# Lesson 10
# Point clouds

GEO1015.2023

Hugo Ledoux

# Appendices of terrain book are important!

Appendix 157

# PC in text files

ASCII

```
x y z
84499.948 446610.324 0.407
84499.890 446609.862 0.434
84499.832 446609.420 0.442
84499.777 446608.987 0.454
84499.715 446608.528 0.444
84499.839 446612.808 0.493
```

PLY

```
ply
format ascii 1.0            ← encoding and ply version number
comment This is an example file!
element vertex 7            ← number of points, start of point record definition
property float x
property float y
property float z
property int custom_attribute     point record definition
end_header
91443.89 438385.69 -0.80 11
91443.94 438386.10 -0.78 43
91444.00 438386.51 -0.79 44
91444.06 438386.94 -0.83 31     point records
91444.11 438387.36 -0.86 31
91443.88 438383.50 -0.83 22
91443.93 438383.91 -0.80 65
```

# PC in binary files: LAS

- LASer file format (LAS)
- most widely used standard for the dissemination of point cloud data.
- designed for datasets that originate from (airborne) lidar scanners.
- classes are fixed (but space for user-defined ones):

| Code | Meaning |
|---|---|
| 0 | never classified |
| 1 | unclassified |
| 2 | ground |
| 3 | low vegetation |
| 4 | medium vegetation |
| 5 | high vegetation |
| 6 | building |
| 7 | low point (noise) |
| 8 | *reserved* |
| 9 | water |
| 13–31 | user-defined |

Format 0

**Table A.1:** LAS Point Data Record Format 0

| Field | Format | Length (bits) | Description |
|---|---|---|---|
| X | int | 32 | X coordinate |
| Y | int | 32 | Y coordinate |
| Z | int | 32 | Z coordinate |
| Intensity | unsigned int | 16 | The pulse return amplitude |
| Return number | unsigned int | 3 | The total pulse return number for a given output pulse |
| Number of returns | unsigned int | 3 | Total number of returns for a given pulse |
| Scan Direction Flag | boolean | 1 | Denotes the direction at which the scanner mirror was travelling at the time of the output pulse. A bit value of 1 is a positive scan direction, and a bit value of 0 is a negative scan direction (where positive scan direction is a scan moving from the left side of the in-track direction to the right side and negative the opposite). |
| Edge of Flight Line | boolean | 1 | Has a value of 1 only when the point is at the end of a scan. It is the last point on a given scan line before it changes direction. |
| Classification | unsigned int | 5 | Classification code |
| Scan Angle Rank | int | 4 | The angle at which the laser pulse was output from the scanner including the roll of the aircraft |
| User Data | unsigned int | 4 | May be used at the user's discretion |
| Point Source ID | unsigned int | 8 | Indicates the file from which this point originated Non-zero if this point was copied from another file |

$$X_{coordinate} = (X_{record} * X_{scale}) + X_{offset}$$
$$Y_{coordinate} = (Y_{record} * Y_{scale}) + Y_{offset}$$
$$Z_{coordinate} = (Z_{record} * Z_{scale}) + Z_{offset}$$

