Automatically repairing invalid polygons with a constrained triangulation

Hugo Ledoux Ken Arroyo Ohori Martijn Meijers

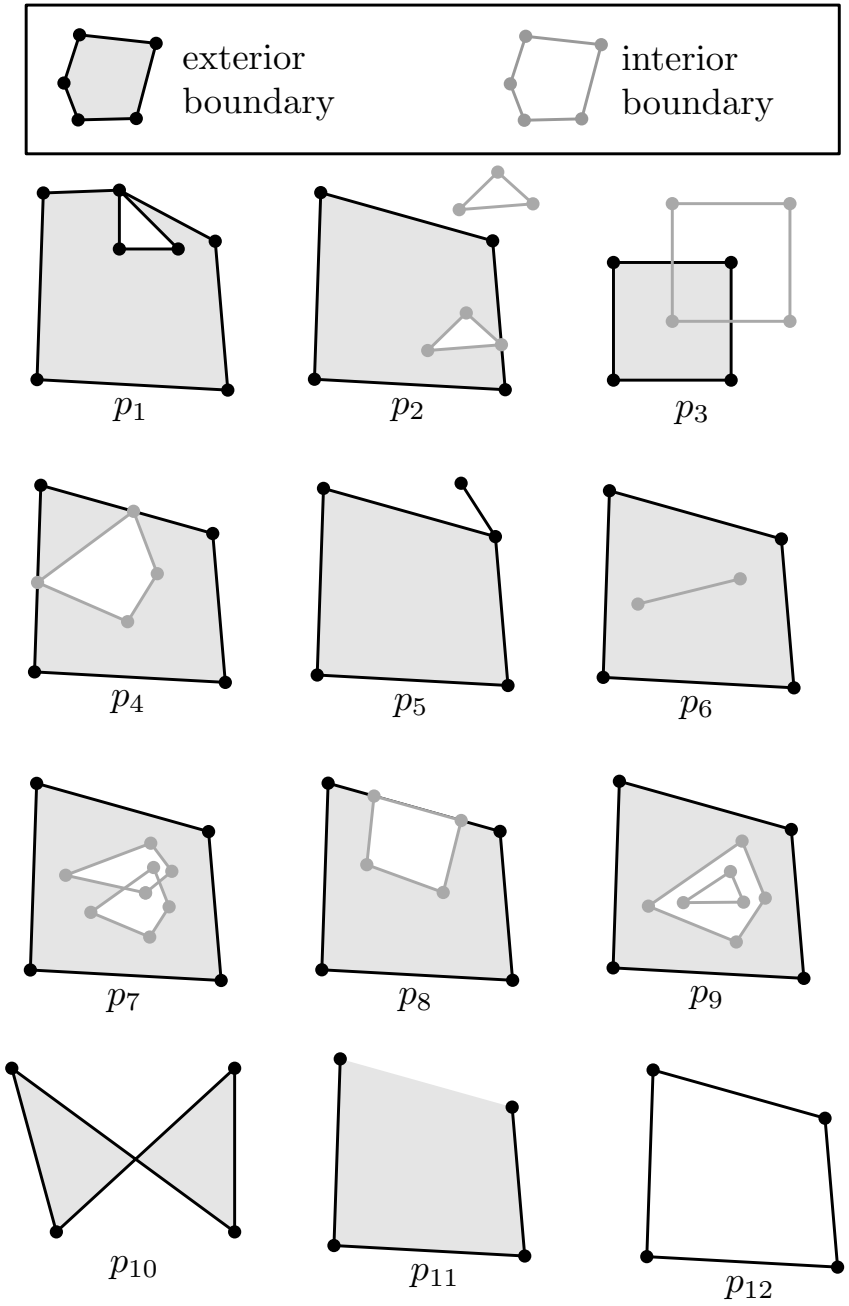
Technische Universiteit Delft

Agile 2012 (Avignon, France)
2012–04–25

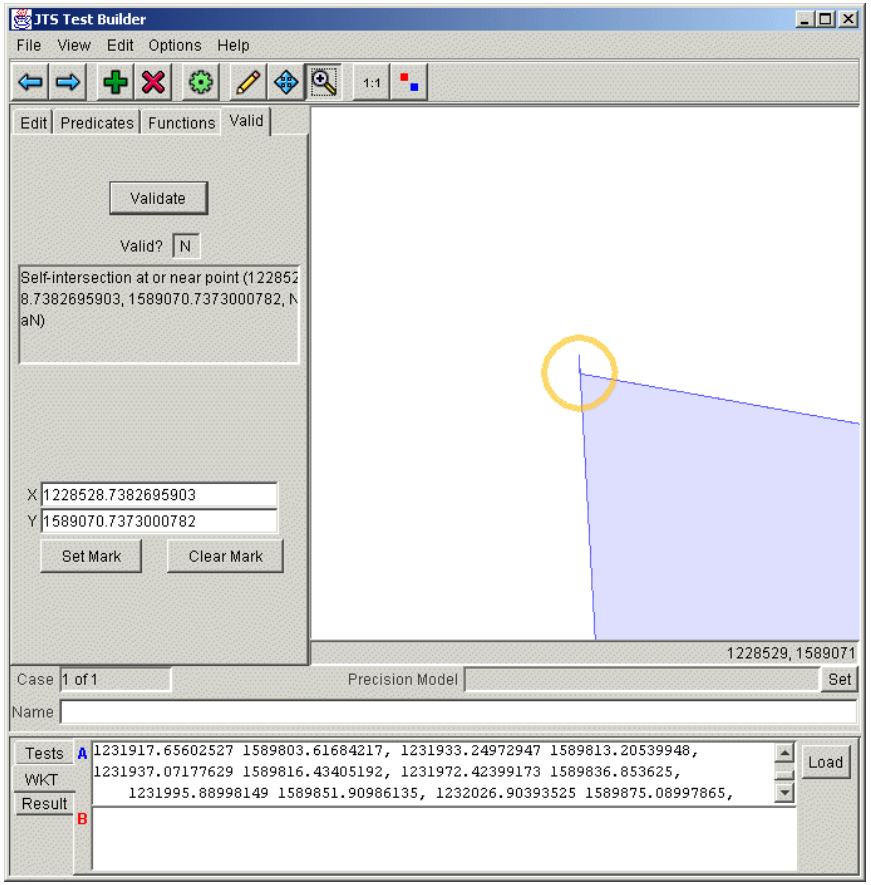
Validation of polygon = a solved problem

OGC Simple Features and ISO19107 rules:

1. no self-intersection
2. closed boundaries
3. rings can touch but not overlap
4. no duplicate points
5. no dangling edges
6. connected interior
7. etc



If it's broken then repair it. But how?



Errors are highlighted, but not repaired. One has to manually fix them.

The problem:

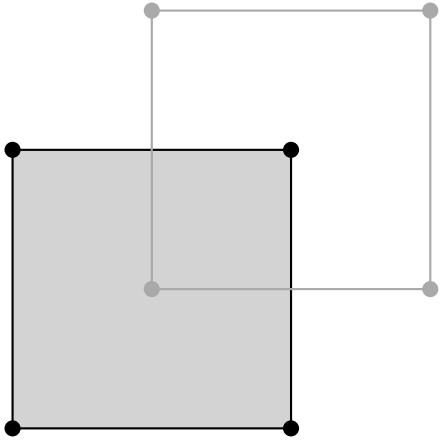
Given an invalid polygon, how to *automatically* repair it?

