

# Validation of planar partitions using constrained triangulations

**Hugo Ledoux** and Martijn Meijers  
(and Ken Arroyo Ohori)



Technische Universiteit Delft

GIS technology group

May 26 2010

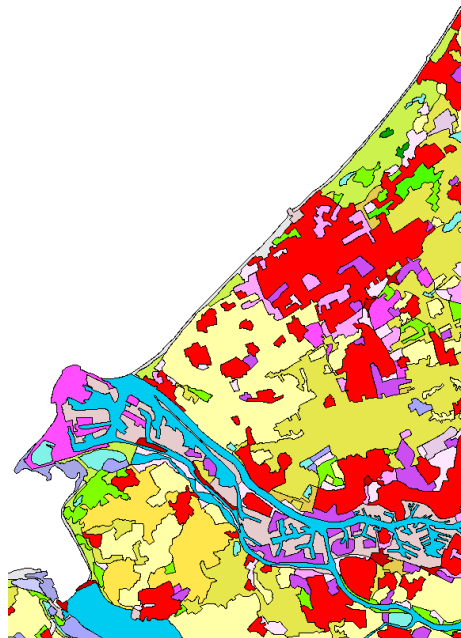
# Introduction

Planar partitions frequently used in GIS:

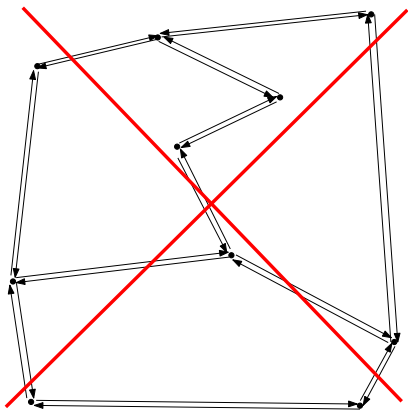
- 1 land cover
- 2 cadastral parcels
- 3 administrative boundaries

The problem:

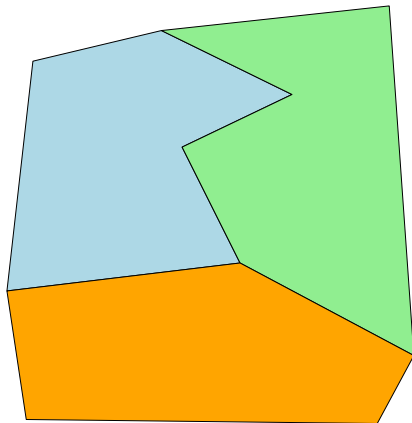
Given a planar partition, we want to *validate* it. And if it's broken *repair* it automatically.



# Planar partitions are often stored with Simple Features

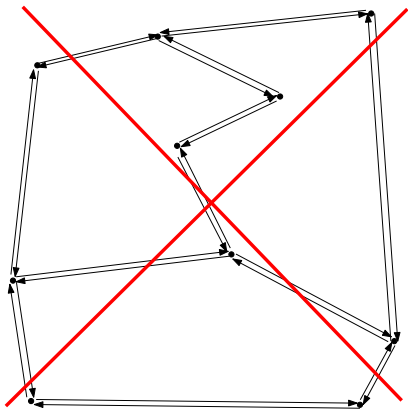


Topological structure

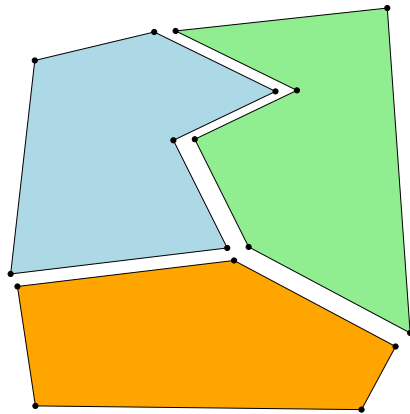


Simple Features (shapefiles)

