Geometric Validation of GML Solids with the Constrained Delaunay Tetrahedralization

Hugo Ledoux, Edward Verbree and Hang Si

Technische Universiteit Delft

GIS technology group

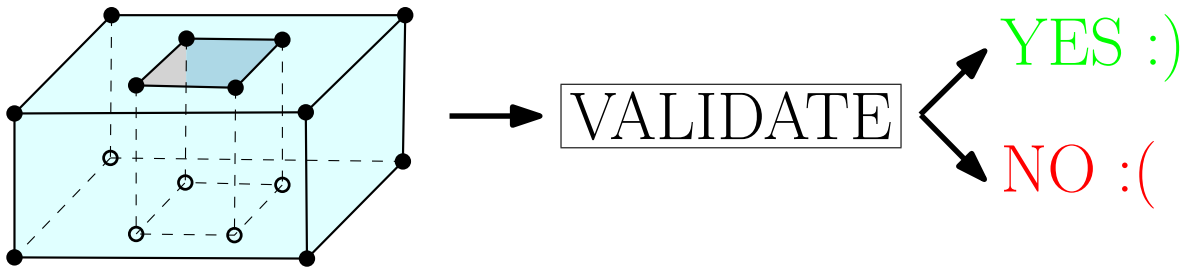
4 November 2009

3DGeoInfo—Ghent

Introduction

The problem:

Given a polyhedron stored with GML, we want to *validate* it



BUT

This is not that simple, because:

1. Definitions of a polyhedron/solid is not 100% clear
2. Generalisation to 3D is difficult
3. 2D problem is not that simple either

Why do we want to validate spatial objects?

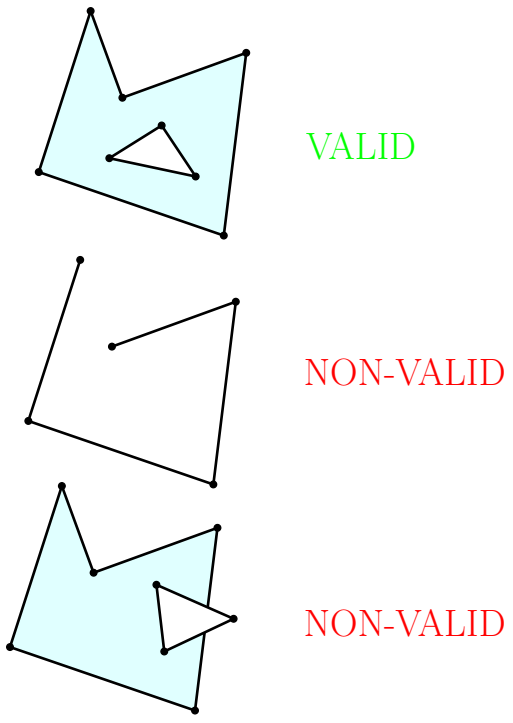
- Non-valid spatial objects are fine if you only want to *look* at them!
- Validation is necessary to guarantee output of *processing* or *manipulation* operations:
  - calculation of area of a polygon
  - creation of a buffer
  - boolean operations such as intersection, touch, contain, etc.
  - conversion to other formats

Validation = *geometric* validation

- Validating a spatial object means ensuring that it respects certain specifications
- Specifications are standardised, such as by the ISO or the OGC
- $\neq$ XML schema validation (*.xsd)

For Polygons:

1. closed boundary
2. no self-intersection
3. ring does not intersect boundary
4. etc



≠ XML Schema Validation (*.xsd)

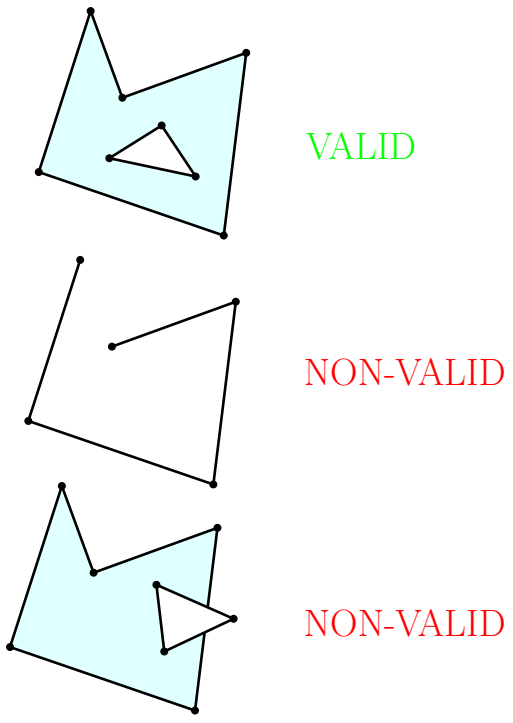
```
<gml:Solid>
 <gml:exterior>
  <gml:CompositeSurface>
   <gml:surfaceMember>
   <!--top surface-->
    <gml:Polygon gml:id="a">
     <gml:exterior>
      <gml:LinearRing>
       <gml:pos>0 0 1</gml:pos>
       <gml:pos>1 0 1</gml:pos>
       ...
      </gml:LinearRing>
     </gml:exterior>
    </gml:Polygon>
   </gml:surfaceMember>
```

Validation = *geometric* validation

- Validating a spatial object means ensuring that it respects certain specifications
- Specifications are standardised, such as by the ISO or the OGC
- $\neq$ XML schema validation (*.xsd)

For Polygons:

1. closed boundary
2. no self-intersection
3. ring does not intersect boundary
4. etc