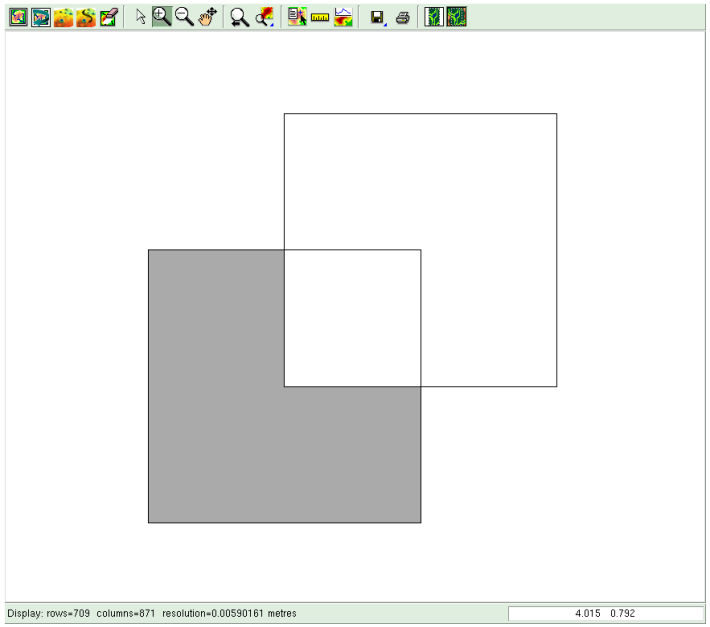
Related work

Surprisingly very little written on the topic:

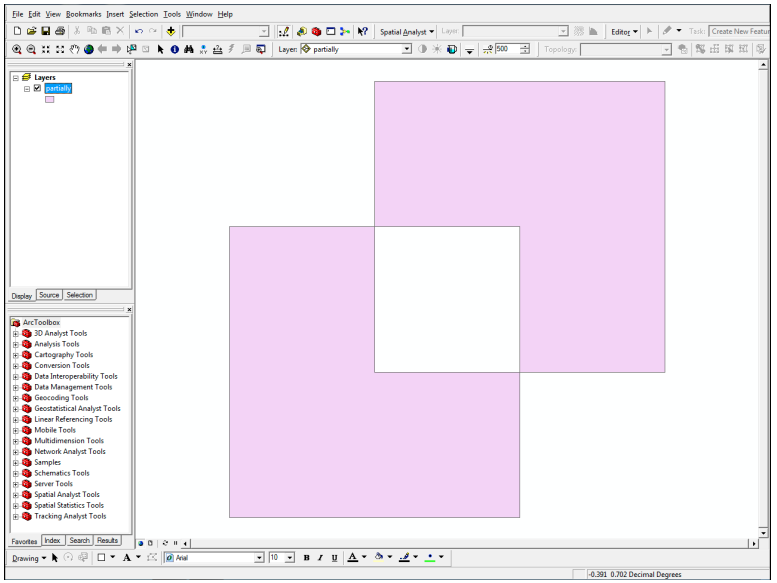
1. "cleaning" for display purposes
2. practitioners say: "buffer-by-0"
3. PostGIS 2.0's ST_MakeValid

