PDAL Algorithm Development Deep Dive

Brad Chambers

PDAL Algorithm Development Deep Dive

> In this talk, we will outline the steps to develop a point cloud filtering algorithm using PDAL's Python extension. We will show how PDAL can be installed via Conda, and in a Jupyter notebook will work through the algorithm development process, showing how the finished algorithm can be distributed and executed using the PDAL command line interface.

PDAL: It's not that hard

> In which we demonstrate that installing PDAL is easy, writing processing pipelines is straightforward, Python facilitates data analysis and algorithm development, and oh yeah, you can write some plugins too.

Overview

- Refresher
- Conda packages
- Development

Binder Links

- https://mybinder.org/
- Runable Jupyter notebooks

Refresher

PDAL Pipeline

- Pipelines are composed of stages
- Stages read, writer, or filter data
- Pipelines are written as JSON

Usage

```
$ pdal pipeline <pipeline>
```

Example #1

Transcoding pipeline

```
{
    "pipeline": [
        "input.las",
        "output.laz"
    ]
}
```

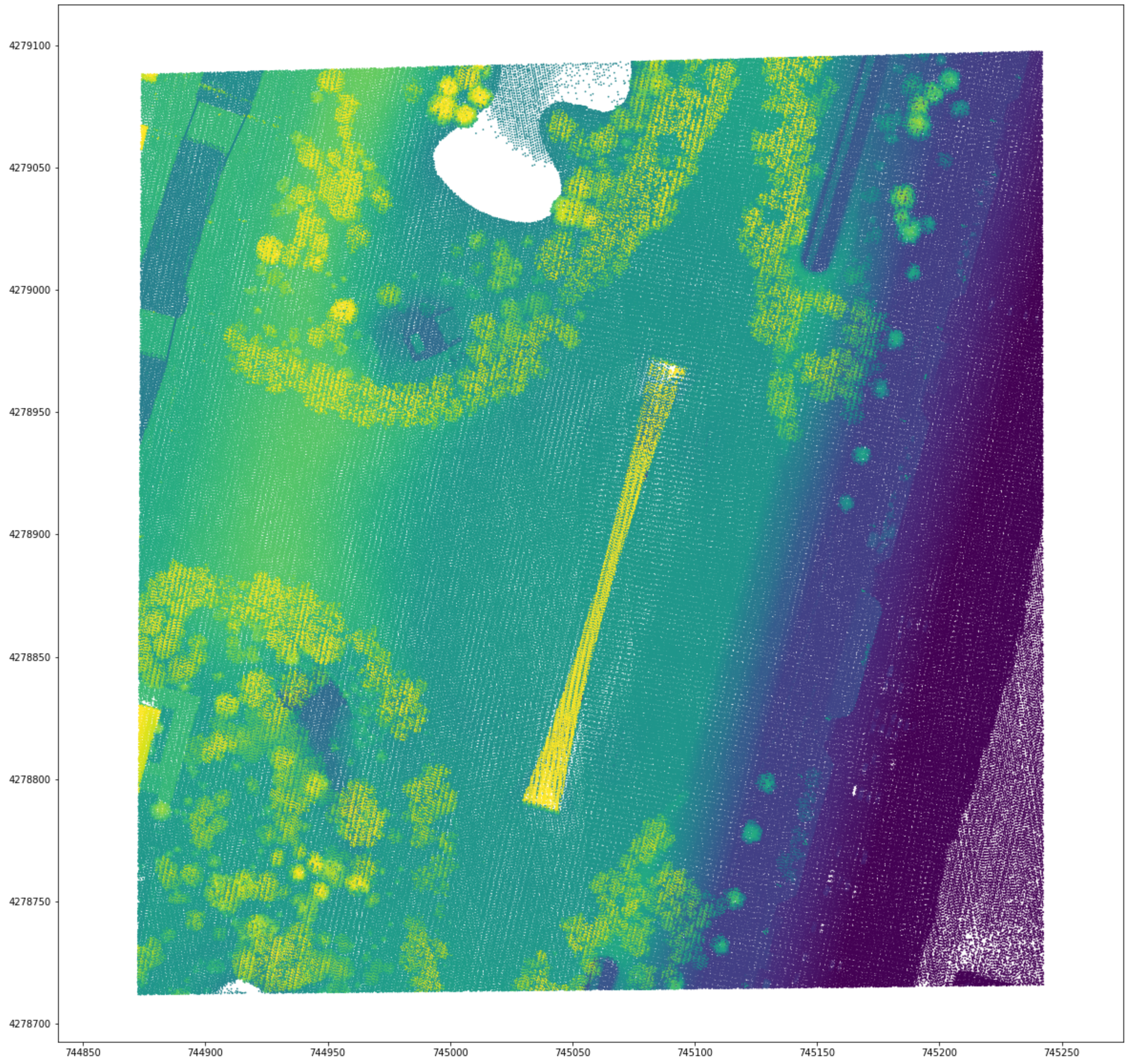
Launch Binder

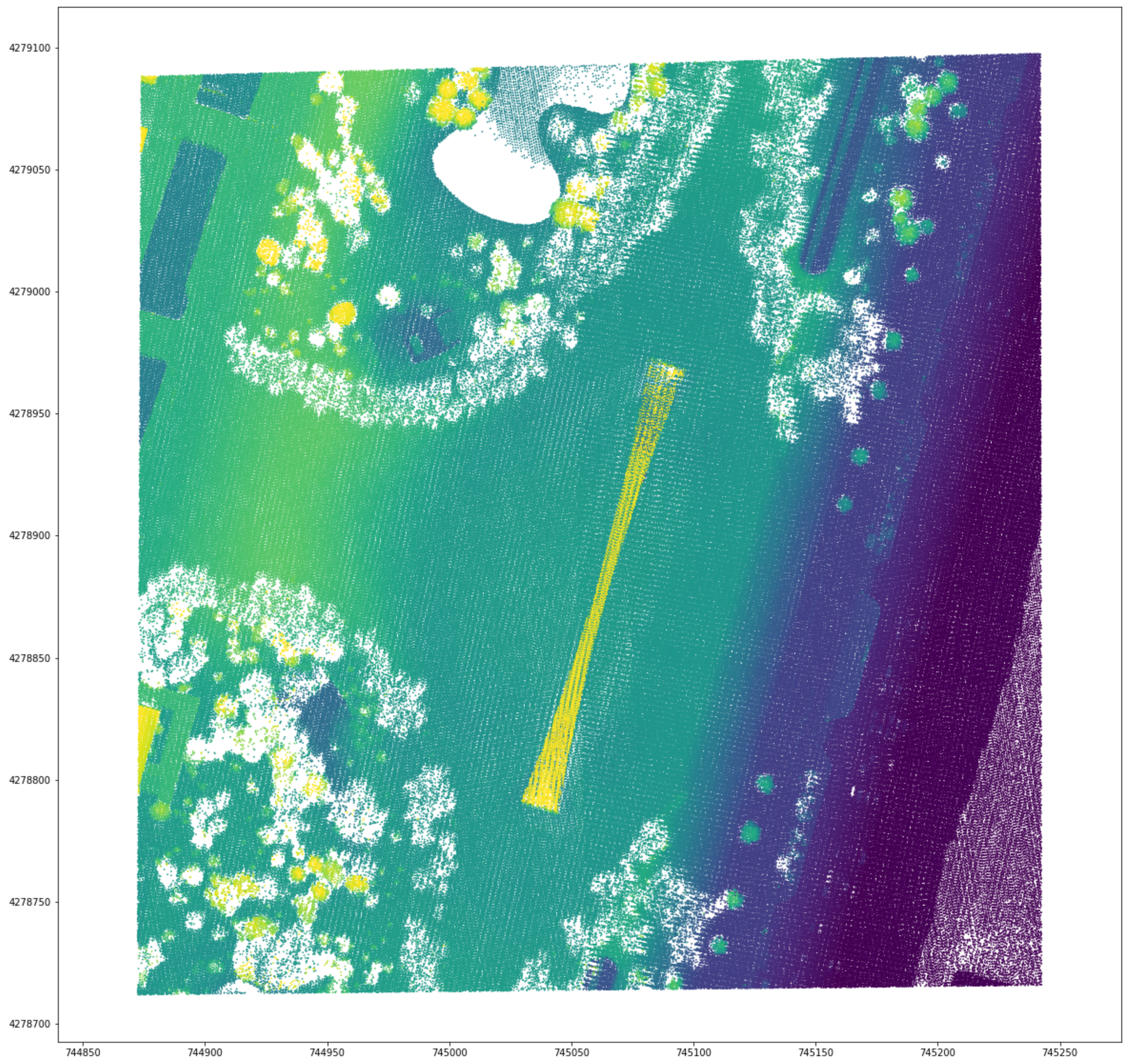
Example #2

Transcoding pipeline with range filtering

```
{
    "pipeline": [
        "input.las",
        {
            "type": "filters.range",