Validation is 2D is not simple

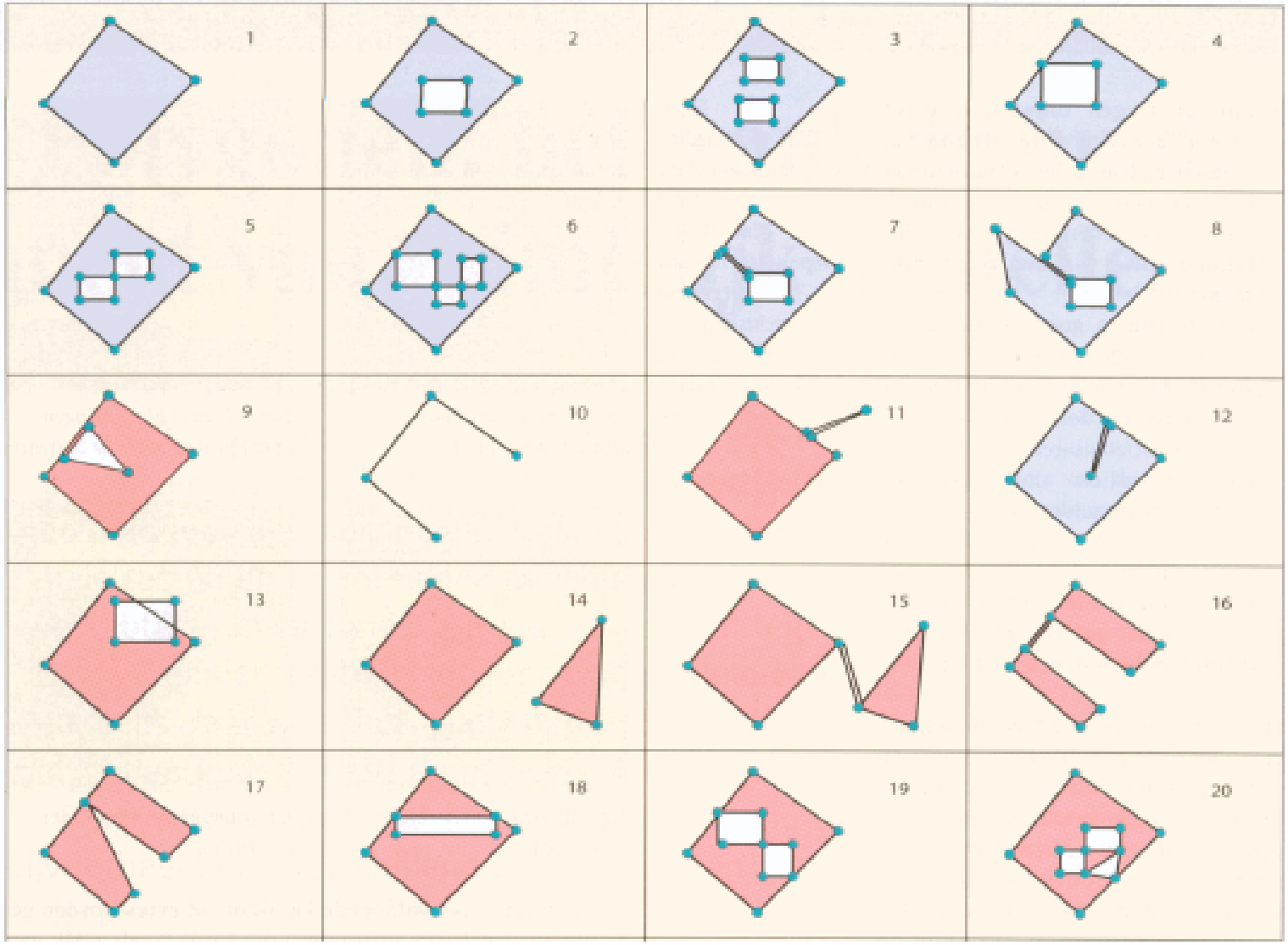


Figure from [vOQT04]

GML Solids

from [OGC07]:

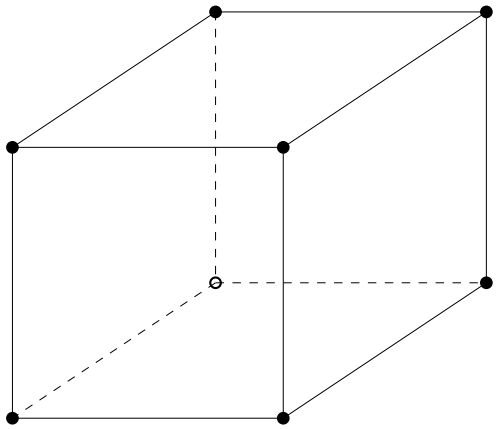
A GML `Solid` "is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces as specified in ISO 19107:2003. `gml:exterior` specifies the outer boundary, `gml:interior` the inner boundary of the solid".

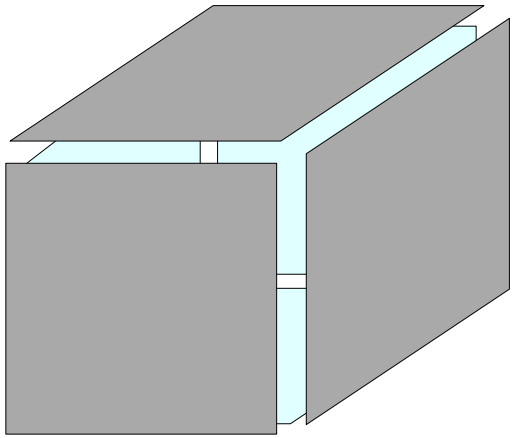
In other words, a GML Solid:

- is represented by its boundaries (`gml:Polygon` embedded in 3D)
- can have "holes" (inner shells) that are allowed to touch each others, or the outer boundary, under certain circumstances.

GML—Geography Markup Language

```
<gml:Solid>
 <gml:exterior>
  <gml:CompositeSurface>
   <gml:surfaceMember>
   <!--top surface-->
    <gml:Polygon gml:id="a">
     <gml:exterior>
      <gml:LinearRing>
       <gml:pos>0 0 1</gml:pos>
       <gml:pos>1 0 1</gml:pos>
       <gml:pos>1 1 1</gml:pos>
       <gml:pos>0 1 1</gml:pos>
       <gml:pos>0 0 1</gml:pos>
      </gml:LinearRing>
     </gml:exterior>
    </gml:Polygon>
   </gml:surfaceMember>
   ...
   ...
  </gml:CompositeSurface>
 </gml:exterior>
</gml:Solid>
```





Some examples of solids

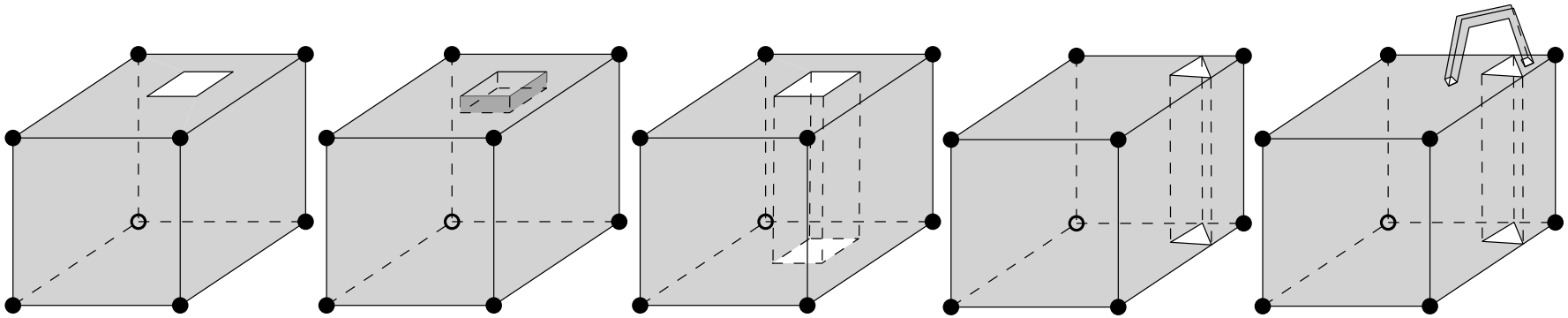
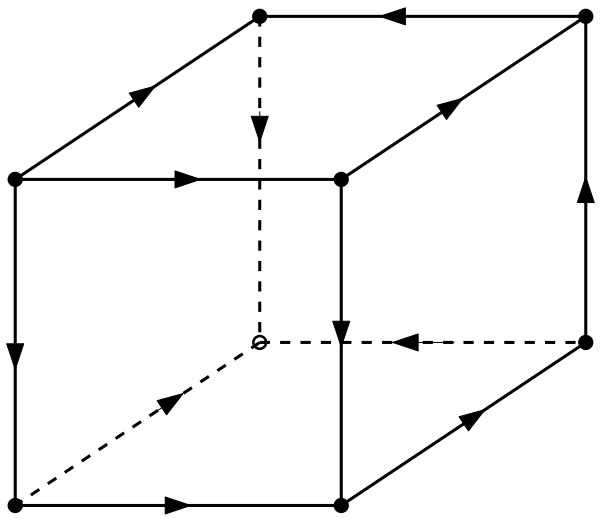


Figure from [KKvOR08]

