

# LiDAR data processing (November 2023)

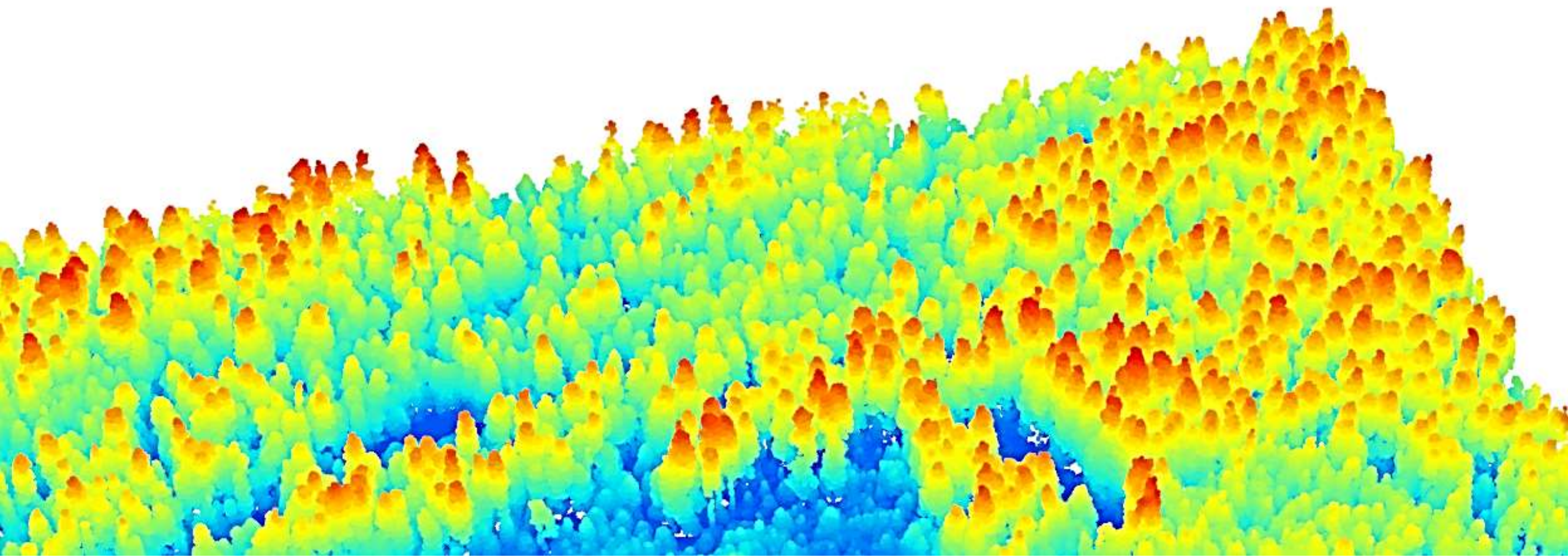


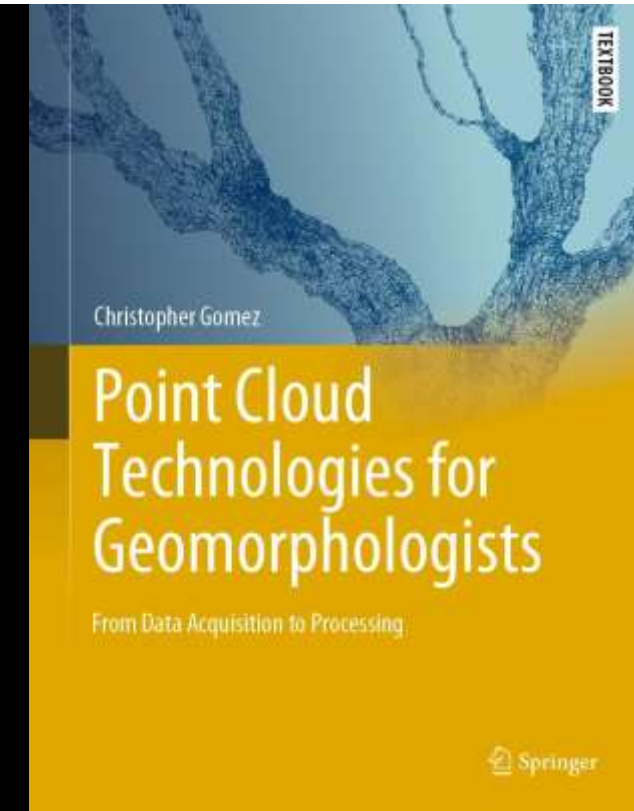
Professor Christopher Gomez

*christophergomez@bear.kobe-u.ac.jp*

SABO Laboratory @ Kobe University (Japan)

PSBA Research Centre & Geography @ UGM University (Indonesia)





# 砂防2 点群データ:入門

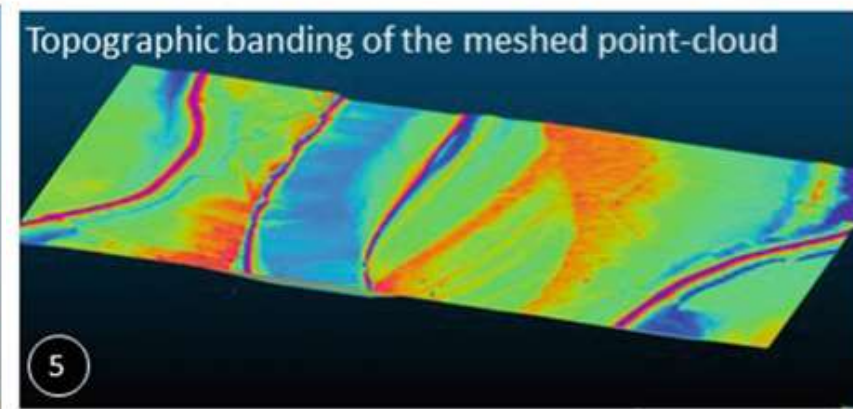
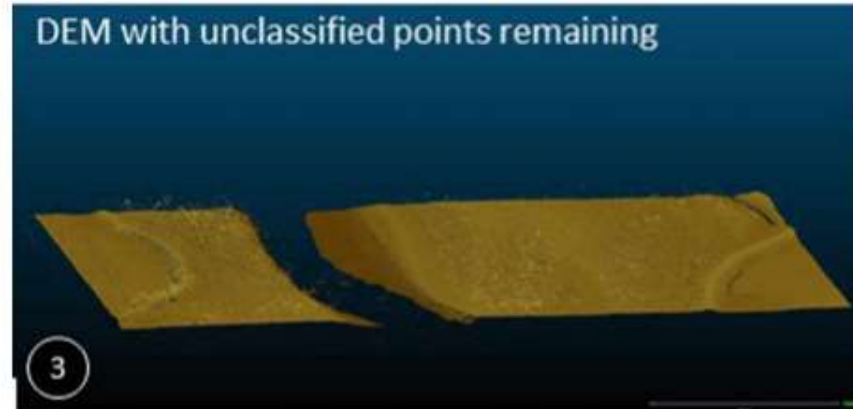
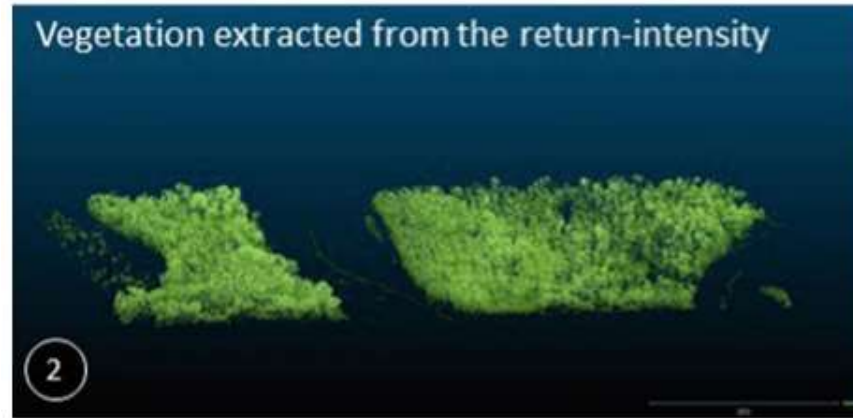
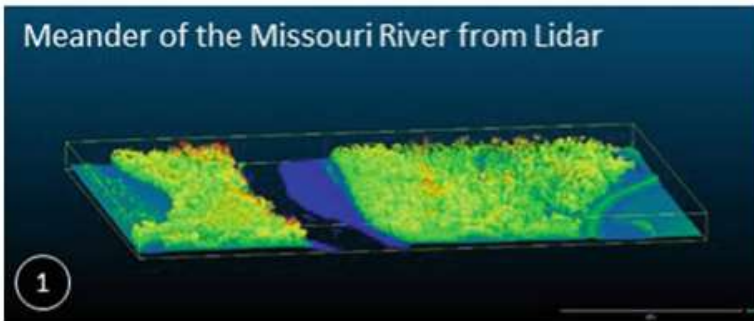
## LiDAR data processing: from points to the world

*Christopher Gomez*



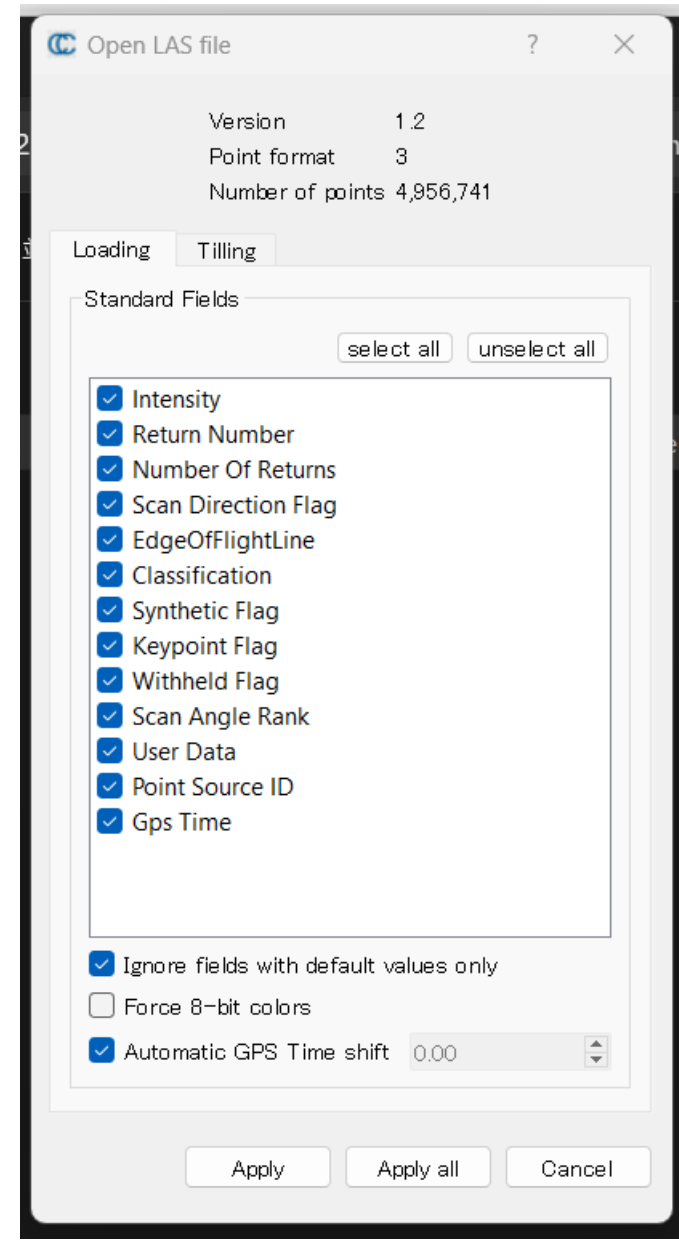
# 1. Extracting elements with CloudCompare

