PROCESSING POINT CLOUDS

Geo1015-Assignment4

Authors
Hsinyu Cheng (6046525)
Xiaoluo Gong (5923476)
Yan Gao (6006175)

Contents

1 Preprocess

1.1 Clip and Buffering

Our original dataset of 1km x 1.25km has been reduced to 500m x 500m. In addition, we also cropped the 600m*600m data set because Laplace interpolation is required later. As the maximum area of buildings in our region is approximately 33m x 33m, in the third step, while searching for the lowest points, the grid size must be set to 40m(more than 33m) to avoid interference from buildings. This ensures that no point within a single cell represents an entire building. Additionally, due to subsequent Laplace interpolation of the dataset, an extra 40 meters is required on all sides to include at least one additional grid cell (if not added, the convex hull of TIN formed by the found lowest points might be smaller than 500m x 500m).

In other words, the initial TIN is generated in the third step, and therefore, the grid must be at least 580m x 580m. To achieve a grid size divisible by the desired 40m, we created a buffer zone of 600m x 600m.

We selected a combination of buildings, terrain, forests, and some water features. Below is an overview of the selected area (see figure1) and the bounding box.(see table 1)

Figure 1: Clipped 500m*500m Area (Top View in CloudCompare)

	x_500	y_500	x_600	y_600
Min	188465	311800	188415	311750
Max	188965	312300	189015	312350

Table 1: Bounding Box of Interested Area

1.2 Outlier Detection and Thinning

We used the **knn distance** method mentioned in the terrain book[1] to do the outlier detection. We create the KDTree with *scipy* and set the k value to 3 as it is shown in the case of the terrain book. The original 500*500 dataset has **5387080** points, and the outlier-removed dataset has **5208693** points. The original 600*600 data has **7634612** points, and the data set after using 0.25 thinning and removing outliers has **1823402** points.

To test the code easier, we also thinned the outlier-moved dataset by randomly keep 25% of the total points.

1.3 Main issues and solutions for preprocessing

Firstly, there are too many packages with KNN, and we initially used sklearn. After seeking advice on Discord, we found that sklearn is not suitable for our use case.

Secondly, when we initially exported the Laz file after removing outliers and loaded it into Cloud Compare, the entire file appeared white. Later, we realized that converting the Laz file to np.array and then writing it back to Laz led to the loss of many attributes. The solution was to improve the code. It's written in the comments.

2 Ground Filtering

2.1 Find the lowest point and generate the initial TIN

Here we use 600m*600m, and after 0.25 thinning, we remove the laz files with outlier. The laz file has 1823402 points.

Firstly, we need to set up a grid and identify the lowest point within each cell. As mentioned earlier, given that the maximum length of buildings in this area is approximately 33 meters, we set up a grid of 40 meters to ensure that the entire grid area is not dominated by buildings, preventing the identification of incorrect lowest points. Finally we get 15*15 (600/40 * 600/40) lowest points. It means that no cell in our grid has completely no data.

After we have finished finding the lowest point of all grids, we will use startinpy to connect the lowest points into a Delaunay TIN.

2.2 Insert additional ground points

We use startinpy.

Then check whether every point that is not the lowest point is a ground point. First, find out whether the point is in the TIN. If it is in the TIN, find out which triangle in the TIN the point falls in. Calculate the d value and alpha value between the point and the triangle plane in the space. If the distance d between the point and the lowest point and the maximum angle alpha are less than threshold, then the point is a ground point.

Setting the threshold is a rather challenging task. Both the code and the image below represent the results of 0.25 thinning. However, initially, we applied the same process to 0.1 thinning, and they both exhibit similar patterns. The threshold for 0.25 thinning was defined based on the results obtained from 0.1 thinning.

During the implementation process, we preserved all data(coordinate, distance, alpha...), which may lead to longer computation times. However, the distribution of distance and alpha data can be observed from the statistical charts (see Figure 2). Initially, we notice a steep decline in distance between 0 and 3, and in alpha between 0 and 5. Let's take a closer look by zooming in (see Figure 3). The distance sharply decreases around 700,000, while alpha is approximately 300,000. Therefore, the estimated point count should be around 700,000 to 800,000.