# Planar partitions are often stored with Simple Features



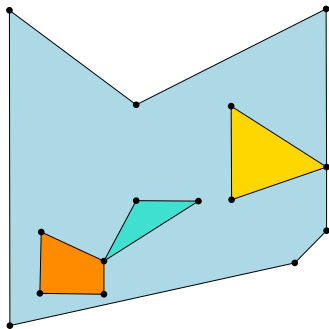
Topological structure



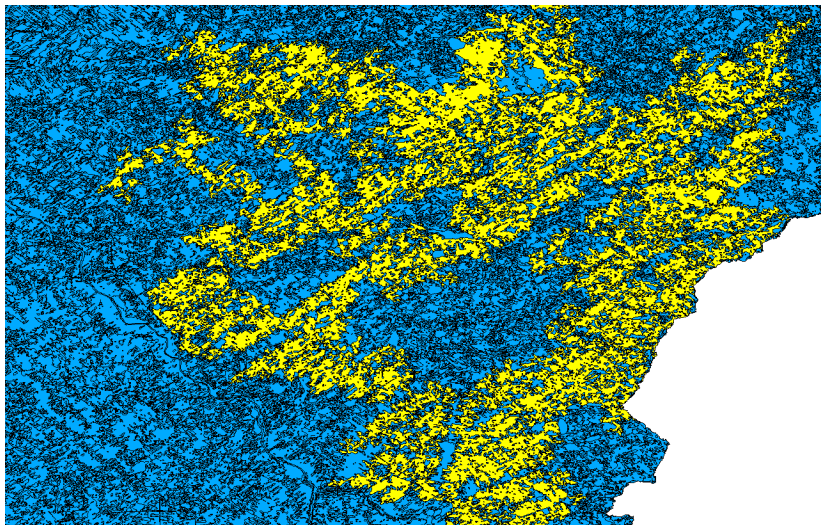
Simple Features (shapefiles)

# Simple Features paradigm

- Points, lines and polygons are stored independently
- Recognised and *used* international standards (by ISO/OGC)
- Geo-DBMS and *shapefile* use it

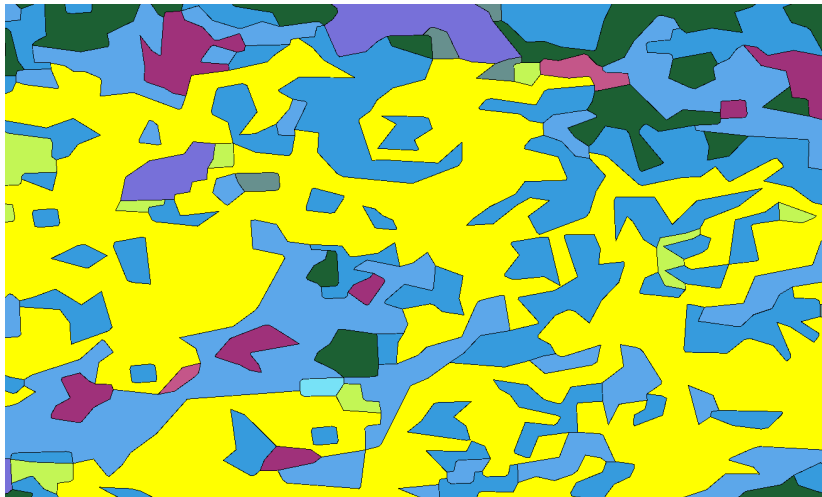


# My favourite polygon!



- Outer boundary has 34 471 points
- Polygon has 3191 holes (inner rings)

# Zoom in on my favourite polygon



# Where could it go wrong?

In practice, errors/mistakes/inconsistencies are often introduced during the construction, manipulation or exchange:

- Overlapping polygons
- Gaps between polygons
- Unconnected polygons
- Tiles of a big datasets do not match