# From different data decompose your pointcloud

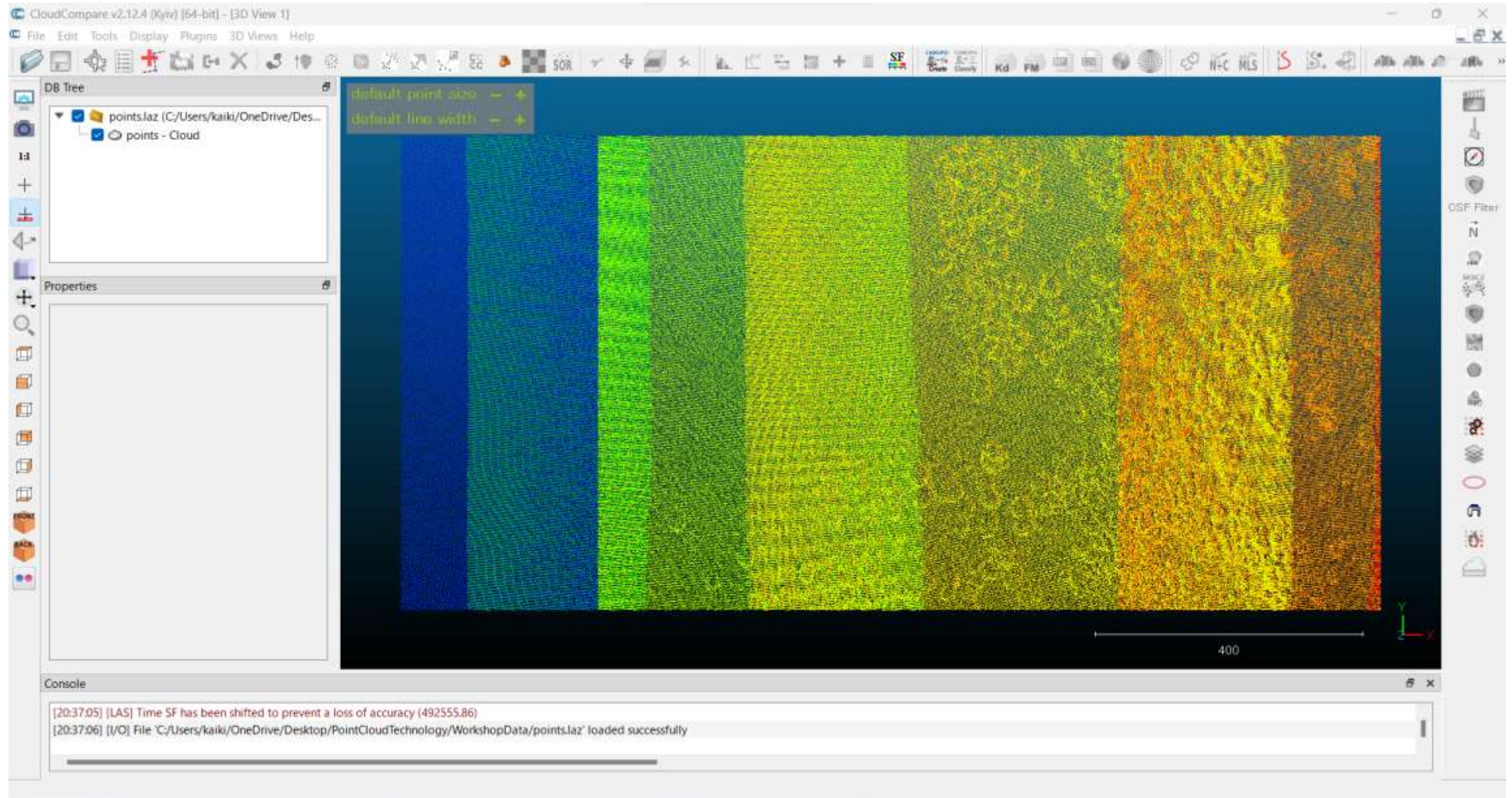


... to do so you use  
the characteristics  
of the pointcloud.

$$P_n = \sum_{i=1}^n [X_i, Y_i, Z_i, Att_i]$$

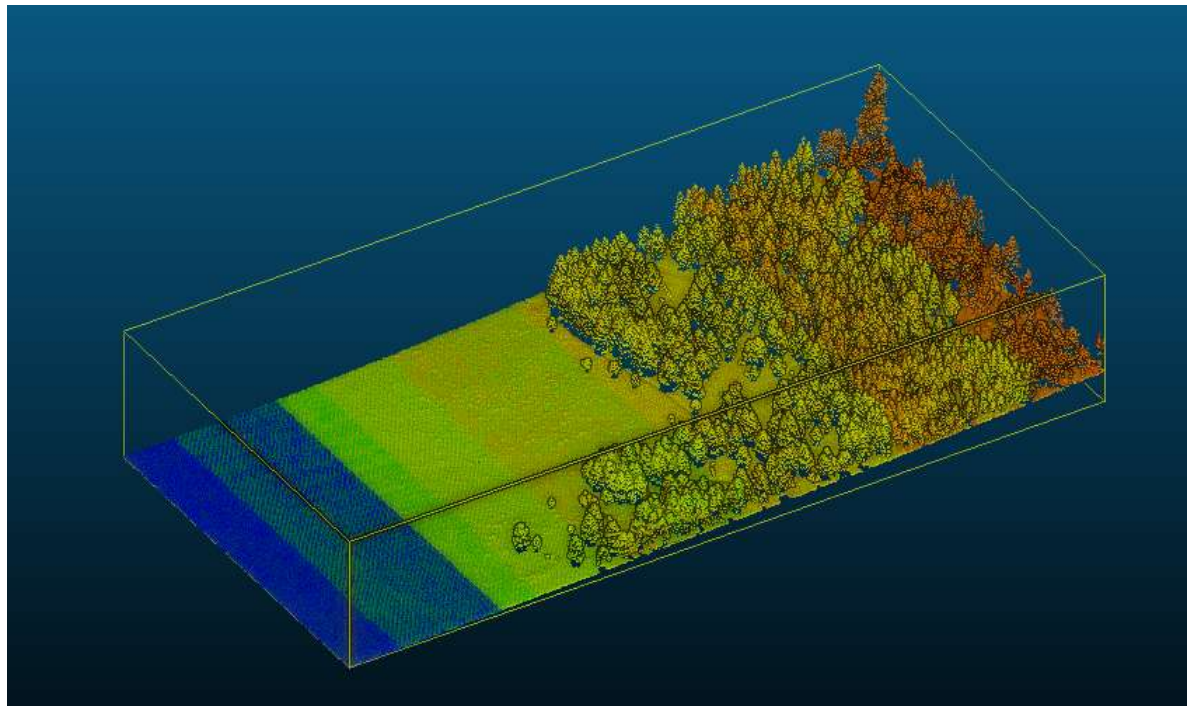
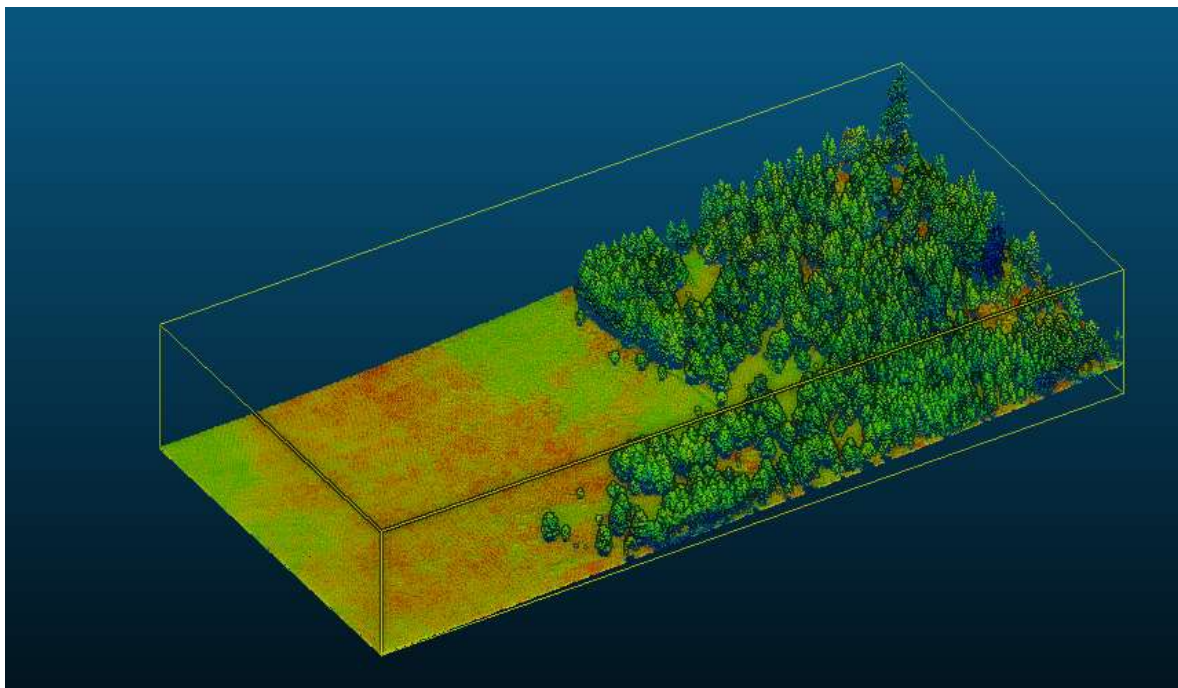


1. Extracting elements



1. Extracting elements

# 1. EDL shader (real time rendering)



Properties

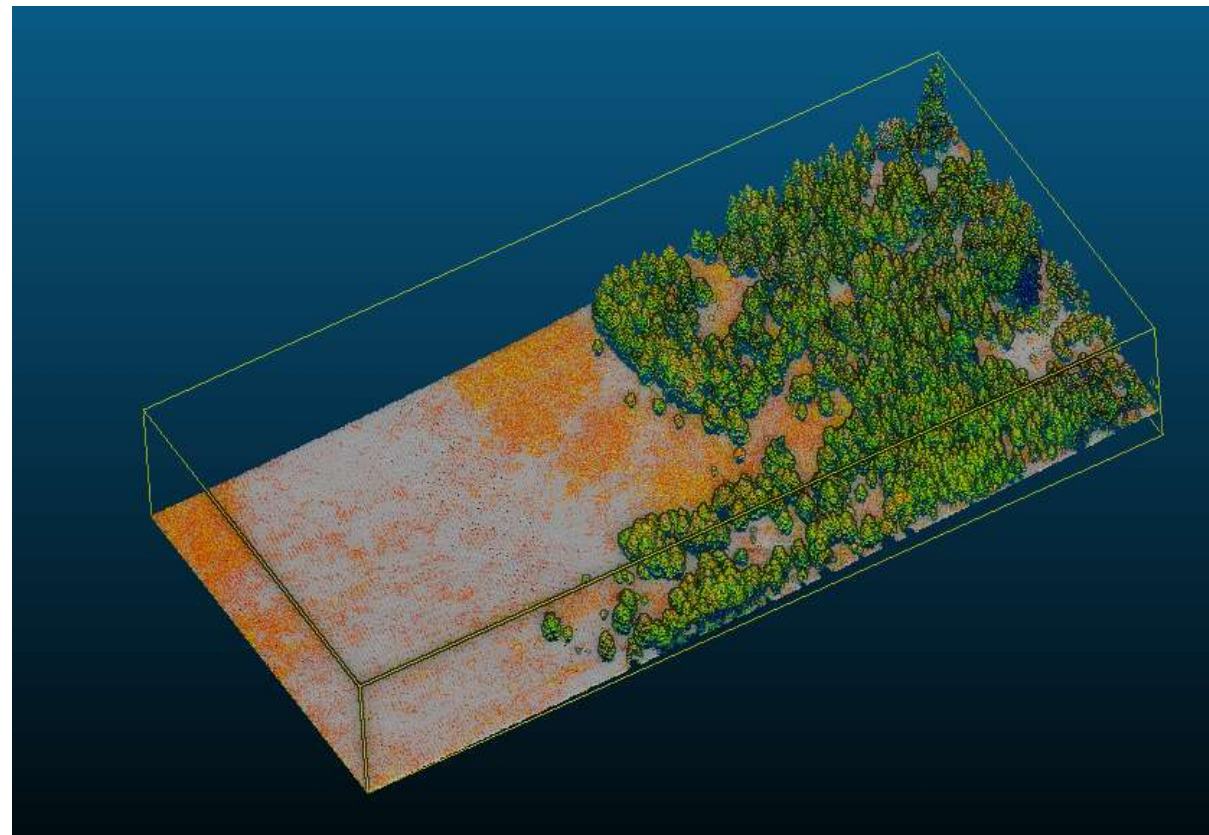
Property	State/Value
CC Object	
Name	points - Cloud
Visible	<input checked="" type="checkbox"/>
Colors	Scalar field
Show name (i...	<input type="checkbox"/>
Box dimensions	X: 1461.46 Y: 706.73 Z: 253.33
Shifted box ce...	X: 436.88 Y: 398.345 Z: 7883.67
Global box ce...	X: 6462436.880005 Y: 2509698.345001 Z: 7883.665039
Info	Object ID: 262 - Children: 0
Current Display	3D View 1
Cloud	
Points	882,106
Global shift	(-6462000.00;-2509300.00;0.00)
Global scale	1.000000
Point size	Default
Scalar Fields	
Count	10