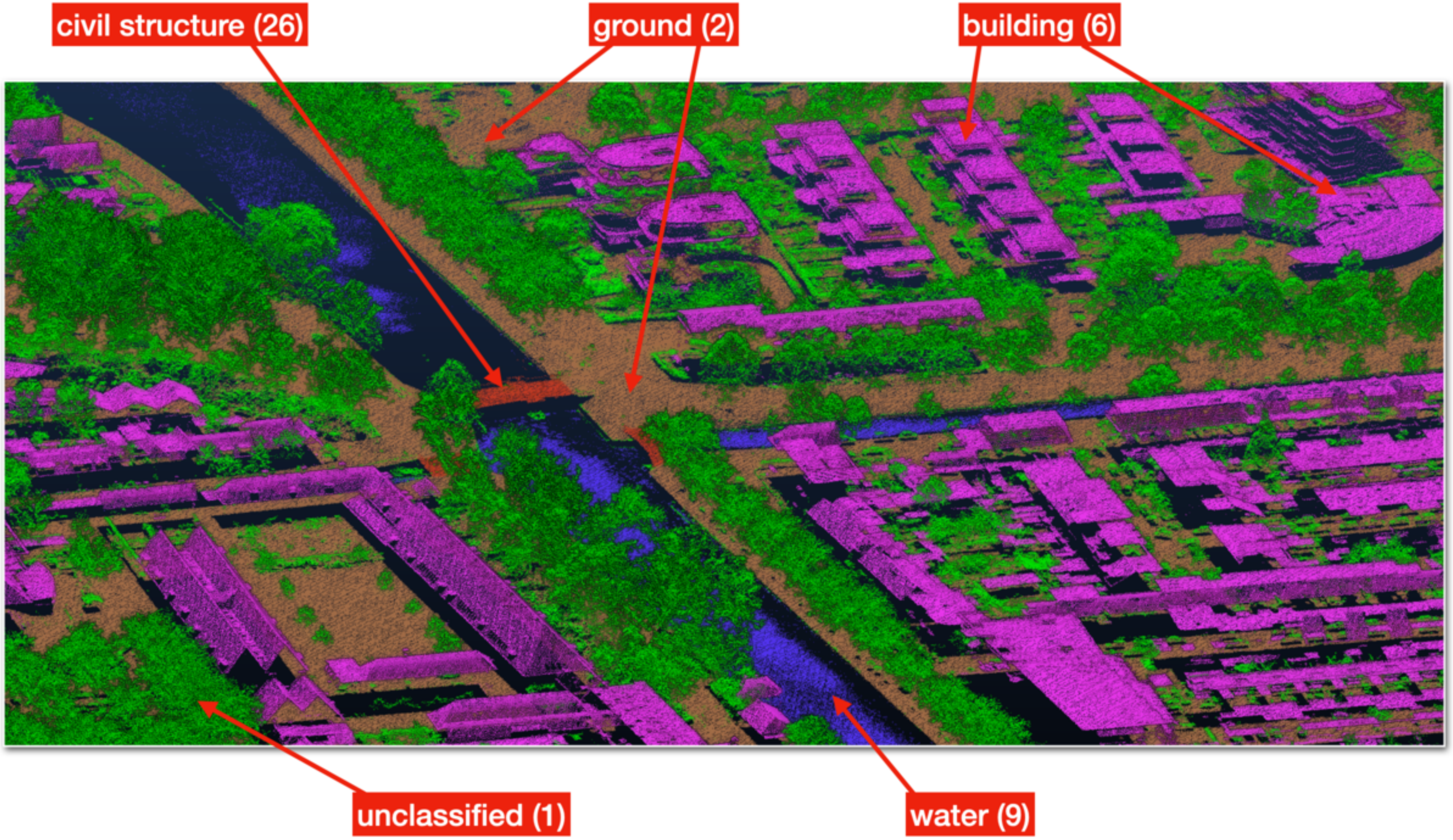
# Compressed LAS == LAZ

- Compression == 10X I'd say: try with AHN4!
- point records are grouped in blocks of 50,000 records
- Each block is individually compressed, which makes it possible to partially decompress only the needed blocks from a file (instead of always needing to decompress the whole file).
- a greater compression factor can often be achieved after spatially sorting the points.
- Read/write is slower…

Not an official standard!

!= zLAS

# AHN4 classification




Class 1
=
unclassified
(includes many things!)

Class 14
=
High-volatage
pylons+cables

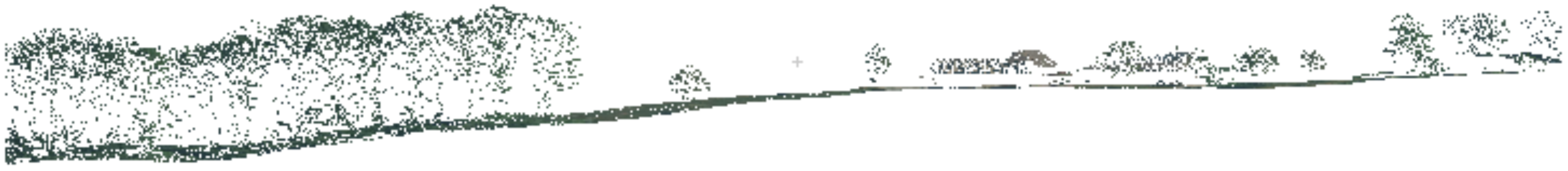
Class 26
=
bridges, statues, and
viaducts

# Thinning

- **random:** randomly keep a given percentage of the points, eg 10%.
- ***n*th-point:** keep only the $n$th point in the dataset. For instance, if $n = 100$, we would keep the 1st, the 101th, the 201th, etc; a dataset with 100 000 points is reduced to 1000 points. This is the quickest thinning method.
- ***n*th-point random:** if there is some structure in the input points (eg if generated from a gridded terrain) then $n$th-point could create datasets with artefacts. The randomised variation chooses randomly in the $n$ points one point.
- **grid:** overlay a 2D or 3D regular grid over the points and keep $m$ points per grid cell. That can be one of the original points, an average of those, or the exact centre of the cell. The thinning factor depends on the chosen cell-size. Notice that the result is often a point cloud with a homogeneous point density on all surfaces (only on the horizontal surfaces if a 2D grid is used).