            "limits": "NumberOfReturns[1:1]"
        },
        "output.laz"
    ]
}
```

Launch Binder


4279100
4279050
4279000
4278950
4278900
4278850
4278800
4278750
4278700
744850
744900
744950
745000
745050
745100
745150
745200
745250


4279100
4279050
4279000
4278950
4278900
4278850
4278800
4278750
4278700
744850
744900
744950
745000
745050
745100
745150
745200
745250

PDAL Translate

- Translations convert data
- Translations apply filter stages sequentially

Usage

```
$ pdal translate <input> <output> [filter...]
```

Example #3

Transcoding translation

```
$ pdal translate input.las output.laz
```

Launch Binder

Example #4

Transcoding translation with range filtering

```
$ pdal translate input.las output.laz range \
    --filters.range.limits="NumberOfReturns[1:1]"
```

Launch Binder

Example #5

Filter-only pipeline through `pdal translate`

```
{
    "pipeline": [
        {
            "type": "filters.range",
            "limits": "NumberOfReturns[1:1]"
        }
    ]
}
```

Usage

```
$ pdal translate <input> <output> [--json pipeline]
$ pdal translate input.las output.laz --json only.json
```

Launch Binder

Conda Packages

Why Conda?

- Multiple platforms (macOS, Linux, and Windows)
- Package management (install/update packages & dependencies)
- Environment management

New Conda Packages

All packages hosted on conda-forge.