Next, by identifying the appropriate thresholds through sorting the distance and alpha values in ascending order, we aimed to find a balanced approach. Initially, referencing AHN's own classification (class 2 as ground points) would result in 1,516,921 points. However, when sorting distance and alpha in ascending order and retrieving the values at the 1,516,921st position, we obtained improbable thresholds: distance = 9.722143221605222, alpha = 76.05652640068469. These values indicate an unrealistic scenario where the point suddenly elevates by nine meters but is still considered ground. Therefore, we considered using half of the points from class 2 as a more reasonable approach.

Finally, we applied the same process to the results of 0.25 thinning. Note: The workflow described above used 0.25 as an example, and the actual point count for 0.1 thinning may differ.

In summary, our process for finding thresholds is:
1. Create bar charts for distance and alpha.

- Identify significant drops in values.

2. Sort distance and alpha in ascending order.

- Return the value at the position of "Number of points in AHN class 2 ground."

3. Compare the two approaches and determine that halving the number of points in AHN class 2 ground is a more suitable threshold.
4. Apply the threshold obtained from 0.1 thinning to 0.25 thinning and compare the results using the same process.

2.3 Main issues and solutions for Ground Filtering

The runtime is the primary challenge in this aspect. Initially, we used scipy but discovered it lacked the capability to insert one point and locally update the TIN, forcing us to redo Delaunay triangulation for each insertion, which proved to be time-consuming. Later, we switched to startinpy and optimized parts of the code, including the calculation of alpha and distance, eliminating the need for redundant plane calculations.

The second challenge lies in finding the threshold. Setting a meaningful threshold is a challenging task. Since the algorithm is a greedy one, different insertion points can lead to different results. Therefore, if the threshold needs to be reset, it requires a complete rerun, adding to the struggle with both space and time considerations.