# Ground filtering

This does not use the classification, only geometry!

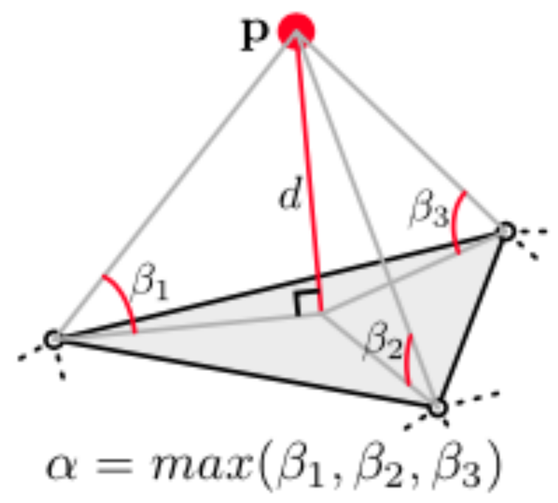


(a) Original point cloud

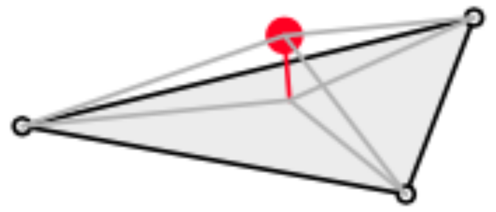
(b) After ground filtering

# Method #1: Ground filtering with TIN (GFTIN)

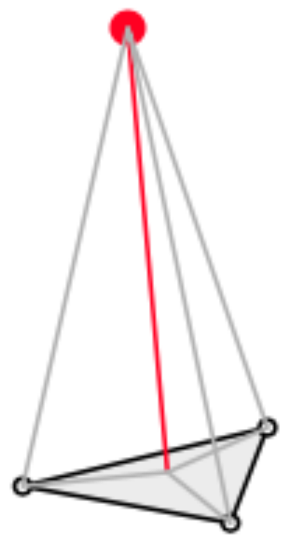
1. construction of a rudimentary initial TIN (usually a Delaunay TIN);
2. computation of two geometric properties for each point that is not already labelled as ground;
3. incremental insertion of points that pass a simple and local 'ground test' based on the computed geometric properties.




**(a)**



**(b)** example ground point



**(c)** example non-ground point

**Figure 11.5:** Geometric properties for a point $p$ in the method for ground filtering based on TIN refinement.

# Method #2: Cloth simulation filter (CSF)

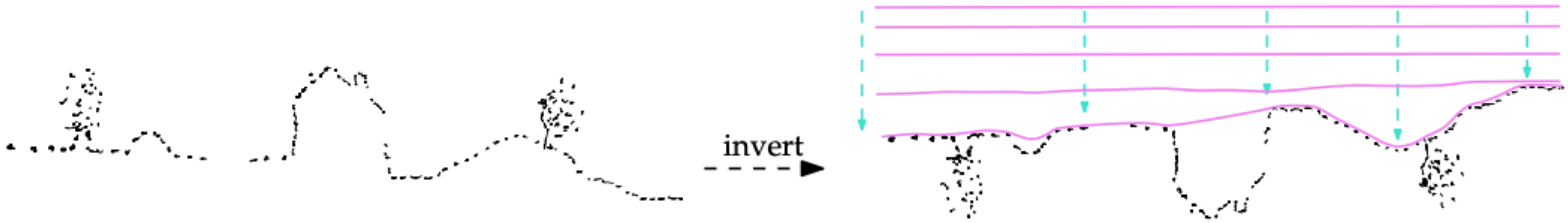



**Figure 11.6:** Basic idea behind the CSF algorithm for ground filtering of a point cloud: inverting the data and letting a cloth fall.