Related work: cleaning for display purposes



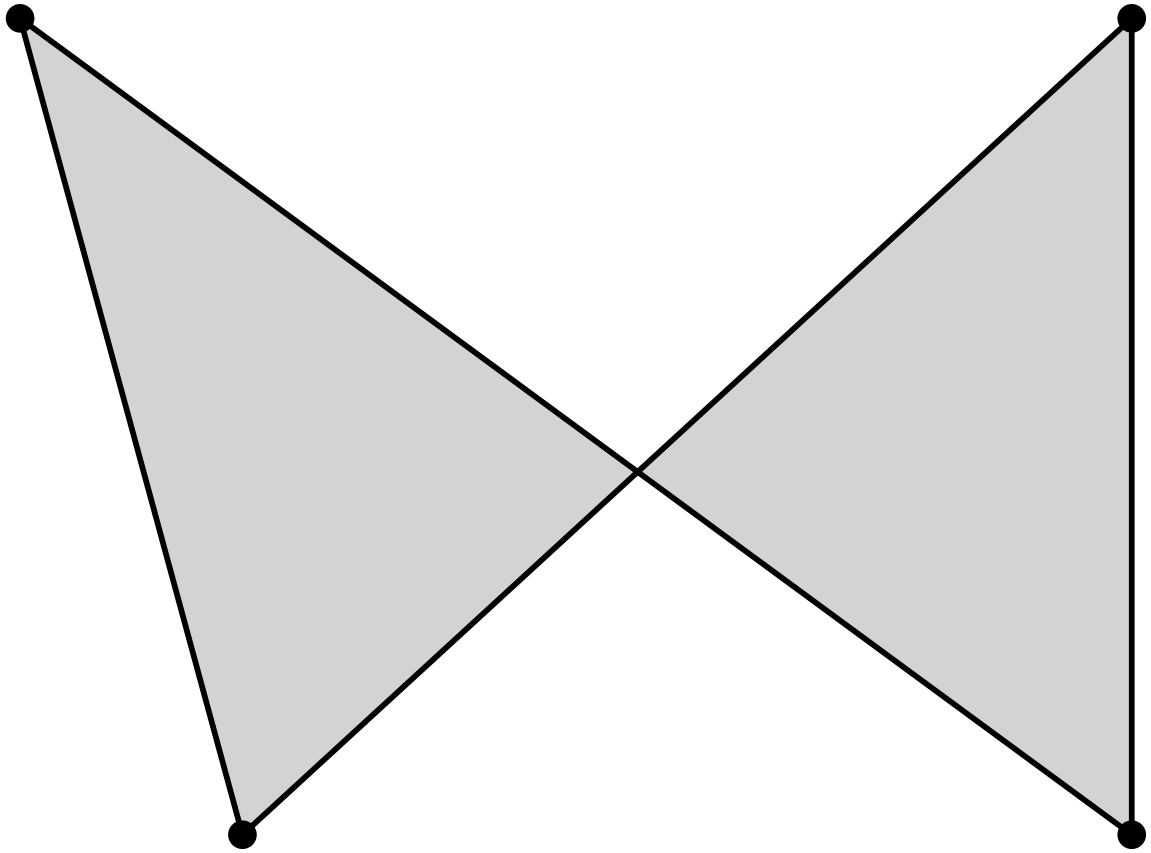


GRASS

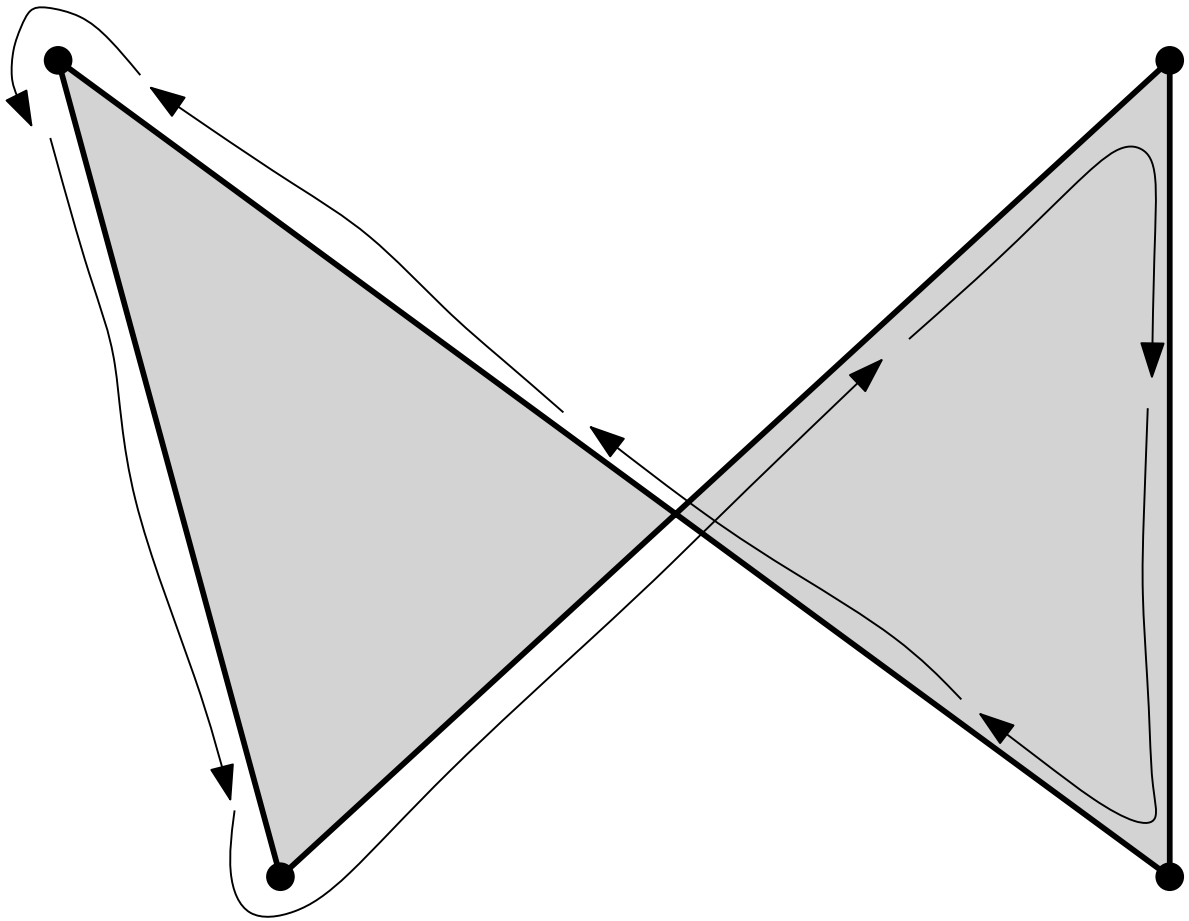


ArcGIS

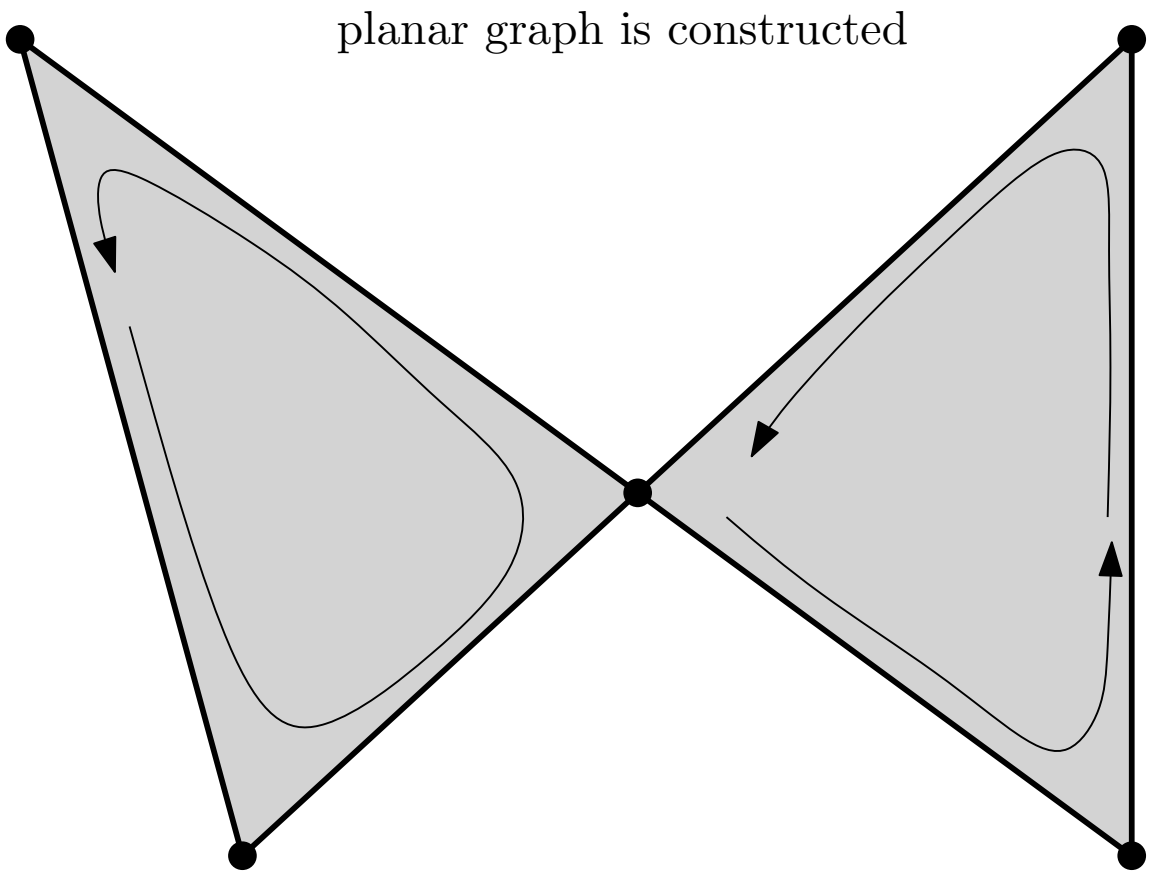
Related work: “buffer-by-0”



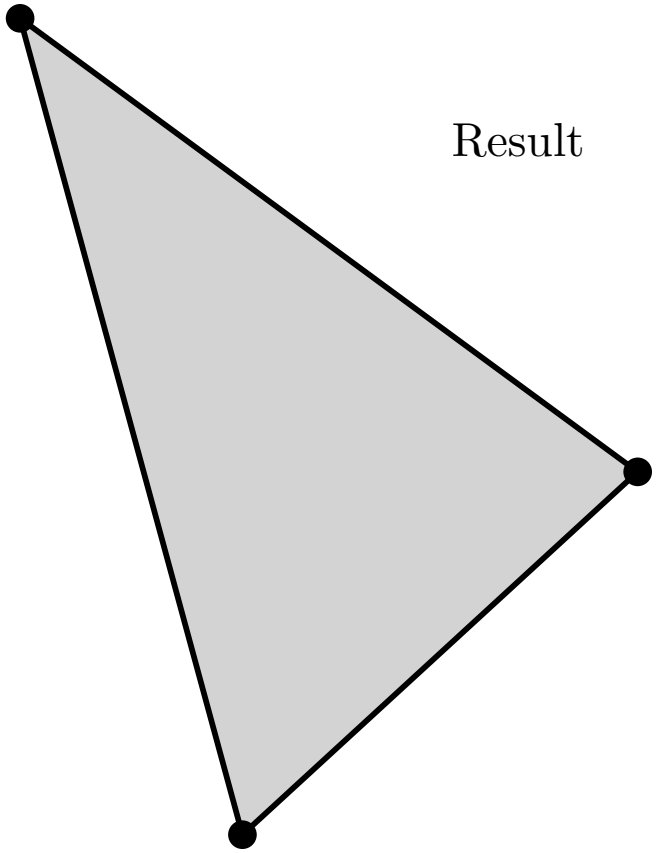
Related work: “buffer-by-0”