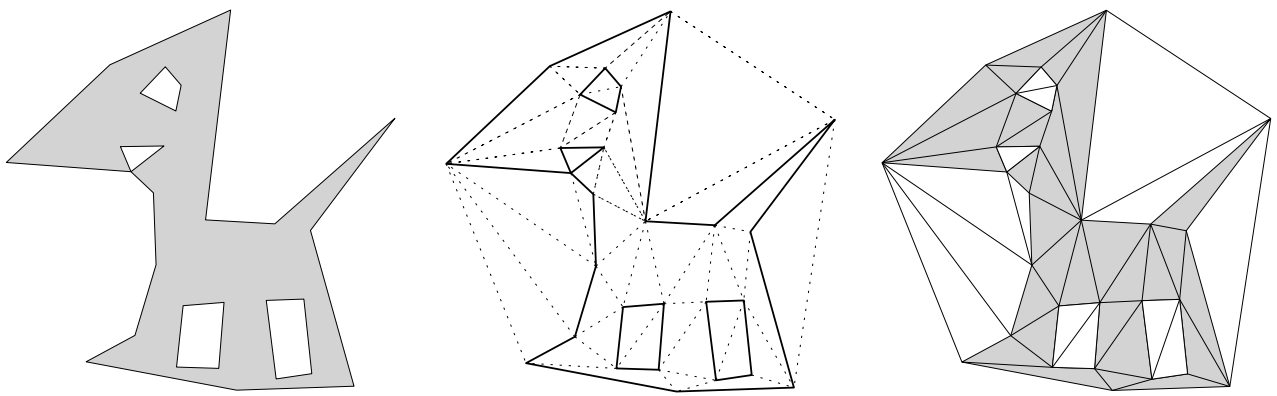
Related Work

- Most work do not allow holes, e.g. [Lie03] or [AF06]
- QS-City 3D tools, by Volker Coors (no publication yet)
- Oracle Spatial 11*g*: use a "graph-based" approach [KKvOR08]



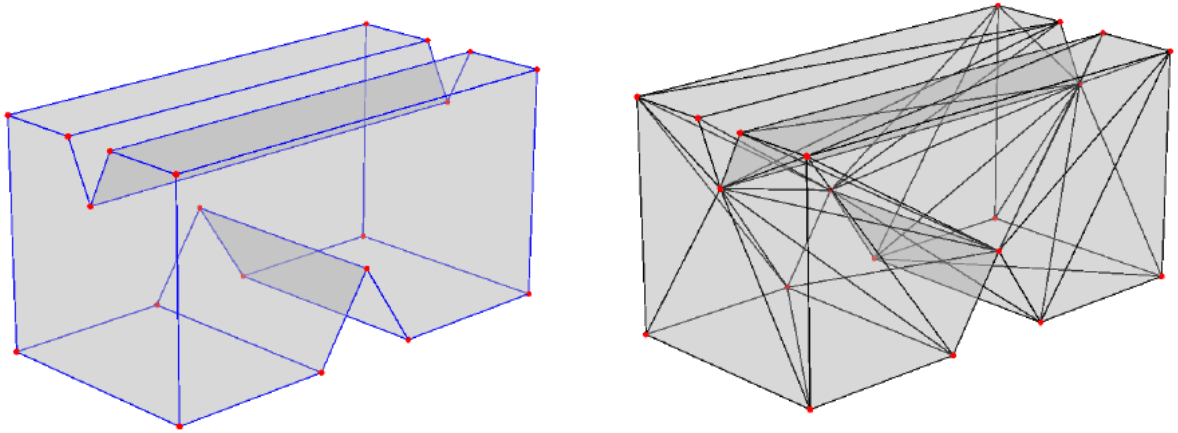
Our approach: constrained Delaunay tetrahedralization

- Using volumetric methods, since we're dealing with volumes
- Raster could be used, but difficulties with boundaries
- So we're using the Constrained Delaunay Tetrahedralization (CDT)
  1. Create CDT for the solid with TetGen [Si04]
  2. Flag each tetrahedron with either IN or OUT
  3. Permits us to easily detect connectedness problems, non-watertight situations, etc.

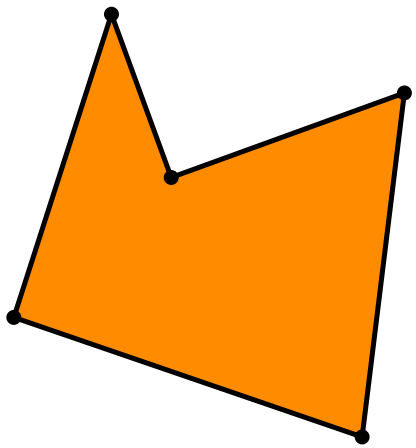


Our approach: constrained Delaunay tetrahedralization

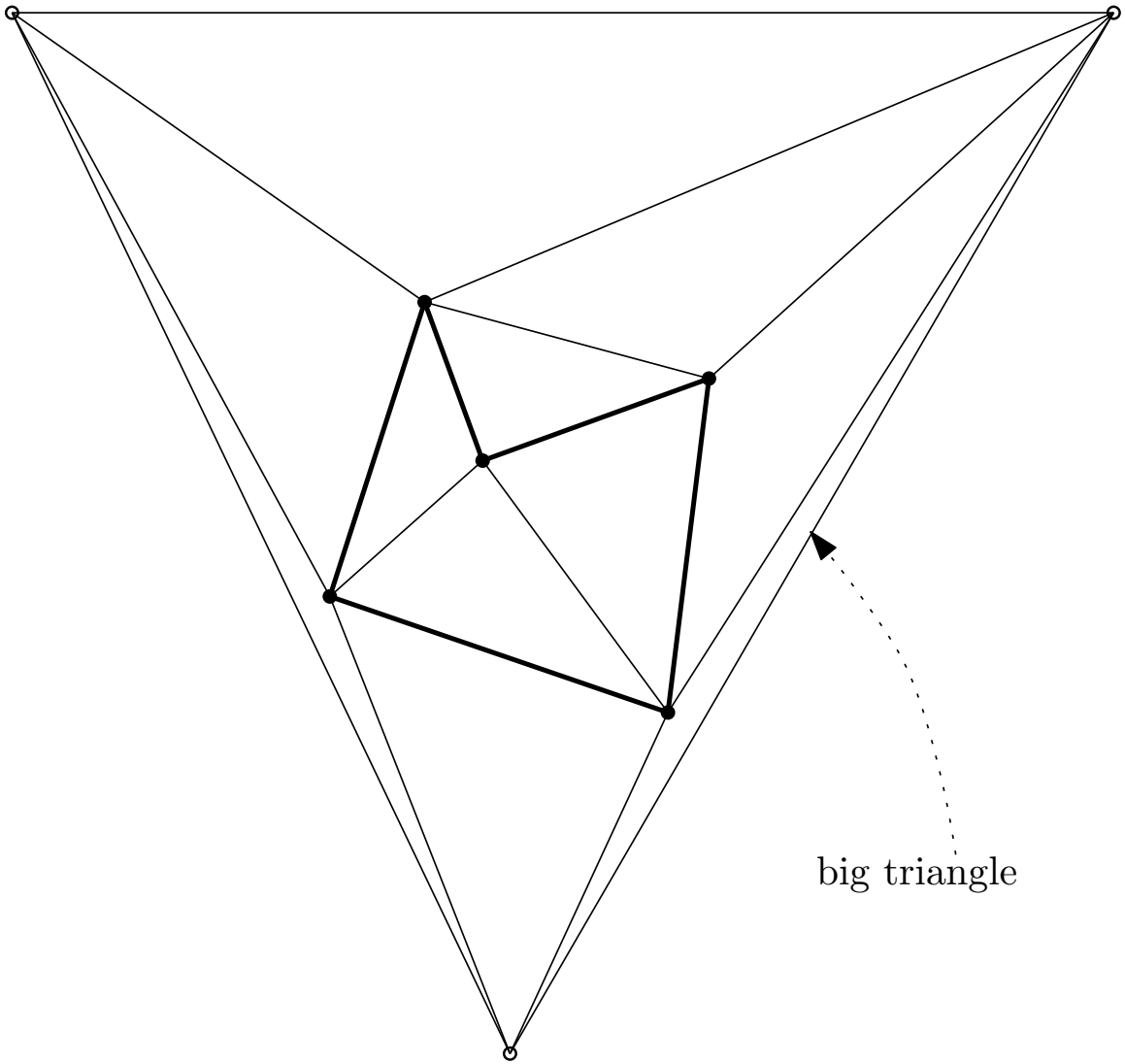
- Using volumetric methods, since we're dealing with volumes
- Raster could be used, but difficulties with boundaries
- So we're using the Constrained Delaunay Tetrahedralization (CDT)
  1. Create CDT for the solid with TetGen [Si04]
  2. Flag each tetrahedron with either IN or OUT
  3. Permits us to easily detect connectedness problems, non-watertight situations, etc.



