ACM SIGSPATIAL 2020

Topology-Preserving Terrain Simplification

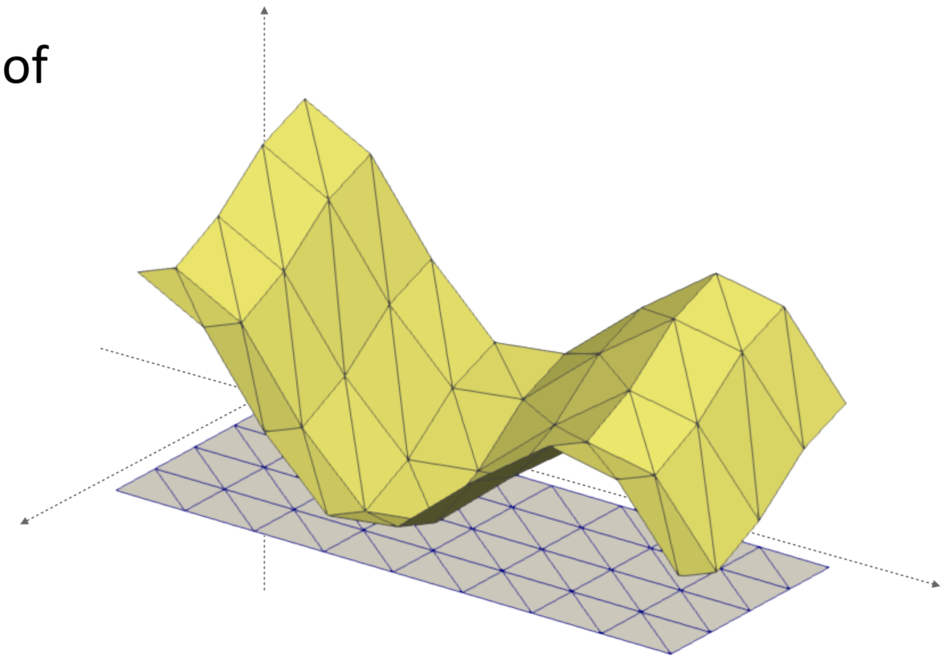
Ulderico Fugacci
CNR - IMATI

Michael Kerber
TU Graz

Hugo Manet
ENS Paris

Motivations

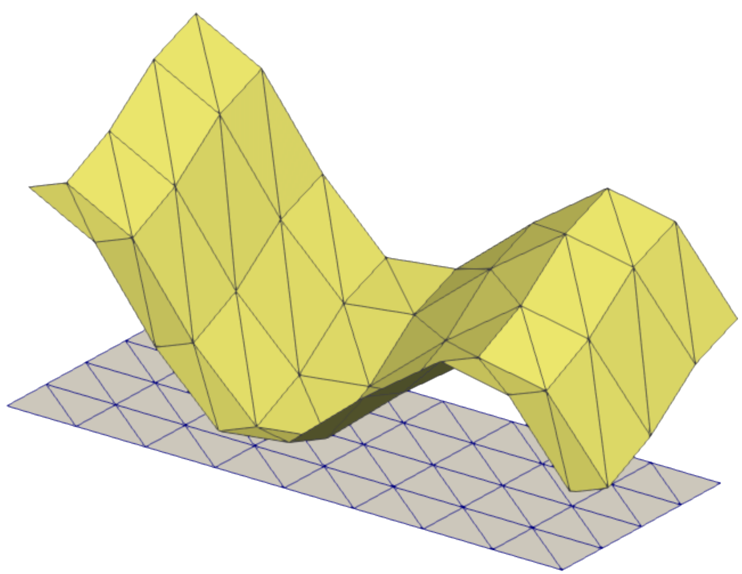
A ***terrain T*** consists of



- a ***triangulation*** of a ***compact polygonal region in*** $\mathbb{R}^2$ on a set of vertices V (possibly with internal vertices)
- endowed with an ***injective scalar function t: V*** $\longrightarrow \mathbb{R}$ called ***height function***