# Method #2: Cloth simulation filter (CSF)

Two factors influence the $z$-value of a particle during the cloth falling process:

1. **external forces:** in this case this is the gravity pulling down a particle;
2. **internal forces:** the tension in the cloth, which is modelled by the interactions between a particle and its neighbours.

As particles fall down, some will reach the ground and become *unmovable*. These will potentially be neighbours to *movable* ones, whose elevation will be controlled by how we define the rigidity of the cloth.

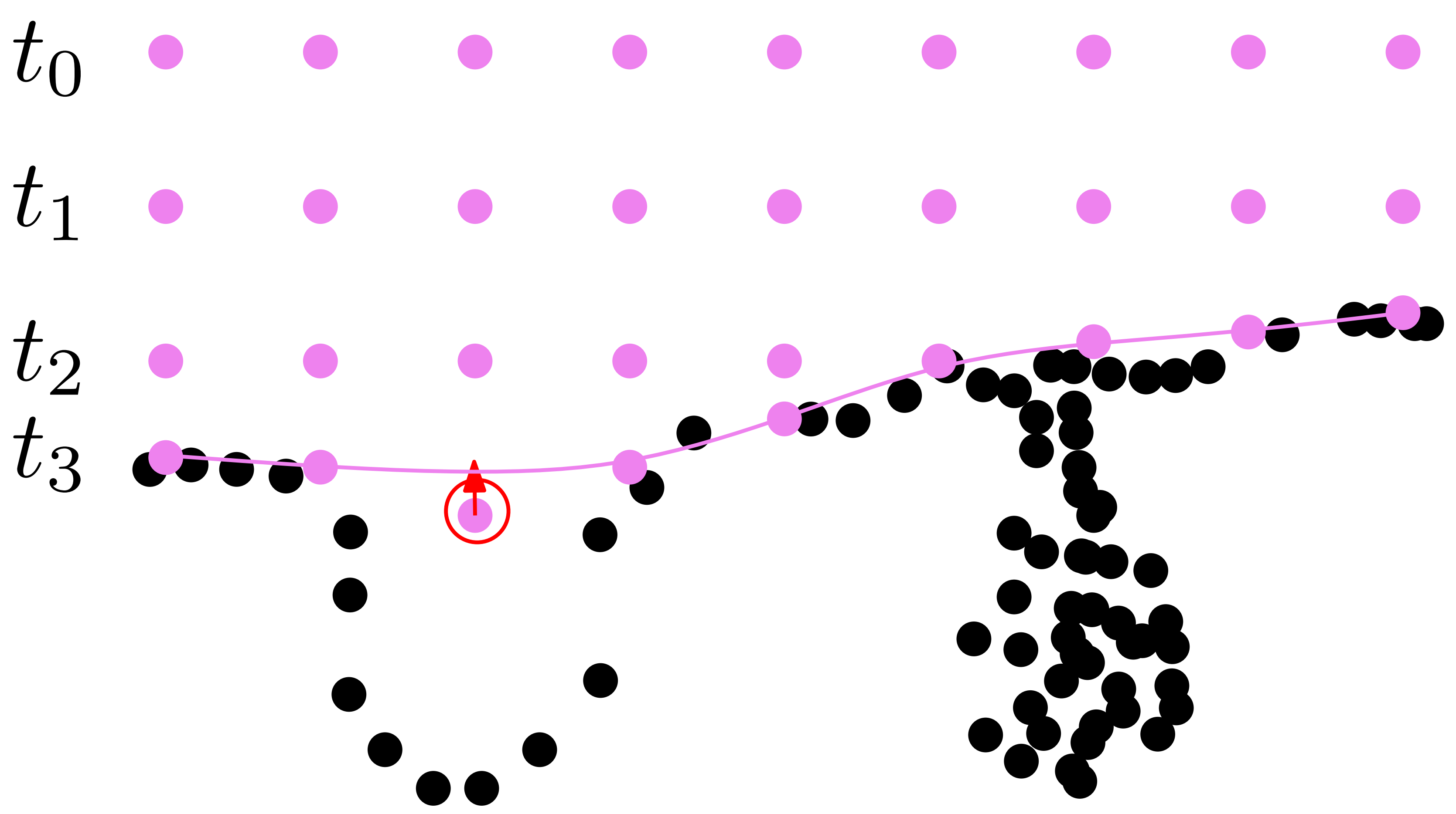
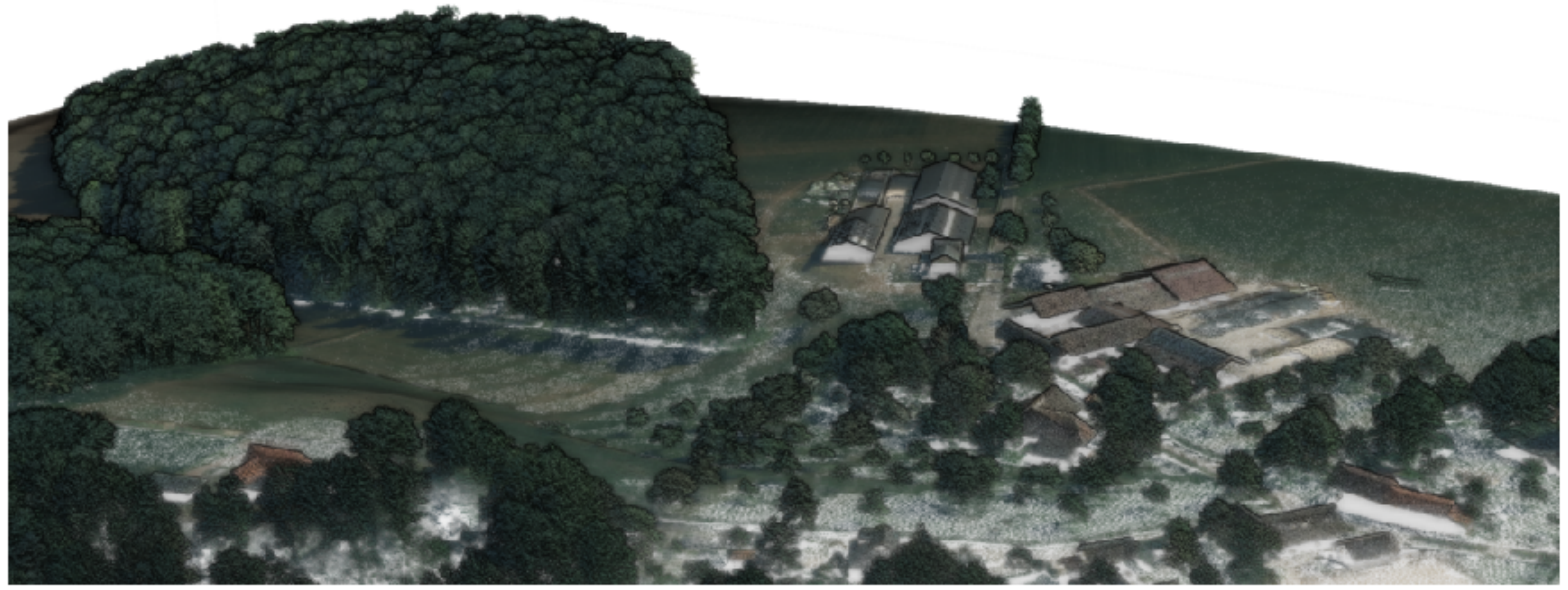
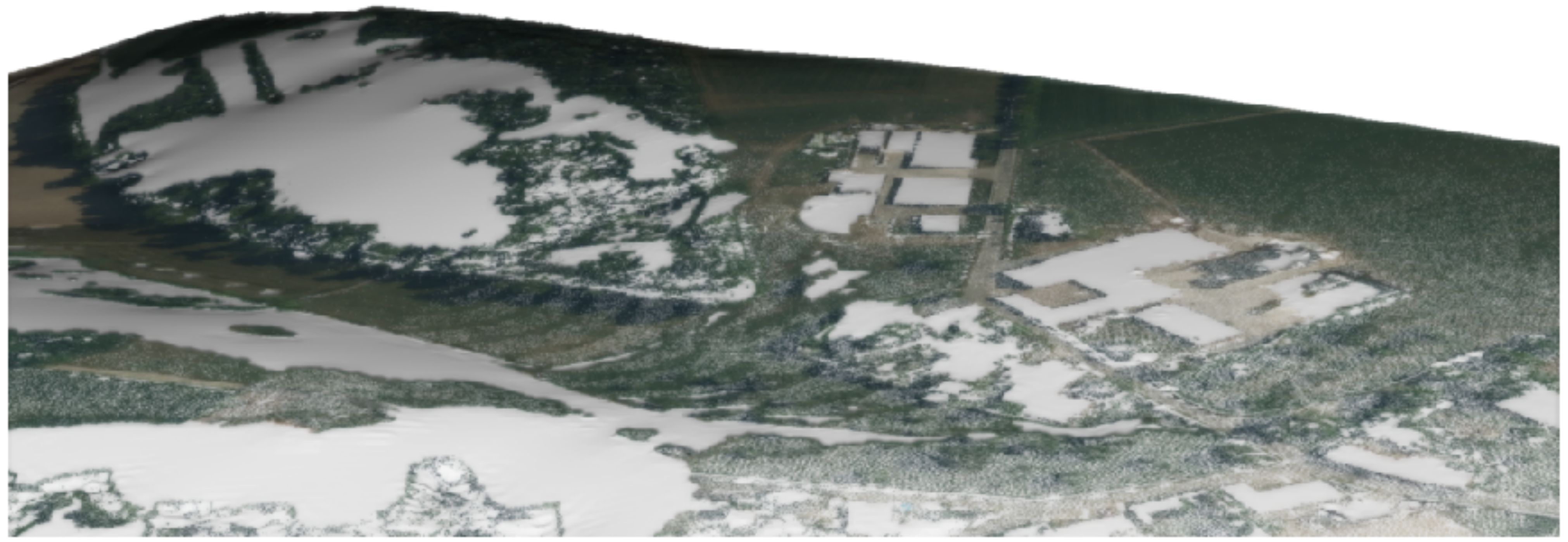



