Click here to hide/show the code

A tour of point cloud processing

FOSS4G Belgium 2017 (http://2017.foss4g.be)

26 october 2017

Mathieu Carette

Notebook available at https://github.com/rockestate/point-cloud-processing (https://github.com/rockestate/point-cloud-processing)

Slides available at https://www.rockestate.be/point-cloud-processing/presentation/ (https://www.rockestate.be/point-cloud-processing/presentation/)

About me

- PhD in Mathematics (ULB, 2009)
- Postdocs (UIUC, UCLouvain, McGill)
- Data Scientist (KBC, Forespell)

- Now working on

ROCKESTATE

(https://www.rockestate.be)

Favorite software stack:

Where do 3D point clouds come from?

Out[1]:

Open LiDAR data for Brussels and Flanders : https://remotesensing.agiv.be/opendata/lidar/ (https://remotesensing.agiv.be/opendata/lidar/)

File formats and software

- LAS (https://www.asprs.org/committee-general/laser-las-file-format-exchange-activities.html) standard file format
- LAZ (https://www.laszip.org/) compressed file format

- PCL (http://pointclouds.org/) **P**oint **C**loud **L**ibrary
  - Open source: https://github.com/PointCloudLibrary/pcl (https://github.com/PointCloudLibrary/pcl)
  - C++
  - Powerful general purpose algorithms

- CGAL (https://www.cgal.org/) **C**omputational **G**eometry **A**lgorithms **L**ibrary
  - Open source: https://github.com/CGAL/cgal (https://github.com/CGAL/cgal)
  - C++
  - State of the art 2D and 3D geometry algorithms

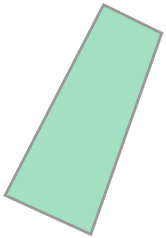
- PDAL (https://www.pdal.io) **P**oint **D**ata **A**bstraction **L**ibrary
  - Open source: https://github.com/PDAL/PDAL (https://github.com/PDAL/PDAL)
  - C++, command-line, python
  - Wraps some PCL functionality
  - For windows users: part of the OSGeo4W (https://trac.osgeo.org/osgeo4w/) distribution (32bit only)

- LAStools (https://rapidlasso.com/lastools/) from RapidLasso
  - Proprietary, preferred pricing for academic use
  - Windows only, runs on wine
  - command-line, GUI
  - Open source laszip (https://www.laszip.org) compression/decompression: https://github.com/LASzip/LASzip (https://github.com/LASzip/LASzip)

Let's process some point clouds

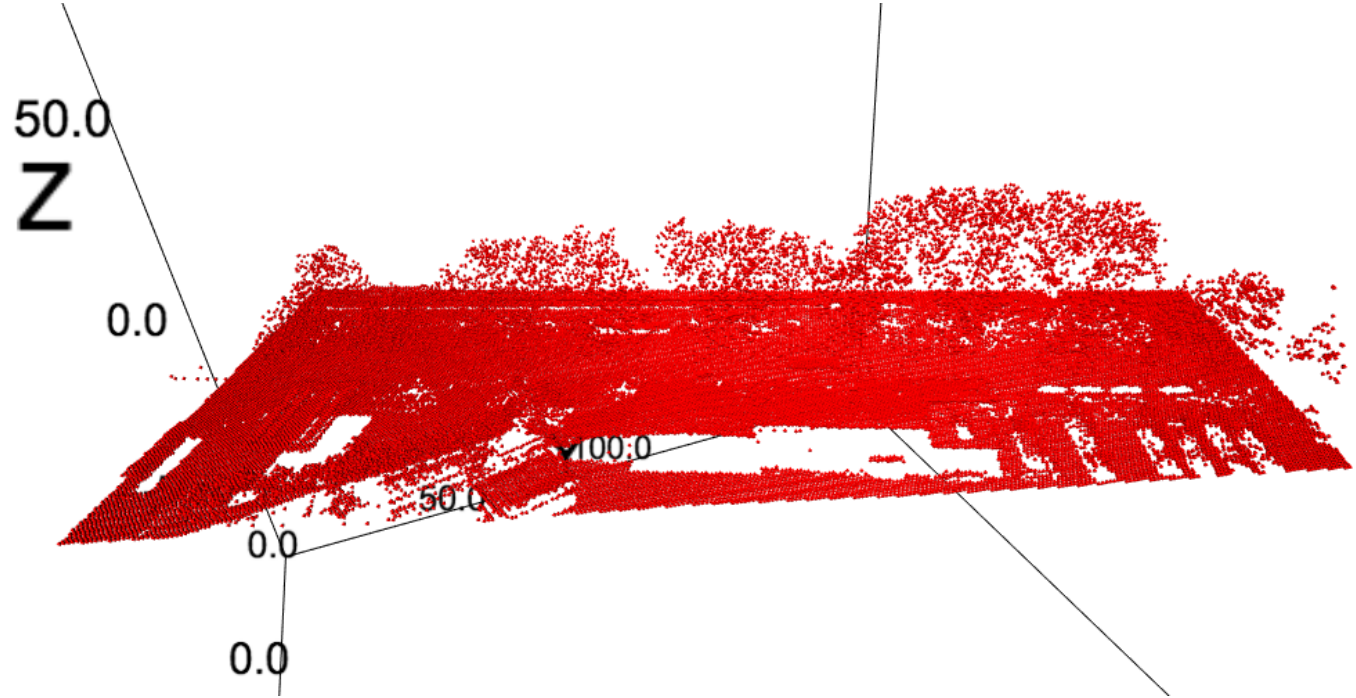
Selecting street portion with a polygon

POLYGON ((148776.2701930787 172919.8420462897, 148817.7993776024 172899.4757044148, 148767.1519177397 172755.6518779453, 148705.9825845991 172783.5876676841, 148776.2701930787 172919.8420462897))