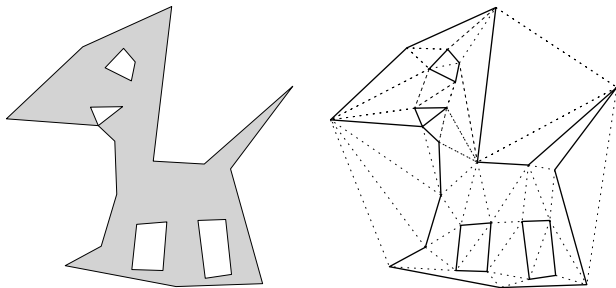
# Potential solutions

- Construct a planar graph (requires cleaning of slivers)
- Define a set of geometric and topologic validation rules on top of graph
- Some commercial solutions:
  - Oracle Spatial Topology
  - ArcGIS
  - 1Spatial Radius Topology

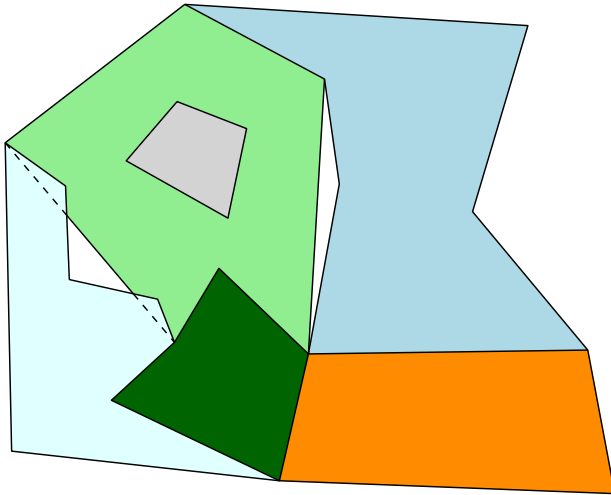
Problem is theoretically easy, but implementation is complicated.

# Our solution = constrained triangulation (CT)

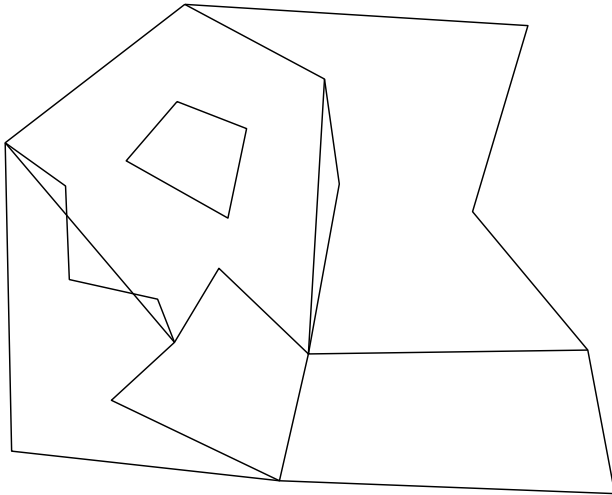
- 1 Construct CT of input polygons (which is a valid planar partition)
- 2 Flag each triangle with ID of its polygon
- 3 Validation made with simple graph-based algos