# 2. Display the intensity return of the Lidar

# 1. Extracting elements



# Intensity-based classification



DB Tree

- points.laz (C:/Users/kaiki/OneDrive/Des...)
  - points - Cloud

Properties

Property	State/Value	
Active	Intensity	
Color Scale		
Current	Blue>Green>Yellow>Re	
Steps	256	
Visible	<input type="checkbox"/>	
SF display params		
Display ranges		
0.00000000	displayed 154.24825175	
0.00000000	saturation 159.59790210	
Transformation history		
Matrix	Axis/Angle	Export
1.000000 0.000000 0.000000 0.000000		
0.000000 1.000000 0.000000 0.000000		
0.000000 0.000000 1.000000 0.000000		
0.000000 0.000000 0.000000 1.000000		

Console



Filter by value

Range 0.00000000 - 154.24824524

Export Split Cancel

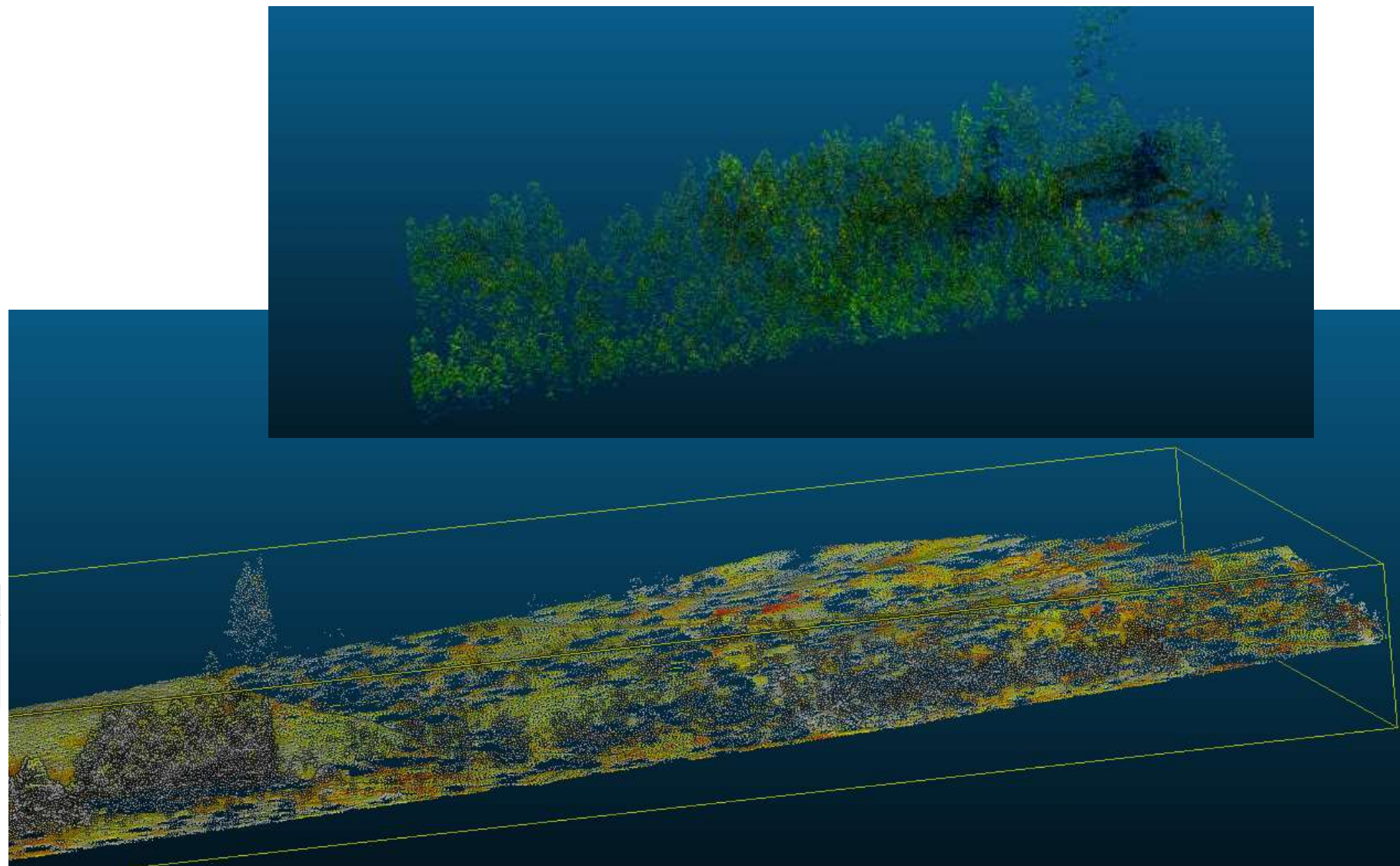
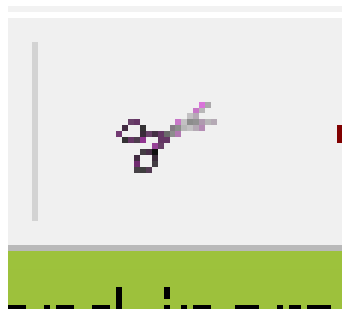
DB Tree

- points.laz (C:/Users/kaiki/OneDrive/Des...)
  - points - Cloud
  - points - Cloud.extract
  - points - Cloud.extract.outside

1. Extracting elements

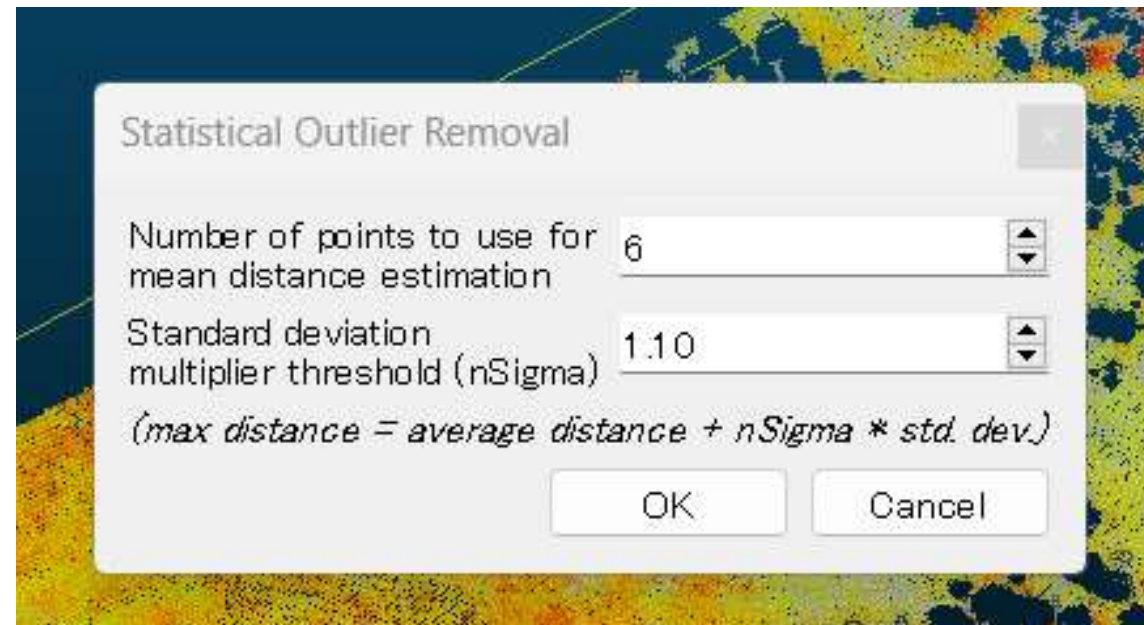


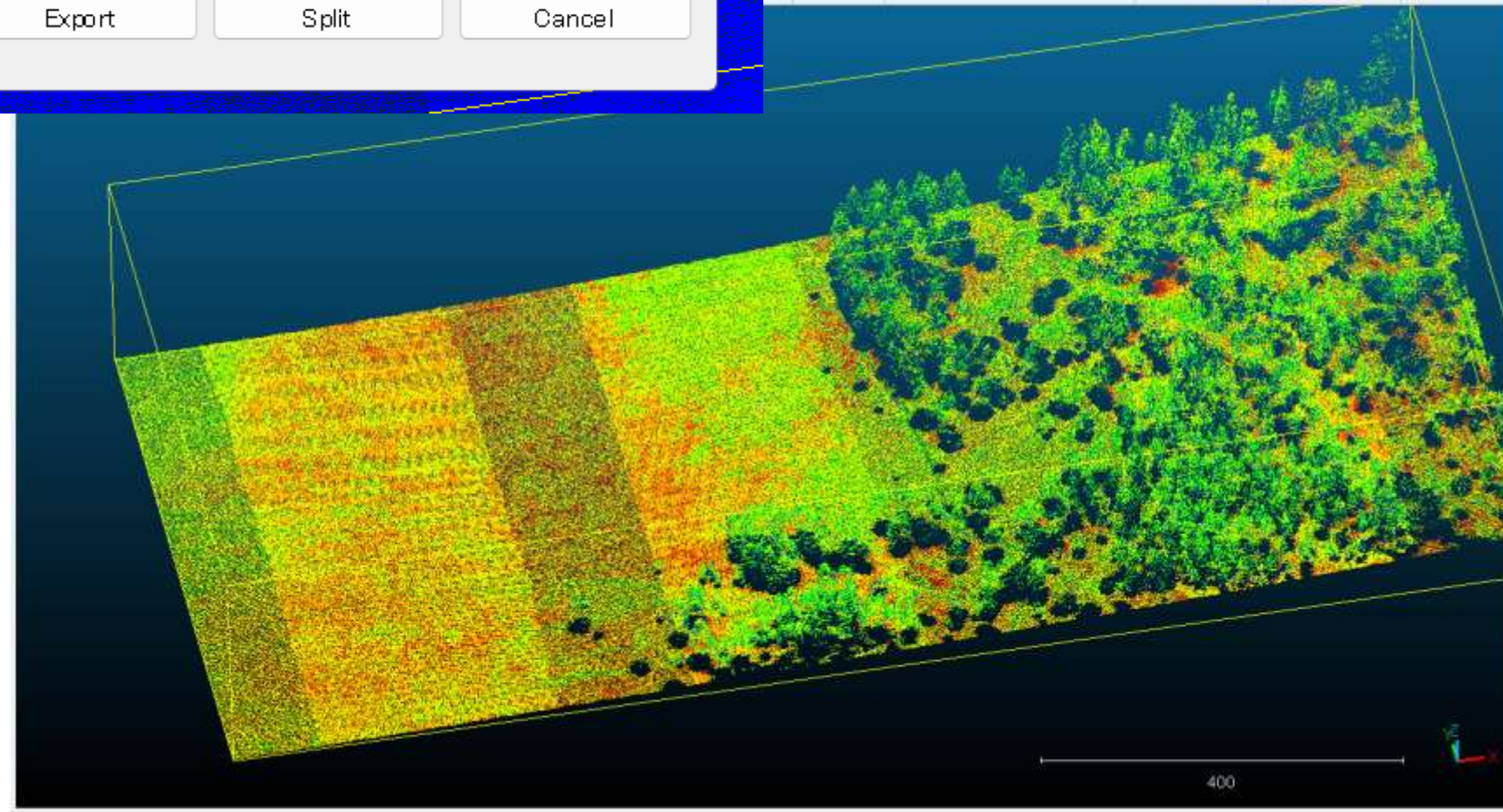
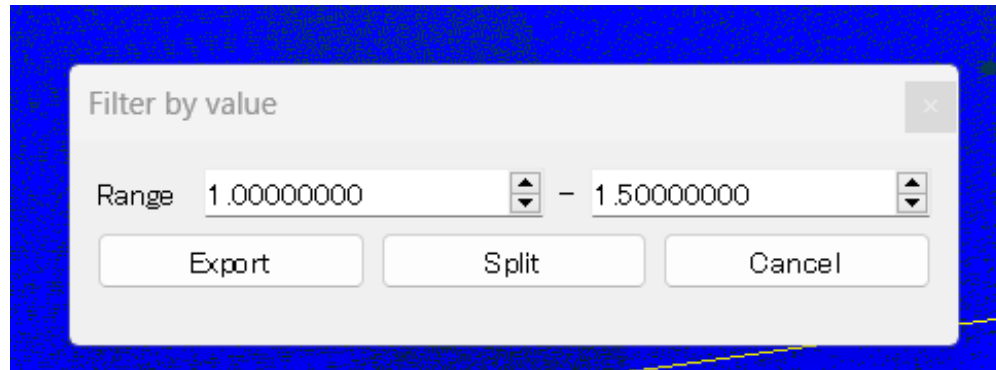
Or cut “manually”