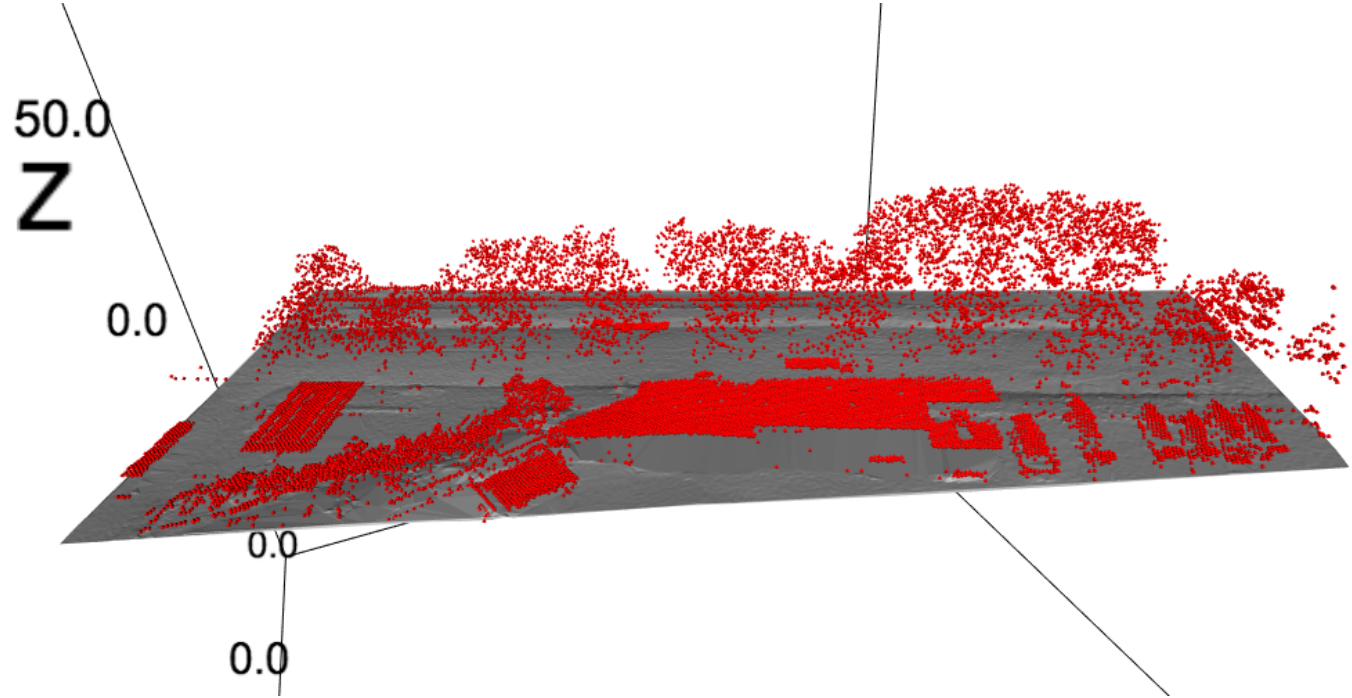


Pipeline selected 48573 points

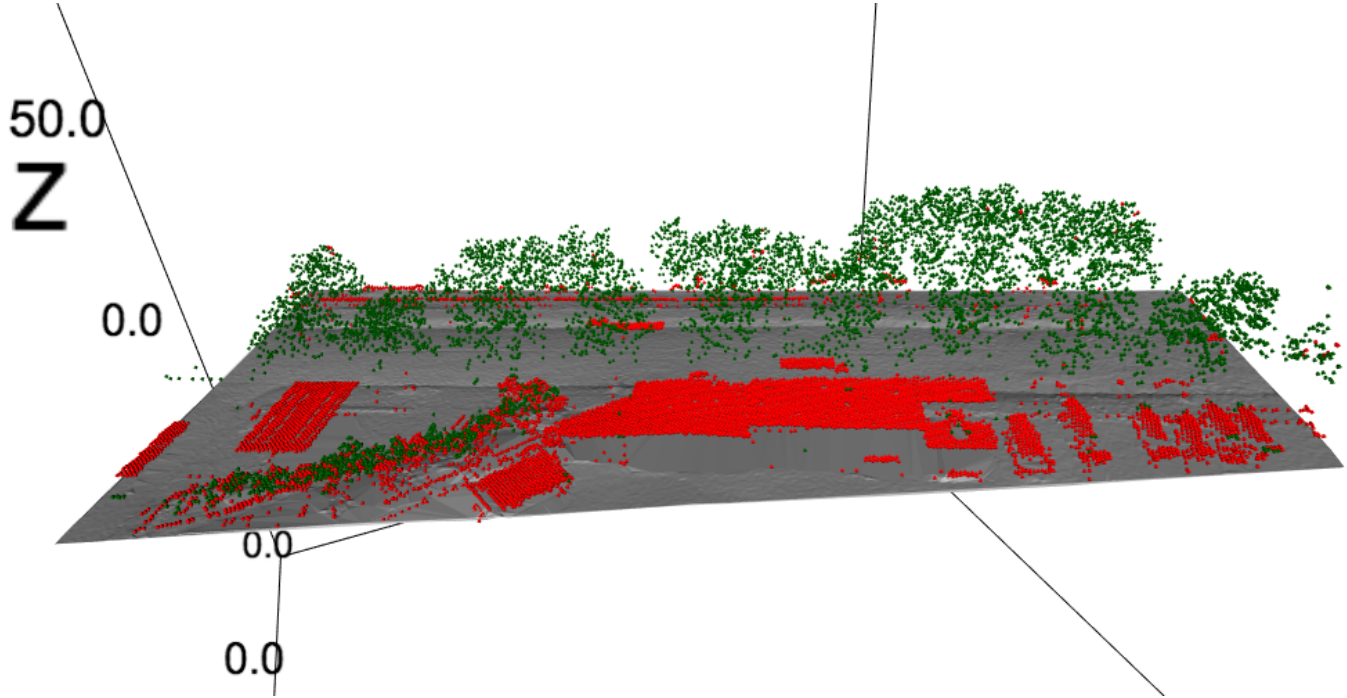
Original data



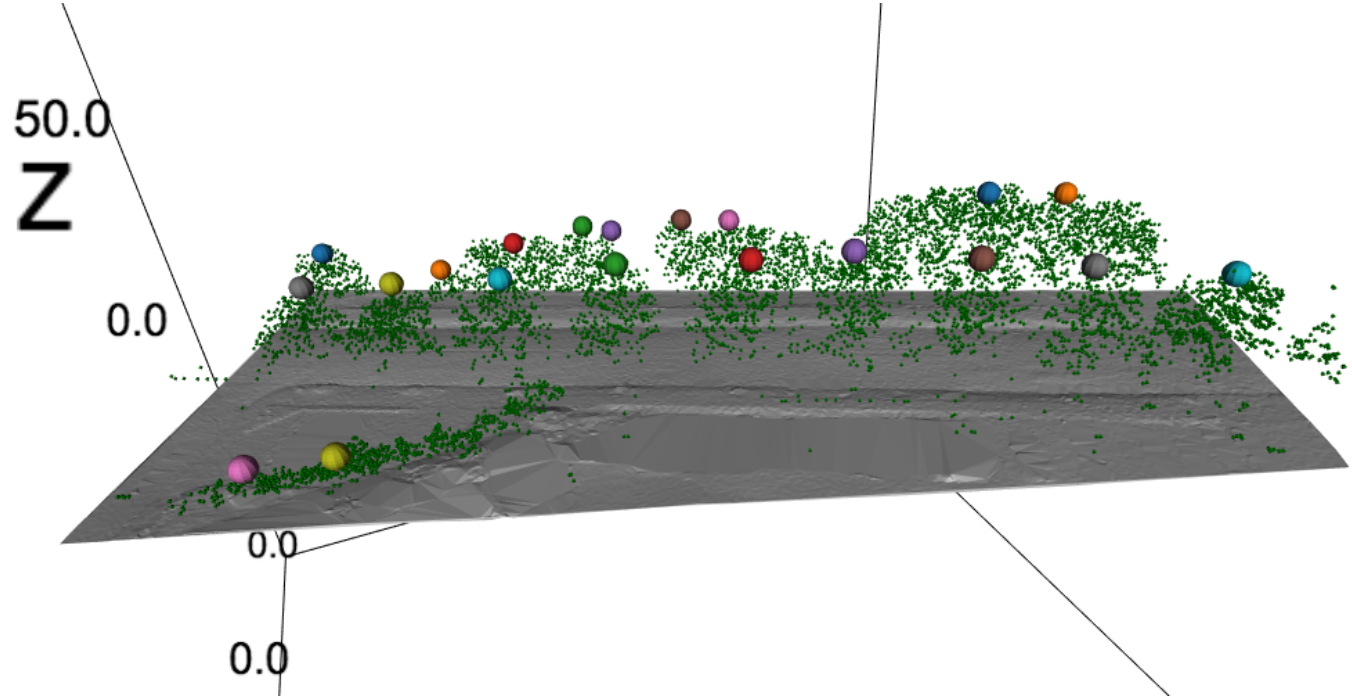