Motivations

By filtering *T* through the height function *t*,

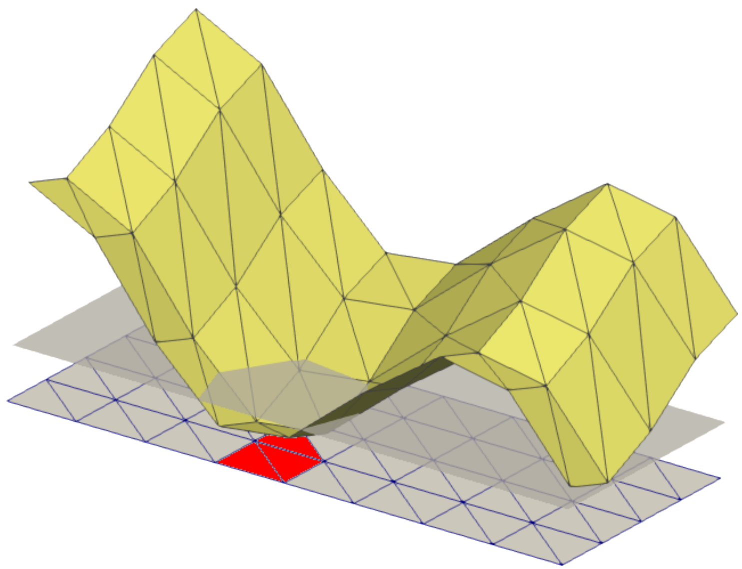


Persistent Homology enables the study of *terrain morphology*

Changes in homology are in correspondence with ***critical points*** of *t*

Motivations

By filtering *T* through the height function *t*,

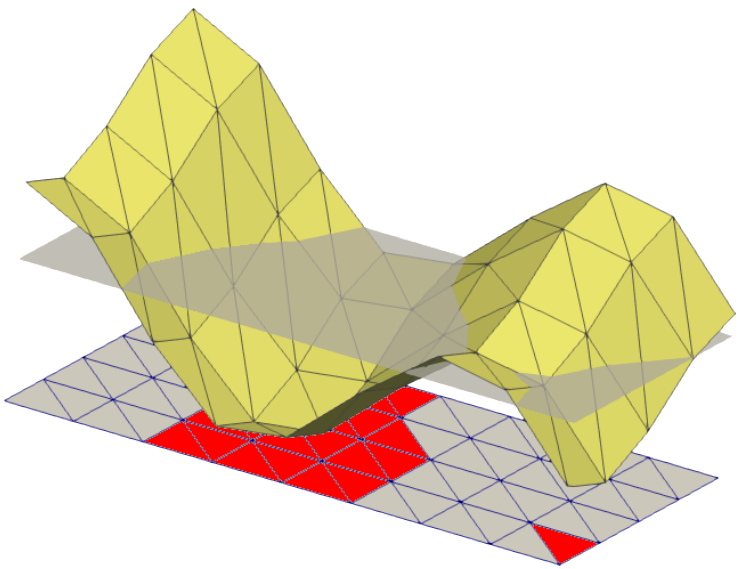


Persistent Homology enables the study of *terrain morphology*

Changes in homology are in correspondence with ***critical points*** of *t*

Motivations

By filtering *T* through the height function *t*,

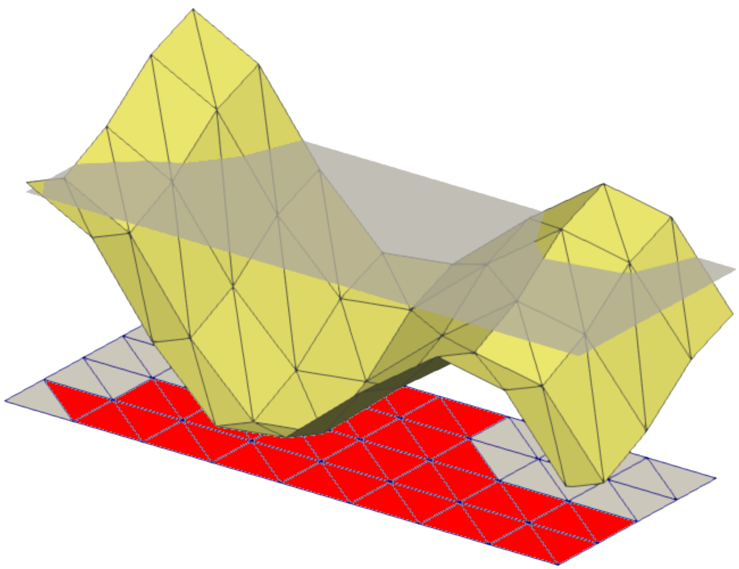


Persistent Homology enables the study of *terrain morphology*

Changes in homology are in correspondence with ***critical points*** of *t*

Motivations

By filtering *T* through the height function *t*,

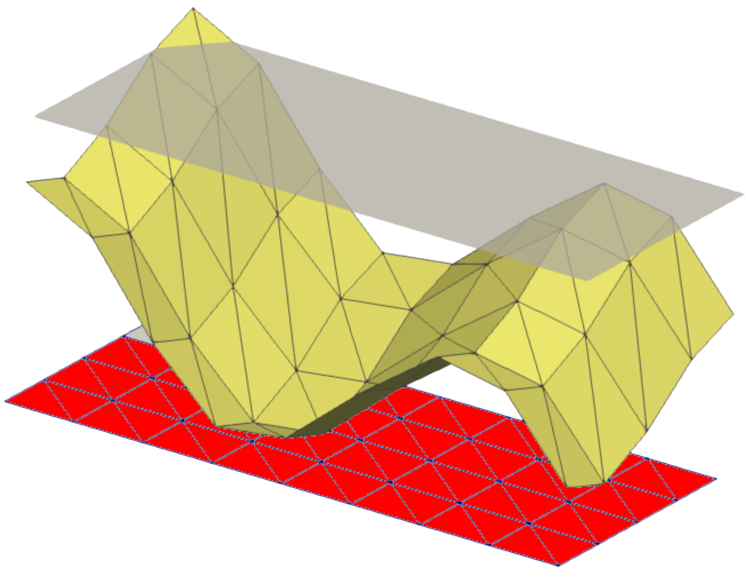


Persistent Homology enables the study of *terrain morphology*

