HW04
Processing Point Clouds

Michele Giampaolo 5865506
Sharath Chandra Madanu 5722101
Vidushi Bhatt 5862124

0 DOWNLOADING AND PREPARING THE DATASET

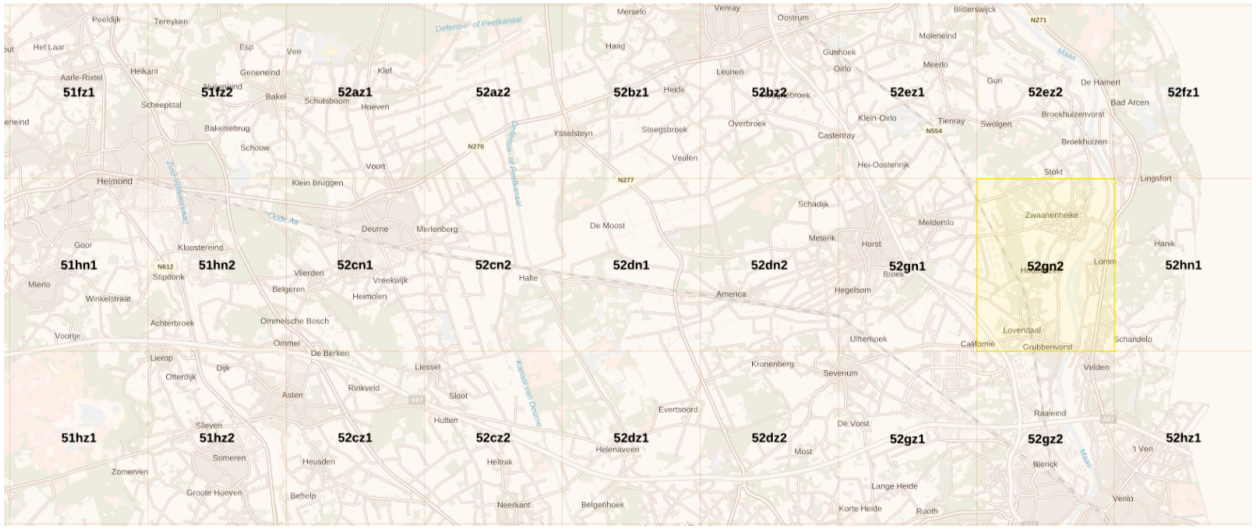


Fig0.1 *AHN3 tile "52gn2" from PDOK.*

We downloaded the LAZ file of the AHN3 tile "*52gn2*" from PDOK, which contains parts of the cities of Lottum, Lomm, Grubbenvorst and Velden in the Limburg province of the Netherlands (fig. 0.1).

We completed the assignment using **Python, QGIS**. We created the file "**user_inputs.py**" where the user has to define the inputs such as the file locations, output directories and various attributes needed for the functions that will follow.

To crop the dataset to a 500mX500m region we mainly used the functions found in the **laspy** library to open and work with the .laz file we downloaded. We defined the output file name and directory of the new cropped file. Then through the **chunk_iterator** function we were able to go through the points in chunks of a requested size (we chose 1,000,000) in order to make processing easier. We created an array of boolean values that defines if each of the points of the original files lies in the boundaries defined by us (Table 1). Then the points to which a True value corresponded were added to a new array and written to a new .laz file (fig. 0.2). We also wrote an output .txt file of the points in the cropped region.

left	208500
right	209000
bottom	385500
top	386000

Table1 *Bounding box of 500m x 500m tile*

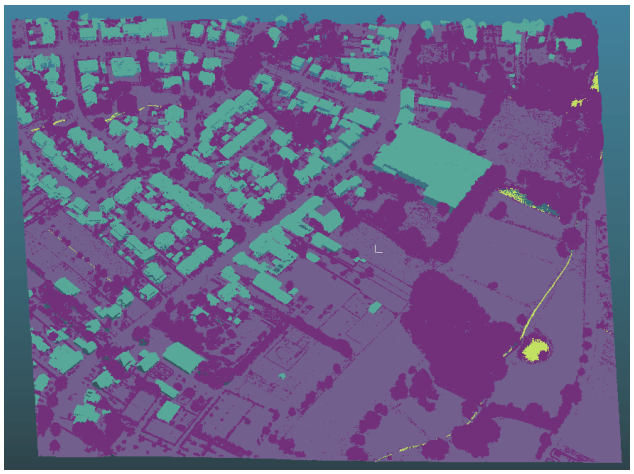


Fig0.2 *The cropped region visualized in CloudCompare (colors correspond to classifications of the .laz file).*