Use gound / non-ground classification



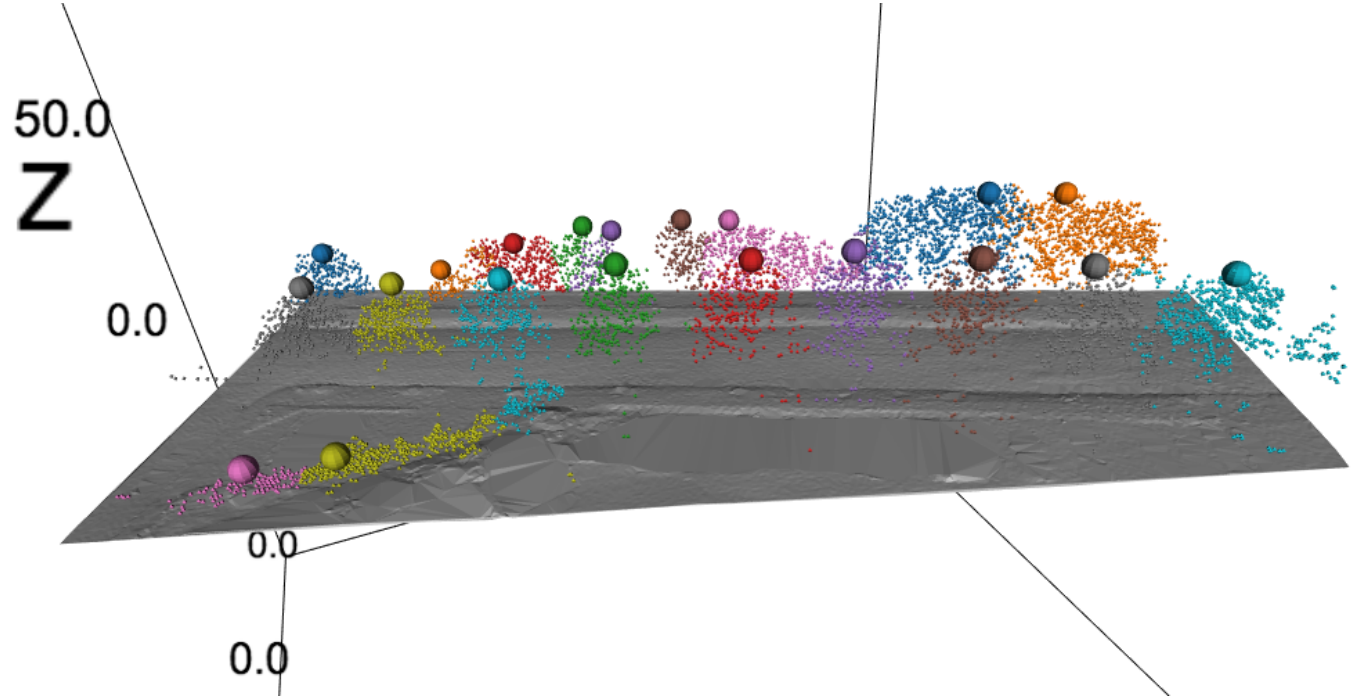
Use point flatness to separate trees from the rest