Related work: “buffer-by-0”



Related work: “buffer-by-0”



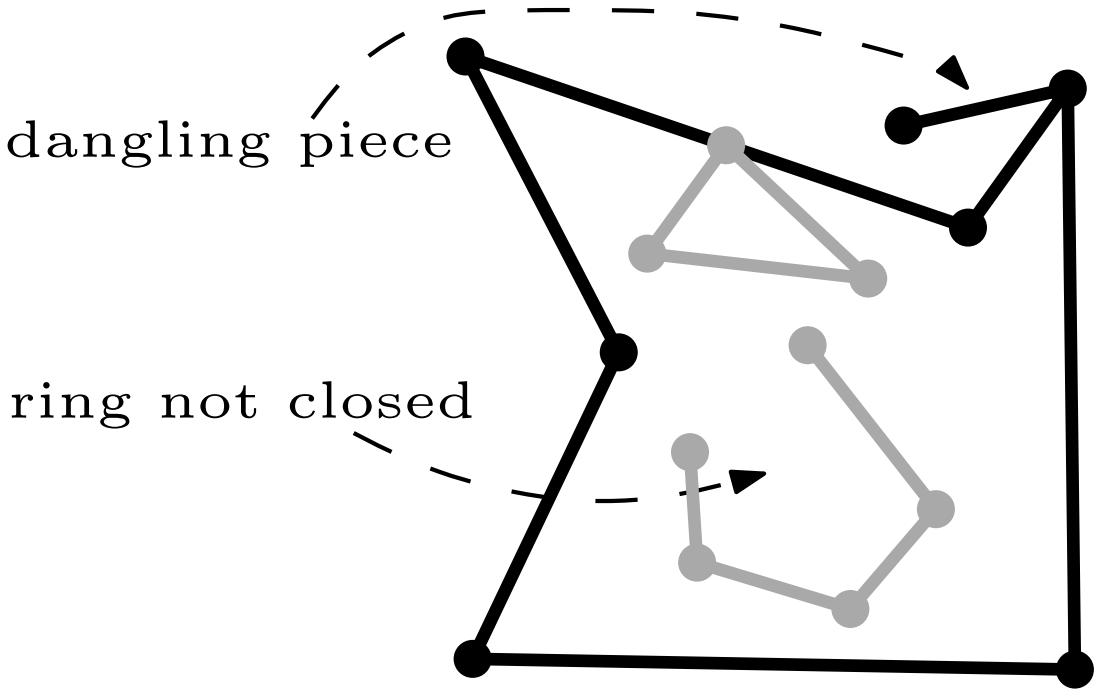
Related work: PostGIS 2.0's ST_MakeValid

- high-level automatic repair function
- diff functions called depending on geometric and topological configurations of rings
- based on construction of planar graph (GEOS is used)
- not documented ("read the code") = predicting behaviour is difficult
- very slow for big polygons

Our approach = constrained triangulation (CT)

Repairing = 3 simple steps:

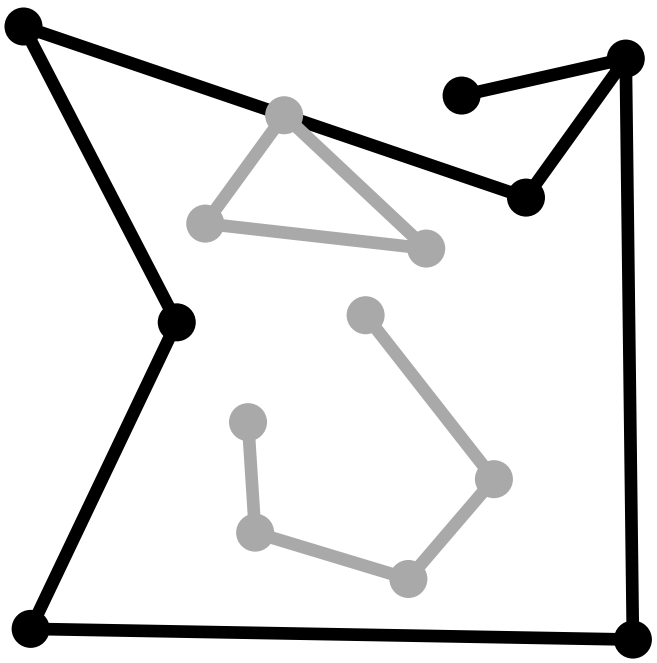
1. CT of input polygon
2. labelling of triangles (*outside* or *inside*)
3. reconstruction of the rings by depth-first search on the dual graph



Our approach = constrained triangulation (CT)

Repairing = 3 simple steps:

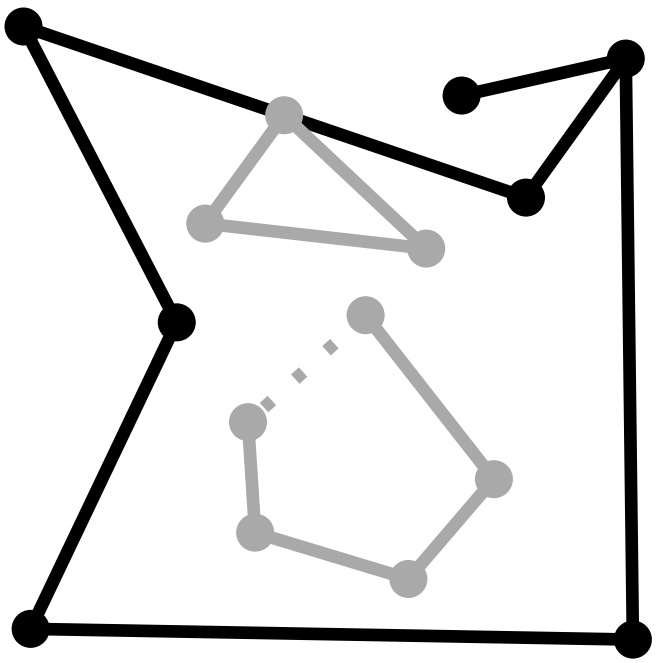
1. CT of input polygon
2. labelling of triangles (*outside* or *inside*)
3. reconstruction of the rings by depth-first search on the dual graph



Our approach = constrained triangulation (CT)

Repairing = 3 simple steps:

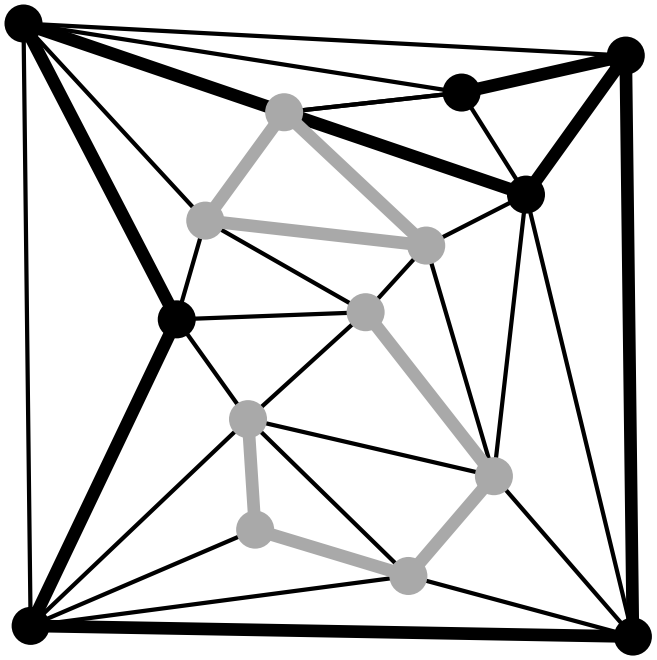
1. CT of input polygon
2. labelling of triangles (*outside* or *inside*)
3. reconstruction of the rings by depth-first search on the dual graph