1 FILTERING THE GROUND USING THE CLOTH SIMULATION FILTER

To use the CSF (Cloth Simulation Filter) we firstly inverted the z value of all the points of our cropped file. In order to create the point grid of our Cloth mesh, we used the numpy function **arange** with the cloth resolution as the given interval for both the x and y coordinates. We concatenated the x and y values to get the coordinates of each point of the Cloth mesh.

We defined the initial values for:

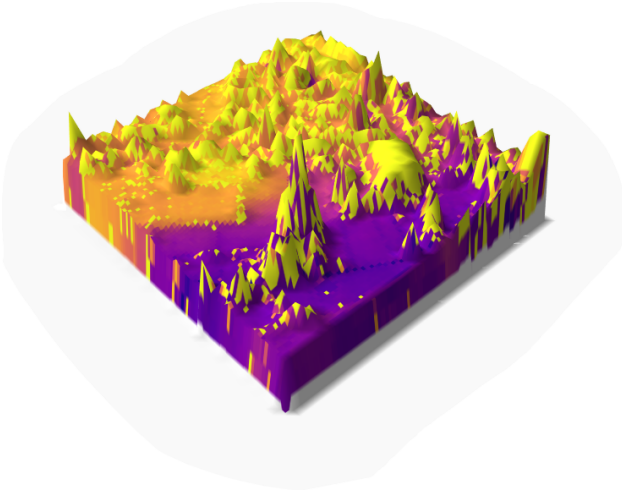
Tension, a user defined parameter to describe how "taut" the cloth will be,
ez_max, a user defined parameter for the displacement of particles, which ends the process.
baby_step, the fall in height due to gravity at each step (we chose 10cm),
Pzmin, the lowest possible elevation of each Cloth point based on its closest (Point Cloud point when projected in the 2D plane),
Pzcur, the current z value of each point, which at the start is at an arbitrary position above the highest point of the point cloud (we chose 3 "baby steps" above the highest point),
Pzprev, the z value of each Cloth point before the current baby step. It can be calculated as Pzcur + baby_step.
delta_z, the maximum displacement of all particles from the previous step, when this value is below ez_max, the process ends

The **external force** of gravity is calculated equally for all points that are considered movable, by decreasing the z value by one baby_step.
The **internal force** is calculated based on if both of the points are movable or not. If only one is movable then it is moved towards the immovable one, by calculating the difference in height between the points multiplied by the tension. If both of them are movable then both of them move towards each other and the distance that they move is equal to half of their height distance multiplied by the tension.

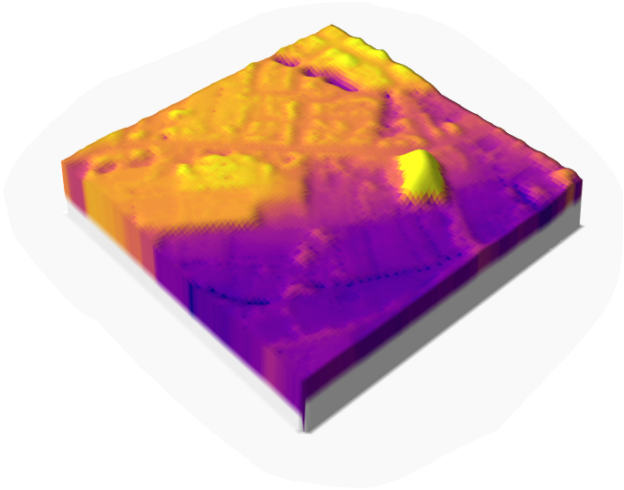
For each iteration the code checks if any points are below their corresponding Pzmin, and if they are, their z value is updated to Pzmin. Then the list of movable points is updated to include only the points that have a z value above their Pzmin. Therefore these previous points are permanently considered immovable from now on.

We repeated the process with different tension values (0.00, 0.25, 0.50, 0.75, 1.00) to see the difference in the final output. The resulting text files that specified the x, y and z value of every point were then imported into QGIS to quickly visualize the outcome, with a vertical exaggeration set to 5 (fig.1.1).