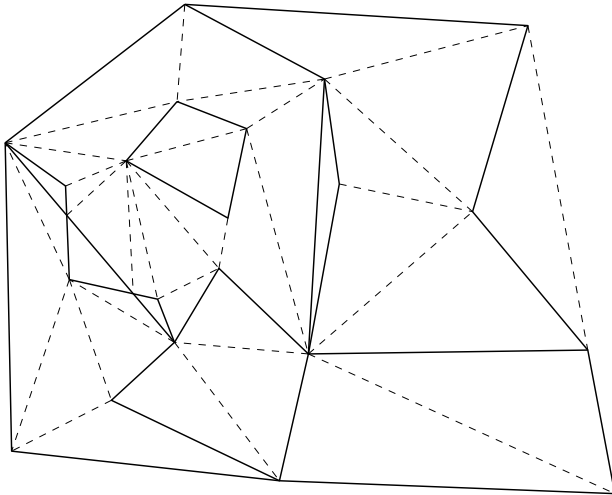
# Steps of our approach



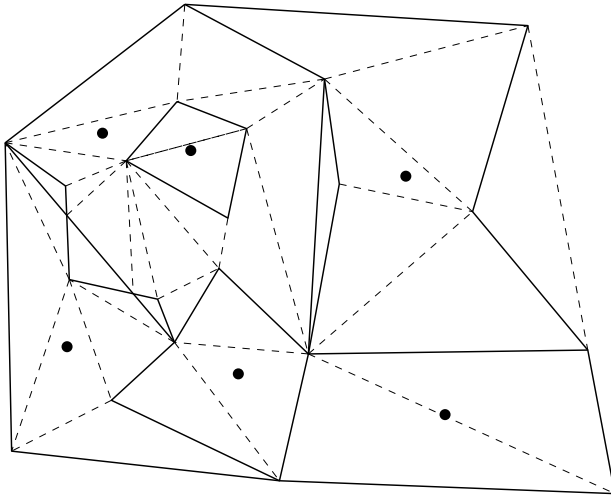
# Steps of our approach



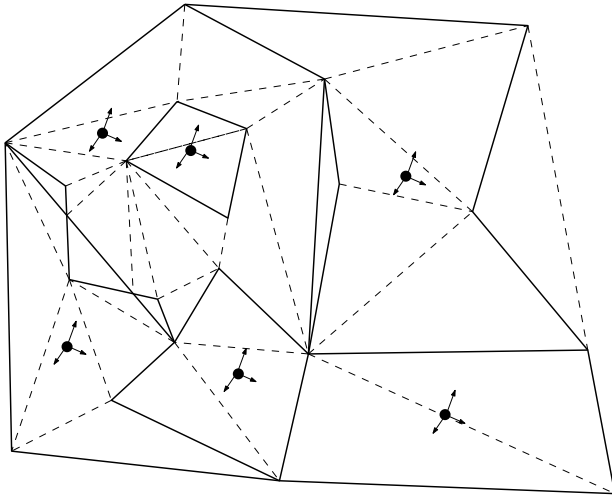
# Steps of our approach



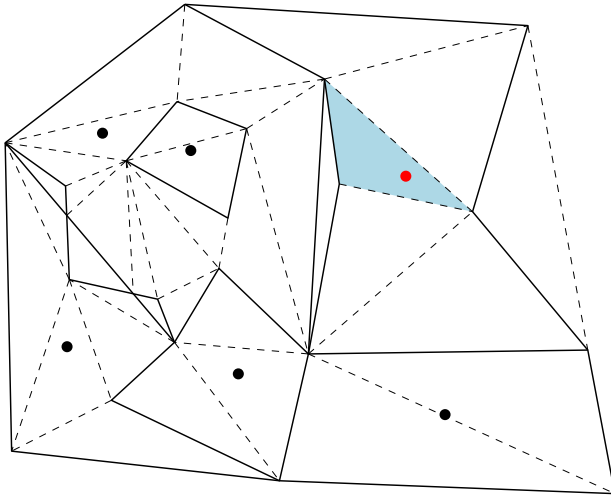
# Steps of our approach



# Steps of our approach

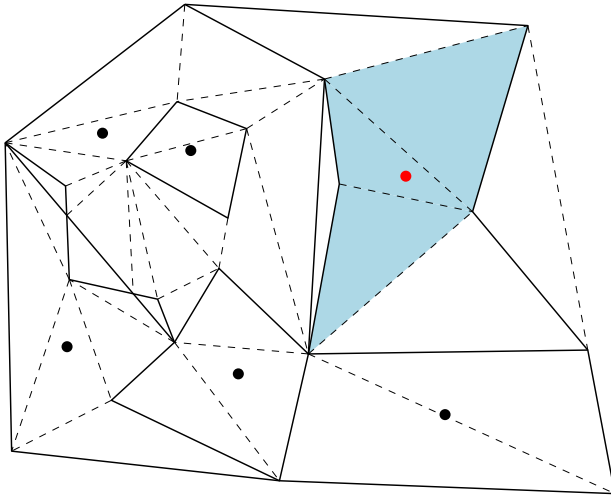


# Steps of our approach

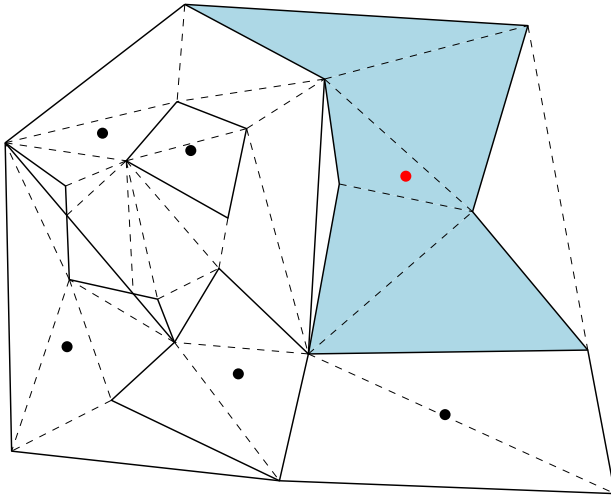




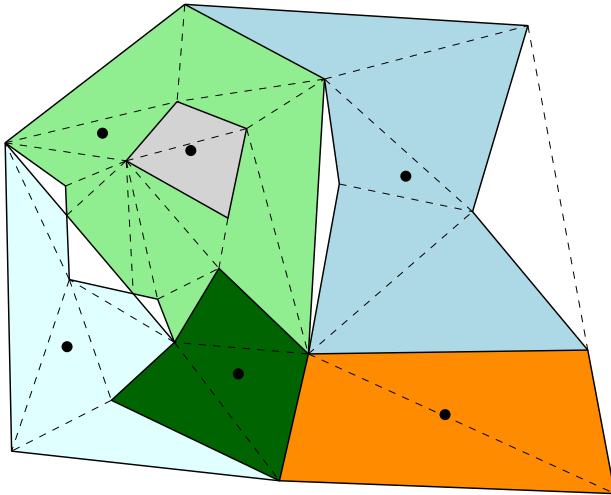