Our approach = constrained triangulation (CT)

Repairing = 3 simple steps:

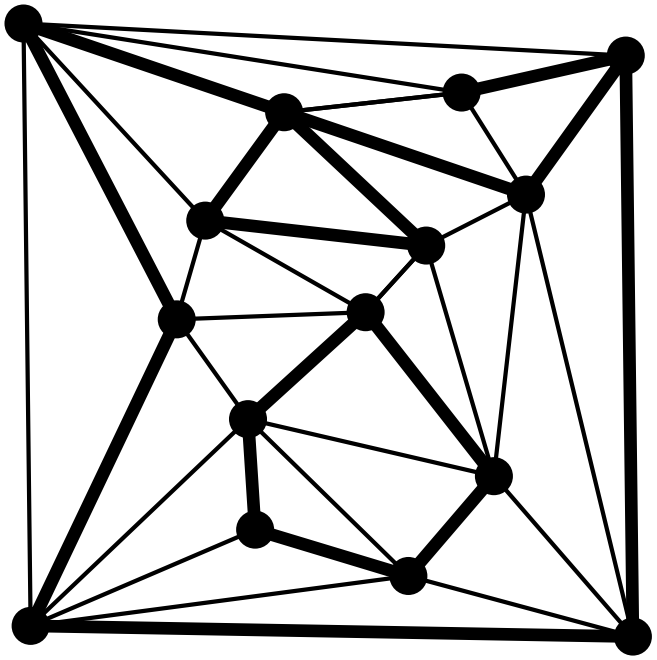
1. CT of input polygon
2. labelling of triangles (*outside* or *inside*)
3. reconstruction of the rings by depth-first search on the dual graph



Our approach = constrained triangulation (CT)

Repairing = 3 simple steps:

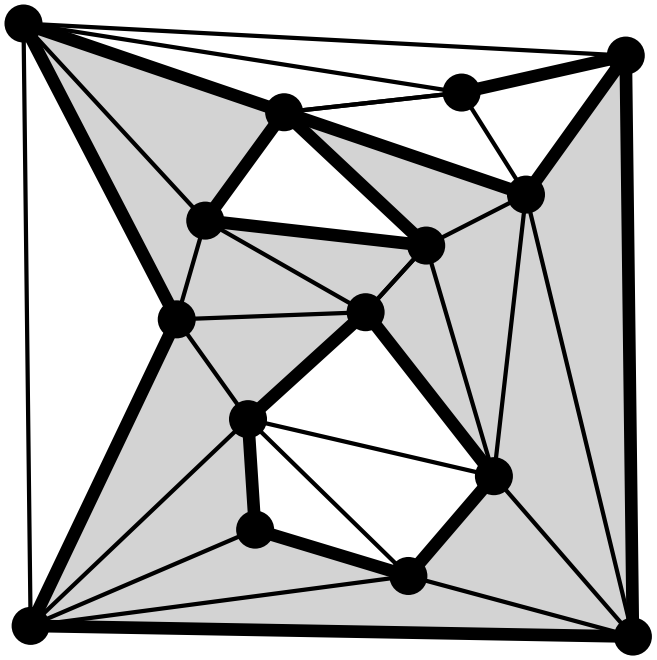
1. CT of input polygon
2. labelling of triangles (*outside* or *inside*)
3. reconstruction of the rings by depth-first search on the dual graph



Our approach = constrained triangulation (CT)

Repairing = 3 simple steps:

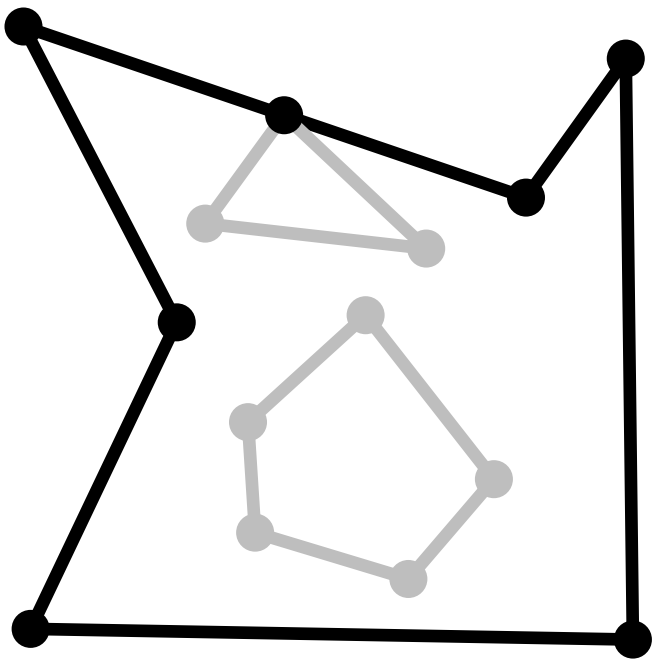
1. CT of input polygon
2. labelling of triangles (*outside* or *inside*)
3. reconstruction of the rings by depth-first search on the dual graph