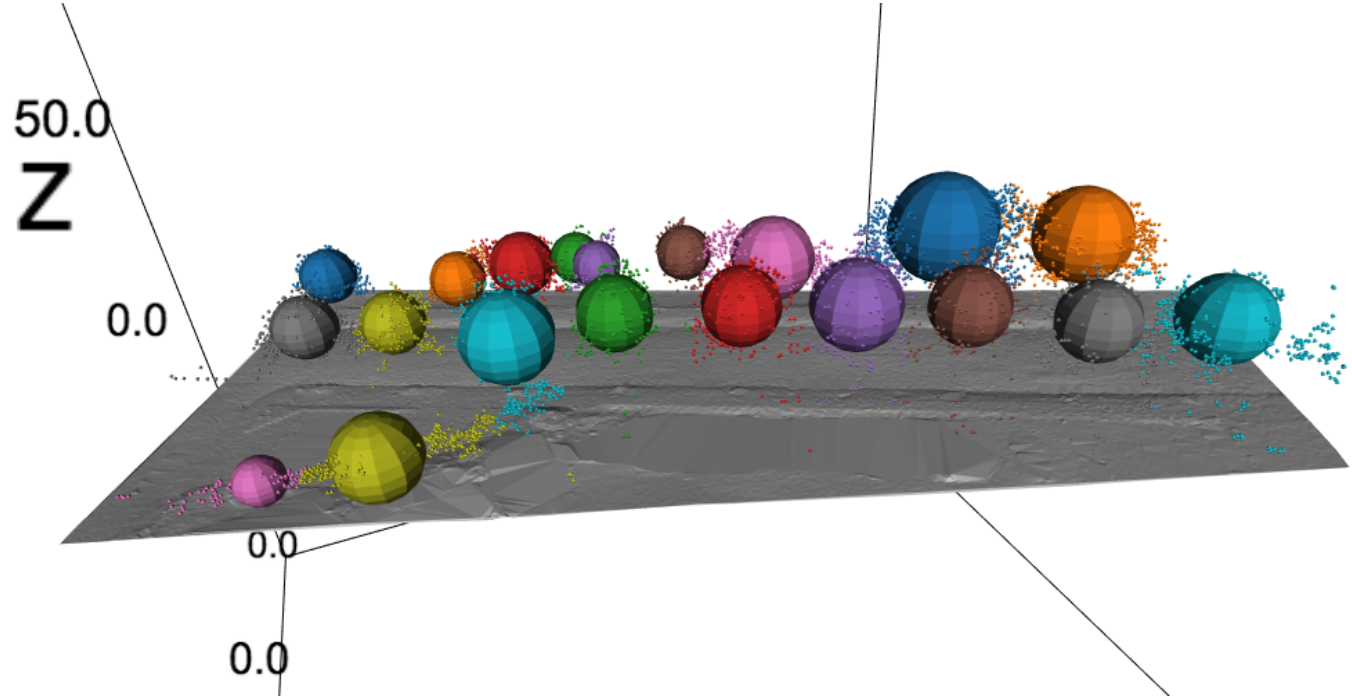
Find treetops as local maxima



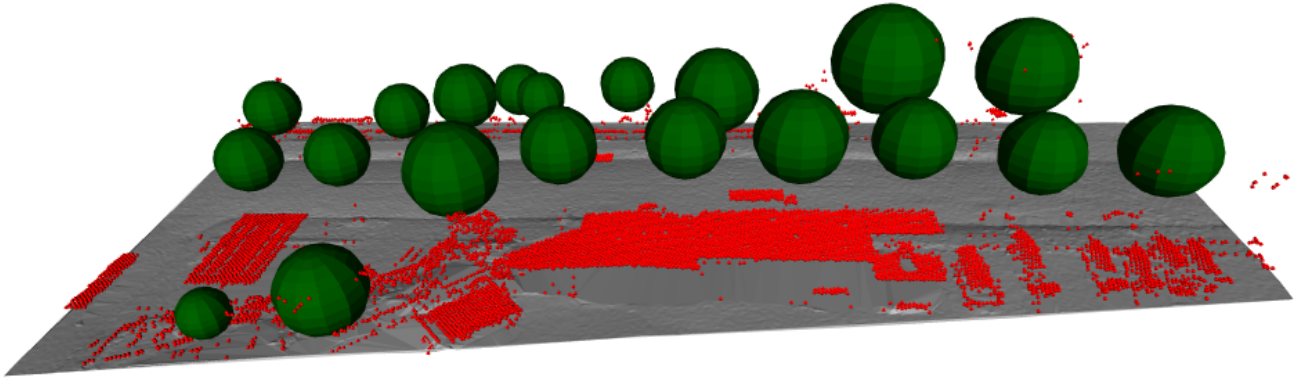
Separate trees using closest treetop



Model each tree individually