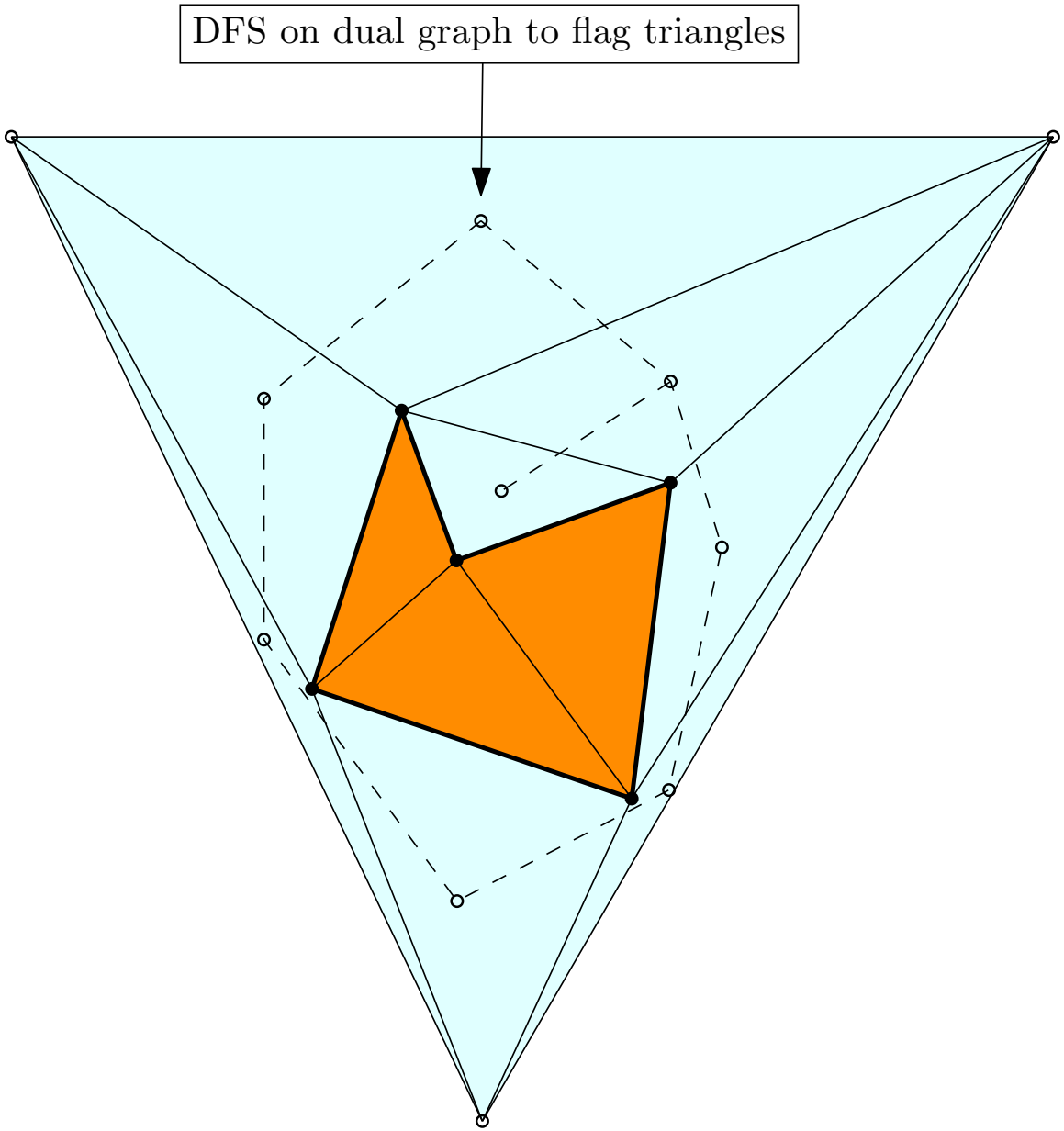
Validation with the CDT



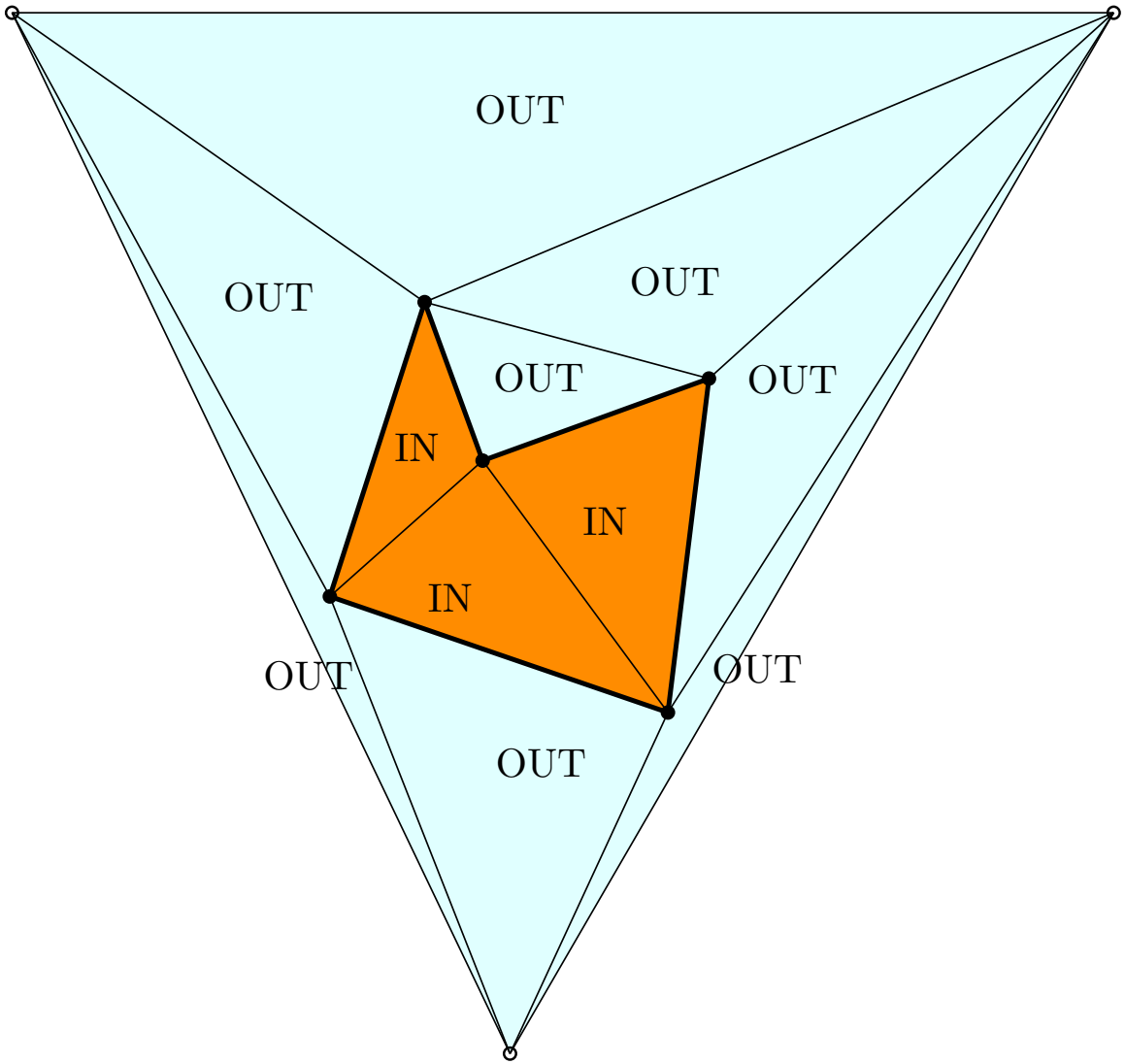
Validation with the CDT



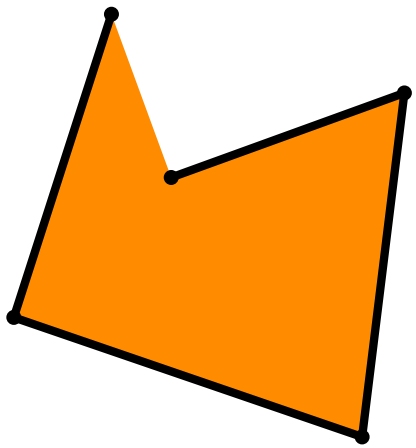
Validation with the CDT