1. Extracting elements

... when you are happy with your classification,  
then you can eliminate the outliers:



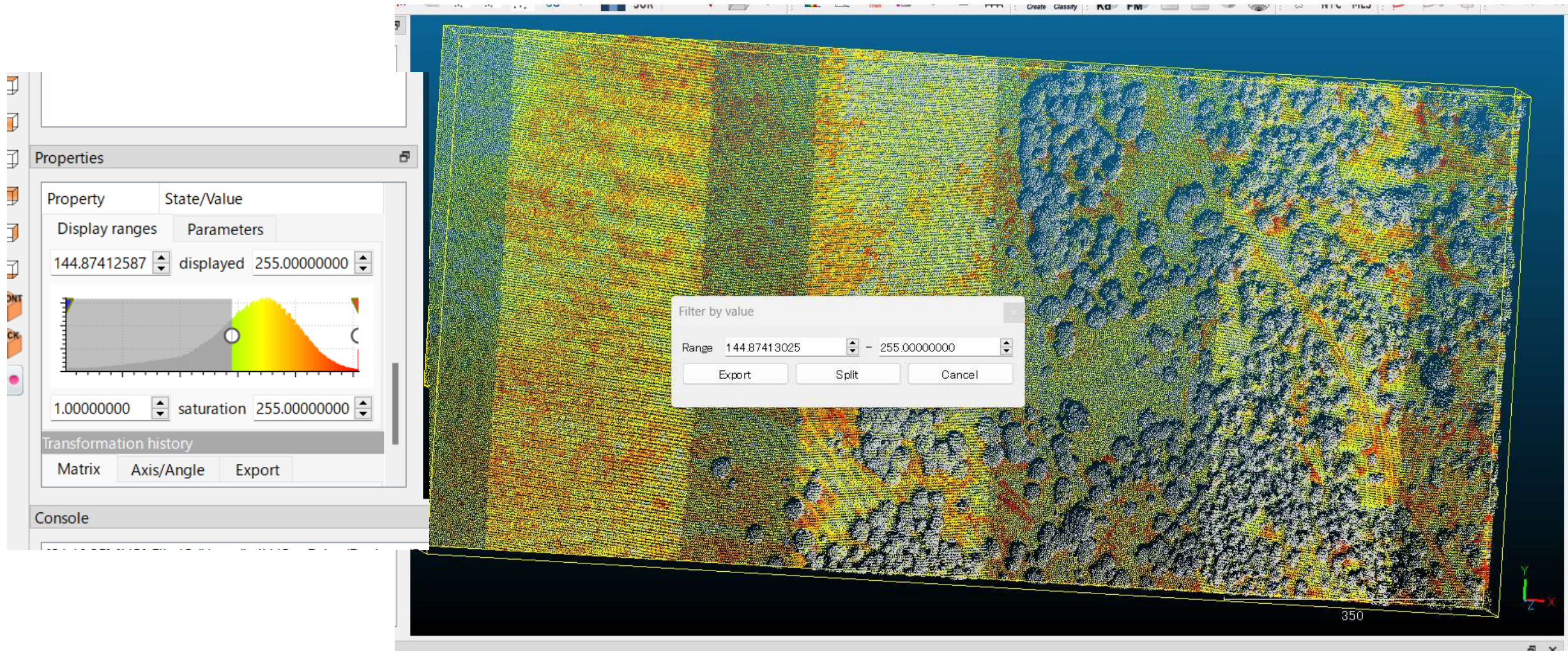


**Other option:**

**Classification  
Based on the  
Number of returns**

1. Extracting elements

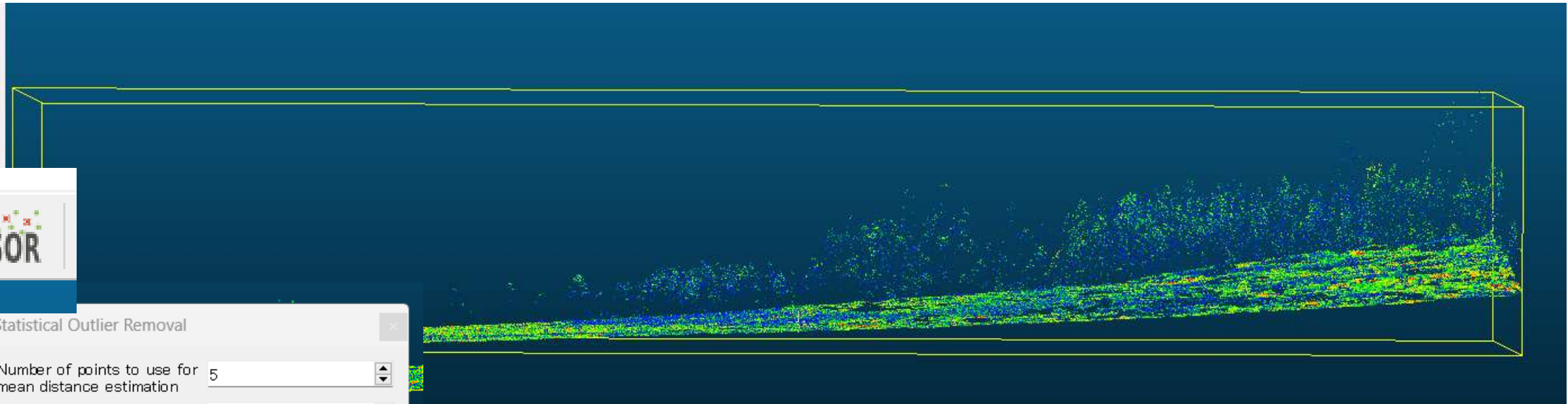
Eliminate the first return on top of the trees:



1. Extracting elements



But it is not perfect yet, we still have a lot of remaining noise.