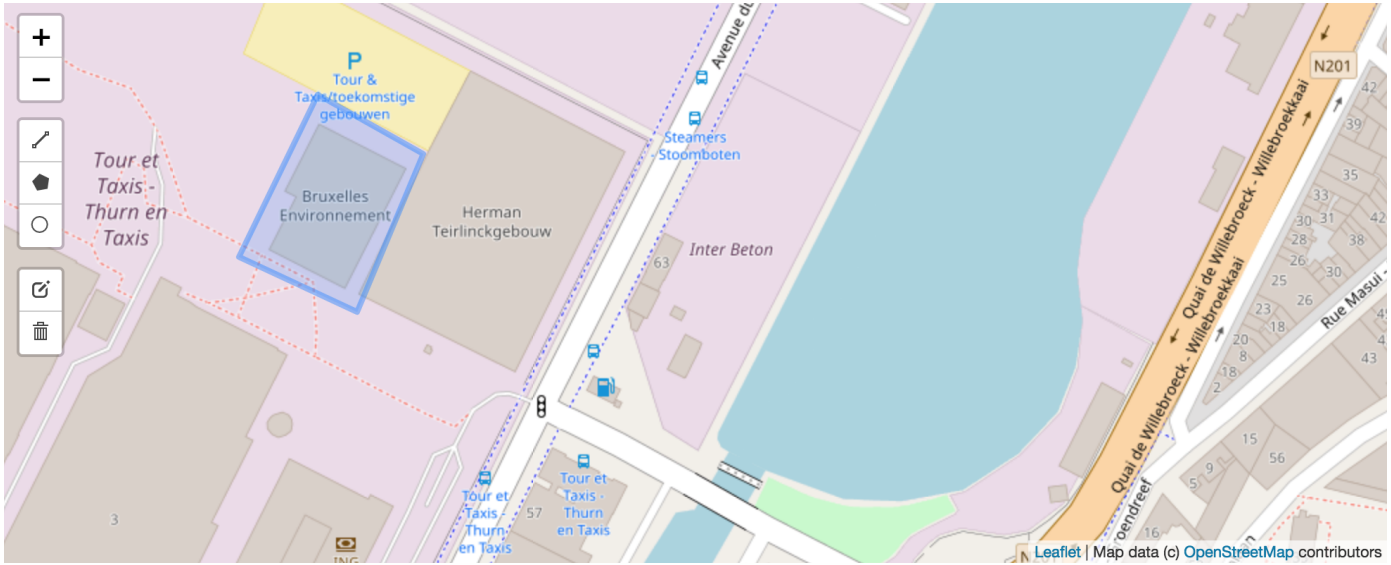


Final street model



Building Modeling

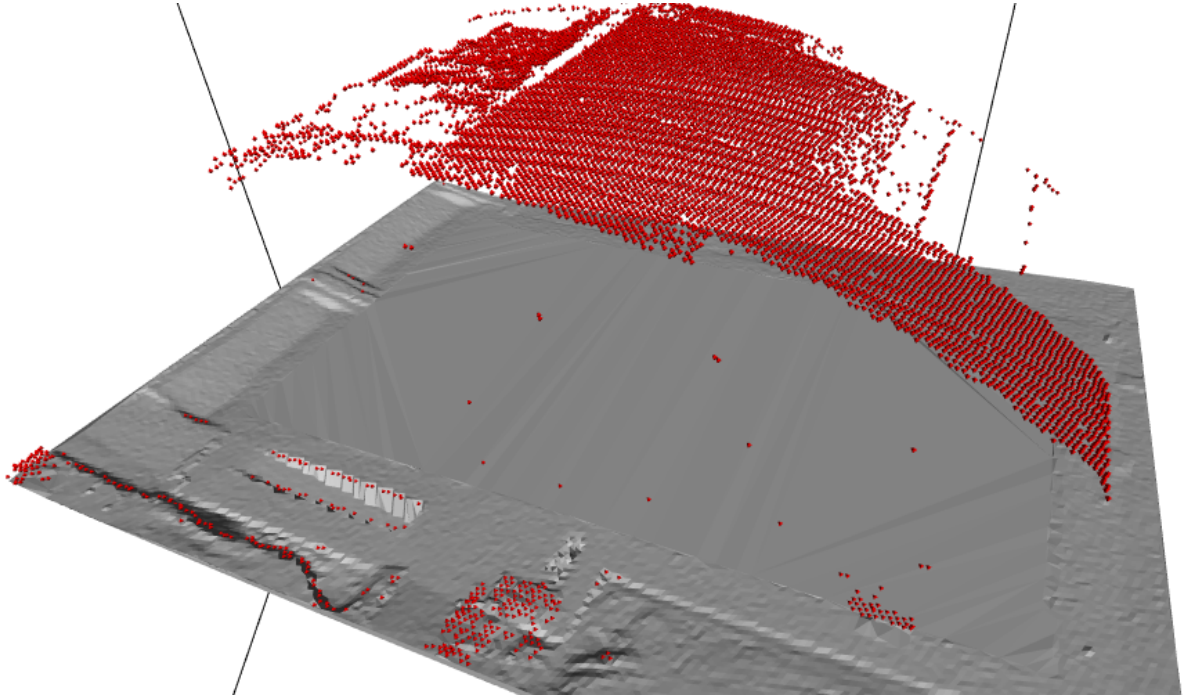
Selecting conference building with a polygon



