A tour of point cloud processing

Mathieu Carette

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

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

(previous versions here)

About me

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

[Link to website]

Favorite software stack:

![Python](https://duke-energy.scene.pages/i/conda.png)

Where do 3D point clouds come from?

```
Out[2]:
```

![YouTube video](https://www.youtube.com/watch?v=dQw4w9WgXcQ)
File formats and software

- LAZ (https://www.laszip.org/) compressed file format

- PCL (http://pointclouds.org/) Point Cloud Library
  - Open source: https://github.com/PointCloudLibrary/pcl (https://github.com/PointCloudLibrary/pcl)
  - C++
  - Powerful general purpose algorithms

- CGAL (https://www.cgal.org/) Computational Geometry Algorithms Library
  - C++
  - State of the art 2D and 3D geometry algorithms

- PDAL (https://www.pdal.io/) Point Data Abstraction Library
  - 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/oseo4w/) distribution

- LASTools (https://rapidlasso.com/lasstools/) from RapidLasso
  - Proprietary, preferred pricing for academic use
  - Windows only, runs on wine
  - command-line, GUI

Let's process some point clouds
Selecting street portion with a polygon

POLYGON ((150687.8518289287 167058.3858805448, 150939.3740217351 167072.33228858, 150980.4548986266 166743.7380920332, 150919.5663185167 166714.6929215267, 150754.0807228128 166712.885090881, 150581.4498102131 166733.4497666787, 150687.8518289287 167058.3858805448))

CPU times: user 15.9 s, sys: 318 ms, total: 16.2 s
Wall time: 16.3 s
Pipeline selected 473184 points (4.4 pts/m²)
Use ground / 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 building with a polygon

POLYGON ((150876.6899425487 166855.1808815384, 150913.4053620266 166873.7645316971, 150904.3812825281 166890.1151175071, 150861.390885691 166886.3253158694, 150876.6899425487 166855.1808815384))

CPU times: user 5.39 s, sys: 880 ms, total: 6.27 s
Wall time: 6.5 s
Pipeline selected 31223 points (28.8 pts/m2)

Original data
Visualize normals

Infer building orientation using normals
Align building orientation along X-Y axes

Model the building using axis-aligned view
Things to look out for

pdal

- Fast point-in-polygon algorithm implemented
- Apache arrow support
- Conda packaging
jupyter

- C++ jupyter kernels
- Jupyterlab

Thank you!