**Figure 11.7**

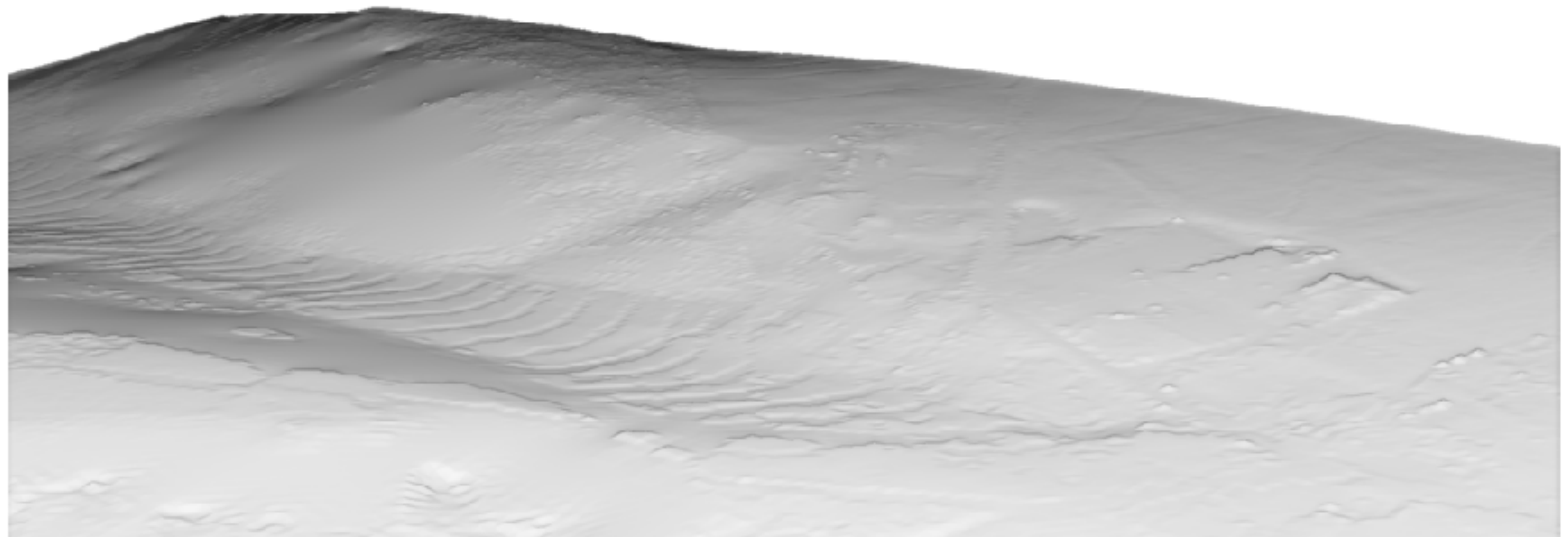
# CSF: 2 outputs



(a) Original point cloud



(b) Output #1: the ground points (with the ground surface shown in grey)



(c) Output #2: the ground surface

**https://3d.bk.tudelft.nl/courses/geo1015/**