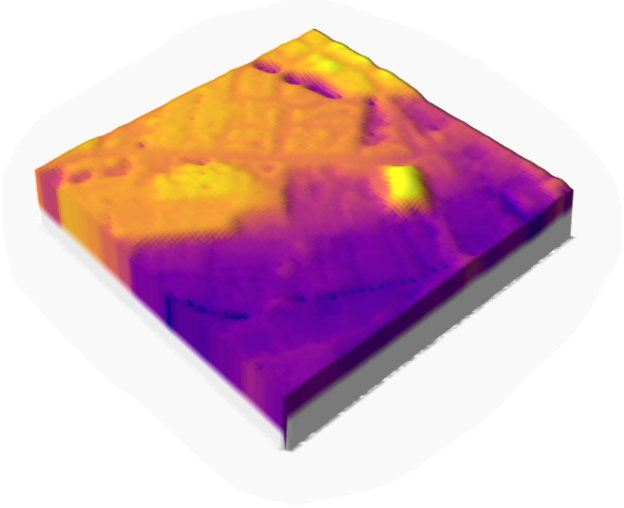
We can see how with a lower tension the taller elements affect more substantially the final values. With a tension of 0.00 the trees in our area create large spikes in the ground since the cloth is "loose" and easily "falls" in the small holes of the point cloud caused by the foliage. With just the next highest tension that we tested (0.25), this problem decreases dramatically, as the higher tension easily guarantees that the cloth does not fall into these smaller "holes". However, large tall and uniform areas, like those of big buildings, can still be seen as higher points even as the tension increases. A small hill remains even with the tension of 1.00, which corresponds to the "hole" caused by the biggest building in our tile, making it hard for the cloth to remain straight across it. It can be seen when we overimposed the satellite imagery, how these bumps correspond to the bigger buildings in the area.