# Steps of our approach



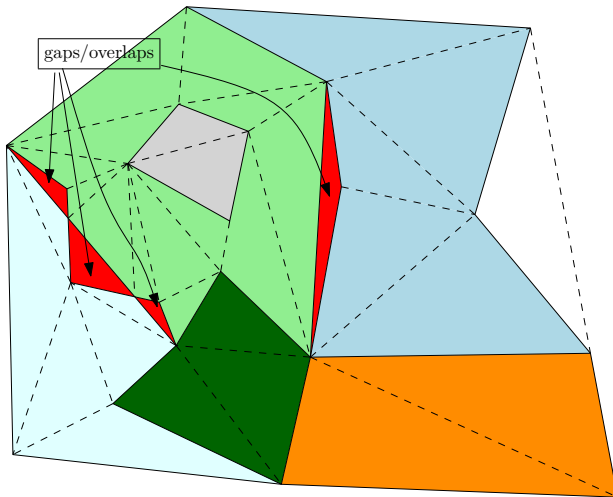
# Steps of our approach



# Steps of our approach

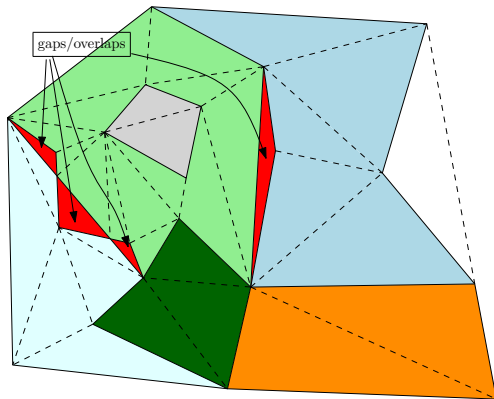


# Steps of our approach



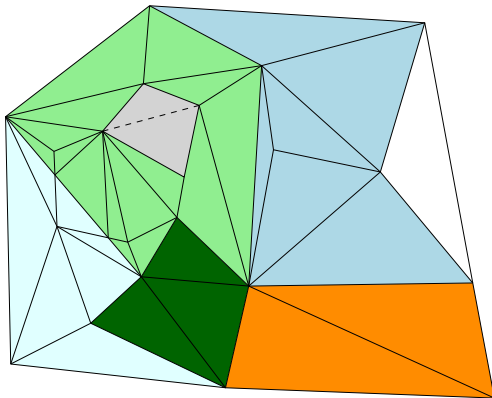
# If it's broken, then fix it

- 1 Repair = simply re-flagging triangles
- 2 No need to modify and update the planar graph (slow operation)



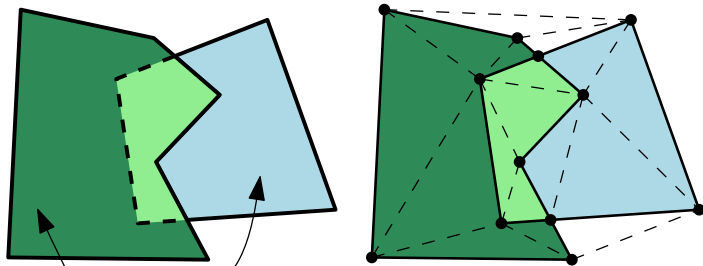
# If it's broken, then fix it

- 1 Repair = simply re-flagging triangles