- hexer v1.4.0
- laszip v3.2.2
- laz-perf v1.1.0
- nitro v2.7 (hobu branch)
- pcl v1.8.1
- pdal v1.7.2
- python-pdal v2.0.0

Enabled Plugins

- Greyhound
- Hexbin
- Icebridge
- NITF
- PCL
- pgpointcloud
- Python
- SQLite

Basic Installation

```
$ conda install -c conda-forge pdal
```

Create Environment

```
$ conda create -n pdalenv -c conda-forge pdal
$ conda activate pdalenv
```

Environment YAML

pdalenv.yml

```
name: pdalenv
channels:
  - conda-forge
  - defaults
dependencies:
  - pdal
  - python-pdal
```

Create Environment from YAML

```
$ conda env create -f pdalenv.yml
$ conda activate pdalenv
```

Verify Installation

```
$ pdal --version
---------------------------------------------------------
pdal 1.7.1 (git-version: Release)
---------------------------------------------------------
```

Development

- Pipeline
- Core stages
- Python

Prerequisites

Read the docs! https://pdal.io/

- PDAL pipelines
- PDAL reader, writer, and filter stages

Pipeline Development

- No code needed, only JSON
- Common tasks
    - Classification
    - Registration
    - Resampling

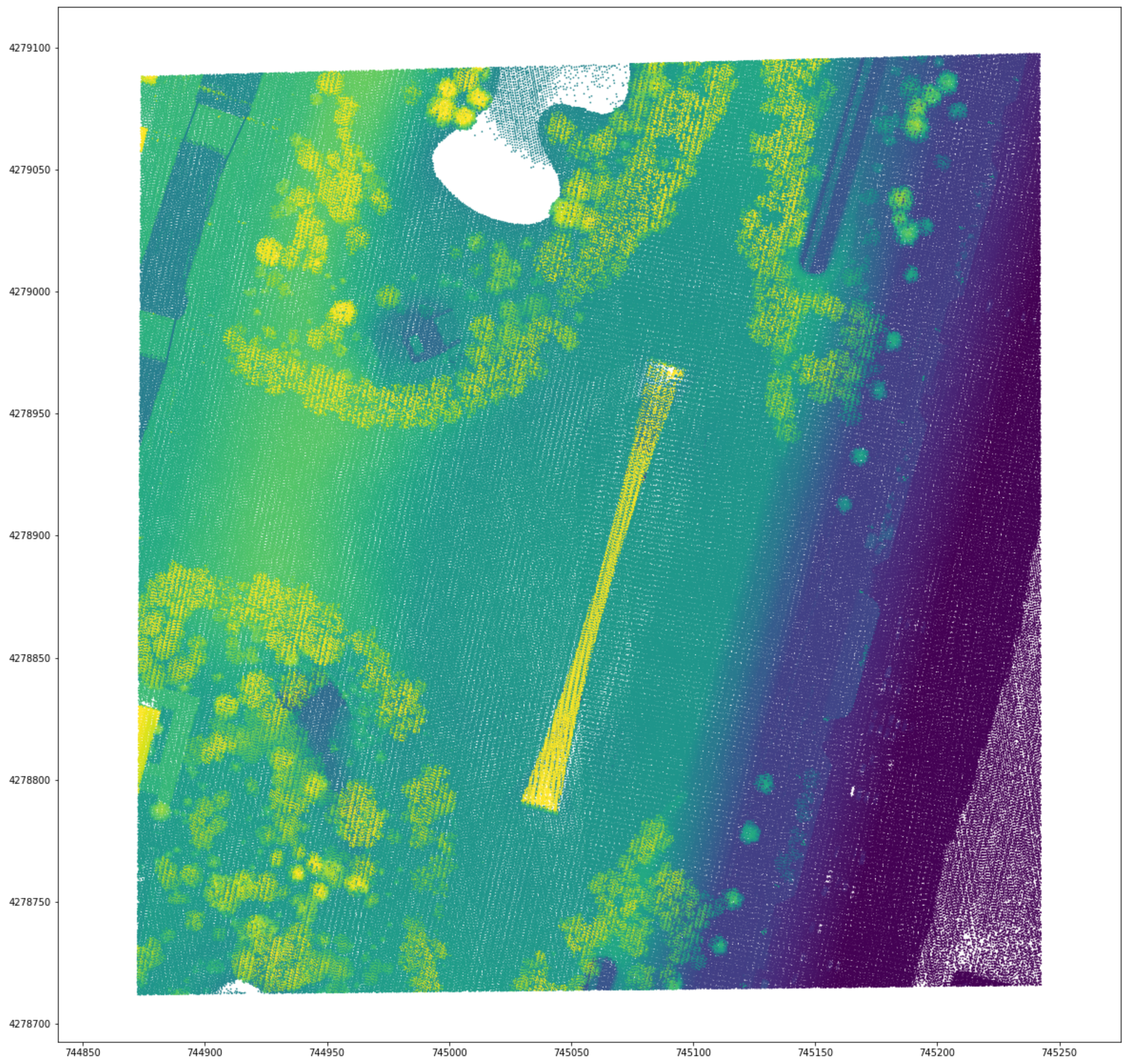
Gadomski, P.J. "Glacier surface velocities from point clouds using an open-source toolchain"