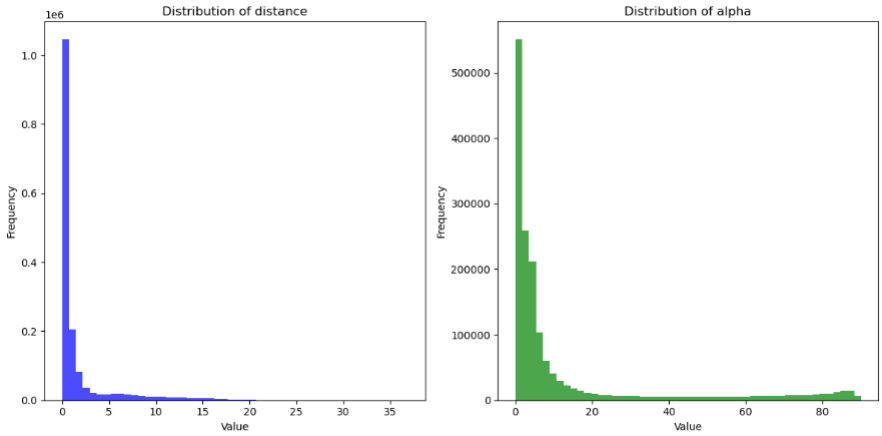

Figure 2: Distribution of distance and alpha data from the statistical chart

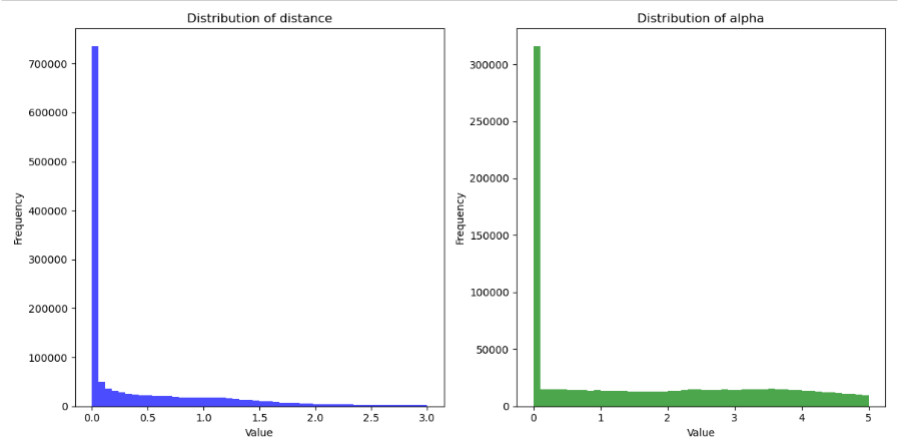

Figure 3: Distribution of distance and alpha data from the statistical chart

	distance	alpha	Note
0.1 thining	0.1565	1.4878	point in half of class2, and will be use as a threshold in 0.25
0.25 thining	0.14402	3.5372	point in half of class2, just for compare.
Conclusion: We can set the larger threshold			

Table 2: Threshold comparison

3 Laplace Interpolation

In this part, we use the ground points extracted from the previous Ground Filtering algorithm to interpolate a grid with resolution of 50cm.

3.1 Grid Generation

As described before, we clip the dataset with the size of 600m*600m to make sure every grid cell in the DTM grid would be interpolated correctly. The bounding box of 500m*500m dataset are applied to generate a grid: [000000, 000000, 000000, 000000] (indicating min_x, max_x, min_y, max_y). Since we also know the resolution is 50cm, there would be 1000*1000 grid cells in total. Center of each cell is calculated to be interpolated.

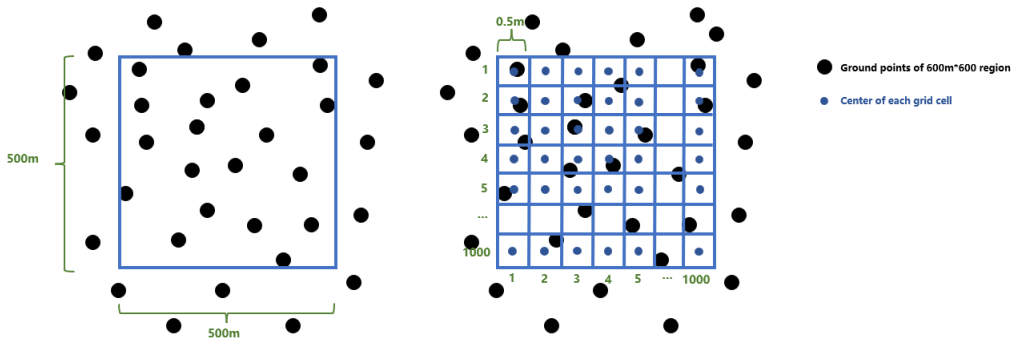

Figure 4: Grid generation

3.2 Interpolation Implementation

To implement the Laplace interpolation, we follow the algorithm introduced in the terrain book. [1] The ground points are used to generate a Delaunay TIN, the convex hull of which surrounding the centers of the grid cells entirely. It is to make sure that every grid cell would get a proper Laplace interpolation result.

The process mainly includes TIN generation, weight calculation(calculation of Delaunay edge and calculation of the Voronoi dual edge) and interpolation value calculation. As Figure3 illustrates, for each grid cell center, we insert it into TIN to obatain its adjacent vertices and incident triangles. Then for each adjacent vertex, we calculate the distance of this vertex to the grid center, and we find the two triangles containing this vertex to calculate the circumcircle centers, thus we could obtain the Voronoi dual edge. Since we acquire both the Delaunay edge and the Voronoi dual edge, interpolation value could be calculated with the equation provided by the terrain book.