- 2 No need to modify and update the planar graph (slow operation)



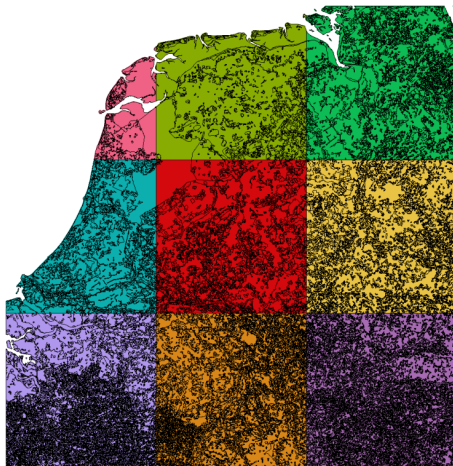
# Implementation

- 1 Fast implementation in C++, with CGAL and OGR
- 2 Open-source code, you can test it
- 3 Numerical and geometric robustness. Points do not move during processing.



2 input polygons that overlap

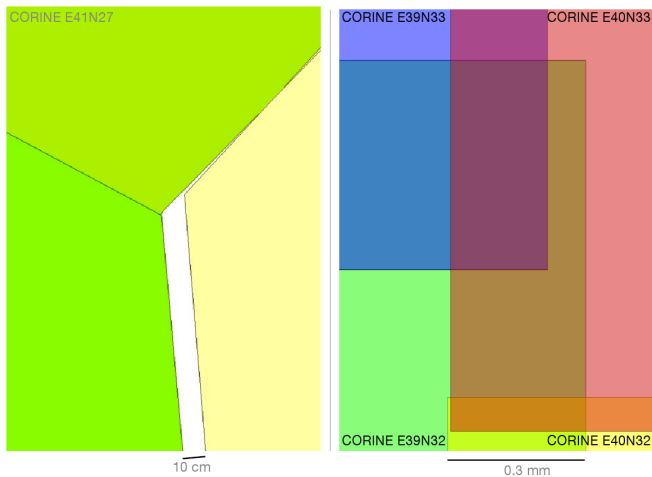
# Experiments with CORINE dataset



Can process around 40 tiles in  $< 1h$   
(around 120 000 polygons with 4GB main memory)