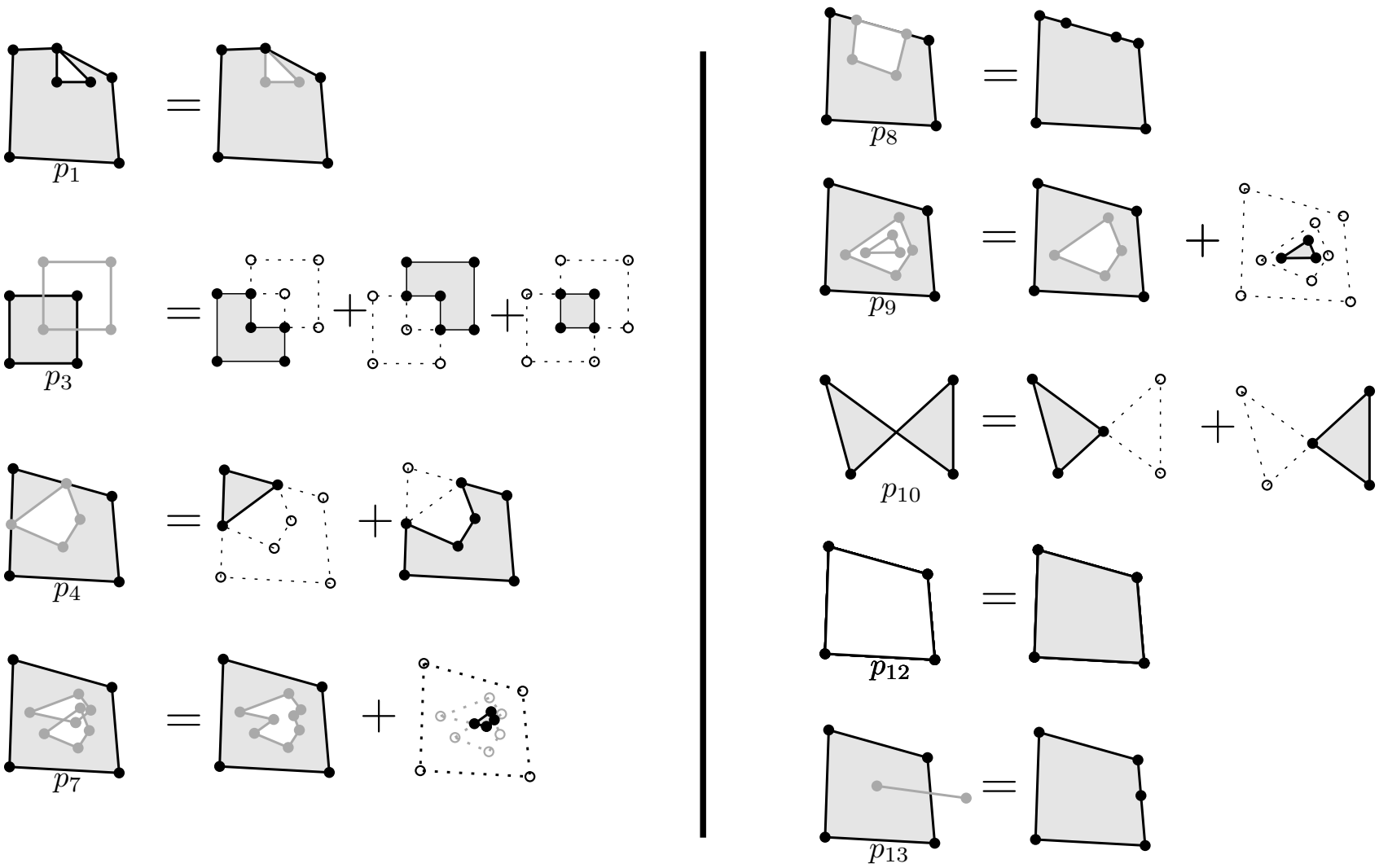
Our approach = constrained triangulation (CT)

Repairing = 3 simple steps:

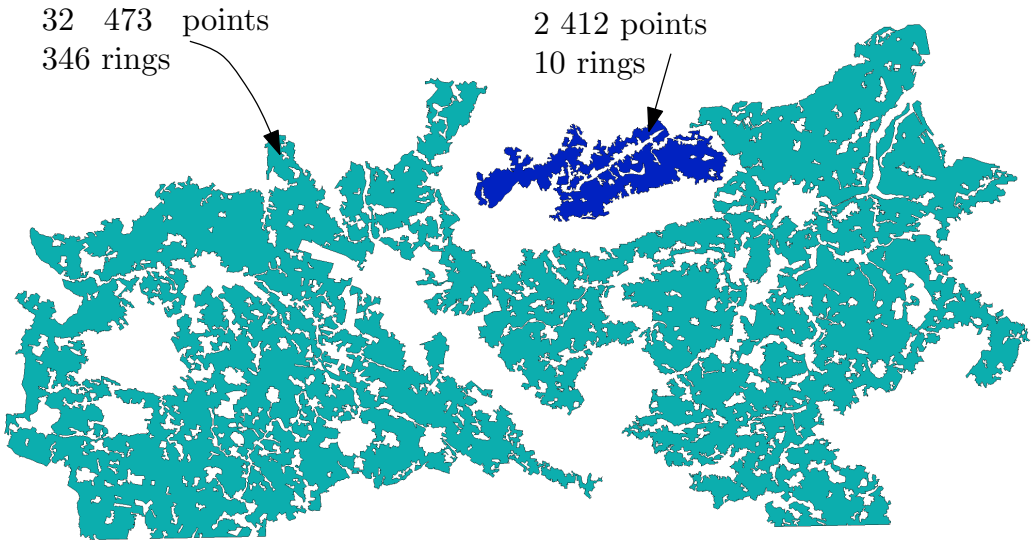
1. CT of input polygon
2. labelling of triangles (*outside* or *inside*)
3. reconstruction of the rings by depth-first search on the dual graph