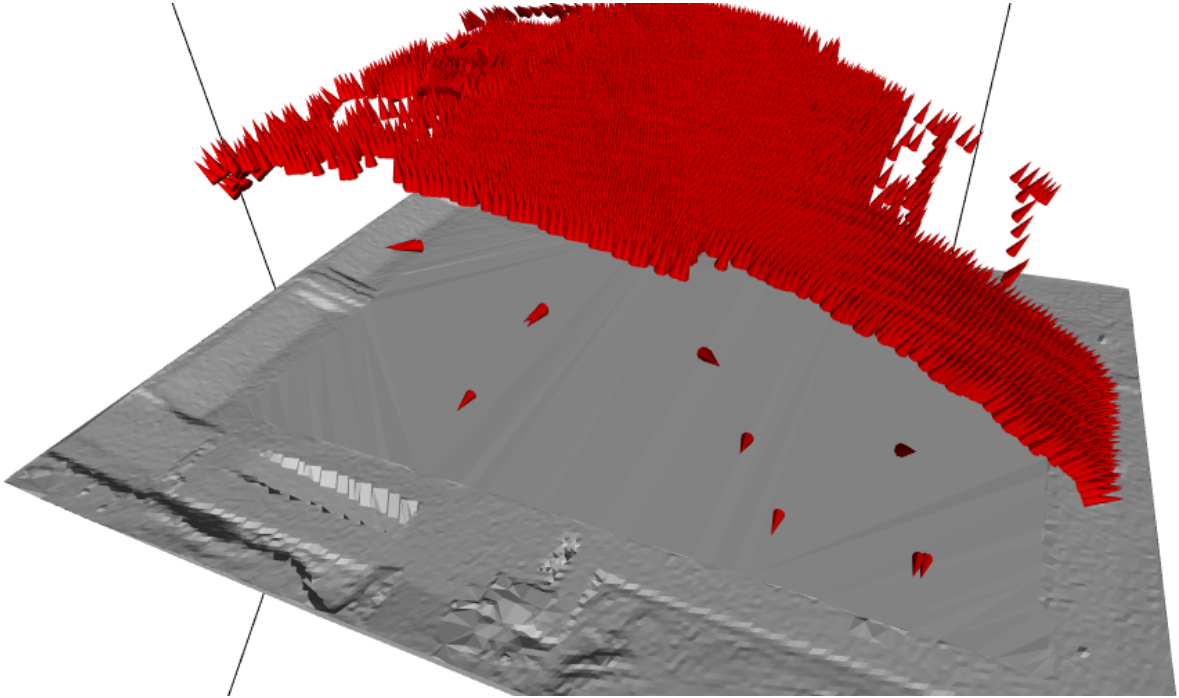

```
POLYGON ((148605.5652729562 172940.129142697, 148662.2259857805 172908.5223111287, 148626.7249616213 1728
24.0989997433, 148563.2344679619 172853.4831915358, 148605.5652729562 172940.129142697))
```

```
Pipeline selected 23525 points
```