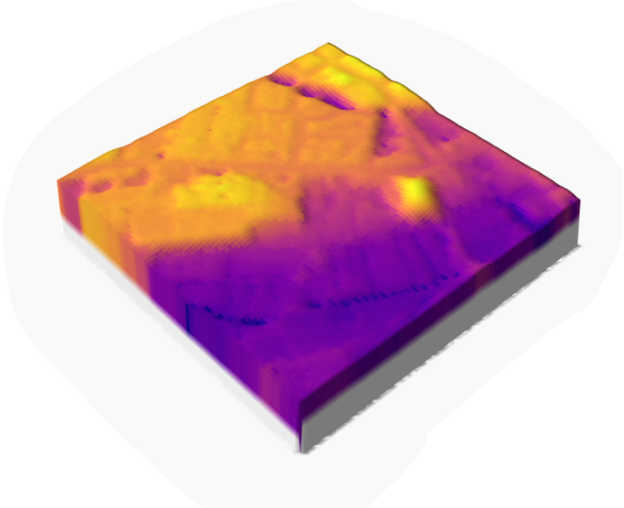
tension = **0.00**



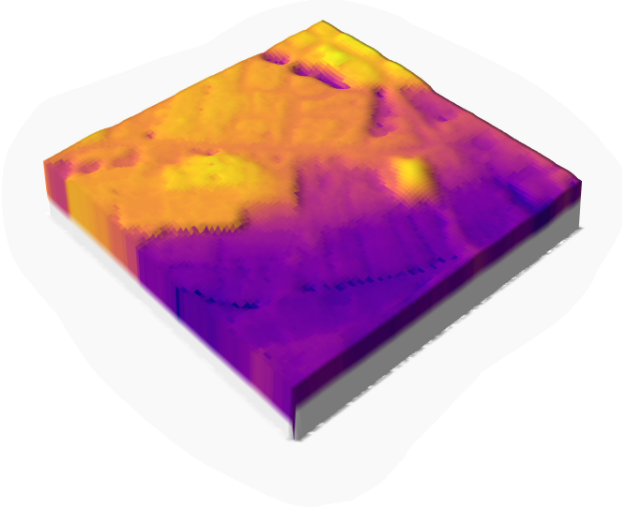
tension = **0.25**



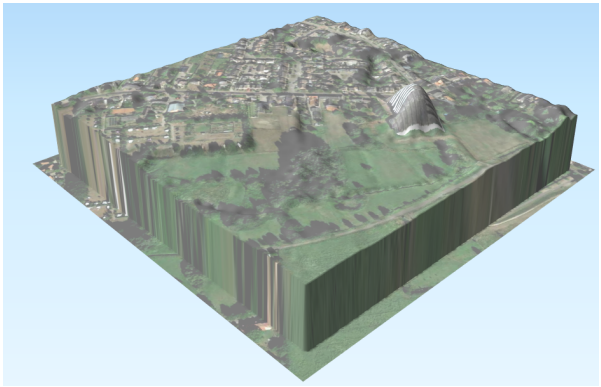
tension = **0.50**



tension = **0.75**



tension = **1.00**



tension = **0.25**, with satellite image (Google) overlayed

Fig1.1 *The result of the cloth simulation with different tension values as visualized in QGIS.*

2 DTM USING THE NATURAL NEIGHBOUR INTERPOLATION

To create the DTM using the data we obtained in the previous step with the cloth simulation, we used the library **startinpy**. We created a Delaunay triangulation class inserting the points of the cloth simulation. We decided to interpolate using the nearest neighbor interpolation, and so used the **.nni** class method to get a value for every point in a grid with 50cm resolution. We visualized the result in QGIS (fig. 3.1).

If the interpolation point is outside the convex hull of the cloth points, then -9999 is returned as interpolation value, to make it easily identifiable. All these outside convex hull points are on the boundaries of the grid.

3 DTM USING THE LAPLACE INTERPOLATION

We firstly classified the points of the point cloud as ground points by comparing them to the points we computed previously with the cloth simulation. Firstly the closest cloth point to every point cloud point was found and the difference in z value between them was calculated. If this distance was smaller than a previously defined **e_tol** (tolerance value), then the points are classified as ground (classification = 2) and written into new .laz and .txt files, otherwise they are labelled as classification = 0.
We created a Delaunay triangulation with only the points that were classified as ground points, by using the **startinpy** library. Then we used the **.interpolate_laplace** class method to get a value for every point in a grid with 50cm resolution. We visualized the result in QGIS (fig. 3.1).

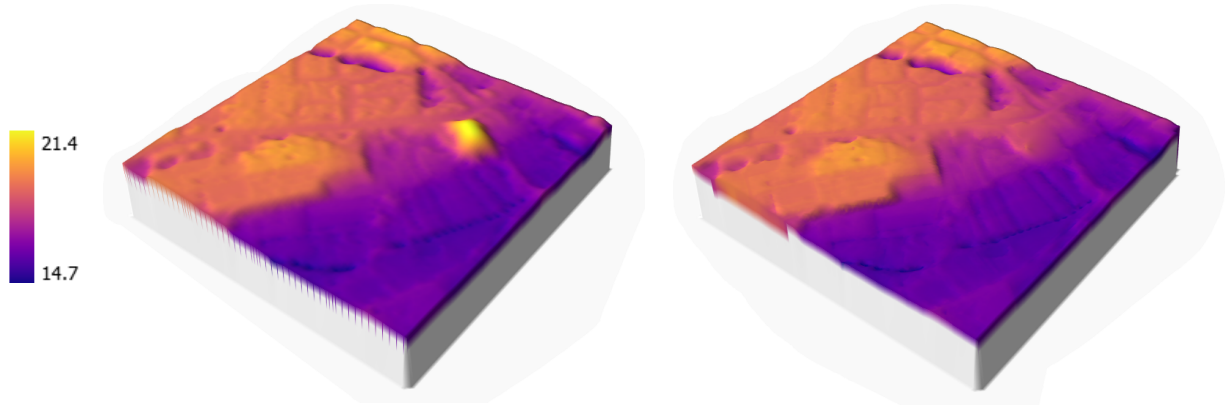


Fig3.1 *The result of the NNI DTM (left) and the Laplace interpolation DTM (right) visualized in QGIS with vertical exaggeration set to 5.*

4 COMPARING THE DTMs

In order to compare the different DTMs that we have acquired we decided to subtract the values of the raster cells in order to see which areas see more change, both positive and negative. We did this through the raster calculator in QGIS.