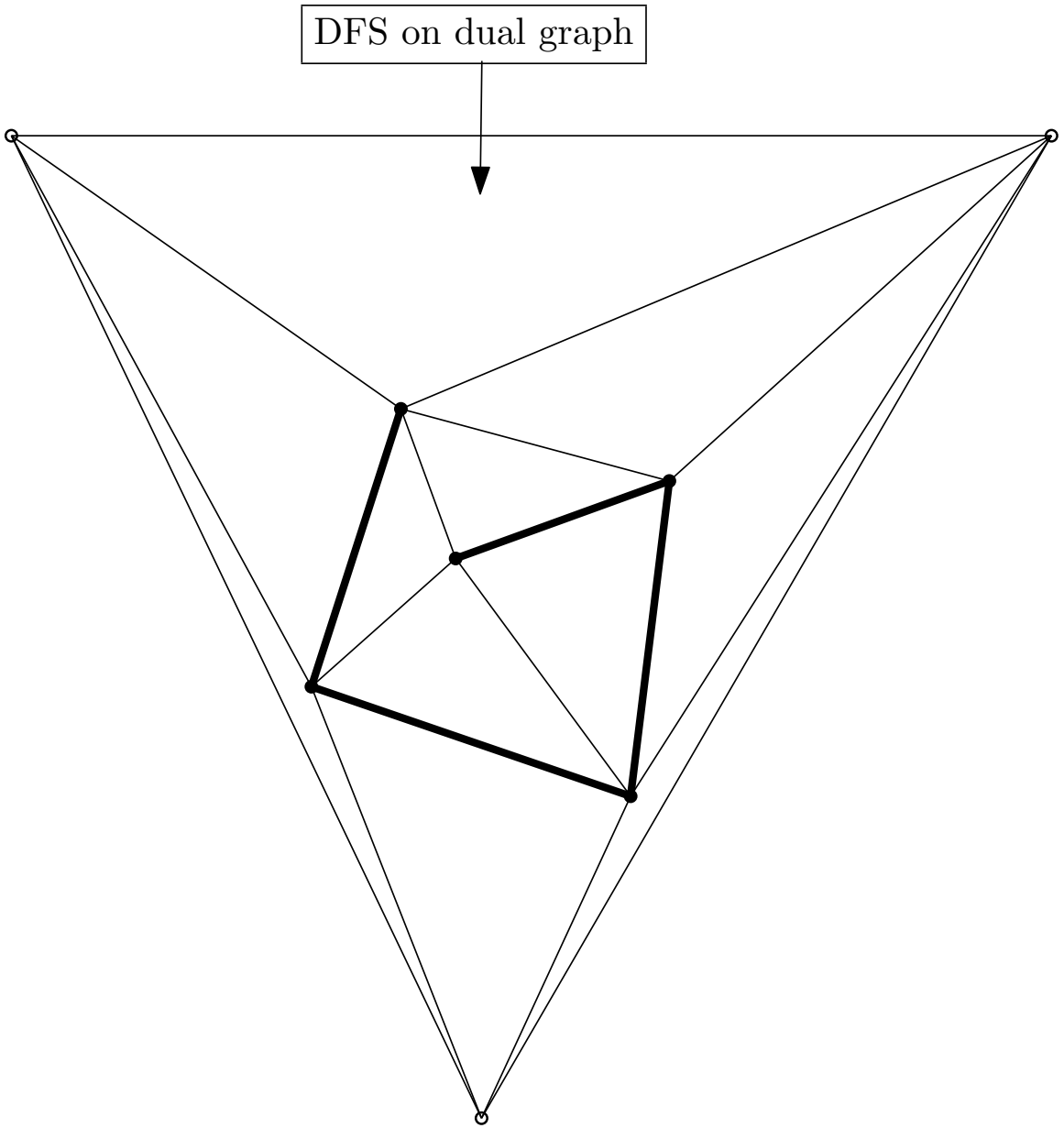
Validation with the CDT



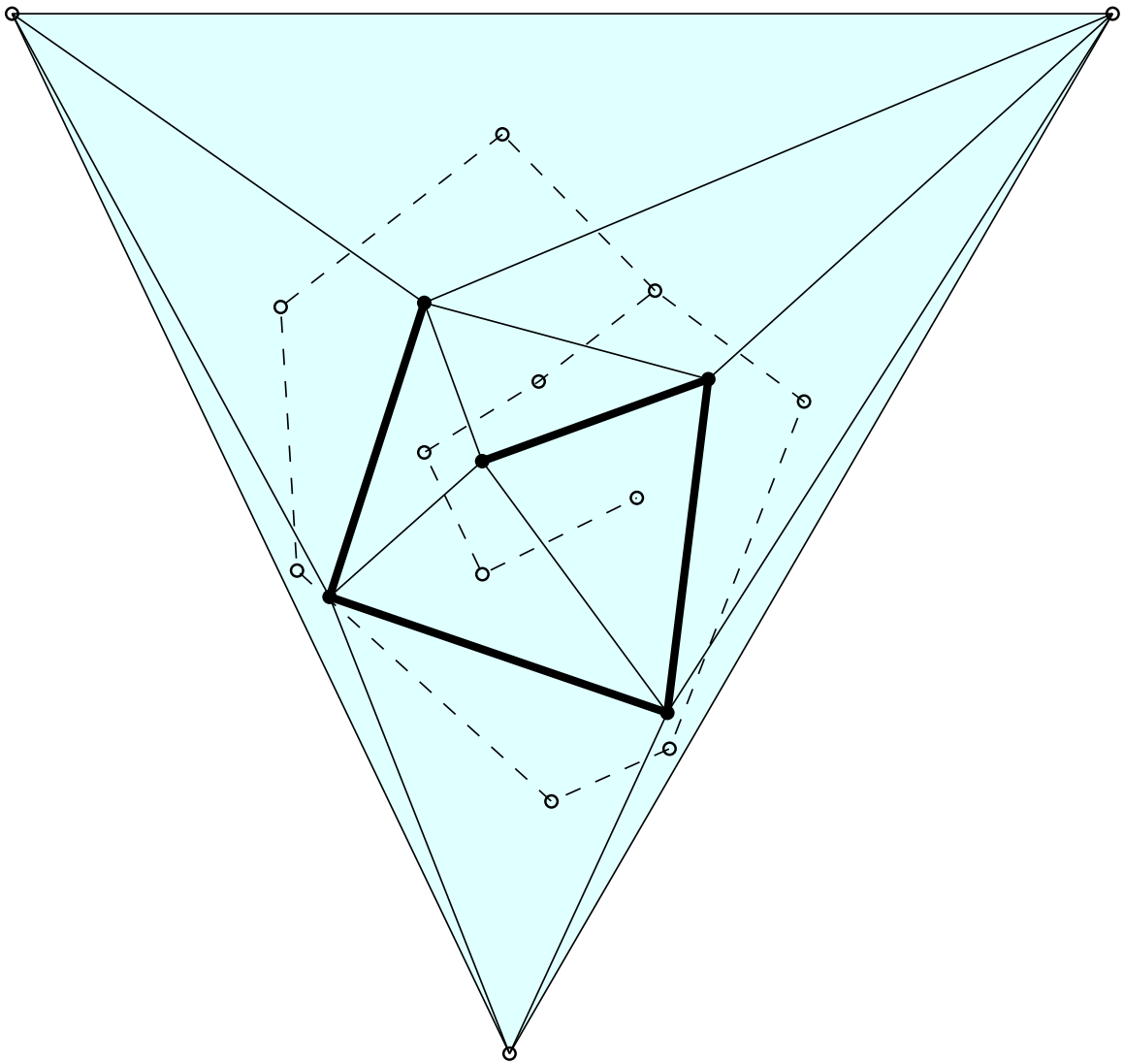
Validation with the CDT



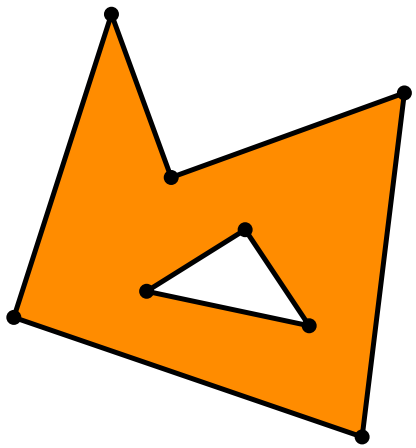