Changes in homology are in correspondence with ***critical points*** of *t*

Motivations

By filtering *T* through the height function *t*,

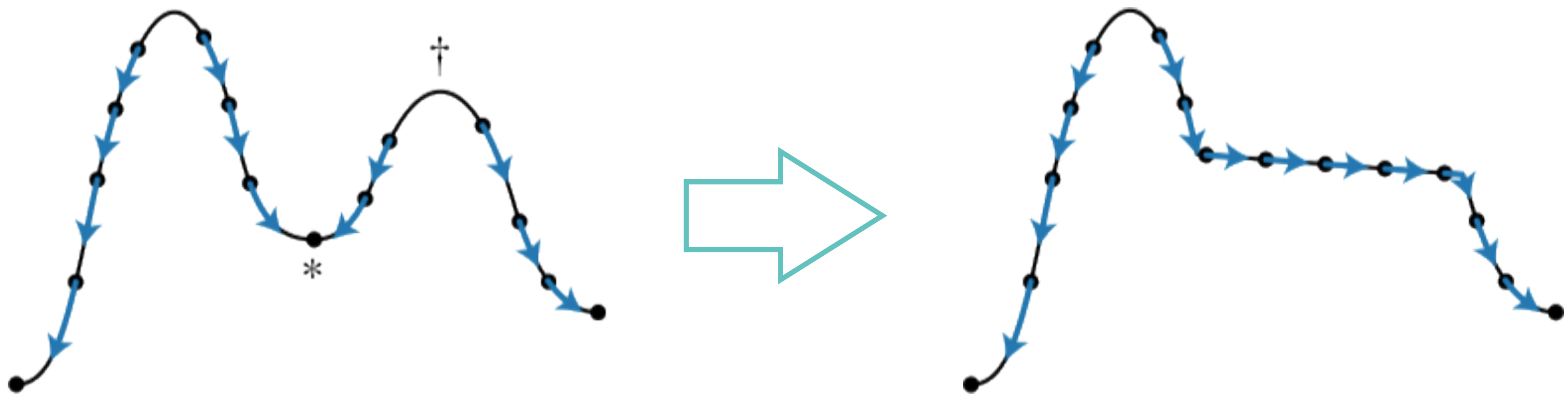


Persistent Homology enables the study of *terrain morphology*

Changes in homology are in correspondence with ***critical points*** of *t*

Motivations

Topological terrain simplification consists in the ***removal*** of



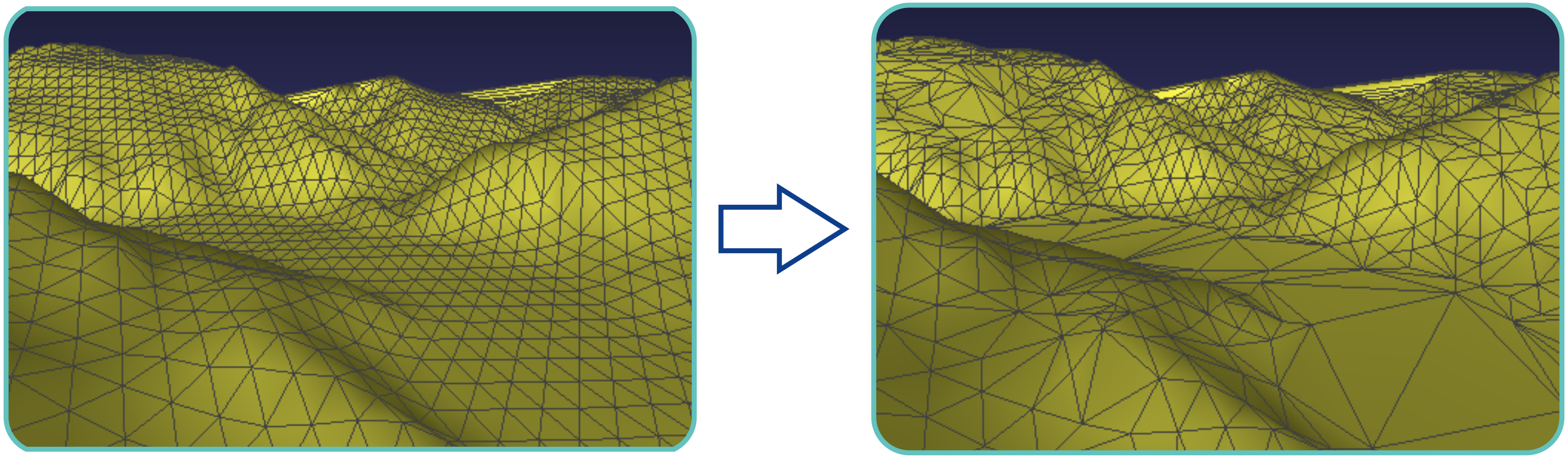
Images from [Bauer et al. 2012]

low-persistence pairs of critical points considered as ***topological noise***

Output can be affected by *geometrical oversampling*

Motivations

We propose a different approach



- ***Reduce*** the ***size*** of the terrain
- ***Maintain*** favorable properties:
  - ***Persistent homology*** (i.e. critical points)
  - ***Geometric closeness*** to the original surface

Simplification Algorithm