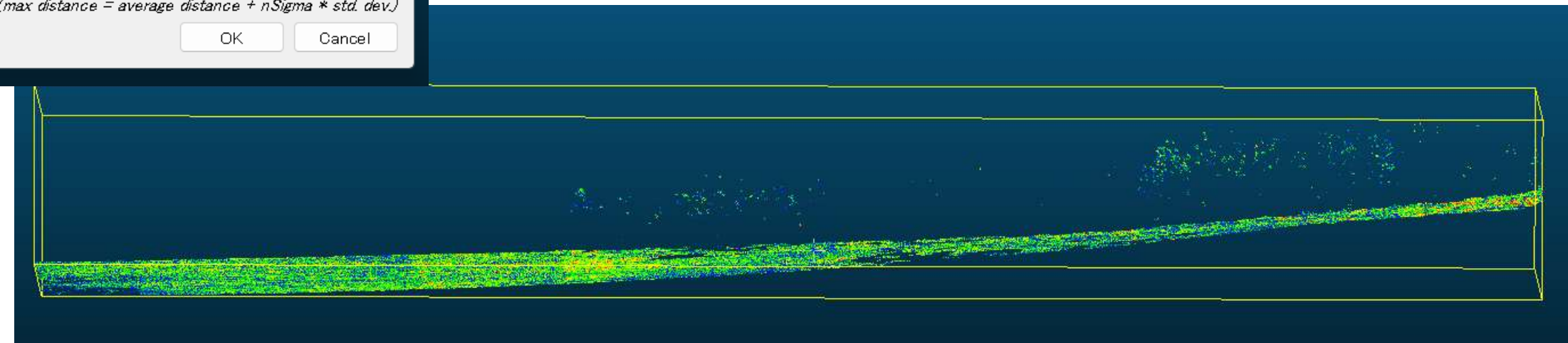
Statistical Outlier Removal

Number of points to use for mean distance estimation: 5

Standard deviation multiplier threshold (nSigma): 0.70

*(max distance = average distance + nSigma \* std. dev.)*

OK Cancel



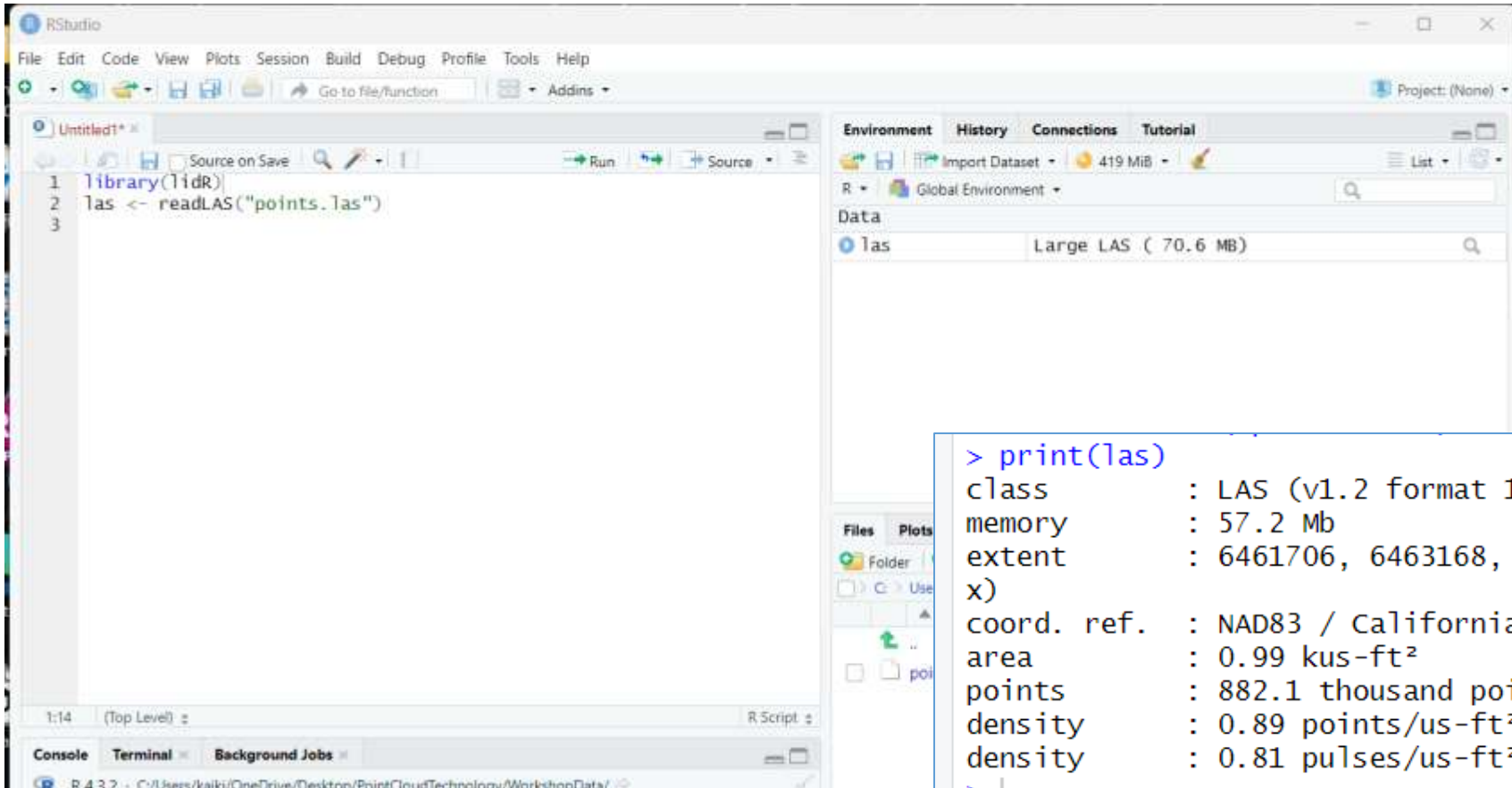
+ some Manual cleaning

1. Extracting elements

## 2. Extracting elements with LidR



# In R-Studio:



```
library(lidR)
las <- readLAS("points.las")
```

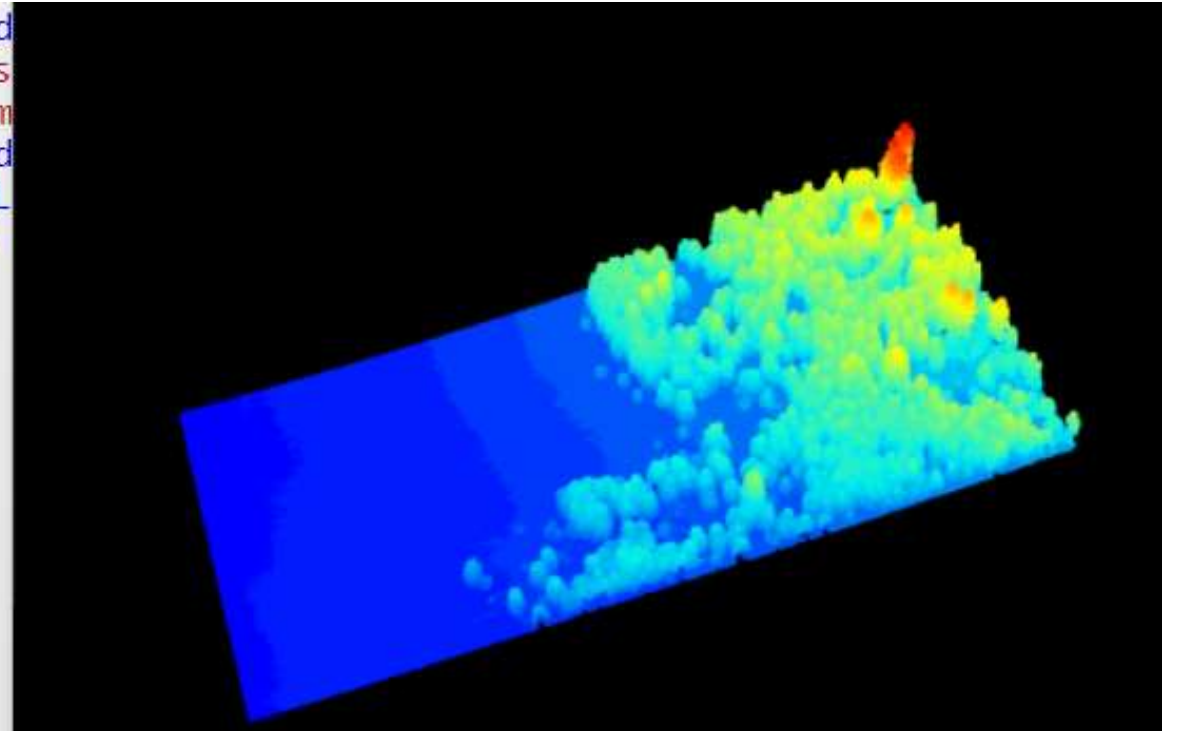
2. Extracting elements with LidR

## ... you can do by coding, what we just did with the GUI:

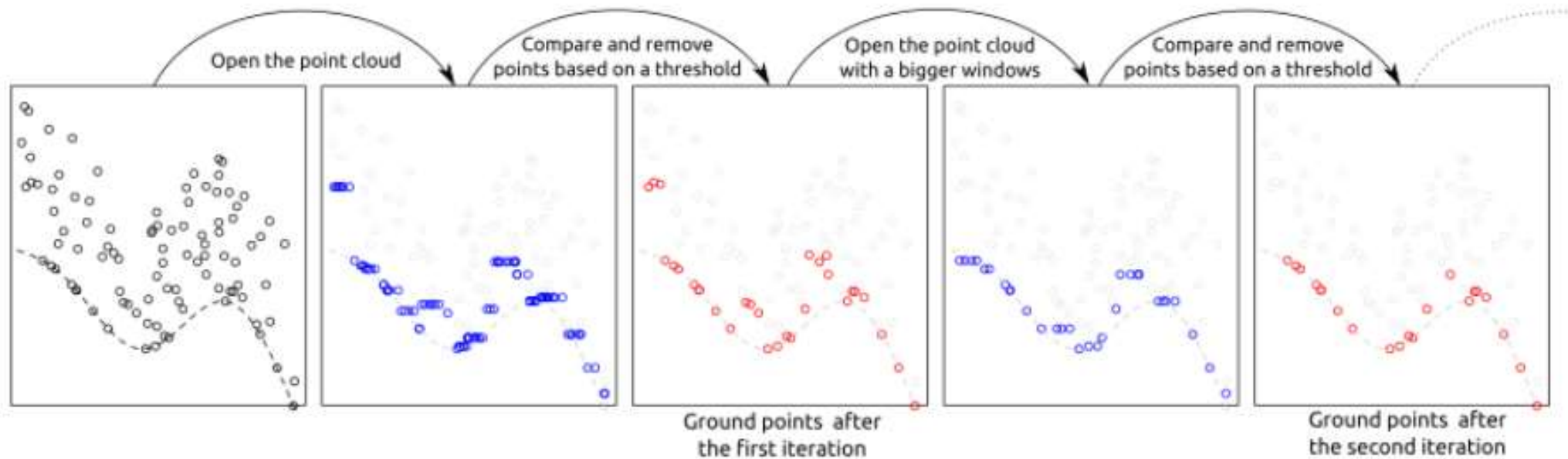
```
las <- readLAS("file.las", filter = "-keep_first") # Read only first returns
```

```
> plot(las)
```

```
> library(lidR)
lidR 4.0.4 us
n <github.com
> library(lidR)
> las <- readLAS("file.las", filter = "-keep_first")
> print(las)
class
memory
extent
x)
coord. ref.
area
points
density
density
5 > plot(las)
> |
```







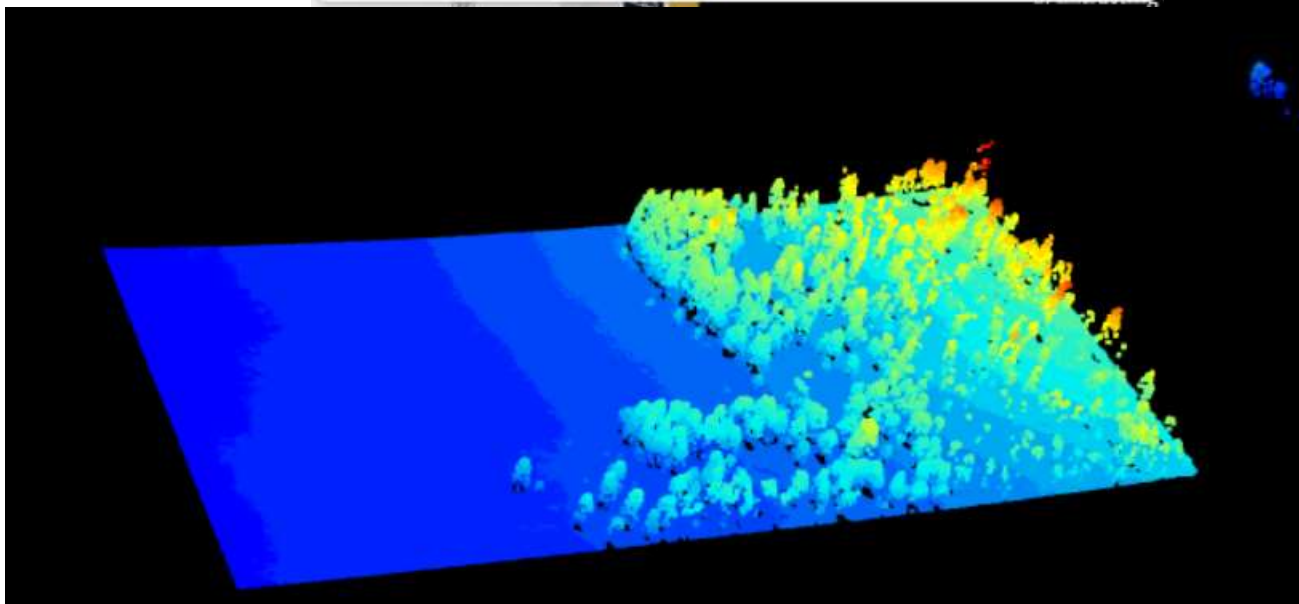
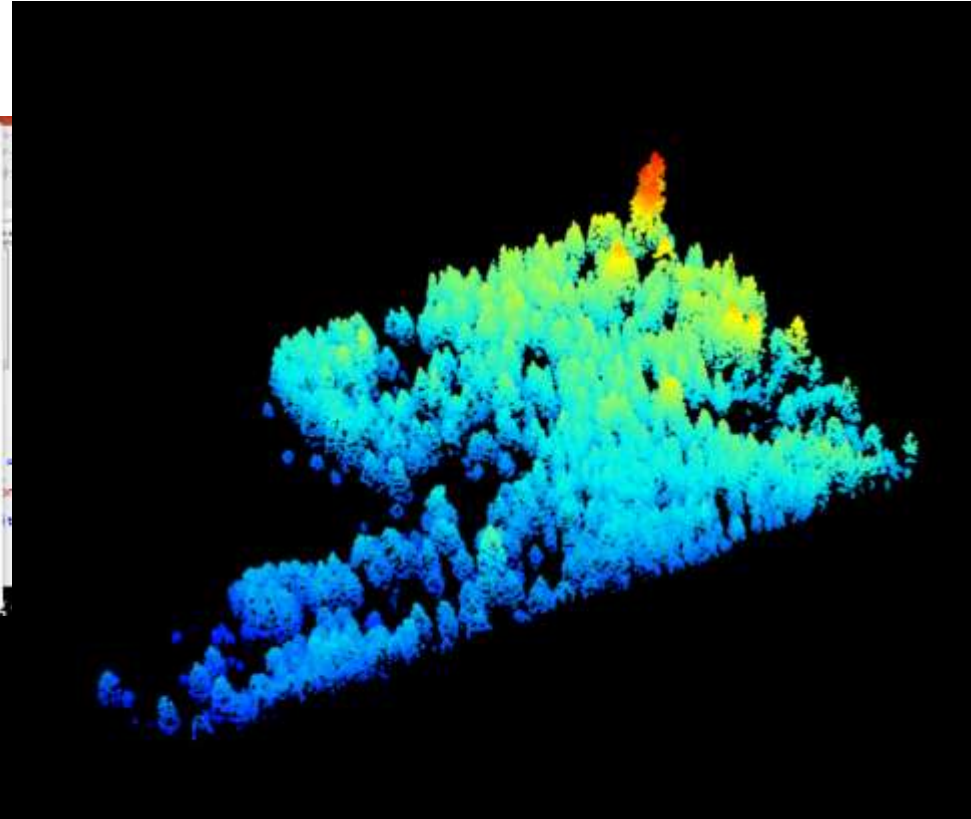
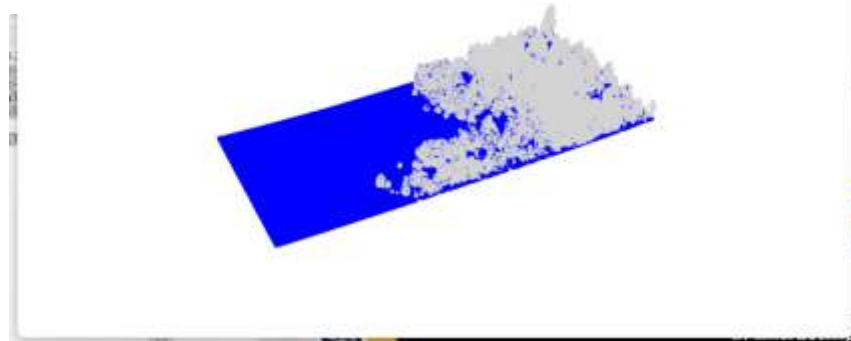
```
las <- readLAS("points.las")
```

```
las <- classify_ground(las, algorithm = pmf(ws = 5, th = 3))
```

