations

• PL matching property
  – Only for interior critical points

• Data pre-simplification [TP12]
  – Only (0, 1) and (d-1, d) pairs

• Triangulation implicit mode
  – More cells to process
Since the release
Experience and feedback

• Internal usage
  – 2 years, 6 papers, 5 grad students
Experience and feedback

• Internal usage
  – 2 years, 6 papers, 5 grad students
    • Skills: C++, no rendering, no GUI
    • Fast and efficient prototypes
Experience and feedback

• Internal usage
  – 2 years, 6 papers, 5 grad students
    • Skills: C++, no rendering, no GUI
    • Fast and efficient prototypes

• Reception
  – O’Reilly daily 4 short links
  – 37k page views
Experience and feedback

• Internal usage
  – 2 years, 6 papers, 5 grad students
    • Skills: C++, no rendering, no GUI
    • Fast and efficient prototypes

• Reception
  – O’Reilly daily 4 short links
  – 37k page views
  – Third party project, user support
Experience and feedback
Experience and feedback
Experience and feedback
Experience and feedback
Improvements

• Releases: matching ParaView’s
Improvements

• Releases: matching ParaView’s

• ChangeLog
  – Improved performances (triangulation, merge trees [GFJ17])
  – New features (Wassertein distances)
  – In-situ support (Catalyst)
  – Bug fixes
Improvements

• Releases: matching ParaView’s

• ChangeLog
  – Improved performances (triangulation, merge trees [GFJ17])
  – New features (Wassertein distances)
  – In-situ support (Catalyst)
  – Bug fixes
  – Major CMake improvements
    • `find_package()`
Next steps

• End user documentation
  – More tutorials, application use cases
Next steps

• End user documentation
  – More tutorials, application use cases
• Teaching platform
Next steps

- End user documentation
  - More tutorials, application use cases
- Teaching platform
- Progress on new features
Next steps

• End user documentation
  – More tutorials, application use cases
• Teaching platform
• Progress on new features
• Community call
Next steps

• End user documentation
  – More tutorials, application use cases
• Teaching platform
• Progress on new features
• Community call
  – Benchmark implementation?
Next steps

• End user documentation
  – More tutorials, application use cases
• Teaching platform
• Progress on new features
• Community call
  – Benchmark implementation?
  – Showcase towards other communities?
Next steps

• End user documentation
  – More tutorials, application use cases
• Teaching platform
• Progress on new features
• Community call
  – Benchmark implementation?
  – Showcase towards other communities?
  – Vector data? Tensor data?
Conclusion
Conclusion

• Open source software platform for topological data analysis
  – PL compliant discrete gradient
  – Cached triangulation
  – Flexible software architecture
Conclusion

• Open source software platform for topological data analysis
  – PL compliant discrete gradient
  – Cached triangulation
  – Flexible software architecture

• End users: ParaView and Python
  – Many online video tutorials
Conclusion

• Open source software platform for topological data analysis
  – PL compliant discrete gradient
  – Cached triangulation
  – Flexible software architecture

• End users: ParaView and Python
  – Many online video tutorials

• Developers: VTK/C++ and C++
Conclusion

- Open source software platform for topological data analysis
  - PL compliant discrete gradient
  - Cached triangulation
  - Flexible software architecture
- End users: ParaView and Python
  - Many online video tutorials
- Developers: VTK/C++ and C++
- Researchers: easily extensible
Thanks!

• Discussions
  – Anonymous reviewers, Attila Gyulassy (Utah), Julien Jomier (Kitware), Joachim Poudreux (Kitware), Will Schroeder (Kitware)
Thanks!

• Discussions
  – Anonymous reviewers, Attila Gyulassy (Utah), Julien Jomier (Kitware), Joachim Pouderoux (Kitware), Will Schroeder (Kitware)

• Contributions
  – Guillaume Favelier (Sorbonne), Charles Gueunet (Kitware), Michael Michaux (Sorbonne), Josh Levine (Arizona)
Thanks!

• Discussions
  - Anonymous reviewers, Attila Gyulassy (Utah), Julien Jomier (Kitware), Joachim Pouderoux (Kitware), Will Schroeder (Kitware)

• Contributions
  - Guillaume Favelier (Sorbonne), Charles Gueunet (Kitware), Michael Michaux (Sorbonne), Josh Levine (Arizona)
  - Maxime Soler (Total), Will Usher (Utah)
Reference

« The Topology ToolKit »
J. Tierny, G. Favelier, J. Levine, C. Gueunet, M. Michaux
IEEE Transactions on Visualization and Computer Graphics
Proc. Of IEEE VIS 2017
Best paper honorable mention award

http://topology-tool-kit.github.io
http://github.com/topology-tool-kit
Internship offers

• 1 or 2 master students
  – Ph.D. thesis (CIFRE with Kitware)

• http://lip6.fr/Julien.Tierny