Validation with the CDT



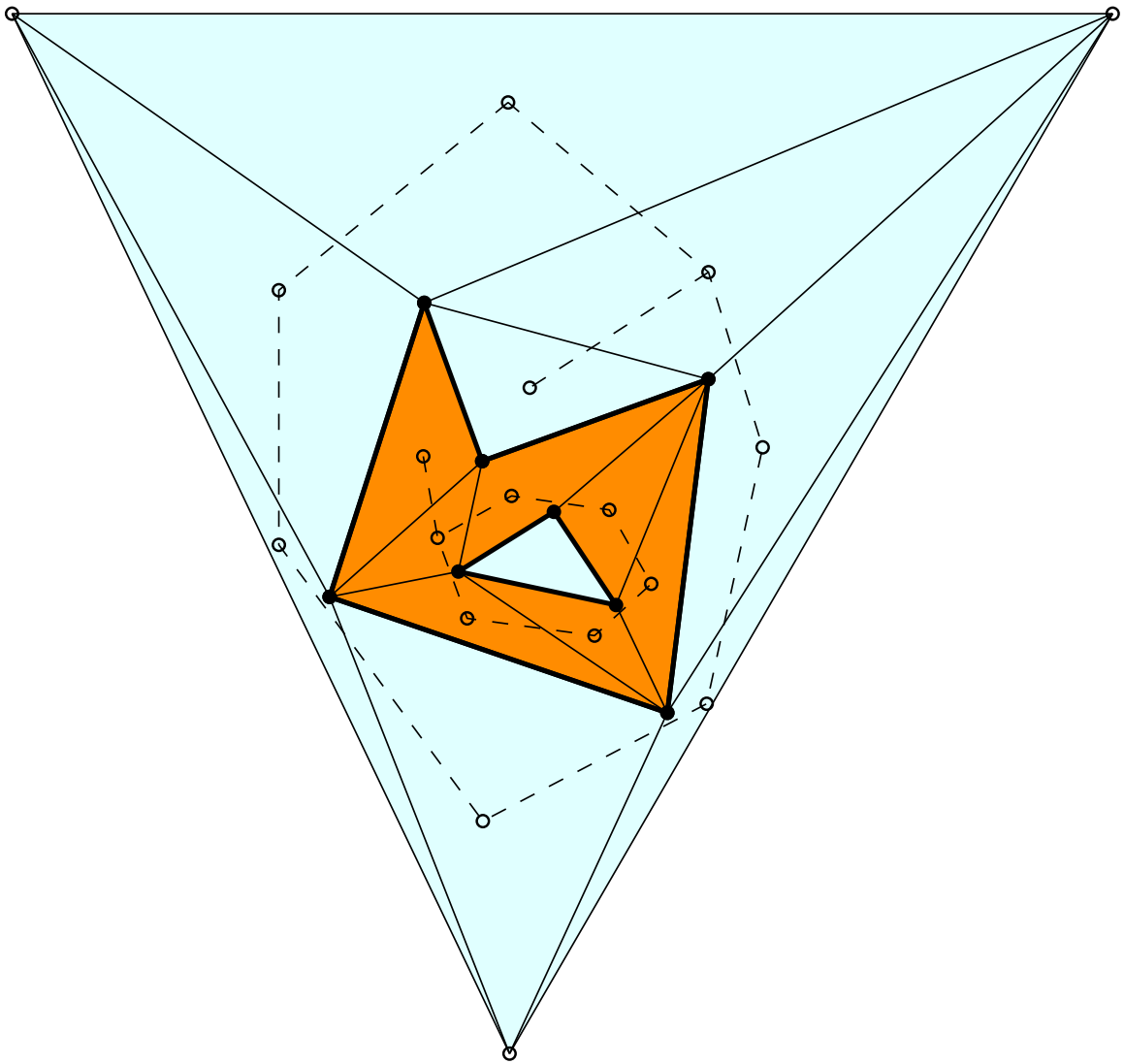
Validation with the CDT



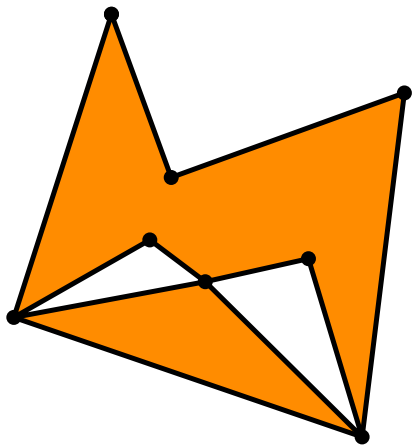
Validation with the CDT



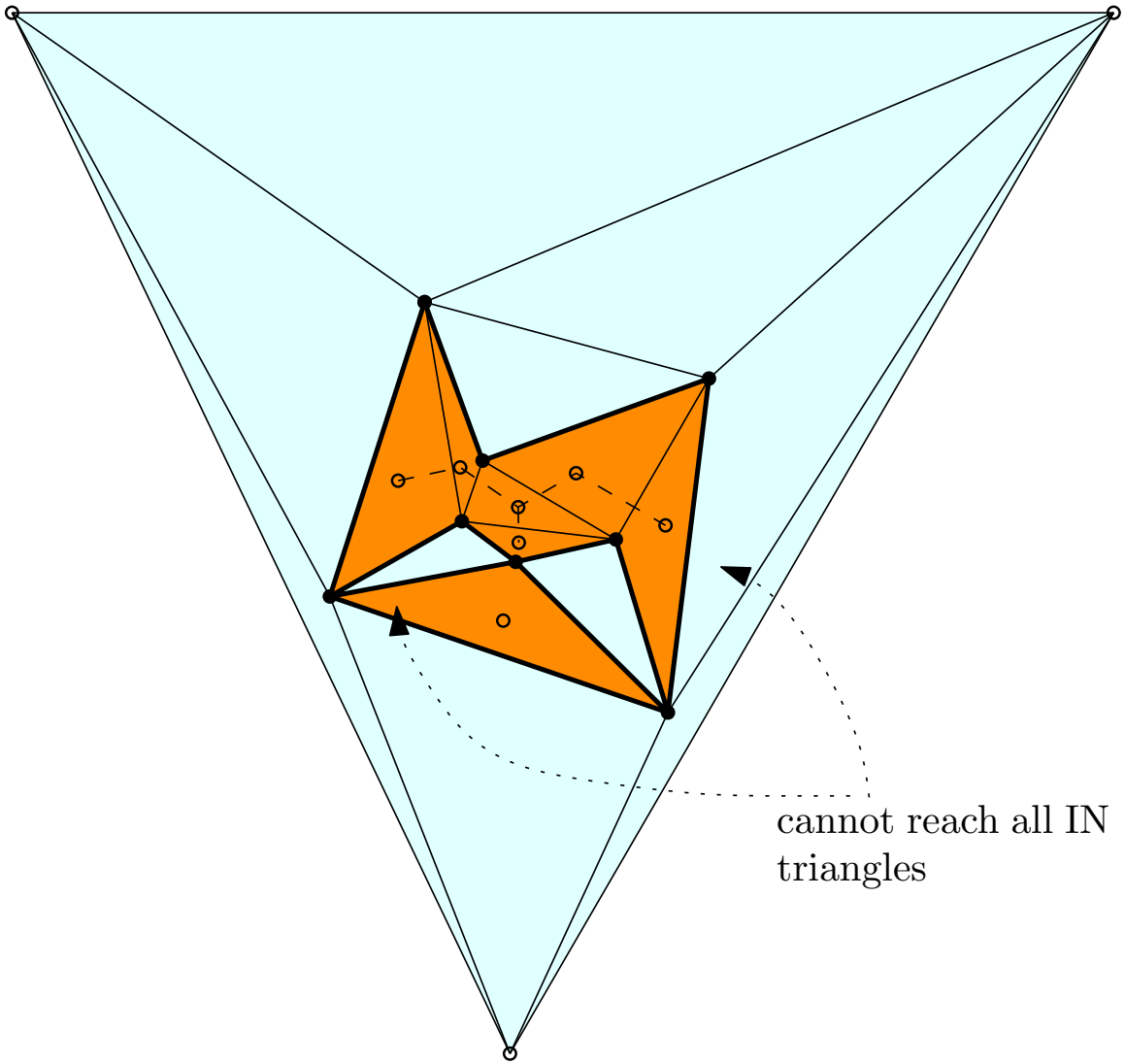
