EXPERIENCES WITH PCLPY ON PUHTI

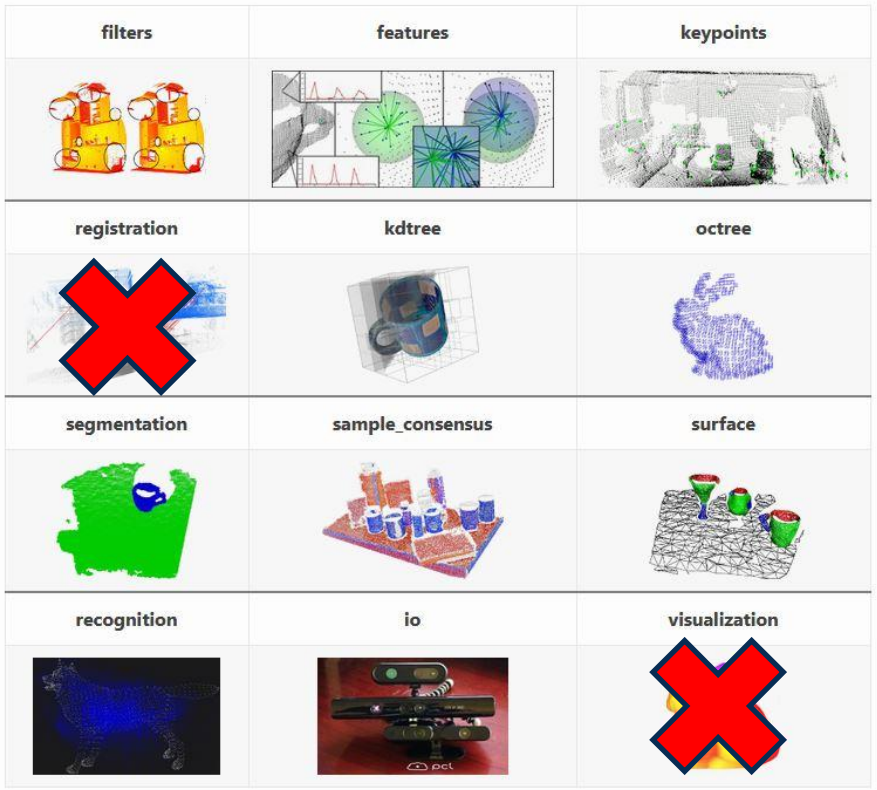
Jiri Pyörälä

University of Helsinki & Finnish Geospatial Institute

PCLPY

- https://github.com/davidcaron/pclpy
- Python bindings for the Point Cloud Library (PCL)
  - https://pointclouds.org
- Uses C++ directly for simple implementation of the original code

PCL VS. PCLPY



Source: https://pointclouds.org

- A large percentage of PCL is covered but not all. Missing libraries:
  - ml
  - people
  - outofcore
  - registration
  - visualization

PCL VS. PCLPY

pclpy

```
import pclpy
from pclpy import pcl

point_cloud = pclpy.read("street.las", "PointXYZRGBA")
mls = pcl.surface.MovingLeastSquaresOMP.PointXYZRGBA_PointNormal()
tree = pcl.search.KdTree.PointXYZRGBA()
mls.setSearchRadius(0.05)
mls.setPolynomialFit(False)
mls.setNumberOfThreads(12)
mls.setInputCloud(point_cloud)
mls.setSearchMethod(tree)
mls.setComputeNormals(True)
output = pcl.PointCloud.PointNormal()
mls.process(output)
```

Source: https://github.com/davidcaron/pclpy

Original pcl

```
pcl::PointCloud<pcl::PointXYZ>::Ptr point_cloud (new pcl::PointCloud<pcl::PointXYZ> ());
pcl::io::loadPCDFile ("bunny.pcd", *point_cloud);
pcl::MovingLeastSquaresOMP<pcl::PointXYZ, pcl::PointNormal> mls;
pcl::search::KdTree<pcl::PointXYZ>::Ptr tree (new pcl::search::KdTree<pcl::PointXYZ>);
mls.setSearchRadius (0.05);
mls.setPolynomialFit (false);
mls.setNumberOfThreads (12);
mls.setInputCloud (point_cloud);
mls.setSearchMethod (tree);
mls.setComputeNormals (true);
pcl::PointCloud<pcl::PointNormal> output;
mls.process (output);
```

Any pcl-documentation found (and there's plenty) is easy to adapt to pclpy

MY EXPERIENCE WITH PCLPY

- I started using PCLpy because of being more familiar with python than C++
  - Need for basic point cloud operations with compatibility to earlier code
  - laspy integration for reading/writing las files
  - Convertible to numpy arrays (e.g., cloud.x or cloud.xyz)
    - Apply methods from scipy, matplotlib, geopandas etc

- Usage: processing of terrestrial point clouds of forests/trees
  - Downsampling and filtering (e.g., kd- or octrees, statistical outlier removal)
  - Classifications and segmentation (e.g., ground points, stem points)
  - Surface/object reconstructions (e.g., cylinder fitting, cloth simulation)