NNI DTM and **PDOK DTM**:
We subtracted the raster values of the NNI DTM from the PDOK DTM (fig. 4.1). We used a color scale that indicates in blue the positive difference, in red the negative difference and in white where there is no difference. Gray areas indicate areas where the PDOK DTM had no data, which correspond to the footprints of the buildings.
These two DTMs appear to have similar values in most areas, however some differences can still be spotted. The areas where the NNI DTM has lower values than the PDOK (in blue) seem to correspond to areas with lines of trees or with roads, whereas the parts where the opposite happens (in red) are fewer and don't seem to correspond to specific types of areas (they can be seen in a cluster of trees as well as in what seems to be flat fields).

To get a better understanding of the differences we also calculated some statistical values: RMSE, NRMSE, R-squared and Mutual Information (Table2).

Fig4.1 *Comparison between Nearest Neighbour DTM and PDOK DTM (left), and over satellite imagery (right).*

Laplace interpolation DTM and **PDOK DTM**:
We subtracted the raster values of the NNI DTM from the PDOK DTM (fig. 4.2). The same color scale was used.
Again, this DTM seems to be close in values to the PDOK DTM, as most of the area of the differences is white when visualized. It also appears to be closer to the PDOK DTM than the NNI DTM, as the areas where a difference can be seen are smaller and lighter in color, and therefore with a smaller difference.
The areas most affected seem to be the same as in the previous comparison: roads and compact trees.

We calculated the same values as before: RMSE, NRMSE, R-squared and Mutual Information (Table2).

Fig4.2 *Comparison between Laplace DTM and PDOK DTM (left), and over satellite imagery (right).*

	Cloth vs PDOK	Laplace (ground class) vs PDOK
RMSE	0.3587	0.1518
NRMSE	0.0603	0.0255
R-squared	0.9324	0.988
Mutual Information	13.7212	13.7603

Table2 *Comparison of PDOK DTM with NNI interpolated Cloth DTM, and Laplace interpolated Ground points*

NNI DTM and **Laplace interpolation DTM**:
We subtracted the raster values of the NNI DTM from the Laplace DTM (fig. 4.3) using the same color scale.

By comparing these two DTMs created by us we can see that similar areas as before are affected, such as the diagonal line of trees in the south, the road to the east and the cluster of trees in the north-west, showing that the NNI from the cloth simulation has lower values in these areas.

However, there are also a lot of other areas that see a difference, that in the previous two comparisons with the PDOK DTM did not appear. Most of these are areas where the NNI DTM has higher values, and we can see by overlaying the satellite image that these points correspond to the buildings. The most visible of these is the big industrial shed on the north right, which has the highest difference of the entire area over almost the entire footprint of the building. This is likely because the NNI DTM has used the values obtained from the cloth simulation, which even with high tensions had difficulty creating flat surfaces for large and uniform tall structures, resulting in the cloth "falling" in the "hole" of the building, while the Laplace DTM has used exclusively points which were labeled as ground.

These differences were not seen in the comparisons with the PDOK DTM, as these inherently do not deal with reconstructing the ground underneath buildings or other tall surfaces, and simply do not have data for these obstructed areas.

We calculated the same values as before: RMSE, NRMSE, R-squared and Mutual Information (Table3).