Original dataset already contains 277737 ground points. These points were reclassified as 'unclassified' before performing a new ground classification.

```
plot(las, color = "Classification", size = 3, bg = "white")
```

- `filtered_points <- subset(las, las$Classification == 2)`
- `plot(filtered_points)`
- `plot(las)`
- `filtered_points <- subset(las, las$Classification == 1)`
- `plot(filtered_points)`



## 2. Extracting elements with LidR

Open Access Article

# An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation

by  Wuming Zhang <sup>1</sup>,  Jianbo Qi <sup>1,\*</sup>  ,  Peng Wan <sup>1</sup> ,  Hongtao Wang <sup>2</sup>,  Donghui Xie <sup>1</sup> ,  
 Xiaoyan Wang <sup>1</sup> and  Guangjian Yan <sup>1</sup> 

<sup>1</sup> State Key Laboratory of Remote Sensing Science, Beijing Key Laboratory of Environmental Remote Sensing and Digital City, School of Geography, Beijing Normal University, Beijing 100875, China

<sup>2</sup> School of Surveying and Land Information Engineering, Henan Polytechnic University, Jiaozuo 454003, China

\* Author to whom correspondence should be addressed.

*Remote Sens.* **2016**, *8*(6), 501; <https://doi.org/10.3390/rs8060501>

**Received: 13 March 2016 / Revised: 19 May 2016 / Accepted: 3 June 2016 / Published: 15 June 2016**

(This article belongs to the Special Issue **Airborne Laser Scanning**)

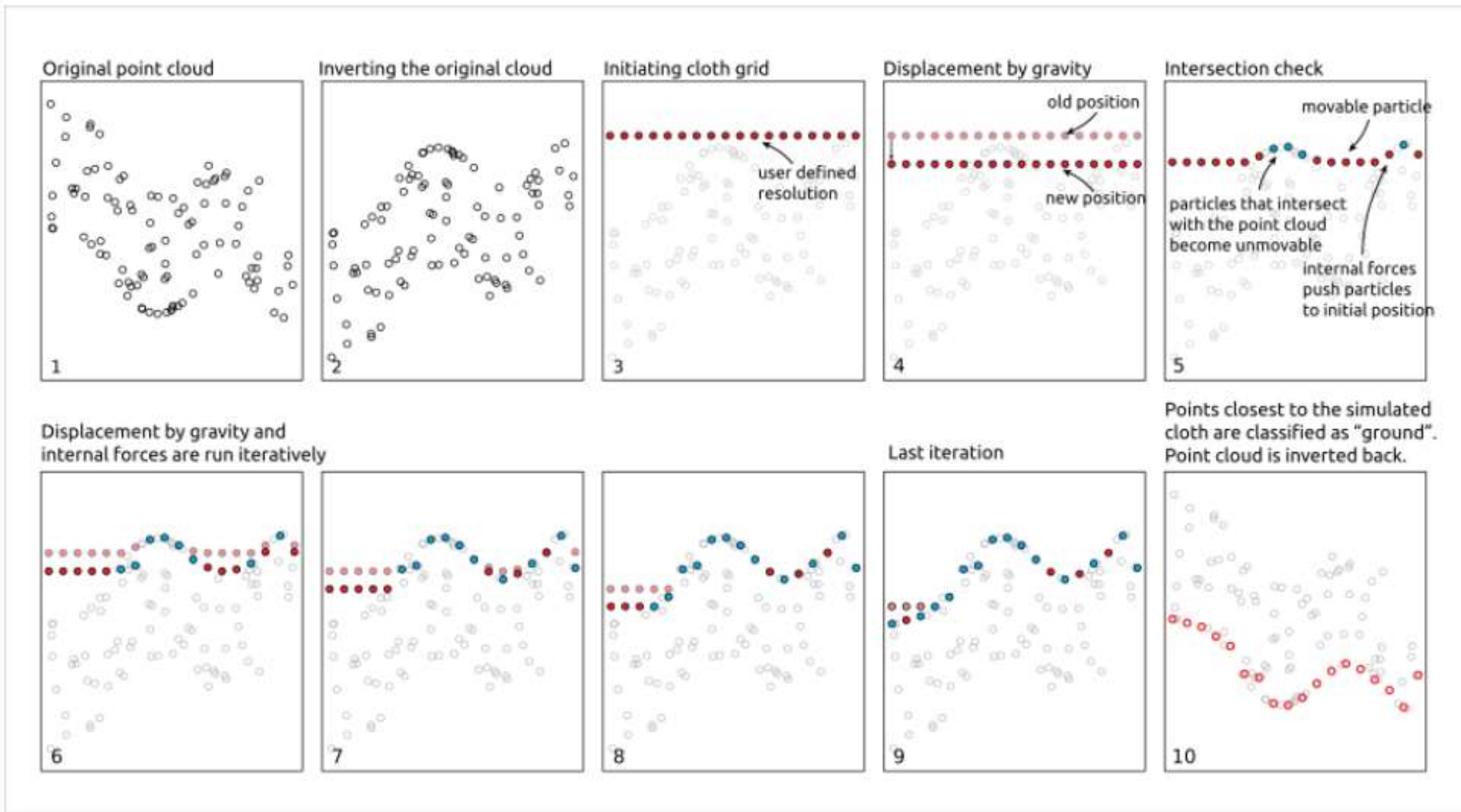
Download 

Browse Figures

Versions Notes



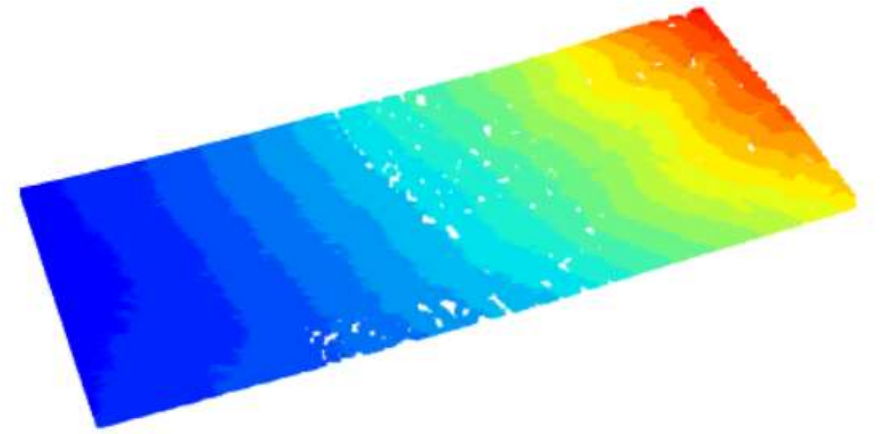
2. Extracting elements with LidR



```
las2 <- classify_ground(las2, algorithm = csf())
```

## 2. Extracting elements with LidR

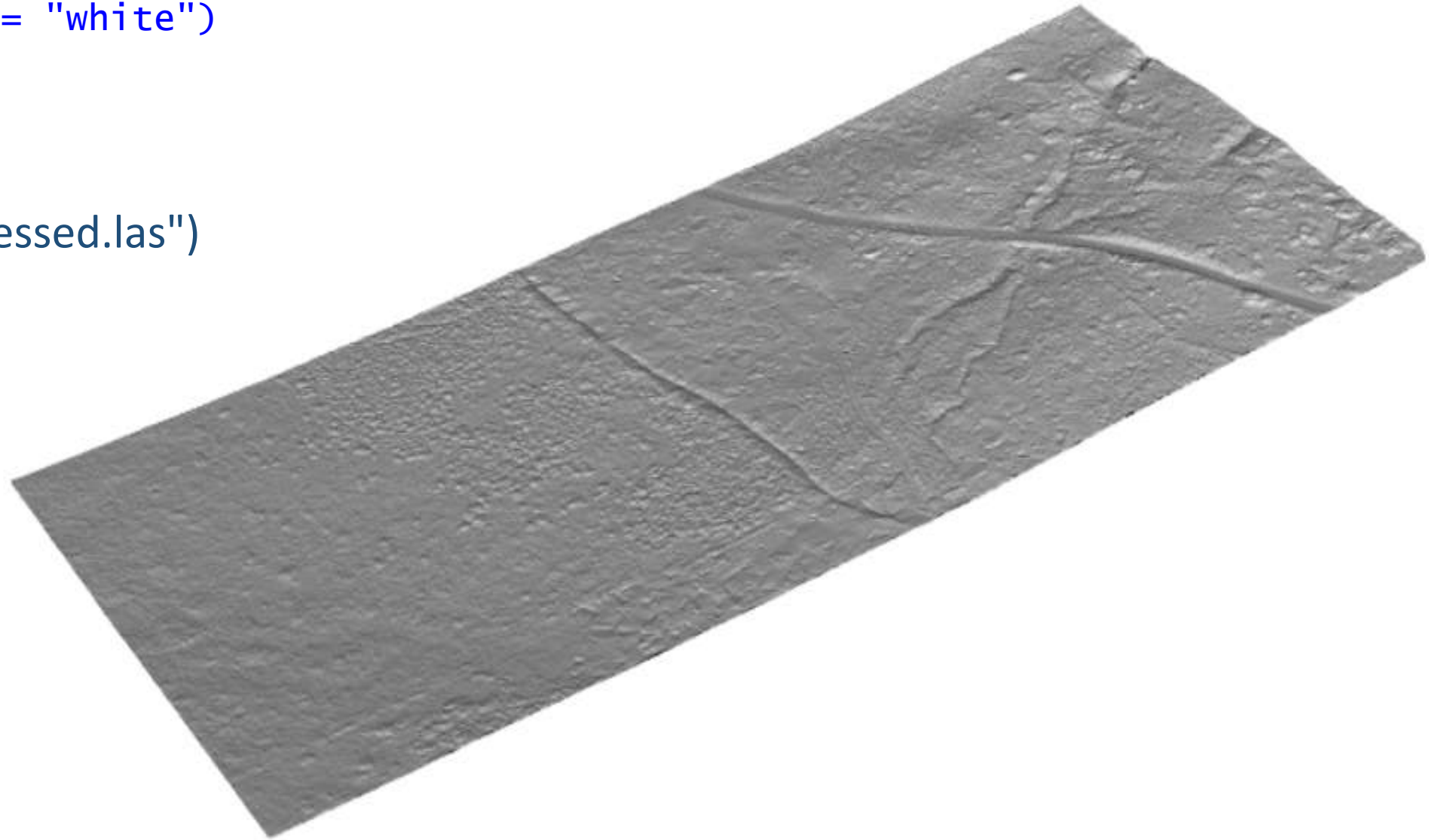
```
gnd <- filter_ground(las2)
plot(gnd, size = 3, bg = "white")
```