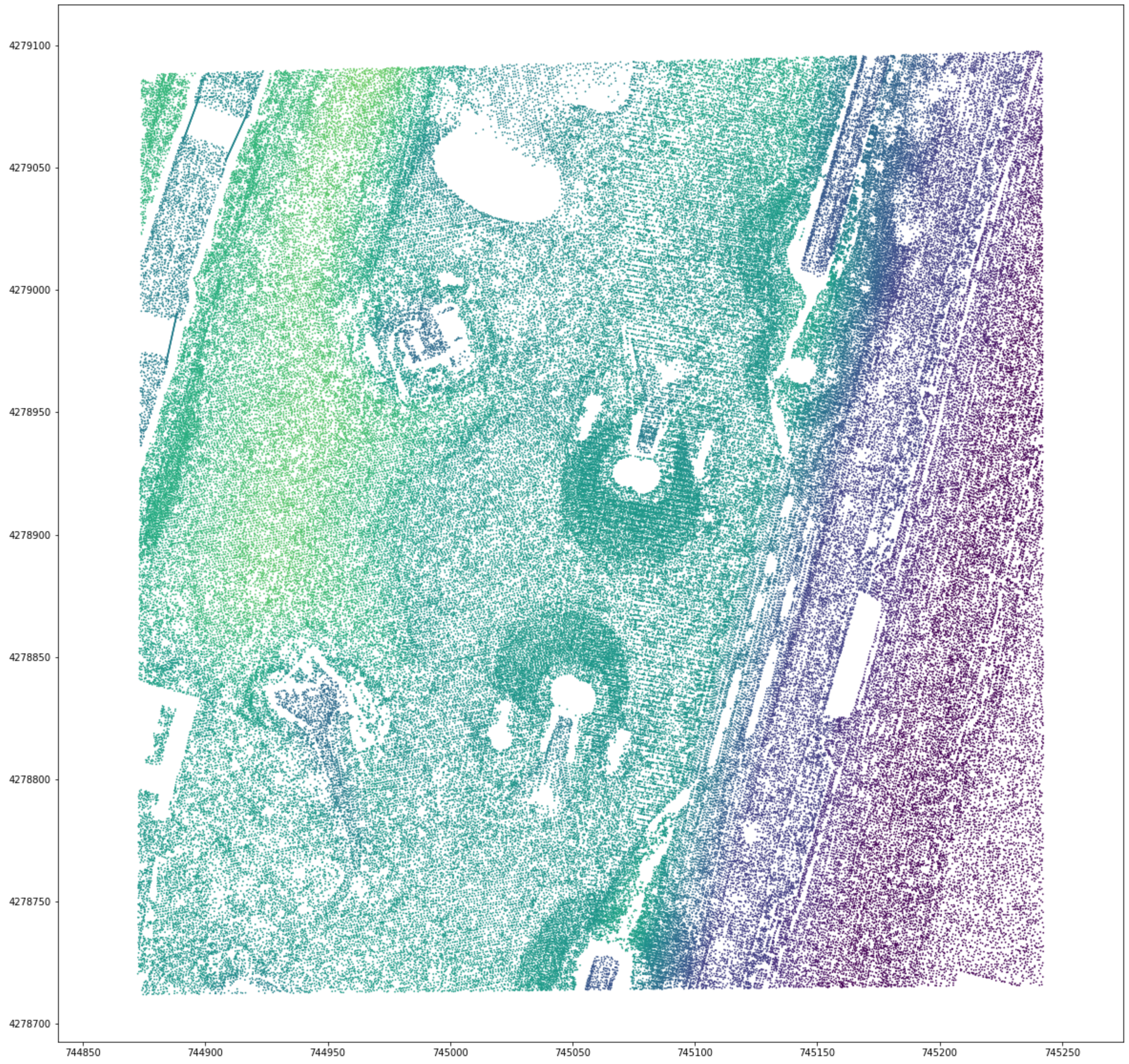
Example #6

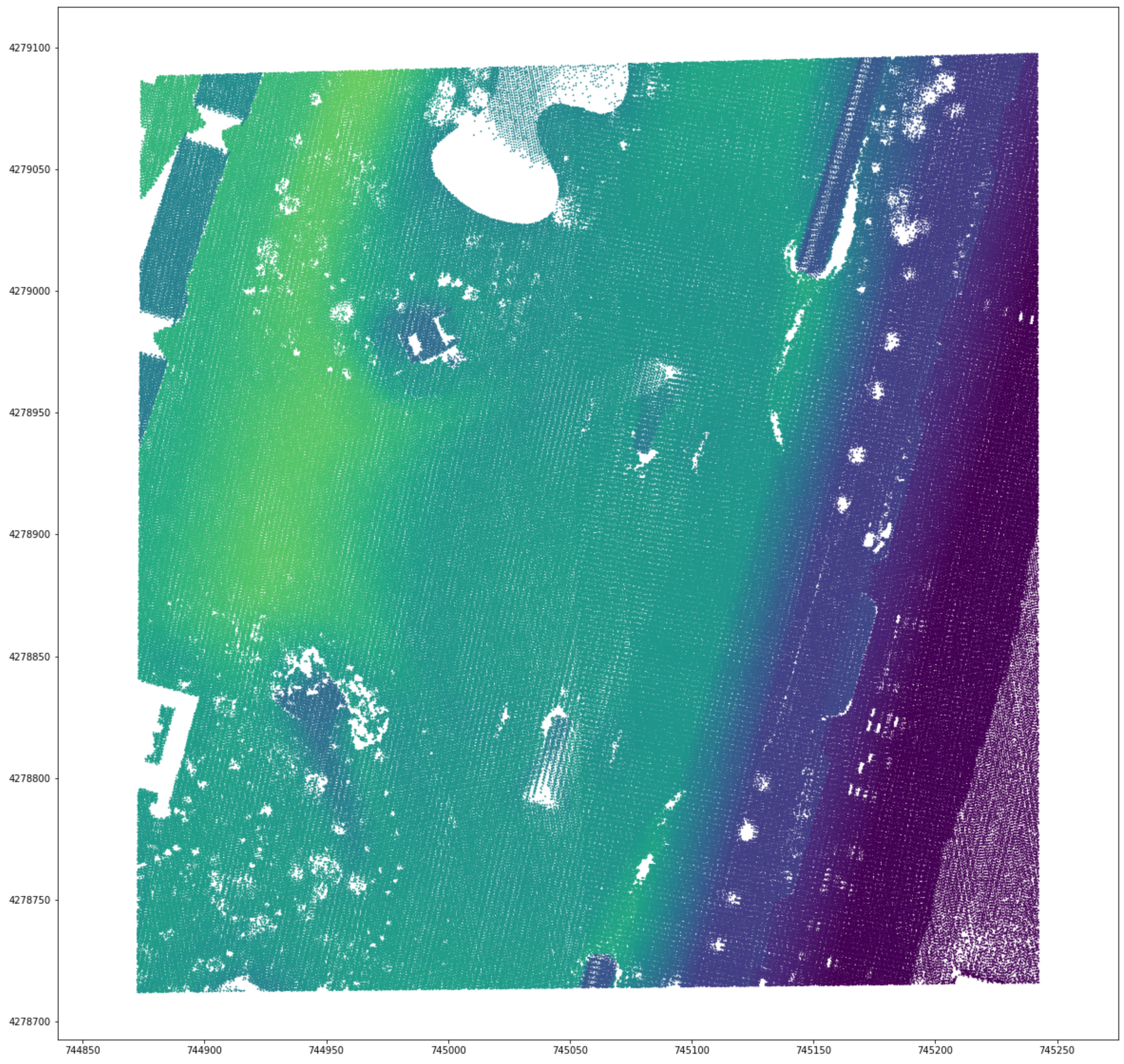
Reset classifications and segment ground returns

```
{
    "pipeline": [
        {
            "type": "filters.assign",
            "assignment": "Classification[:]=0"
        },
        {
            "type": "filters.smrf"
        }
    ]
}
```

Launch Binder


4279100
4279050
4279000
4278950
4278900
4278850
4278800
4278750
4278700
744850
744900
744950
745000
745050
745100
745150
745200
745250


4279100
4279050
4279000
4278950
4278900
4278850
4278800
4278750
4278700
744850
744900
744950
745000
745050
745100
745150
745200
745250


4279100
4279050
4279000
4278950
4278900
4278850
4278800
4278750
4278700
744850
744900
744950
745000
745050
745100
745150
745200
745250

Core Stage Development

- C++ experience required
- PDAL architecture
- Beyond the scope of this talk
- Generally used when an algorithm is matured
- Many examples in the codebase
- Core stages and shared library plugins

Why Build Plugins?

- Incompatible (possibly proprietary) license
  - PDAL/PRC (LGPL) will generate PDF with embedded point cloud

Python Development

- Python experience required
- PDAL Python capabilities and limitations
- Stages
- Extension

Python Stages

- `filters.python`
  - Embedded Python `source`
  - Path to Python `script`
- `readers.numpy`
- `writers.numpy` coming in PDAL v1.8

Flaxman, M. & Zwitch, R. "Taming Billions of LIDAR Points with GPU Database MapD"

Example #7

"Last of many" returns Python filter

```python
import numpy as np

def filter(ins,outs):
    rn = ins['ReturnNumber']
    nr = ins['NumberOfReturns']

    rets = np.logical_and(np.equal(rn, nr),
        np.greater(nr, 1))

    outs['Mask'] = rets
    return True
```