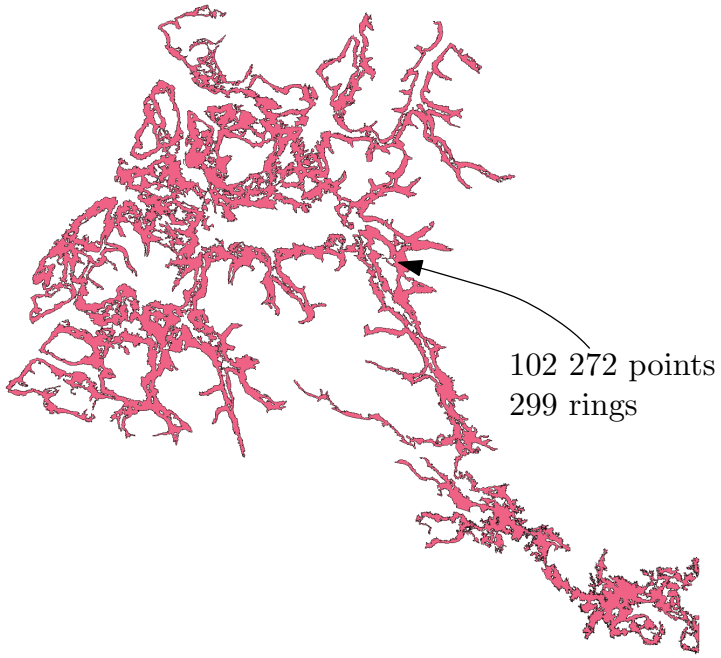


Examples of polygons automatically repaired

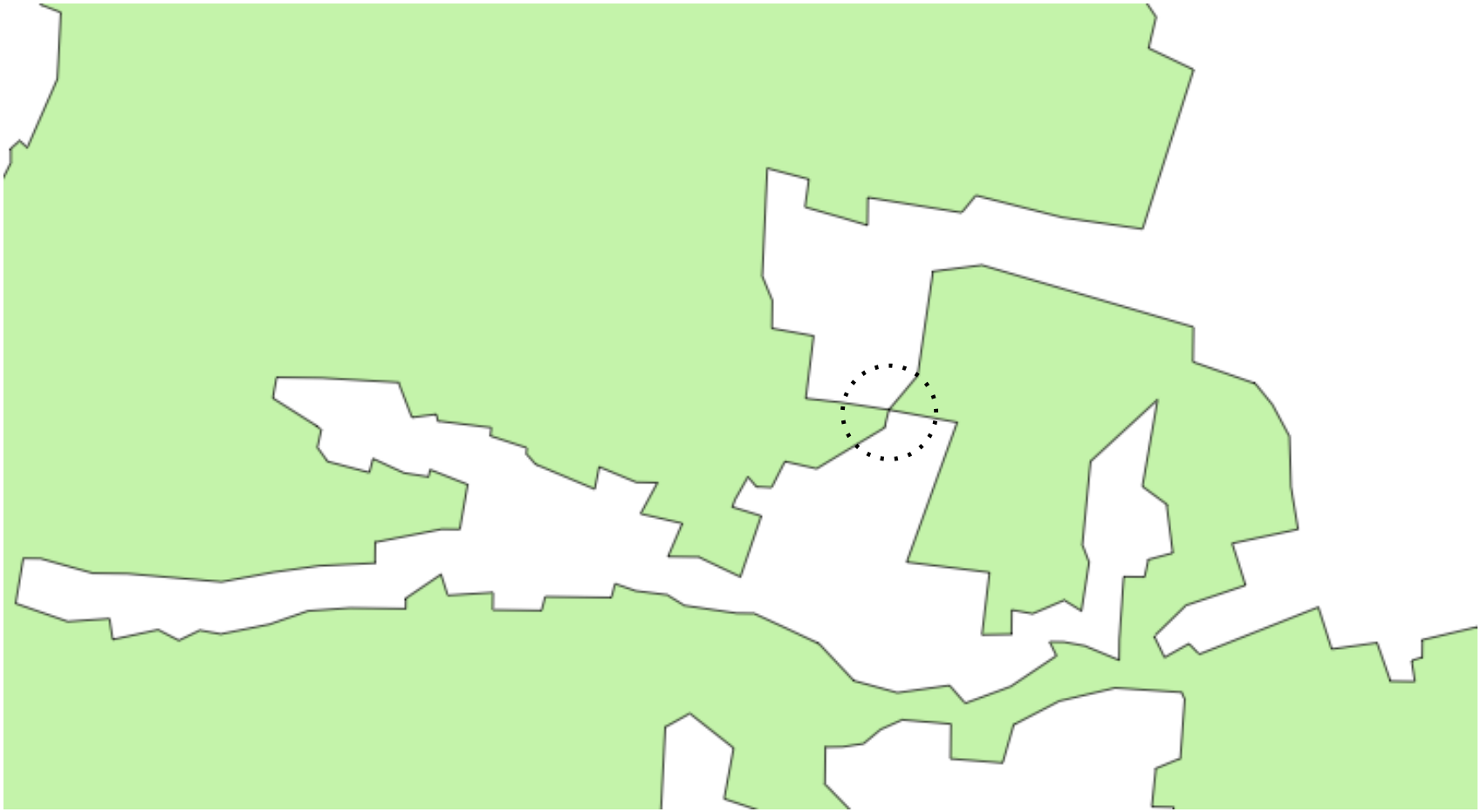


Experiments with big polygons: CORINE2006

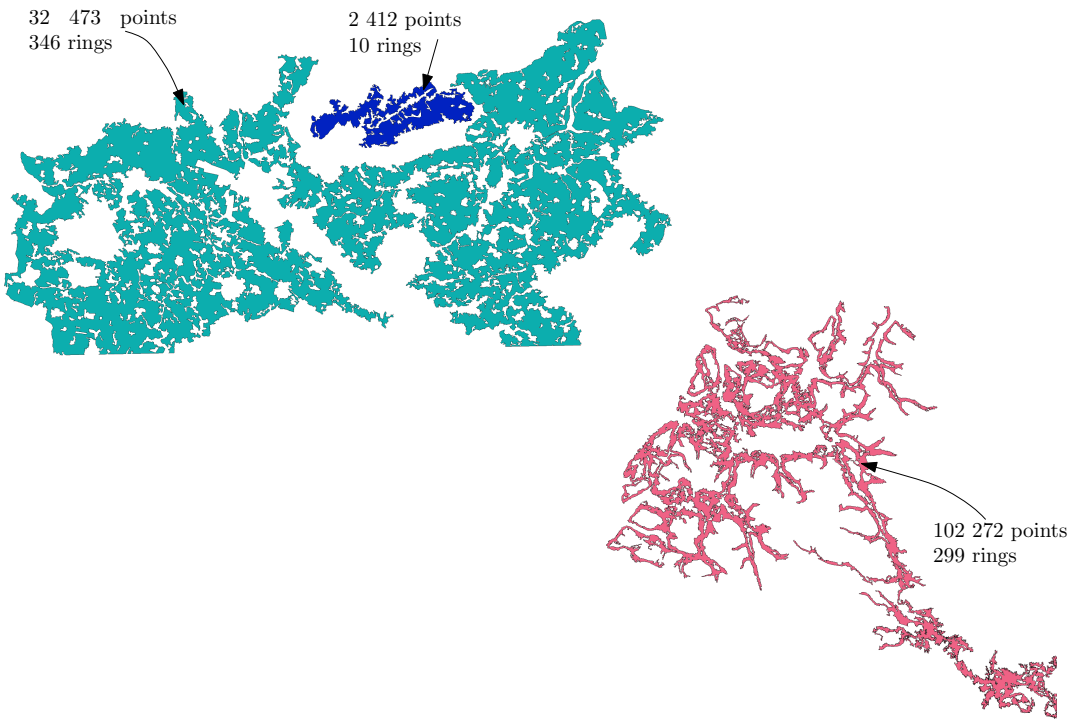




Experiments with big polygons: CORINE2006



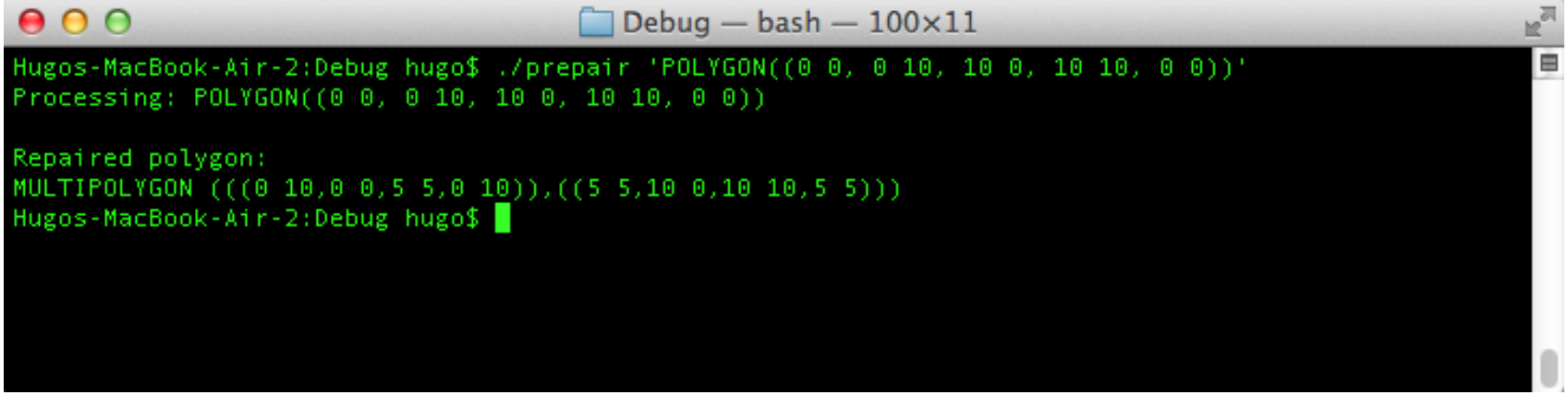
Experiments with big polygons: CORINE2006



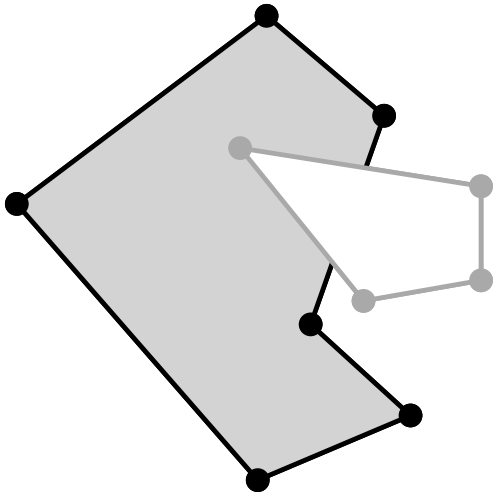
	points	rings	prepair	ST_MakeValid
EU-47552	2 412	10	0.5s	0.8s
EU-47997	32 473	346	11.4s	314.0s
EU-180927	102 272	299	52.2s	740.2s

The code is freely available: help us improve it!