## 2. Extracting elements with LidR

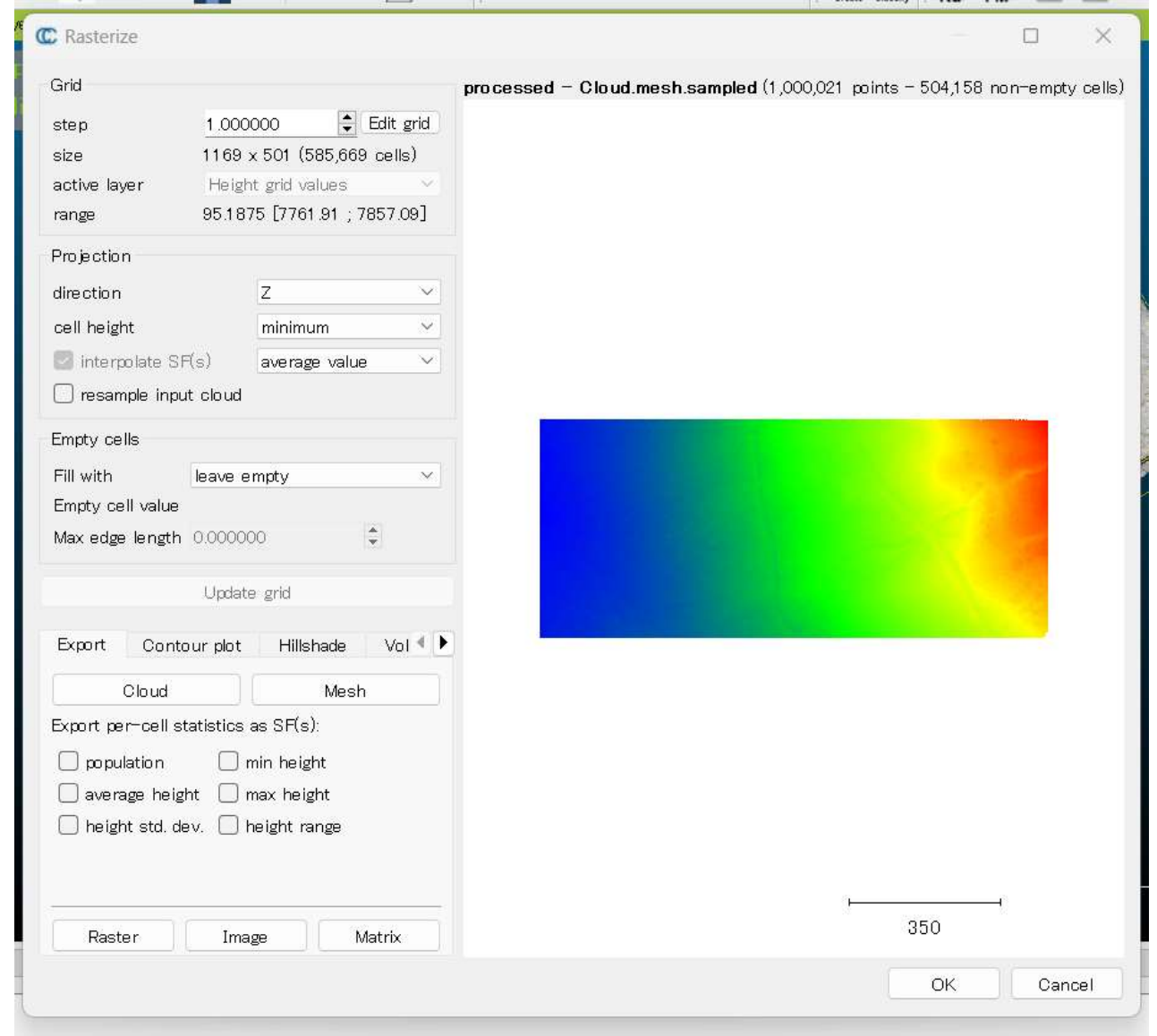
```
dem <- rasterize_terrain(gnd, res = 1, algorithm = tin())  
plot_dtm3d(dem, bg = "white")
```

```
writeLAS(gnd, "processed.las")
```



1. Mesh Delauney 2.5
2. Point-sampling on Mesh
3. From Pt to DEM

... then you can also add population for your DEM to assess the quality.