Filter embedded as script

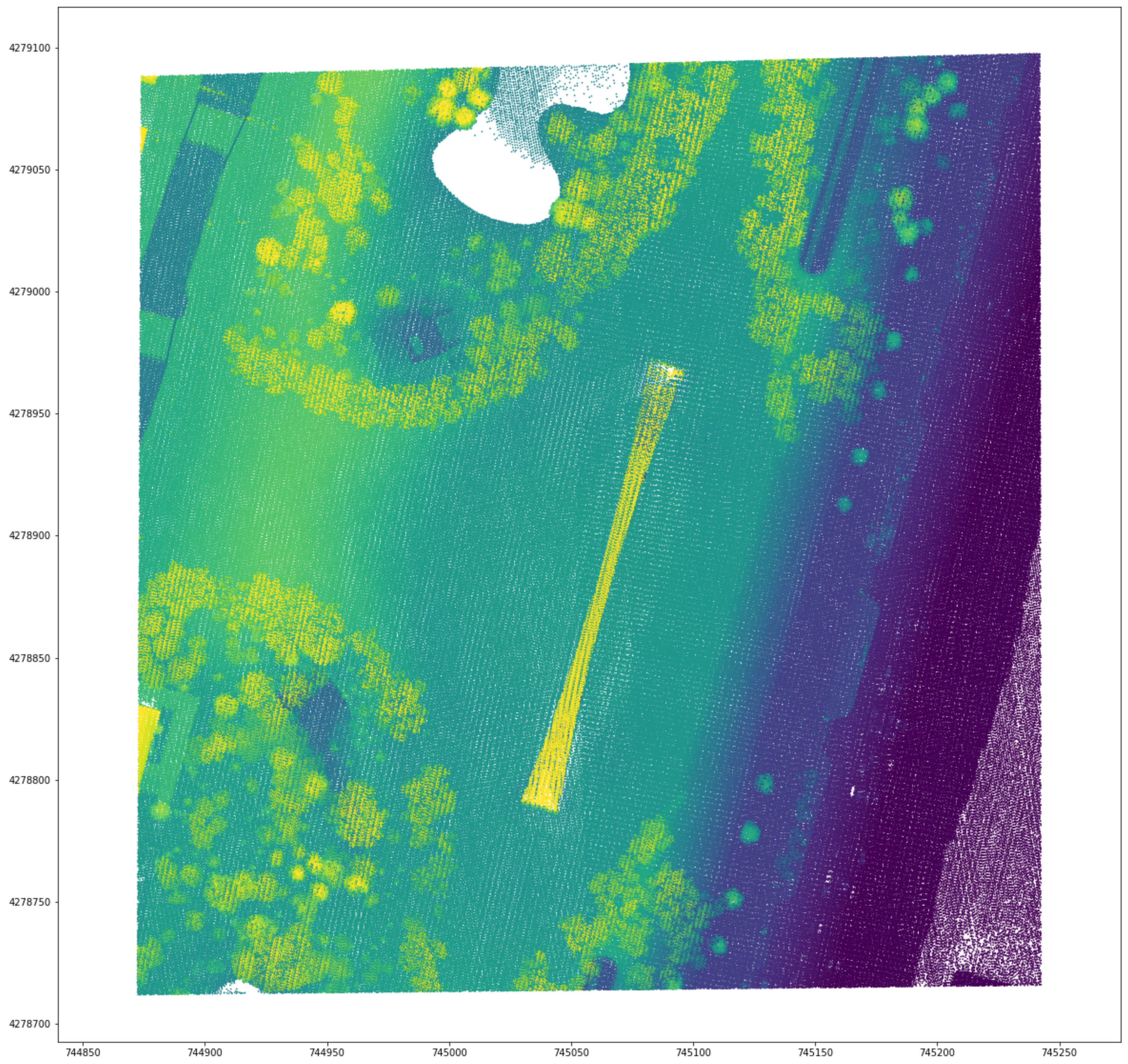
```
{
    "pipeline":[
        {
            "type": "filters.python",
            "function": "filter",
            "script": "last-of-many.py"
        }
    ]
}
```