Problem:

Given $\varepsilon > 0$, a base terrain B and a terrain T with $\| B - T \|_\infty \leq \varepsilon$, compute a terrain S with fewer vertices than T, such that $\| B - S \|_\infty \leq \varepsilon$ and S has the ***same persistent homology*** of T

Algorithm:

Initialize $S \leftarrow T$ and $C \leftarrow$ *Vertices of T*

`while` $C \neq \varnothing$,

pick a vertex v in C (uniformly at random) and *remove* it from C

`if` ***link(v)*** can be ***re-triangulated maintaining*** the condition on ***persistent homology*** and ***L_∞-distance*** `then`

remove v from S and *re-triangulate* its link by the found triangulation

insert the vertices of *link(v)* in C (if not already contained)

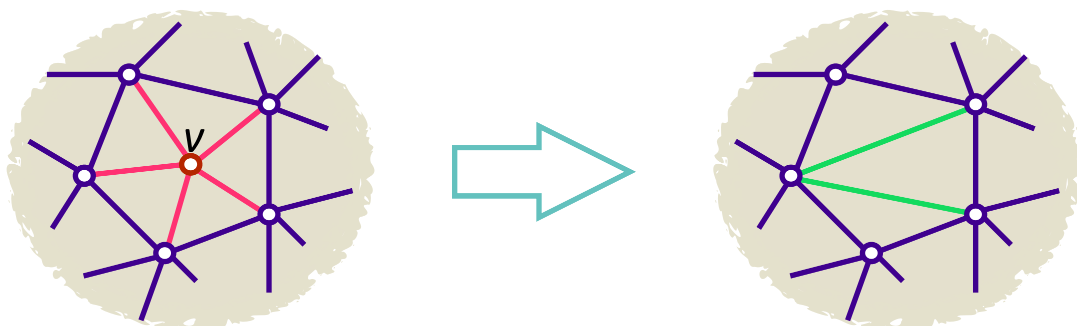
`return` S

Simplification Algorithm

Theorem:

Given a vertex *v* of *S*, a ***re-triangulation of its link maintains*** the condition on:

- ***L_∞-distance if*** all the ***diagonals*** and the ***triangles*** of the re-triangulated link are ***L_∞-aware***
- ***persistent homology iff*** *v* is ***not a critical point*** and all the ***diagonals*** of the re-triangulated link are ***persistence-aware***



A vertex satisfying the above conditions is called ***removable***

Simplification Algorithm

L_∞-Awareness:

Let *u*, *v*, *w* be vertices of *S*,

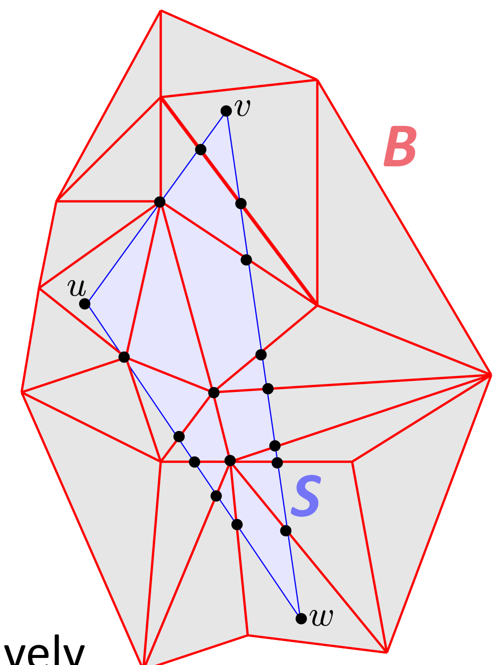
- *uv* is called ***L_∞-aware*** if for any point $x = \lambda u + (1-\lambda)v$ of *uv*,

$$|(\lambda s(u) + (1-\lambda)s(v)) - b(x)| \leq \varepsilon$$

- *uvw* is called ***L_∞-aware*** if for any point $x = \lambda u + \mu v + (1-\lambda-\mu)w$ of *uvw*,

$$|(\lambda s(u) + \mu s(v) + (1-\lambda-\mu)s(w)) - b(x)| \leq \varepsilon$$

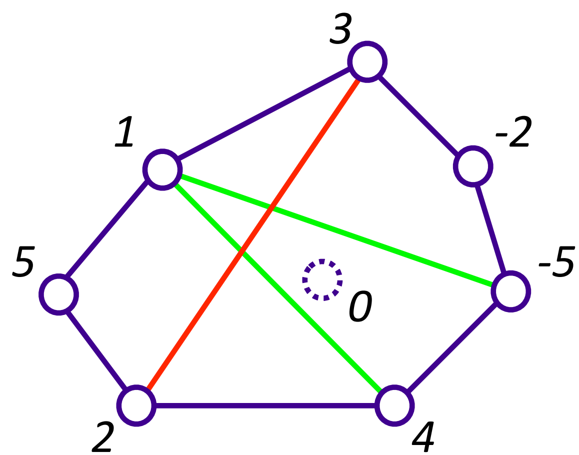
where *s* and *b* denote the height functions of *S* and *B*, respectively



Simplification Algorithm

Persistence-Awareness:

Let *v* be a vertex of *S* with two adjacent vertices *u*, *w* such that *s(u) < s(w)*, *uw* is called ***persistence-aware*** if one of the following conditions holds:

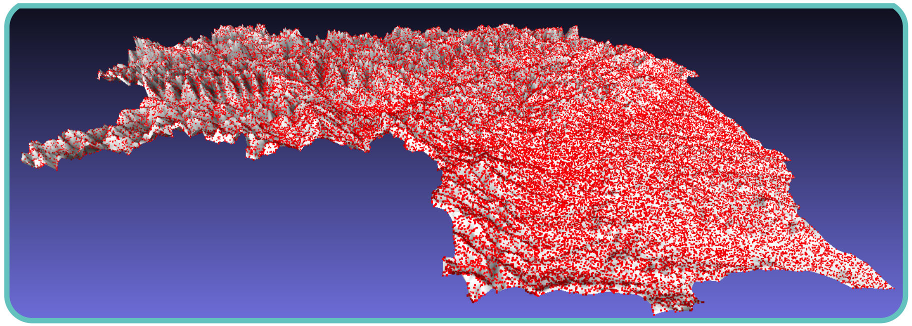


- ***s(u) < s(v) < s(w)***
- ***s(w) < s(v)*** and there is a path on *link(v)* from *u* to *w* with ***maximal height s(w)***
- ***s(v) < s(u)*** and there is a path on *link(v)* from *u* to *w* with ***minimal height s(u)***