Validation with the CDT



Validation with the CDT



Validation with the CDT



Some examples of solids

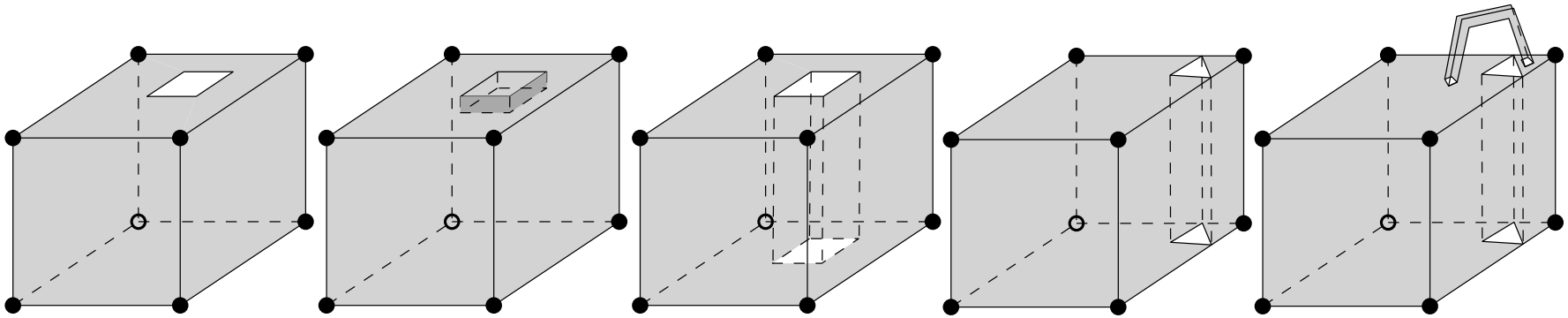
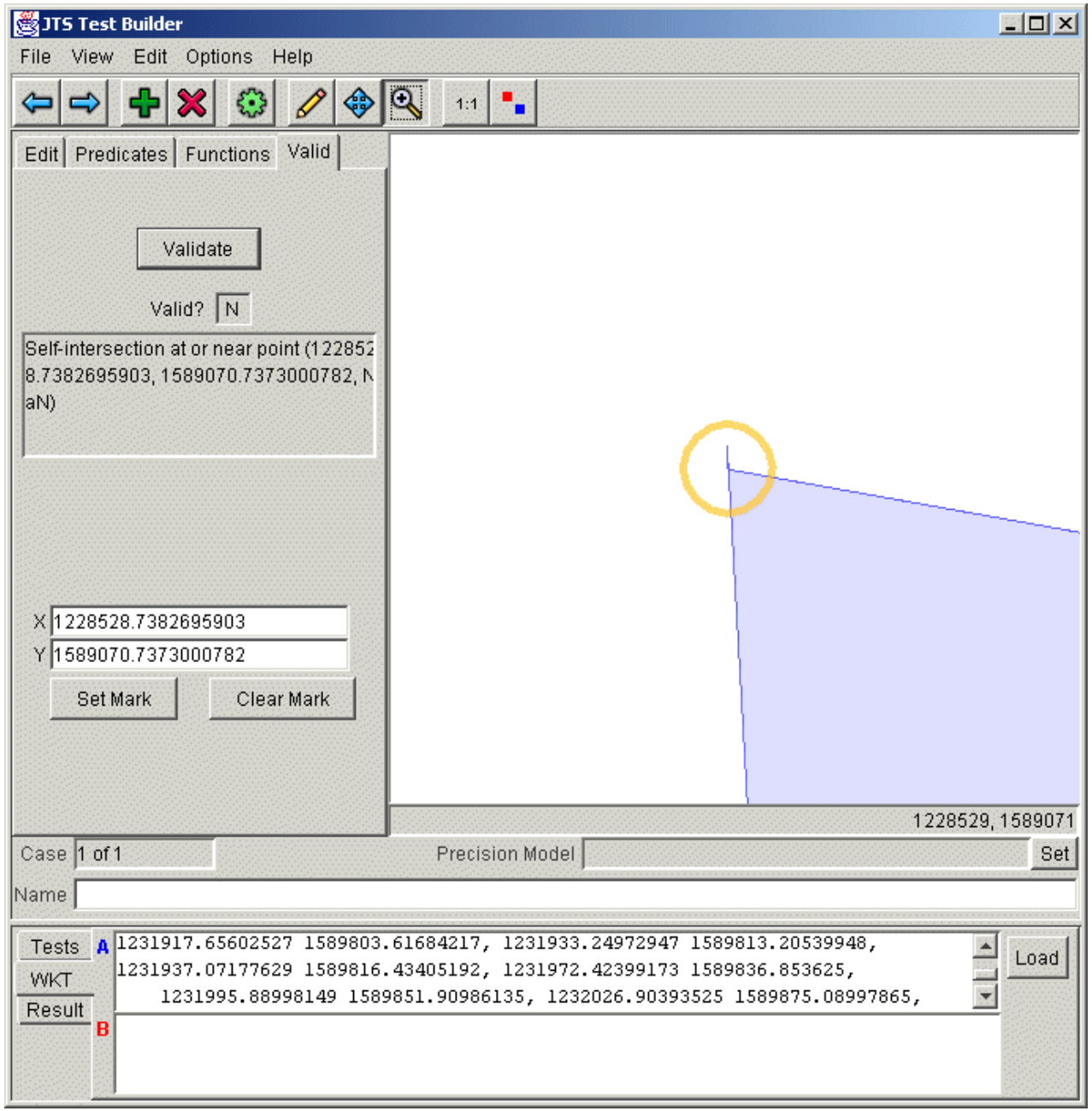