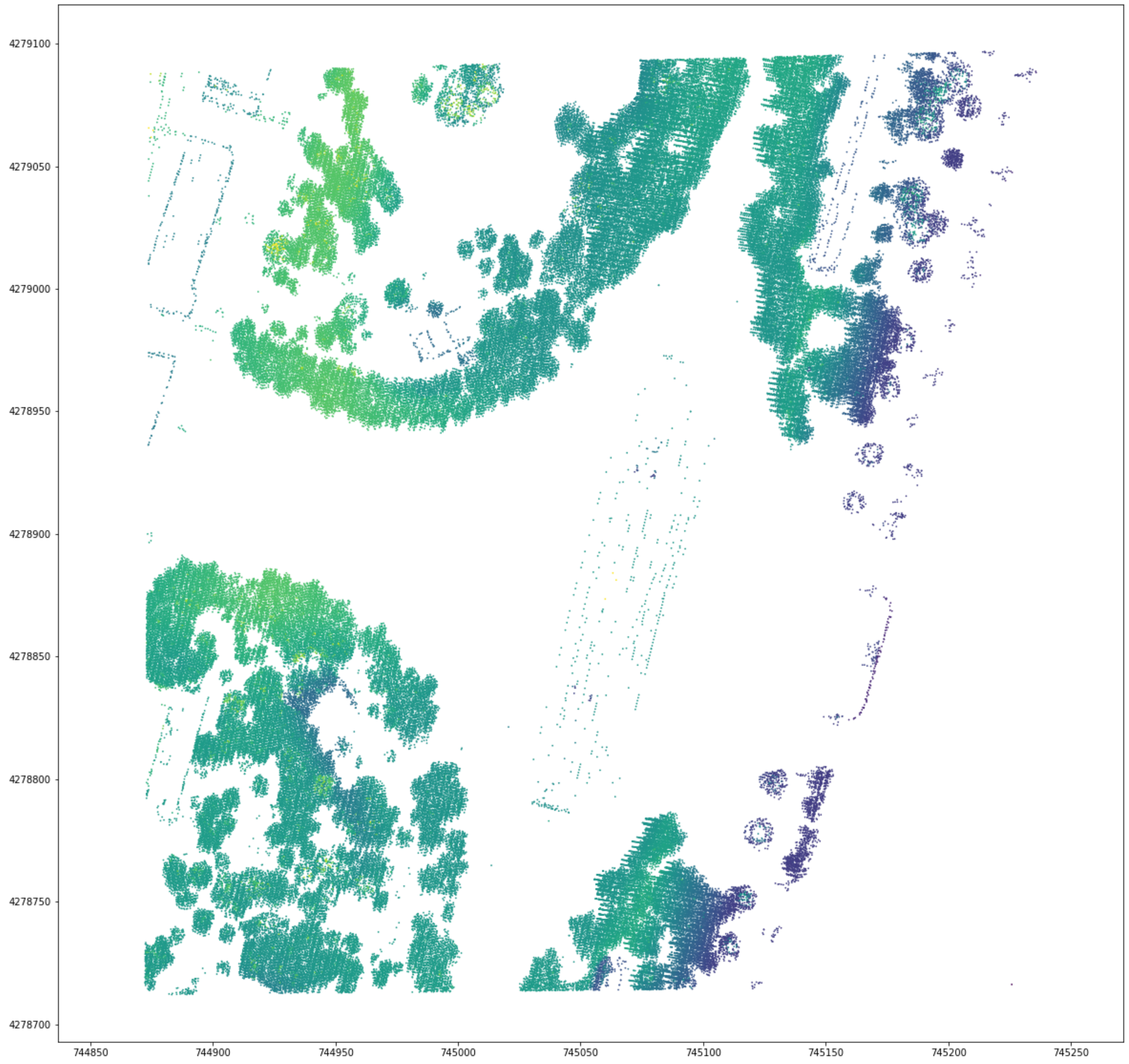
Launch Binder

Filter embedded as source

```
{
    "pipeline": [
        {
            "type": "filters.python",
            "function": "filter",
            "source": "import numpy as np

def filter(ins,outs):
    rn = ins['ReturnNumber']
    nr = ins['NumberOfReturns']

    rets = np.logical_and(np.equal(rn, nr),
        np.greater(nr, 1))

    outs['Mask'] = rets
    return True"
        }
    ]
}
```

Launch Binder


4279100
4279050
4279000
4278950
4278900
4278850
4278800
4278750
4278700
744850
744900
744950
745000
745050
745100
745150
745200
745250


4279100
4279050
4279000
4278950
4278900
4278850
4278800
4278750
4278700
744850
744900
744950
745000
745050
745100
745150
745200
745250

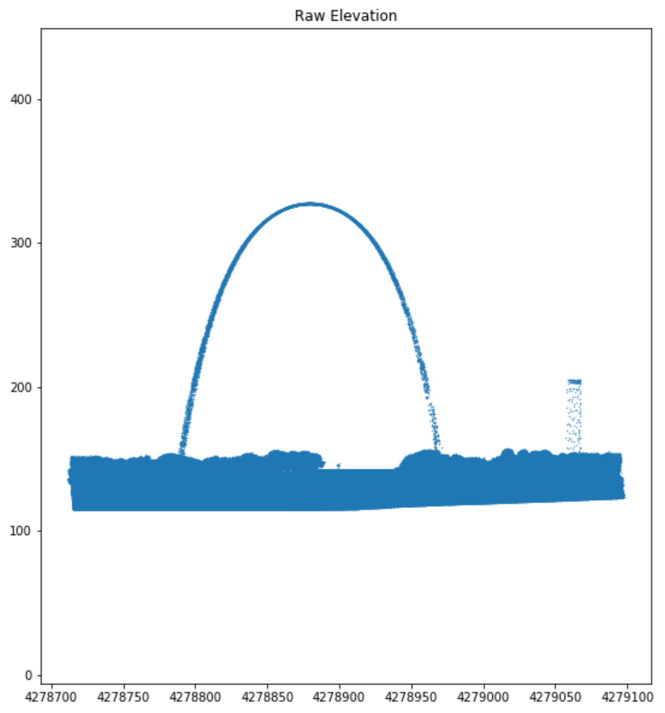
Python Extension

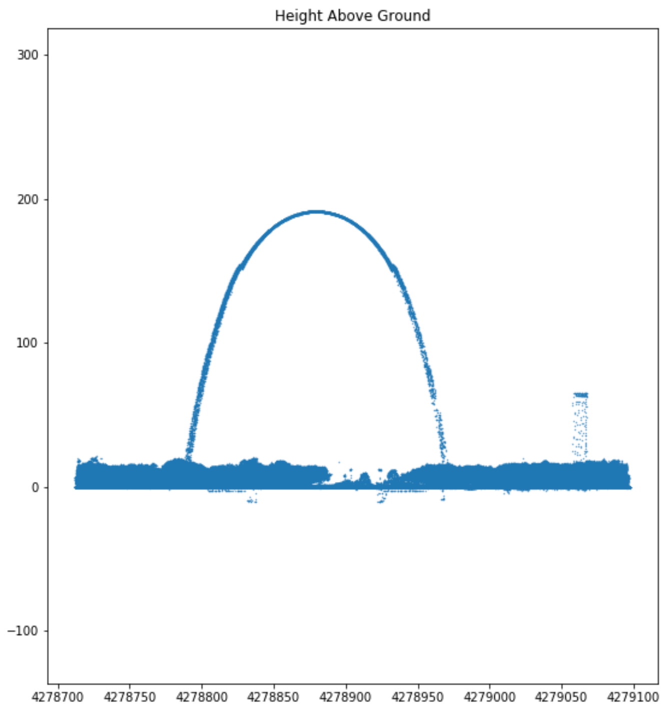
- Validate and execute PDAL pipelines
- End result is available as Numpy array