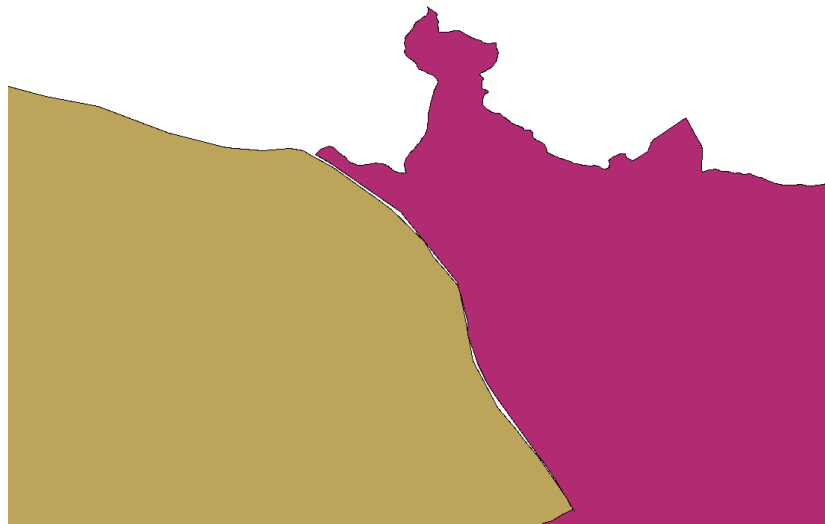


# Experiments with CORINE dataset



**None** of the tiles fit perfectly with their neighbours.

## Future work: edge matching

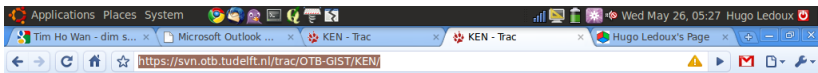


# More Information?

Hugo Ledoux

h.ledoux@tudelft.nl

<https://svn.otb.tudelft.nl/trac/OTB-GIST/KEN>

 Search[Login](#) | [Settings](#) | [Help/Guide](#) | [About Trac](#)[Wiki](#)[Timeline](#)[Roadmap](#)[Browse Source](#)[View Tickets](#)[Search](#)[Start Page](#) | [Index by Title](#) | [Index by Date](#) | [Last Change](#)

## Validation and Automatic Repair of Planar Partitions

From the paper:

Planar partitions—full tessellations of the plane into non-overlapping polygons—are frequently used in GIS to model concepts such as land cover, cadastral parcels or administrative boundaries. Since in practice planar partitions are often stored as a set of individual objects (polygons) to which attributes are attached (e.g. stored with a shapefile), and since different errors/mistakes can be introduced during their construction, manipulation or exchange, several inconsistencies will often arise in practice. The inconsistencies are for instance overlapping polygons, gaps and unconnected polygons. We present in this paper a novel algorithm to validate such planar partitions. It uses a constrained triangulation as a support for the validation, and permits us to avoid different problems that arise with existing solutions based on the construction of a planar graph. We describe in the [paper](#) the details of our algorithm, our implementation, how inconsistencies can be detected, and the experiments we have made with real-world data (the CORINE2000 dataset).

Since the submission of the paper, the algorithm has been extended and improved in order to:

- Obtain more information about the invalid situations that can occur.
- Maintain geometric and numerical robustness.
- Validate and repair individual polygons.
- Automatically repair polygons according to predefined criteria.

A fast implementation of the algorithm has been written in C++, using the [OGRE](#) and [CGAL](#) libraries. It is available for download [here](#).

### More Information