Figure from [KKvOR08]

Other things to test

- Each polygon embedded in 3D must:
  1. have distinct vertices
  2. its rings must be "closed" (first and last are the same)
  3. its inner rings must have opposite orientation
  4. must be planar
  5. must not self-intersect
  6. its inner-rings must not intersect outer ring
- Also:
  1. solid must be watertight
  2. inner shells must be solid themselves
  3. connectedness of the interior of the solid
  4. do inner-outer shells intersect?
  5. orientation of surfaces (normale pointing outwards)

Future work = automatic repair



That's all, thanks for your attention!

References


Marco Attene and Bianca Falcidieno.
ReMESH: An interactive environment to edit and repair triangle meshes.
In *Proceedings IEEE International Conference on Shape Modeling and Applications*, page 41, Matsushima, Japan, 2006.


Baris M. Kazar, Ravi Kothuri, Peter van Oosterom, and Siva Ravada.
On valid and invalid three-dimensional geometries.
In P. van Oosterom, S. Zlatanova, F. Penninga, and E. Fendel, editors, *Advances in 3D Geoinformation Systems*, Lectures Notes in Geoinformation and Cartography, chapter 2, pages 19–46. Spinger, 2008.

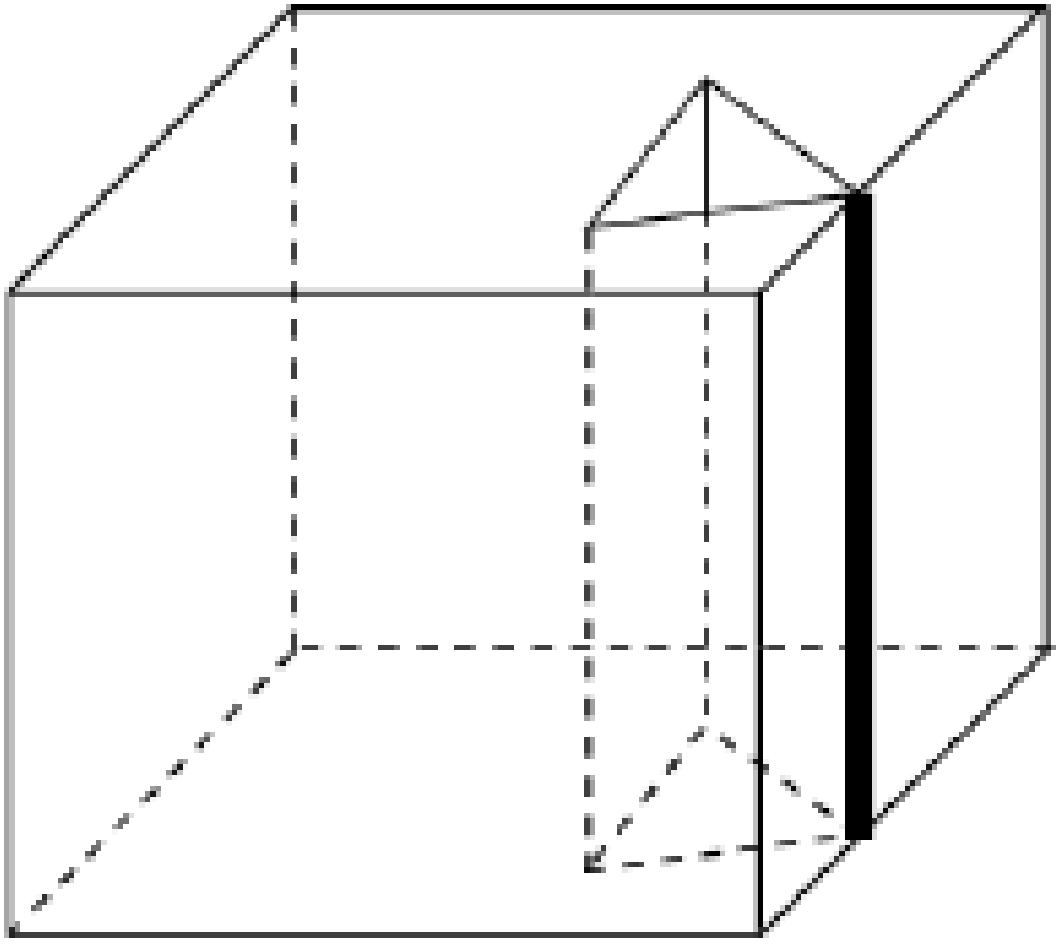

Peter Liepa.
Filling holes in meshes.
In *Proceedings 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 200–205, 2003.


OGC.
Geography markup language (GML) encoding standard.
Open Geospatial Consortium inc., 2007.
Document 07-036, version 3.2.1.


Hang Si.
Tetgen: A quality tetrahedral mesh generator and three-dimensional Delaunay triangulator.
User's manual v1.3 9, WIAS, Berlin, Germany, 2004.


Peter van Oosterom, Wilko Quak, and Theo Tijssen.
About invalid, valid and clean polygons.
In Peter F. Fisher, editor, *Developments in Spatial Data Handling—11th International Symposium on Spatial Data Handling*, pages 1–16. Springer, 2004.

Ambiguities in ISO/GML



What do we have to check?

1. closedness of the rings of every surface: easy, just check if 1st and last point given are the same. Done outside tetgen, as a pre-processing.
2. distinct vertex: easy to check, and must be done outside tetgen as pre-processing.
3. orientation of points within a surface: to make sure that the outer and the inner ring(s) have opposite orientation (CCW and CW). This must be done outside tetgen as a pre-process, and involves finding a single point that defines a hole for the inner rings.
4. planarity of surfaces: this is checked when tetgen makes a surface mesh, it's a requirement of tetgen. Done pre-processing right now.
5. non-self intersection of surfaces: this is checked with tetgen surface mesh also.

What do we have to check?

6. non-overlapping inner rings on a surface. That would be checked automatically by tetgen also when meshing each surface. [again, that might not be necessary for same reason as rule #3]
7. closedness of the solid: Builtin tetgen.
8. Inner shells are a volume This is to make sure that one inner shell is not a line segment for instance. That wouldn't be a problem for tetgen, so we have to take extra care for that one... I guess that would be done when checking the closedness in rule #7.
9. connectedness of the volume of the solid: rather easy to perform, once we have the CDT. Start at a point inside a IN tetra, and breath-first search, counting tetra visited on the way. Constraints obviously stops you. Then compare number of visited tetra with total number of IN tetra.

What do we have to check?

10. inner-outer check of shells , ie make sure that inner shells do not intersect outer shells. The triangle-triangle intersection series of tests will test that.
11. orientation of surfaces : probably the last step of the algorithm. Once we have the CDT with all tetra flagged as IN or OUT, we can then make sure that for each face, the tetra on each side are all consistent. Two cases: the point given to define the interior was based on a wrongly orientated faces; point given was based on a correctly orientated face.