STEM EXTRACTION

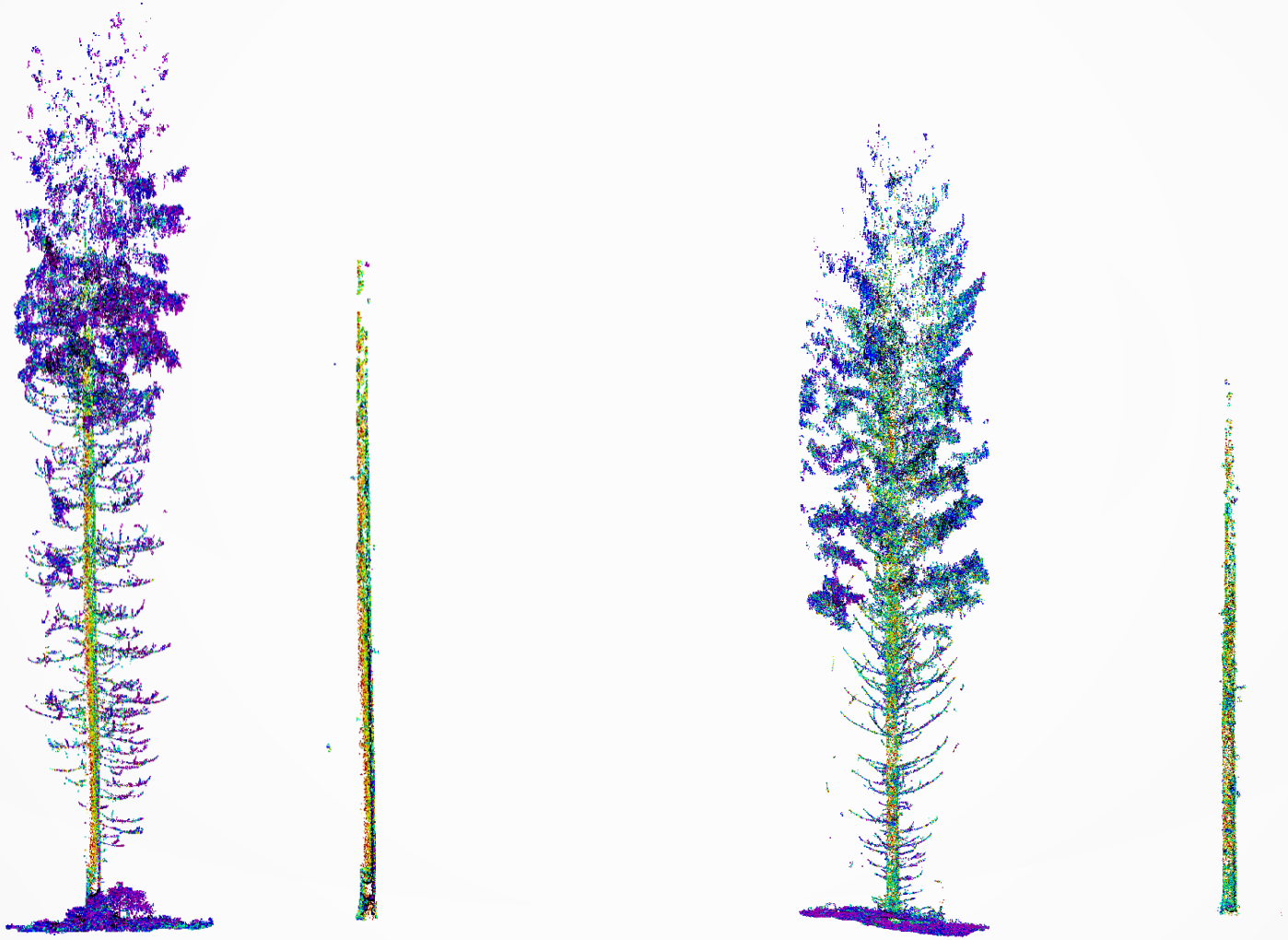
Example of stem extraction from point cloud of single tree:

- Estimate normals
- Segment point cloud based on normal
- Fit cylinders to segments using Random Sample Consensus (RANSAC)
- Save stem points to new point cloud

```
tree = pclpy.read_las(file.las, point_type="PointXYZI") #read lasfile of single tree
normals = pc.compute_normals(radius=0.035) #compute normal for tree
seg = pcl.segmentation.SACSegmentationFromNormals.PointXYZI_Normal()
#initiate segmentation from normals
seg.setInputCloud(tree) #set the tree point cloud as input for segmentation
seg.setInputNormals(normals) #set the normals as input for segmentation
seg.setModelType(pcl.sample_consensus.SACMODEL_CYLINDER) #set
segmentation model type to cylinder
seg.setMethodType(pcl.sample_consensus.SAC_RANSAC) #set segmentation
iteration type to RANSAC
seg.setMaxIterations(1000)
seg.setDistanceThreshold(0.025)
seg.setRadiusLimits(min_radius=0.025, max_radius=0.35)
inliers = pcl.PointIndices() #empty vector for inlier (stem point) indices
coefs = pcl.ModelCoefficients() #empty vector for segmentation coefficients
seg.segment(inliers, coefs) #run segmentation

extract=pcl.filters.ExtractIndices.PointXYZI() #empty class vector to extract inliers
extract.setInputCloud(pc)
extract.setIndices(inliers)
extract.setNegative(False) #FALSE=filter out inliers
stem = pcl.PointCloud.PointXYZI() #empty point cloud for stem points
extract.filter(stem) #save stem points to the empty point cloud
```

STEM EXTRACTION EXAMPLES



PCLPY ON PUHTI

- Version 0.12.0 available in module ”PCL” for Python 3.8
- Parallel processing/ array jobs
- ~500 trees (each 200-500 MB) took ~4 hours to process:
  - Stem & branching models

```
#!/bin/bash -l
#SBATCH --account=project_xxxxxxx
#SBATCH --job-name xxx
#SBATCH –time xx:xx
#SBATCH --ntasks 1
#SBATCH --mem-per-cpu=2G
#SBATCH --array=1-99
#SBATCH --partition small

module load pcl

treeList="/sampleTrees.txt"

### read a filepath to the .las file given in the list of files
inputfilepath=$(sed -n "$SLURM_ARRAY_TASK_ID"p $treeList)

python /projappl/project_xxxxxxx/Scripts/branchModeling/run.py $inputfilepath
```