Implementation & Experiments

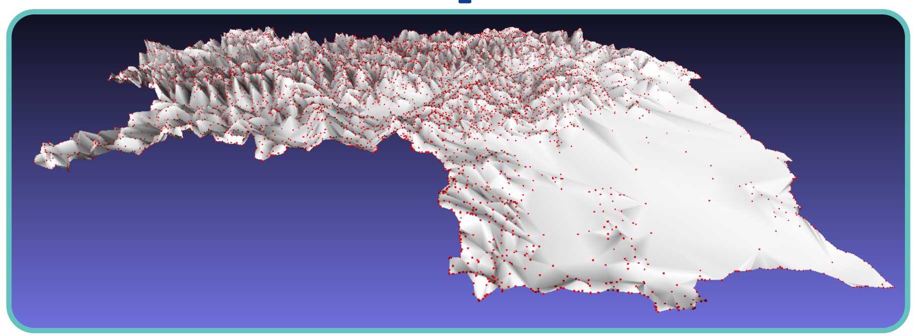
Based on the proposed simplification, an algorithm has been ***implemented in C++*** and it is available in a ***public repository*** (https://bitbucket.org/mkerber/terrain simplification)

Given an ***input terrain B*** and some **$\varepsilon > 0$**, the algorithm returns a ***terrain S*** that is ***ε-close to B in L_∞-distance*** and has the ***smallest number of critical points possible***



Input

Output

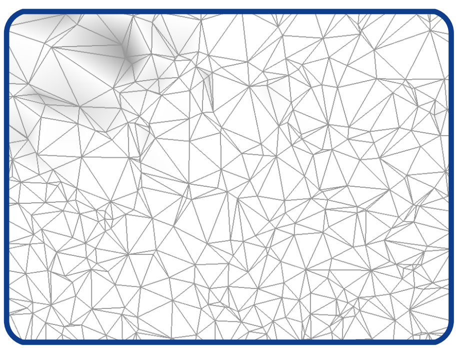
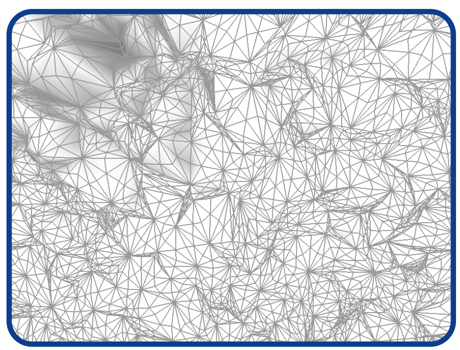
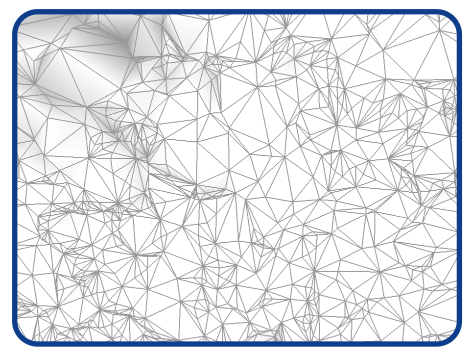


Implementation & Experiments

Based on the proposed simplification, an algorithm has been ***implemented in C++*** and it is available in a ***public repository*** (https://bitbucket.org/mkerber/terrain simplification)

Given an ***input terrain B*** and some **ε > 0**, the algorithm returns a ***terrain S*** that is ***ε-close to B in L_∞-distance*** and has the ***smallest number of critical points possible***

- Apply an improved version of the ***topological simplification*** of [Bauer et al. 2012] on *(B,ε)* to get a terrain *T* with the specified properties