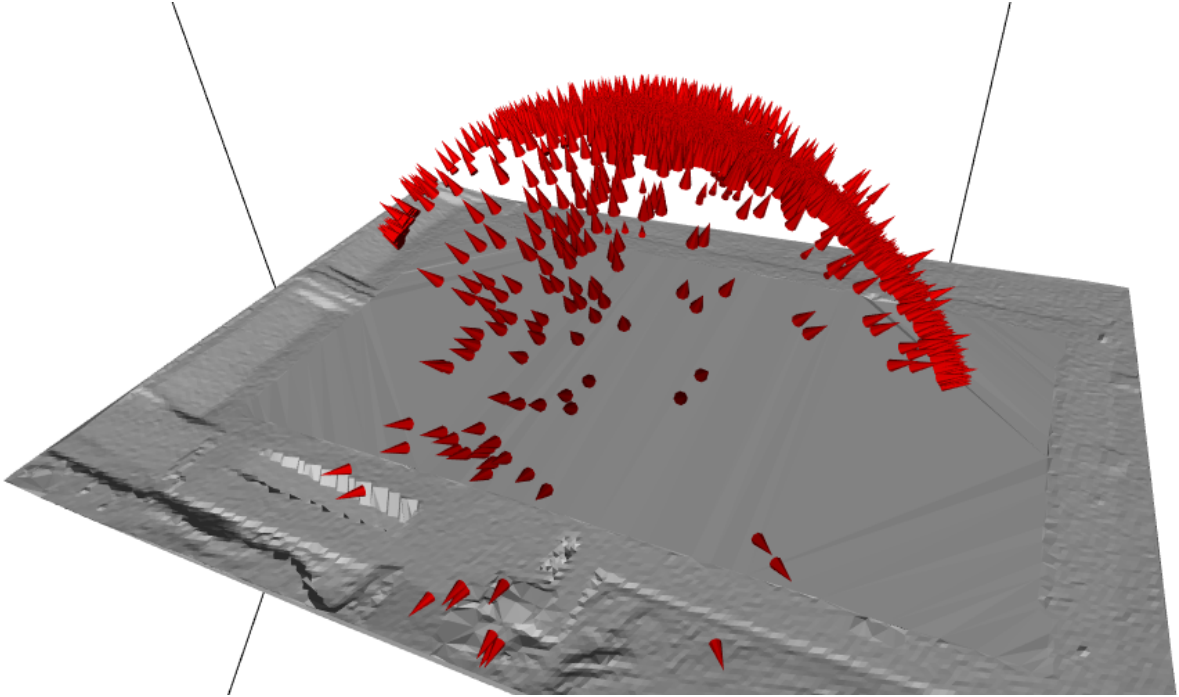
Original data



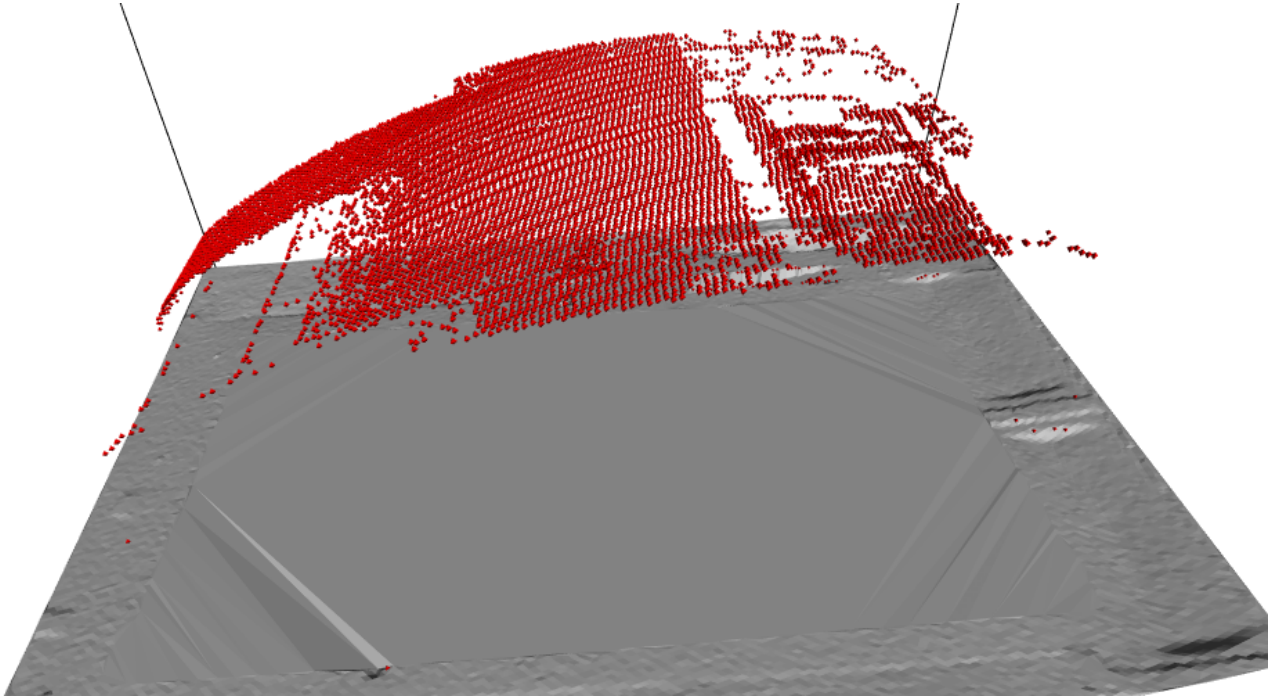
Visualize normals



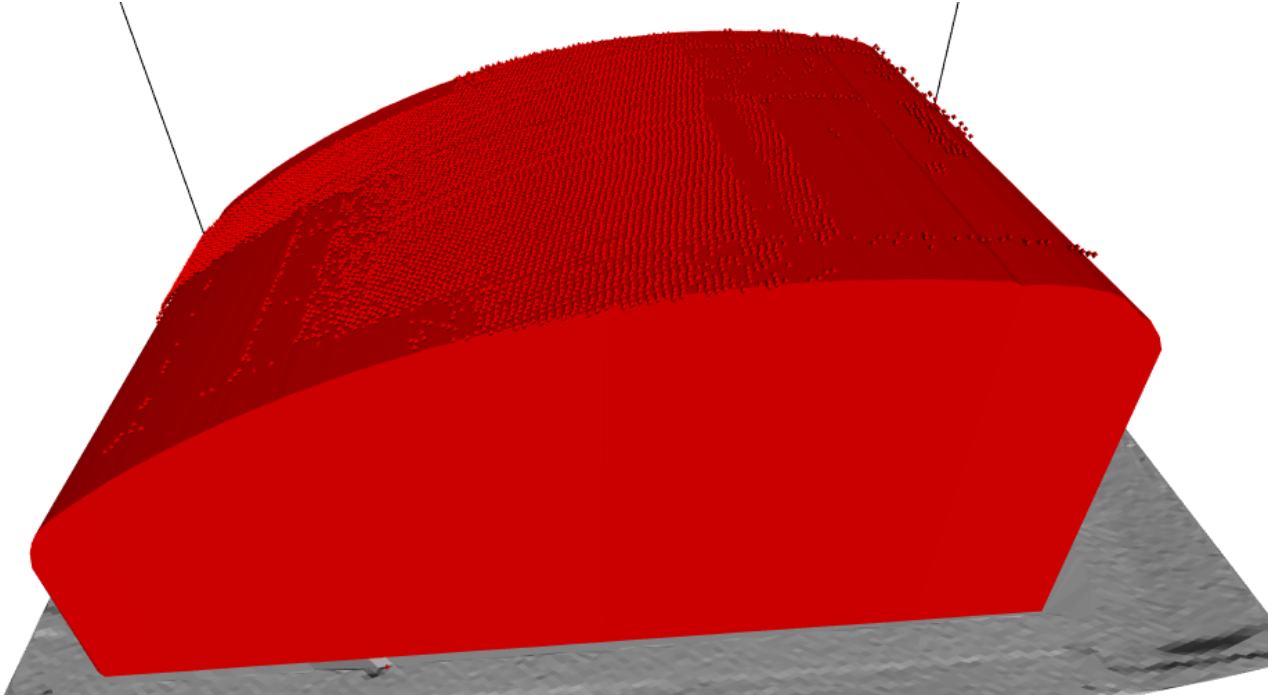
Infer building orientation using normals