Example #8

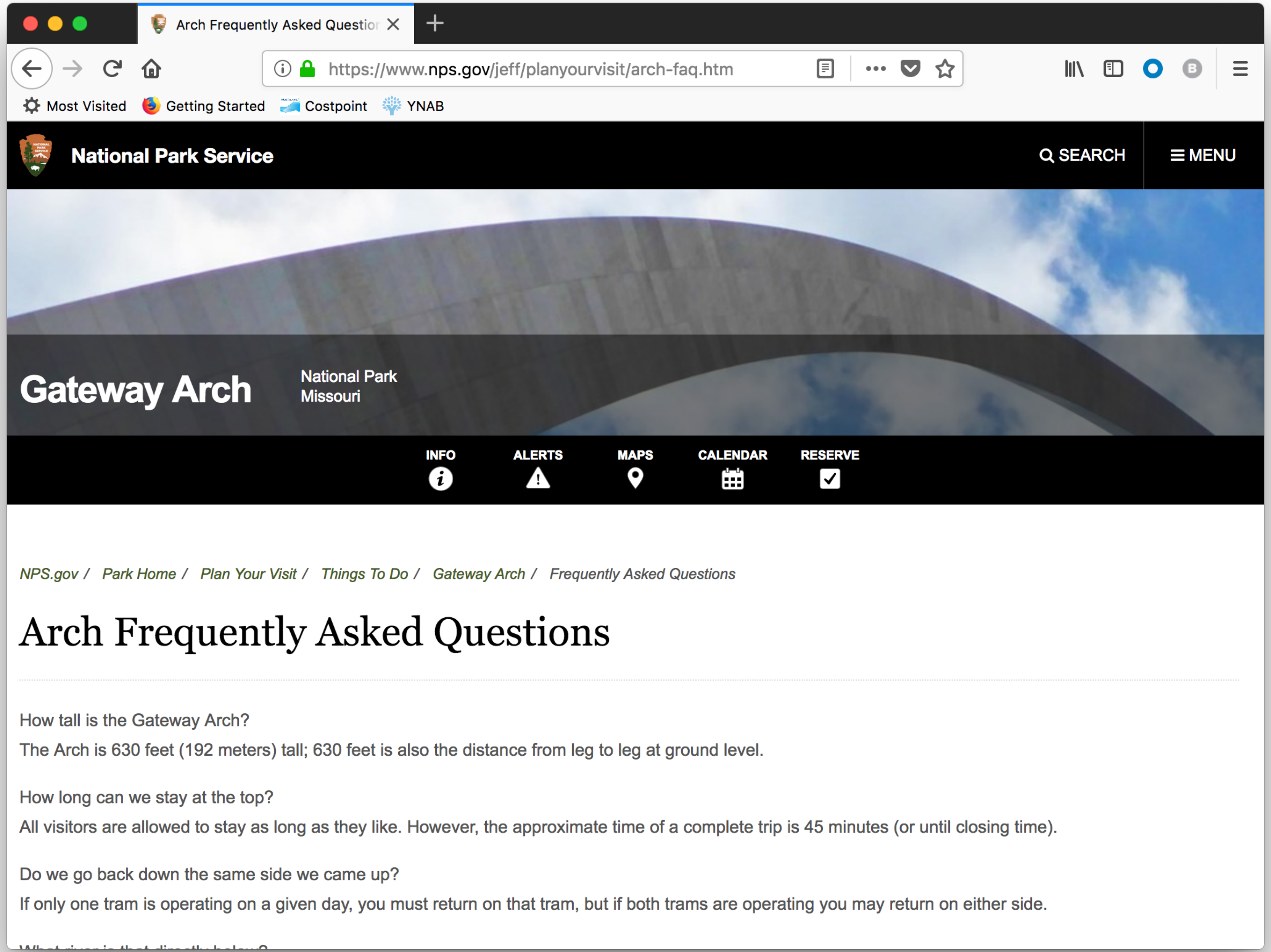
```
import json
import pdal

pipeline = {
    "pipeline": [
        "arch.las",
        {
            "type": "filters.hag"
        }
    ]
}

p = pdal.Pipeline(json.dumps(pipeline))
p.validate()
p.execute()
```

Launch Binder





`p.arrays[0]['HeightAboveGround'].max()` shows that apex is 191.61 meters above ground.



NPS.gov / Park Home / Plan Your Visit / Things To Do / Gateway Arch / Frequently Asked Questions

Arch Frequently Asked Questions

How tall is the Gateway Arch?
The Arch is 630 feet (192 meters) tall; 630 feet is also the distance from leg to leg at ground level.

How long can we stay at the top?
All visitors are allowed to stay as long as they like. However, the approximate time of a complete trip is 45 minutes (or until closing time).

Do we go back down the same side we came up?
If only one tram is operating on a given day, you must return on that tram, but if both trams are operating you may return on either side.

Issues

- PDAL project
  - PDAL/PDAL
- Conda packaging issues
  - conda-forge/pdal-feedstock
  - conda-forge/python-pdal-feedstock

How do I …?

https://pdal.io/community.html

- Mailing list
- Gitter, Keybase, IRC links
- StackOverflow

Thank You!

https://mybinder.org/v2/gh/chambbj/foss4gna-2018-binder/master

@chambbj