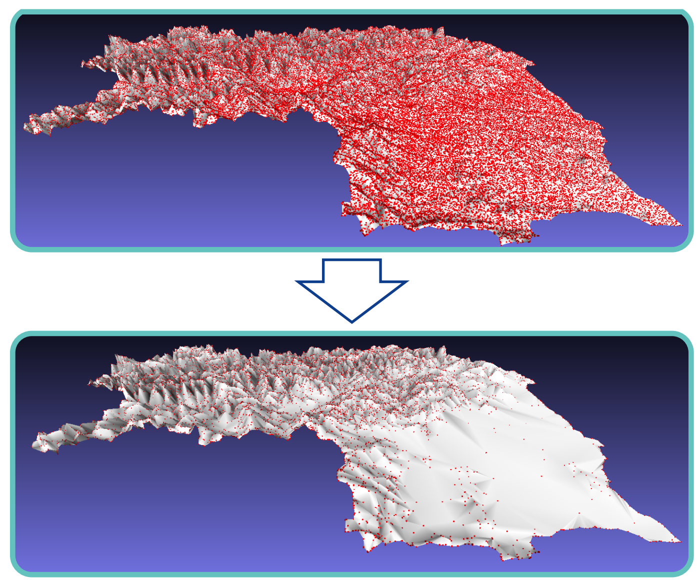




Implementation & Experiments

Based on the proposed simplification, an algorithm has been ***implemented in C++*** and it is available in a ***public repository*** (https://bitbucket.org/mkerber/terrain simplification)

Given an ***input terrain B*** and some **ε > 0**, the algorithm returns a ***terrain S*** that is ***ε-close to B in L_∞-distance*** and has the ***smallest number of critical points possible***

- Apply an improved version of the ***topological simplification*** of [Bauer et al. 2012] on *(B,ε)* to get a terrain *T* with the specified properties
- Apply our ***reduction strategy*** on *(B,T,ε)* to get a smaller terrain *S* with the same properties

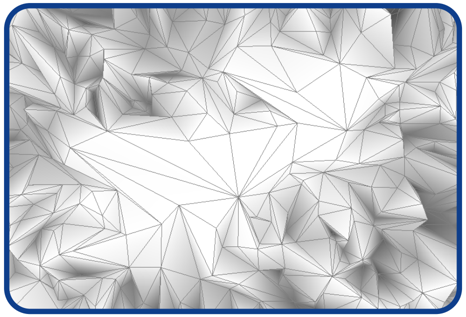


Implementation & Experiments

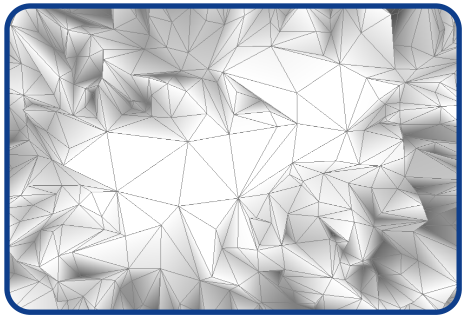
Based on the proposed simplification, an algorithm has been ***implemented in C++*** and it is available in a ***public repository*** (https://bitbucket.org/mkerber/terrain simplification)

Given an ***input terrain B*** and some **$\varepsilon > 0$**, the algorithm returns a ***terrain S*** that is ***ε-close to B in L_∞-distance*** and has the ***smallest number of critical points possible***

- Apply an improved version of the ***topological simplification*** of [Bauer et al. 2012] on *(B,ε)* to get a terrain *T* with the specified properties



- Apply our ***reduction strategy*** on *(B,T,ε)* to get a smaller terrain *S* with the same properties



- Apply a ***mesh improvement*** algorithm on *(B,S,ε)*

Implementation & Experiments

Benchmark Results for Styria Dataset:

I	S	C	O	T
10K	17K	407	4100.6 (±9.6)	8.90 (±0.18)
20K	33K	543	6114.2 (±49.2)	19.96 (±0.37)
40K	64K	547	8654.2 (±27.2)	38.96 (±0.36)
80K	124K	597	11223.2 (±74.8)	81.19 (±1.79)
160K	239K	611	13160.2 (±65.8)	168.54 (±3.74)
320K	455K	637	15019.0 (±61.0)	342.40 (±6.12)

I: size of the input *(number of vertices)*
S: size of the output of the topological simplification algorithm *(number of vertices)*

C: critical points of the output of the topological simplification algorithm
O: size of the final output *(number of vertices)*
T: running time *(in seconds)*

Conclusions

In Summary:

Based on necessary and sufficient criteria for elementary operations in a terrain to preserve the persistent homology,

we have developed a method to reduce the total size of a topologically simplified terrain

overcoming a major drawback of previous simplification methods without giving up on its guarantees

Future Developments:

- *Investigate how far our output is from an optimal solution*
- *Extend our implementation to piecewise-linear functions on triangulated surfaces*
- *Find other interesting application domains*

Thank you

Ulderico Fugacci

CNR - Imati