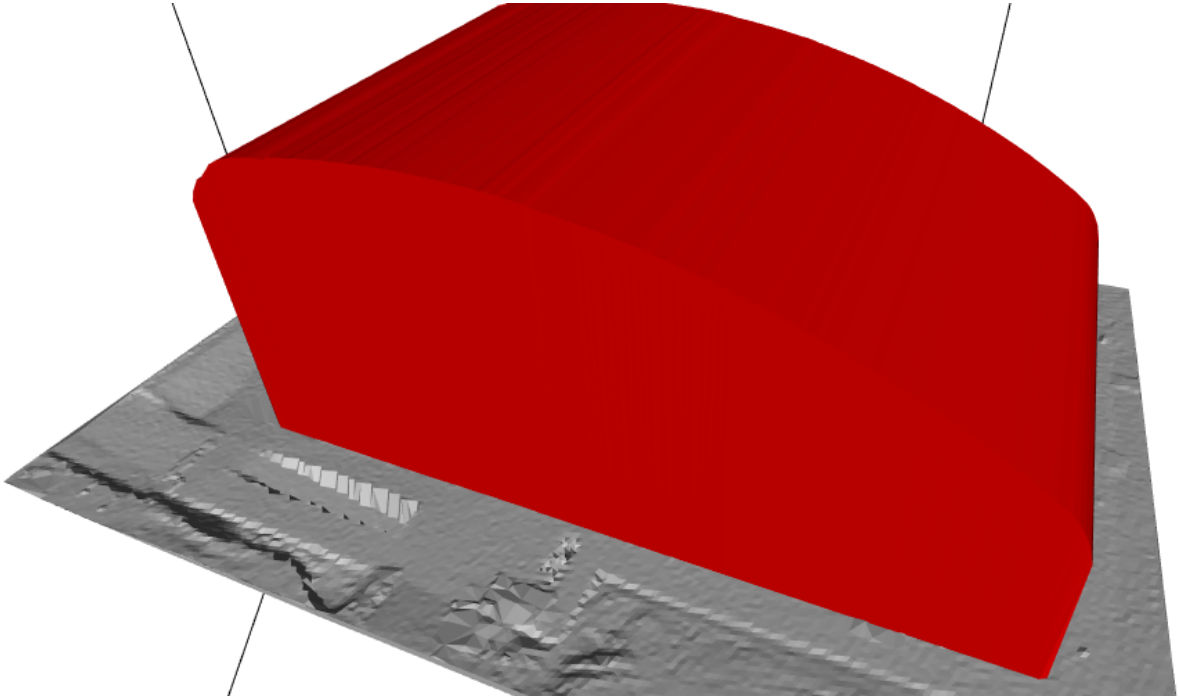
Align building orientation along X-Y axes



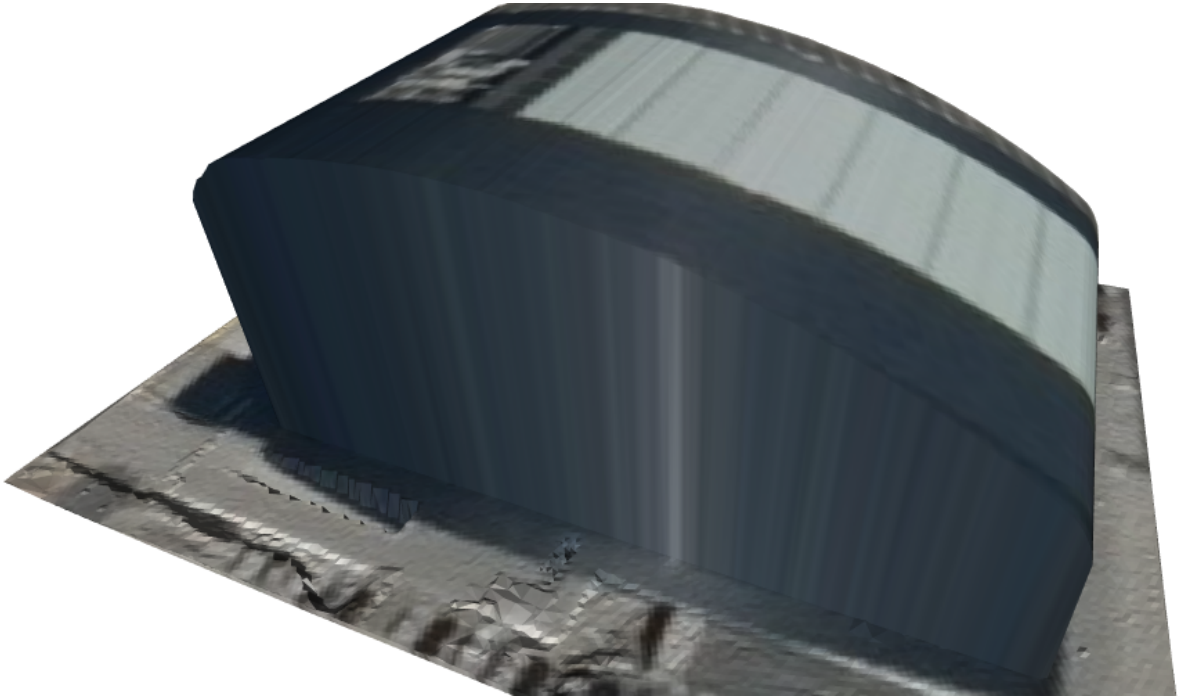
Model the building using axis-aligned view



Rotate back to original coordinate system



Add texture from aerial imagery



Thank you!