## 2. Extracting elements with LidR

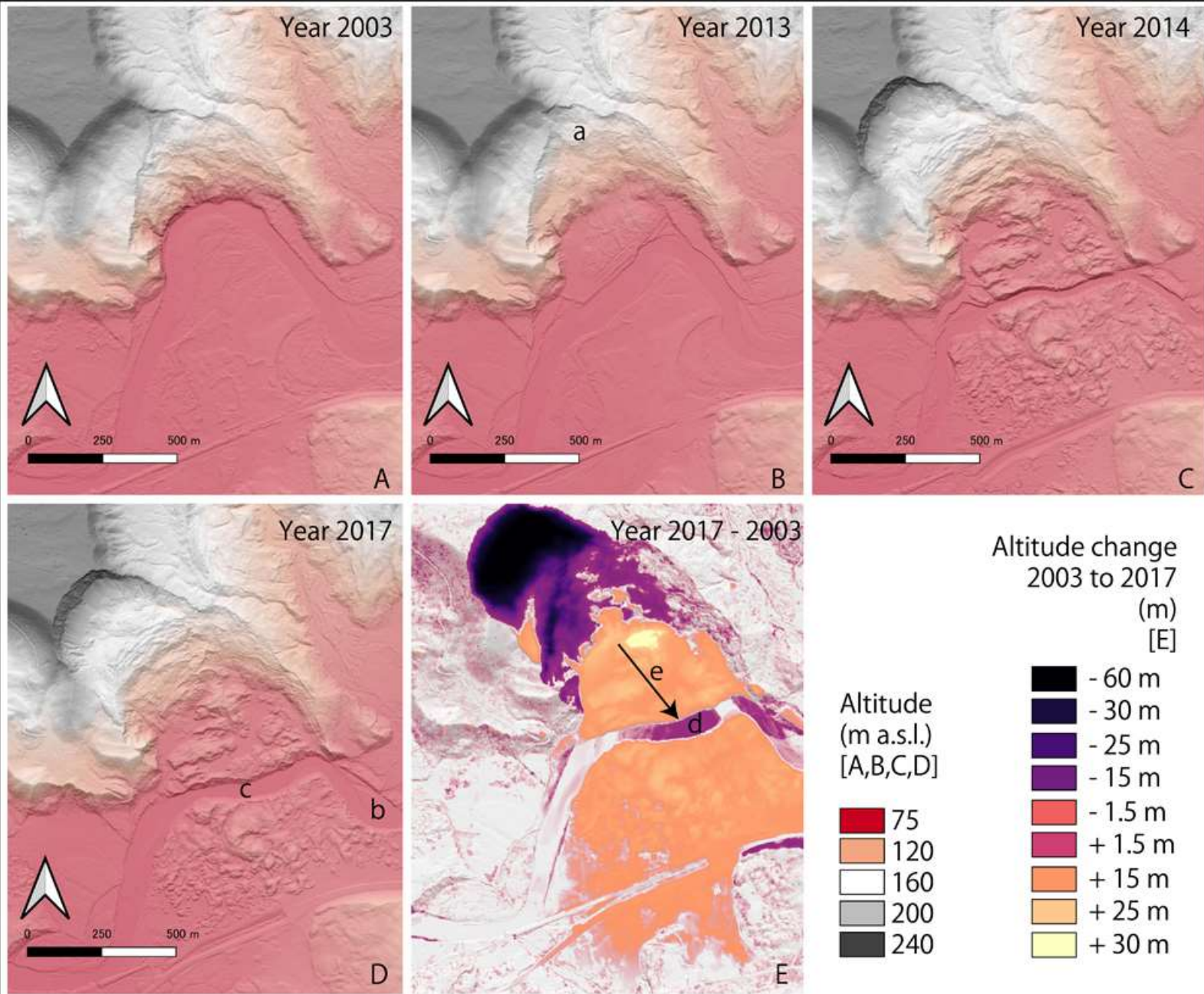


### 3. Comparing pointclouds

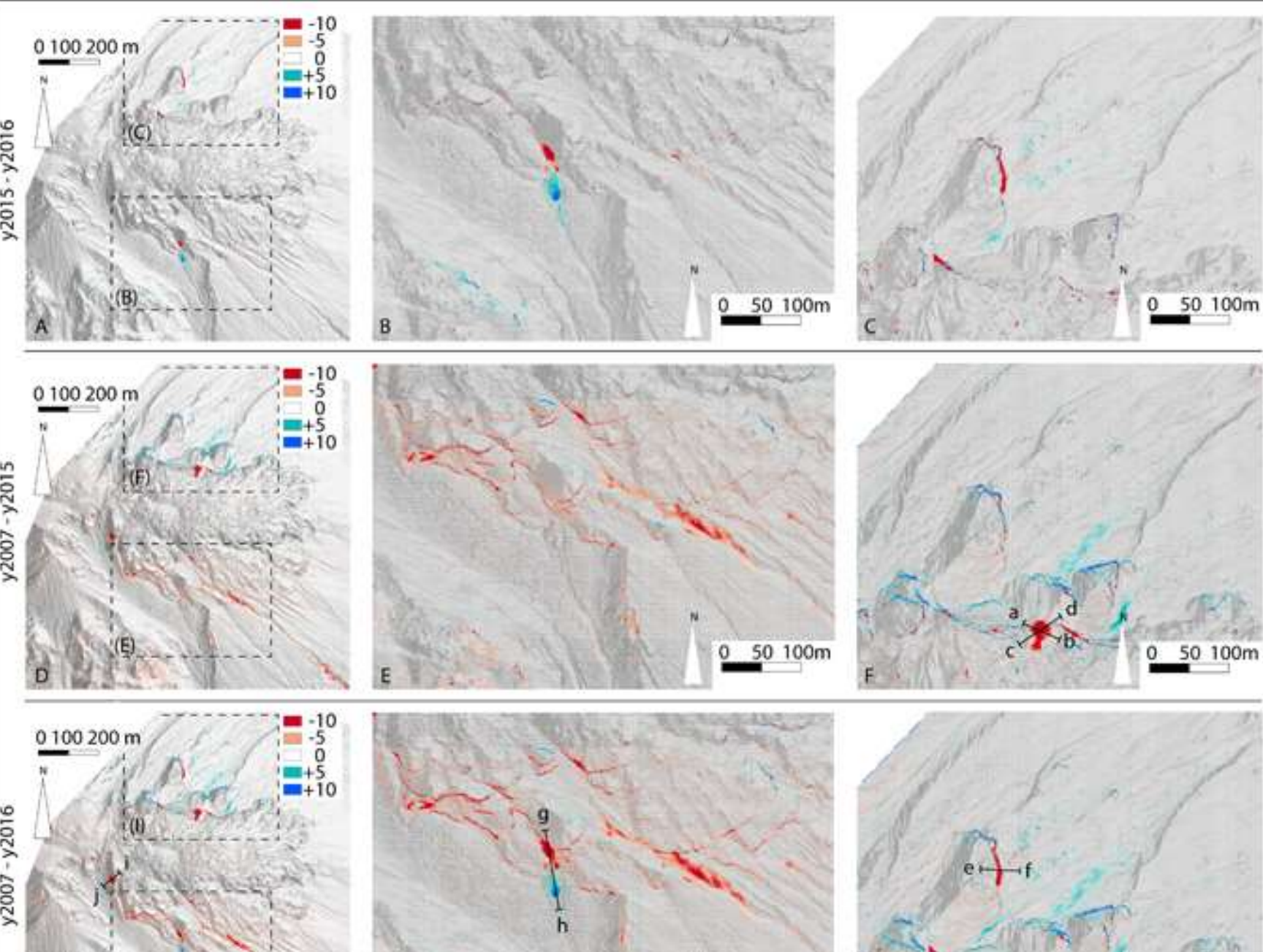


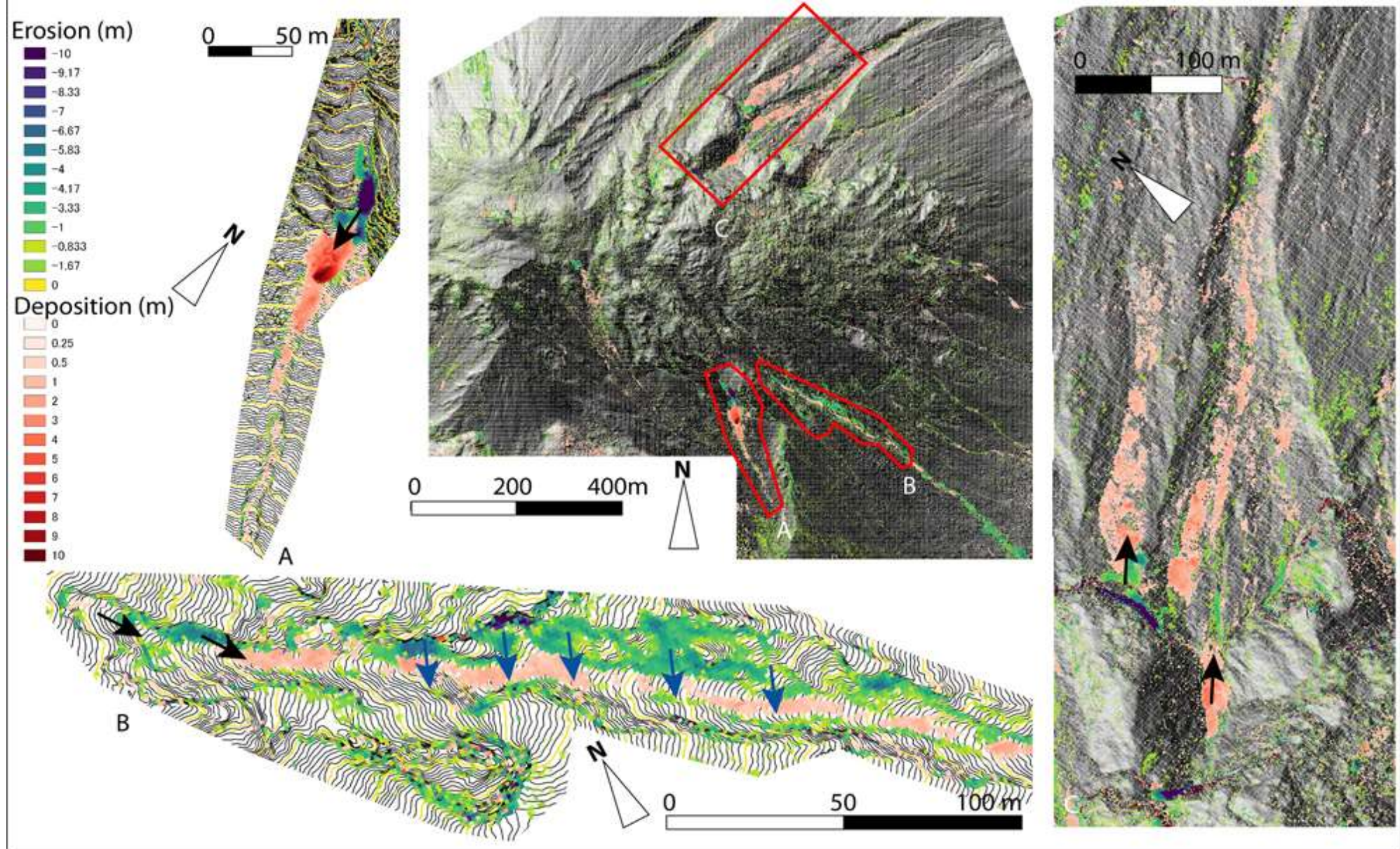


Temporal evolution from HRT (High-resolution Topography) PCL and DEMs differentiation at Oso Landslide in USA (LiDAR from the USGS and the OpenTopography community).



### 3. Comparing pointclouds





## 2. Comparing pointclouds

## The cone filtering method

$$\forall P_j \in A: hp_i - \Delta h(d(p_i, p_j), m) \leq hp_j$$

## The adaptive-cone filtering method

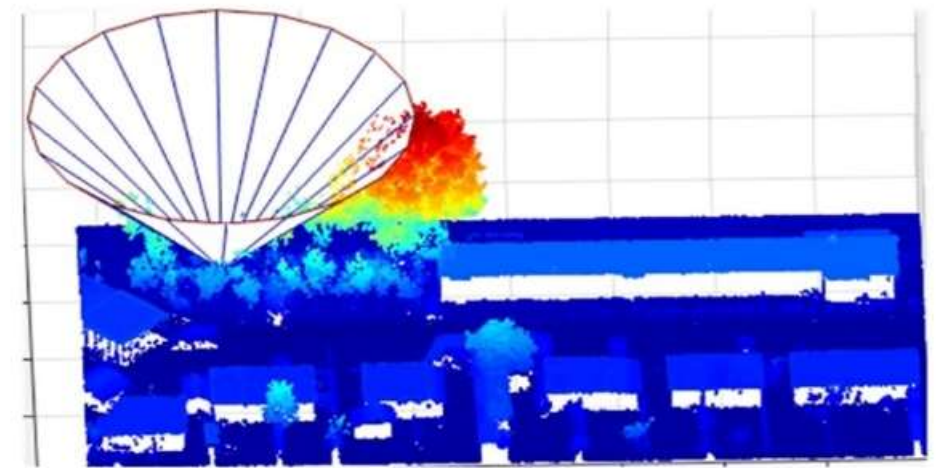
$$\forall P_j \in A: hp_i - \Delta h(d(p_i, p_j), m_i) \leq hp_j$$

Where,  $P_j$  is a point in the dataset within the window excluding the point being examined;

$hp_i$  and  $hp_j$  are the heights of the  $i$  points in the window and of the  $j$  point being examined,

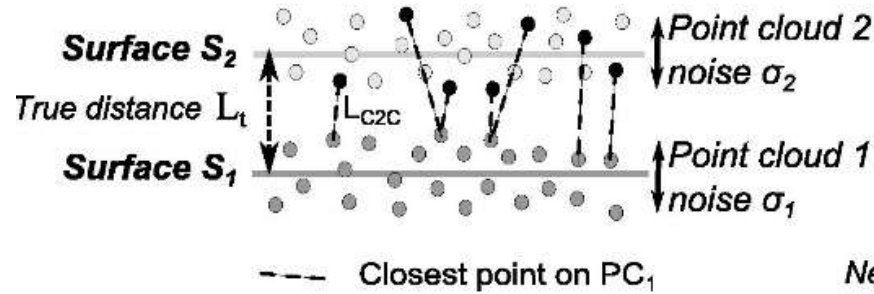
$m$  is the absolute value of the cone's gradient, and the negative value of  $m$  is the height of the cutoff plane,

while  $A$  is the set of points to be filtered out.

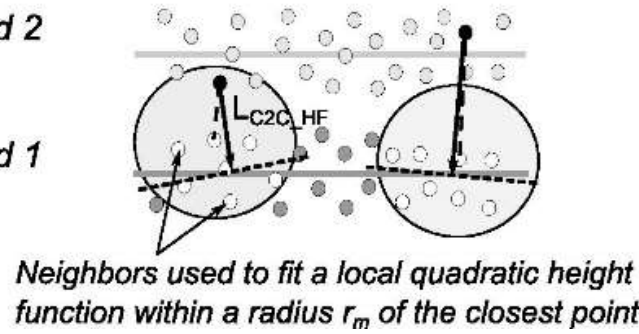


Figures after Mahphood and Arefi, 2020

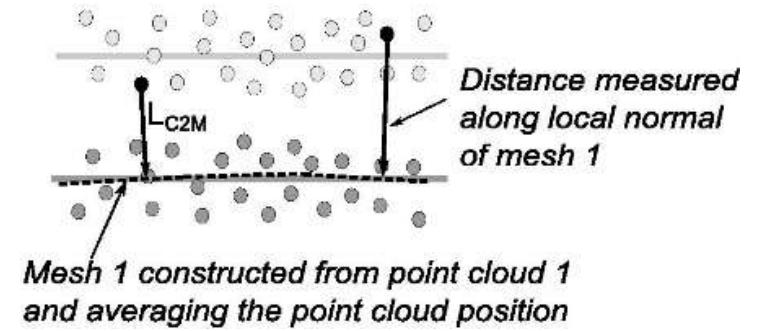
**A: Closest point distance  $L_{C2C}$**



**B: Closest point with local height function  $L_{C2C\_HF}$**

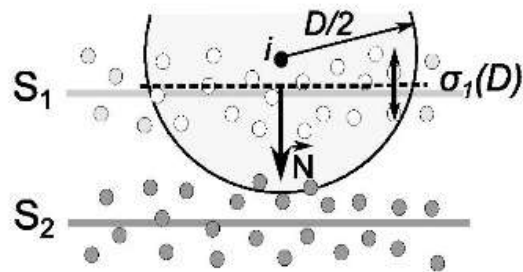


**C: Point to mesh distance  $L_{C2M}$**

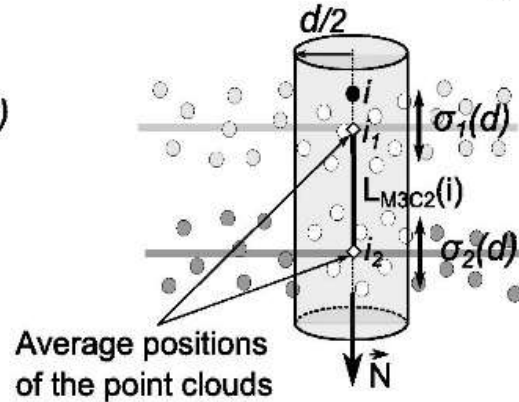


**a Principle of the Multiscale Model to Model Cloud Comparison M3C2**

Step 1: Calculation of normal  $\vec{N}$  at a scale  $D$  around the core point  $i$ .

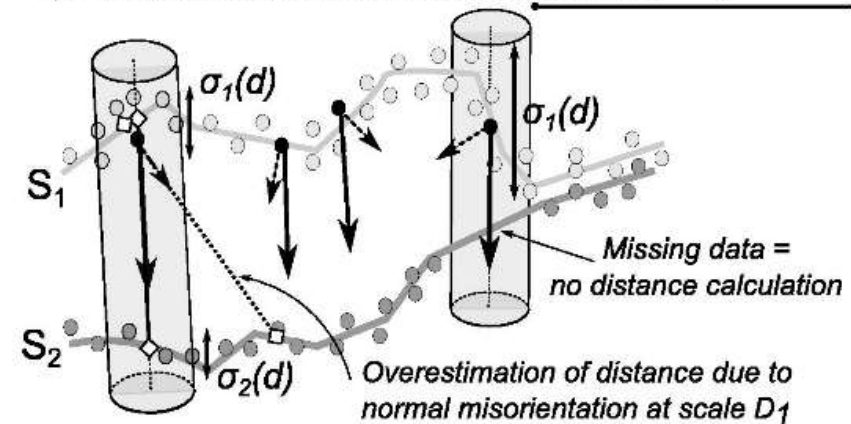


Step 2: Average distance between the two clouds measured at a scale  $d$  along  $\vec{N}$



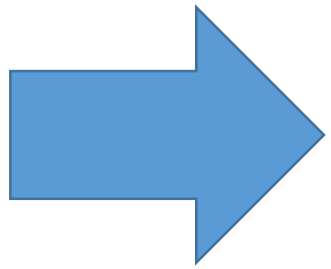
**b M3C2 on complex topography**

Normal at scale  $D_1$  affected by roughness  $\frac{D_1}{D_2}$   
 Normal at scale  $D_2$  not affected by roughness  $\frac{D_2}{D_2}$



Figures from: D. Lague et al., 2013. ISPRS Journal of Photogrammetry and Remote Sensing 82, 10-26.





We will learn how to do this second part, during the afternoon workshop with Mr. Rikuto Daikai.

## What you should know:

- (a) Understand what the LiDAR data are made of
- (b) Know how to manipulate them in CloudCompare and with LidR
- (c) Know how to extract elements from the dataset
- (d) Be able to transform your raw data from the LiDAR into a DEM and a DSM