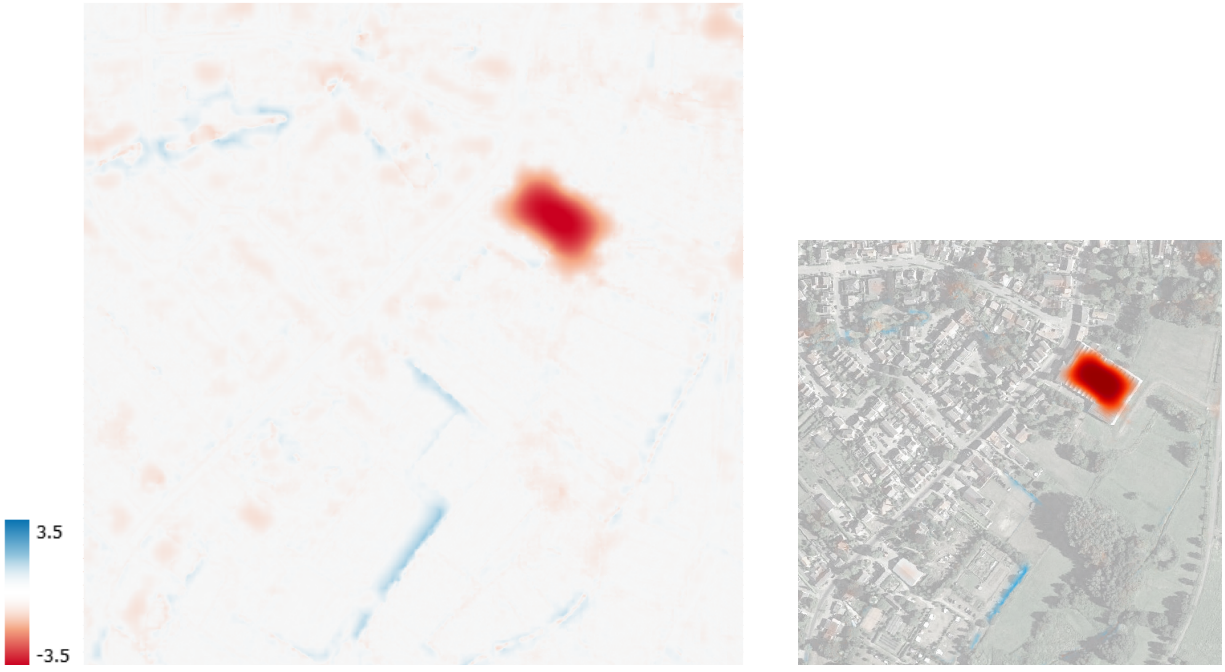


Fig4.3 *Comparison between Laplace interpolation DTM and Nearest Neighbor interpolation DTM (left), and over satellite imagery (right).*

	Cloth vs Laplace (ground class)
RMSE	0.29
NRMSE	0.04
R-squared	0.96
Mutual Information	13.77

Table3 *Comparison of NNI interpolated Cloth DTM vs Laplace interpolated ground classified points*

5 THINNING

We implemented two different thinning methods: **nth point** and **random percentage** (fig. 5.1).

The **nth point** method:

> *"keep only the nth 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."* (Ledoux, Ohori, Peters, Pronk, *Computational modelling of terrains*, 113)

By using the numpy function **arange** we created an evenly spaced out array with a given interval (we chose 2), so that we had an array of every 2nd index to have a 50% thinning. Then the points of the clipped .laz file with those indices were chosen and written into new .laz and .txt files.

The **random percentage method**:

> *"randomly keep a given percentage of the points, eg 10%."* (Ledoux, Ohori, Peters, Pronk, *Computational modelling of terrains*, 113)

We used the numpy function **random.choice** to generate a random sample from the size of our point (we chose 10%) array, therefore giving us the indexes of the chosen points. We specified the "Replace" parameter to be False so that a point cannot be chosen more than once. Then we selected the points with the chosen indexes and wrote them into new .laz and .txt files.