- `http://tudelft-gist.github.com/prepair`
- only around 300 lines of code
- BSD license
- (CGAL is GPL)

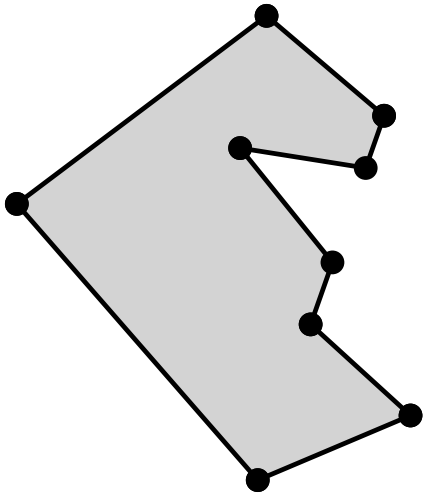


Future work: alternative repair paradigms



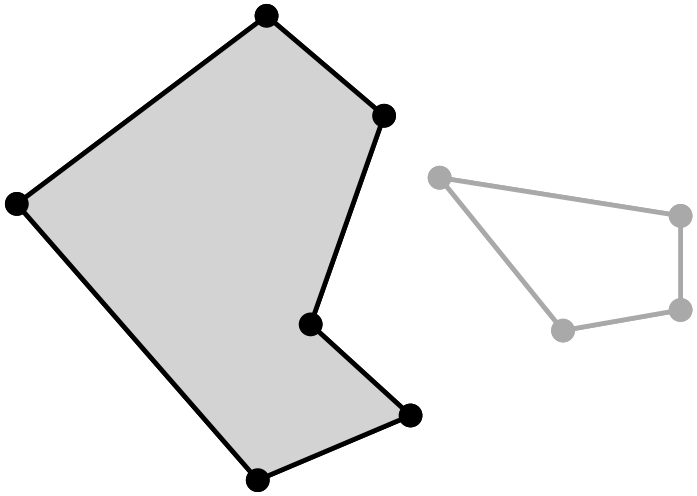
Point-set topology: $p = outerring \setminus \{innerrings\}$

Future work: alternative repair paradigms



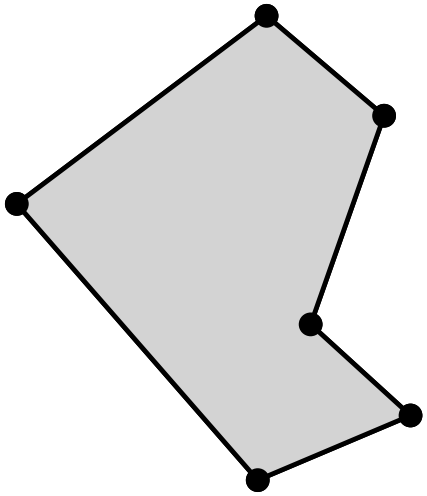
Point-set topology: $p = outerring \setminus \{innerrings\}$

Future work: alternative repair paradigms



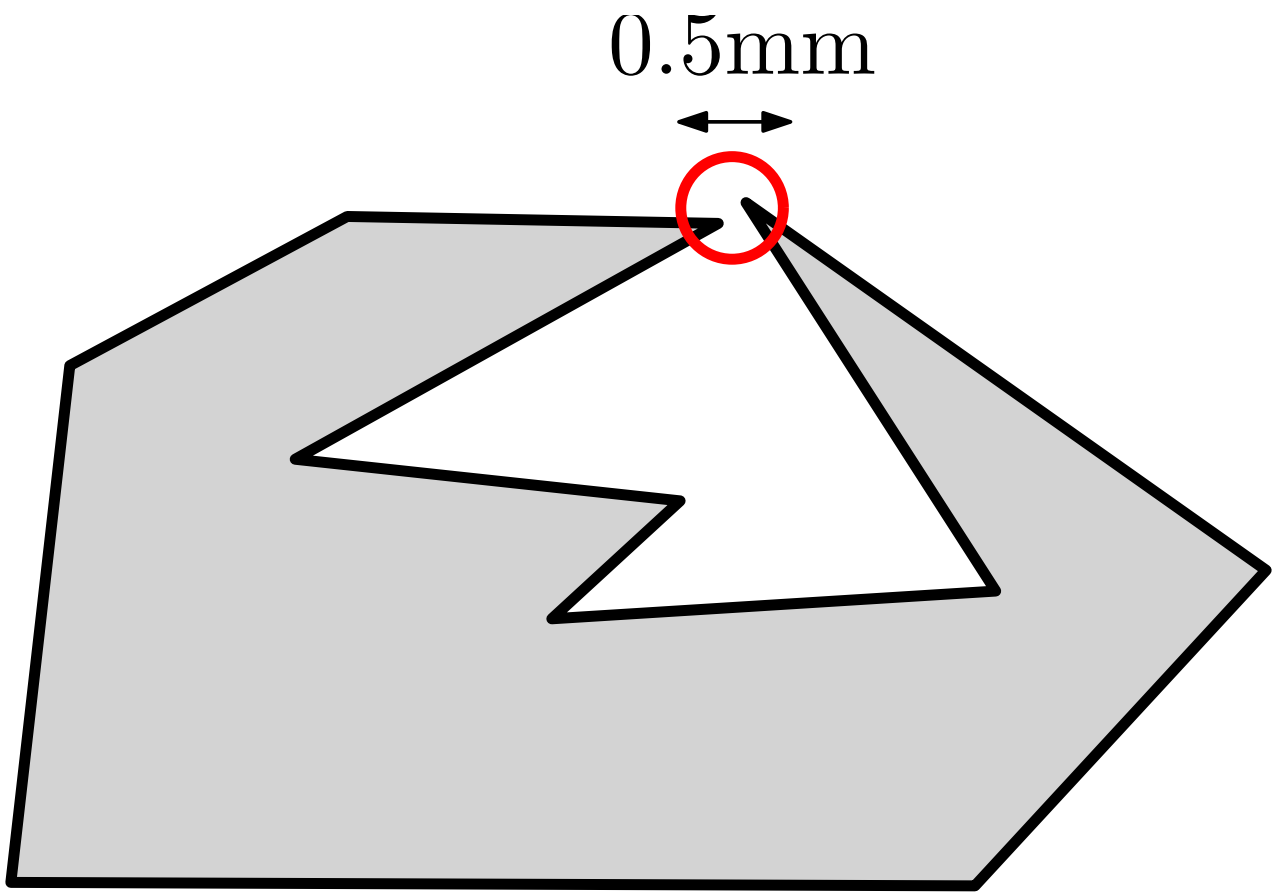
Point-set topology: $p = outerring \setminus \{innerrings\}$

Future work: alternative repair paradigms



Point-set topology: $p = outerring \setminus \{innerrings\}$

Future work: validation with a tolerance



Thanks for your attention

Hugo Ledoux
h.ledoux@tudelft.nl

Ken Arroyo Ohori
g.a.k.arroyoohori@tudelft.nl

Martijn Meijers
b.m.meijers@tudelft.nl

tudelft-gist.github.com/prepair