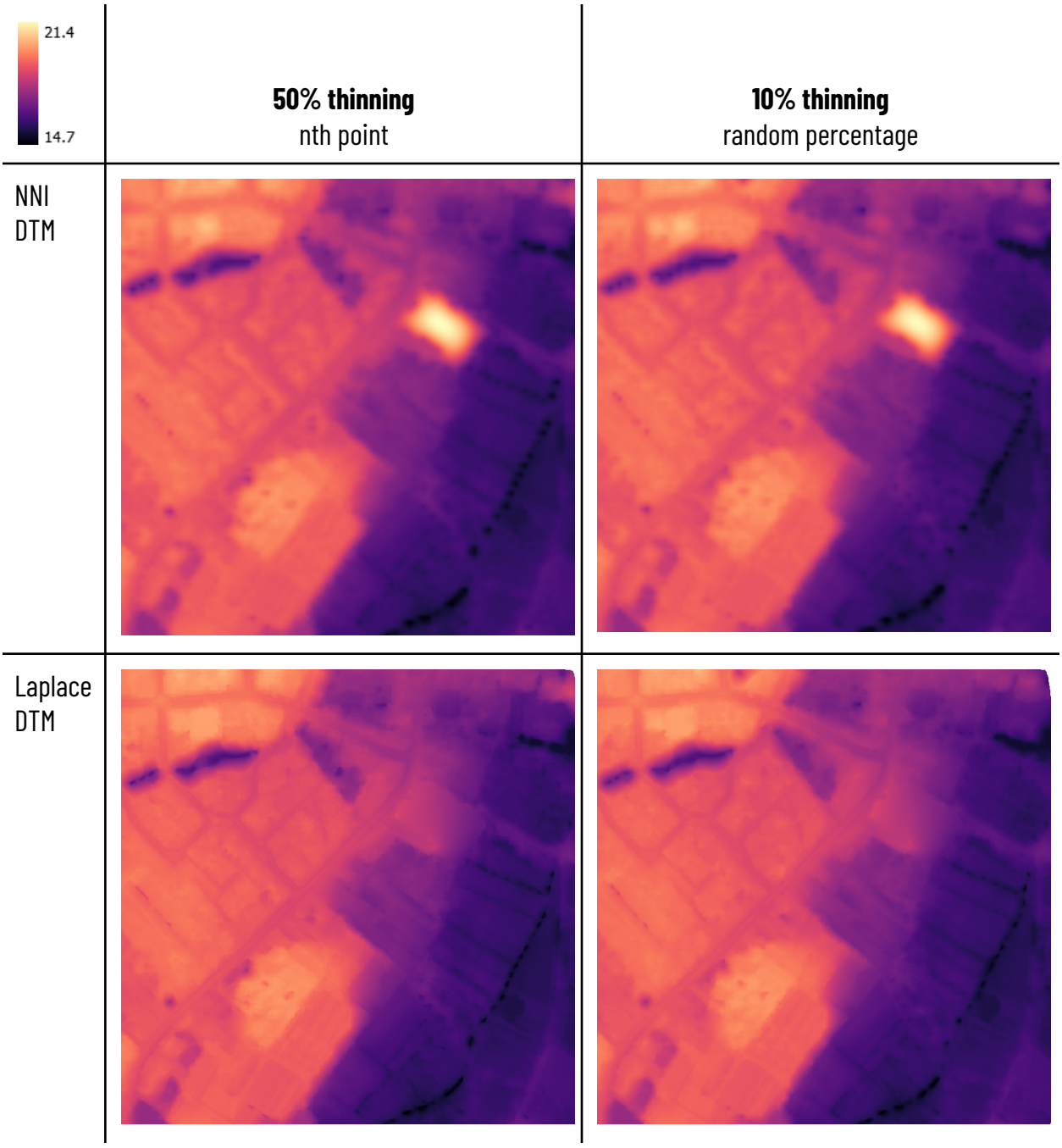


Fig5.1 *The resulting thinned DTMs with the two interpolation and two thinning methods.*

	RMSE	NRMSE	R - squared	Mutual Informatiom
50% thinning	0.0856	0.0125	0.9963	13.7525
10% thinning	0.085	0.0124	0.9963	13.7513

Table4 *Full data cloth comparison with cloths of different thinning.*

	RMSE	NRMSE	R - squared	Mutual Informatiom
50% thinning	0.0924	0.0161	0.9955	13.8138
10% thinning	0.0918	0.016	0.9956	13.8130

Table5 *Laplace interpolated full data DTM comparison with Laplace interpolated thinned DTMs*

	RMSE	NRMSE	R - squared	Mutual Informatiom
Cloth vs PDOK	0.3514	0.0591	0.9351	13.7215
Laplace vs PDOK	0.163	0.0274	0.9861	13.7603

Table6 *50% thinning cloth, and Laplace ground comparison with PDOK*

	RMSE	NRMSE	R - squared	Mutual Informatiom
Cloth vs PDOK	0.3421	0.0575	0.9385	13.7222
Laplace vs PDOK	0.1599	0.0269	0.9866	13.76

Table7 *10% thinning cloth, and Laplace ground comparison with PDOK*

6 EXTRACTING ISOLINES

To extract the isolines:

1. the **startinpy** library was used to create Delaunay Triangulation of the DTM points.
2. Triangles in DT were handled one by one and z values of each vertex was compared to identify points of intersection with isolines.
3. After identifying the two points of intersection in a DT (it was assumed after analysis that there cannot be a singular point of intersection of isolines with a triangle) LINESTRINGS were created and stored for each triangle and appended to a file.
4. The compiled list of linestrings along with their isoline values was written into a text file which was then opened in qgis as Delimited Text (figure below).

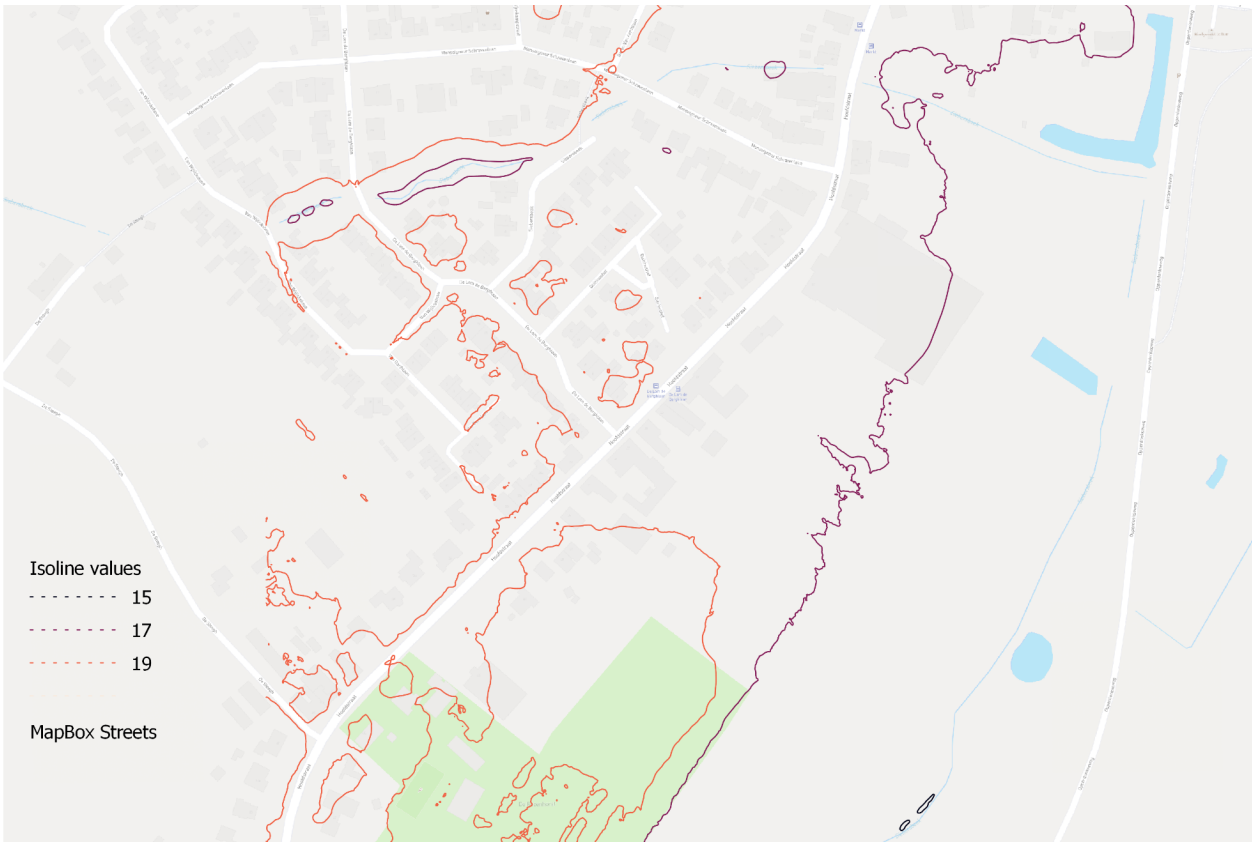
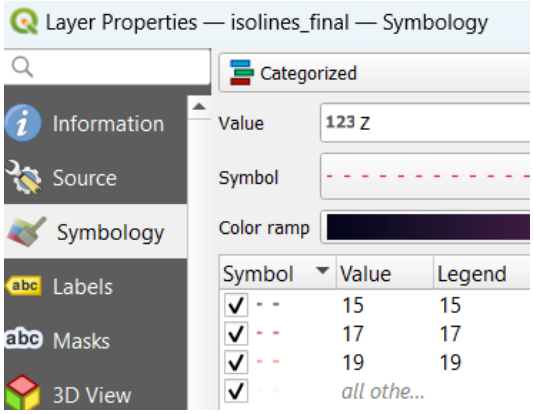


Fig6.1 *Isolines viewed on QGIS.*

In the range of 15 to 25 for interval 2, we got 3 contour lines:



7 WHO DID WHAT

Michele - Comparison of DTMs, Report layout
Sharath - Cloth Simulation, Thinning
Vidushi - Isolines