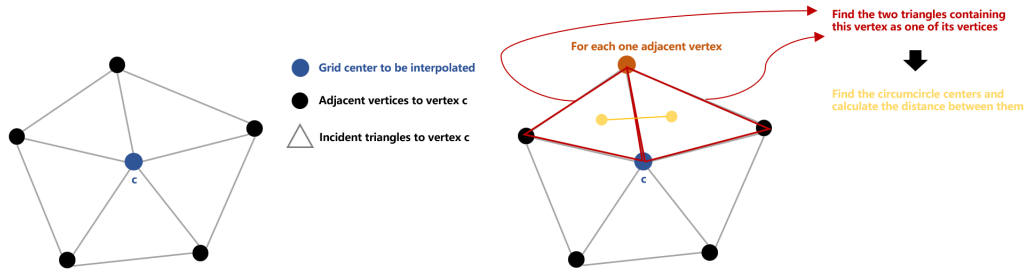

Figure 5: Main implementation process

Also, in special case that when cell center is exactly equal to one of the sample points, the value of that sample point will be returned.

4 Vegetation Extract and Create Grid

In this part, we extract the vegetation points with three steps of filtering, including classification, number of returns, and NDVI. The input dataset is the selected 500m*500m LAZ file after outlier extraction, and the output dataset is an raster data with 0.5m resolution.

4.1 Filtering with Classification

Since we are allowed to use the AHN4 classification in this step, we firstly look up to which classifications we have in the selected area with following code and its output:

```
print(np.unique(las.classification))
#output :[ 1  2  6  9  26]
```

According to the AHN4 classfication code[2], the dataset is consisted of unclassified(1),ground(2), building(6),water(9),and civil structure(26). The unclassified classification contained the vegetation as well as other other objects such as street furniture.So it needs the further processing.In our dataset. Also, we need to calculate the height of ground if there is no vegetation, so we selected both the class 1 and 2 for the next step. All selected points are split by 50cm*50cm grids.

4.2 Filtering with Number of Returns

When the laser pulses goes down through the vegetation, it would have multiple returns. Thus, the number of returns attribute can be used to distinguish the vegetation and the objects that can't be penetrated by laser. We just select the points with more than one return of laser pulse as the potential vegetation points.

4.3 Filtering with NDVI and Output

The AHN4 data is an laz file of format 8.The RGB and Near Infrared parameters are included for each point[3]. Normalized Difference Vegetation Index (NDVI) is a widely-used metric

for detecting and assessing the vegetation with remote sensing data, which is calculated by the following formula:

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

We calculate the average NIR and RED for each 50cm*50cm grid, then calculate the NDVI for this grid. If the NDVI value is higher than 0.2[4], we would consider that this grid has vegetation and get the max z value of points in this grid. Otherwise, the grid will be considered as non-vegetation grid and assigned with the average height value of the ground points in the grid. Since there are about 10-14 points per m^2 in AHN4 data[2], the 50cm * 50cm area usually has 2-4 points. So we don't encounter problems in NA pixels in the point cloud as mentioned in the lidR documentation[5]. All the extracted z values of a grid are stored by a list and reshaped to a matrix to input to raster.

4.4 Other Attempts

Inspired by the Assignment 4 in Sensing Technologies (Geo1001), we also tried to apply Density-based spatial clustering of applications with noise (DBSCAN) to detect the vegetation points. DBSCAN is a density-based clustering non-parametric algorithm which groups together points that are closely packed together, which can also help to detect the vegetation, especially the trees. However, there are multiple types of vegetation including both bushes and trees, it will be hard to find suitable eps and min-point parameters that can work for all kinds of vegetation. So it is not very effective to apply DBSCAN here.

5 Results and Discussion

5.1 DTM

We get the result of DTM at 50cm resolution and plot it both in 2D (Figure6) and 3D (Figure7). From the observed data, the elevation values range from approximately 109 to 144 meters with no abrupt changes in elevation.

5.2 Vegetation DSM

In step4, we get the DSM of vegetation. Below is the overview for the result in QGIS with an OSM basemap (Figure8). We can see there are some grids formed in belts or clusters are higher than their surrounding area, which indicates there would be some vegetation.

5.3 CHM

Comparing with vegetation DSM we get in step4, the Canopy Height Model (CHM) focuses on the height of vegetation and the influence of terrain is removed. The grids with vegetation height lower than 5cm are set translucent int the map (Figure9).There are **800111** grids having vegetation out of the total 1000000 grids, which means there is approximately 80% of land covering with vegetation. The average height of vegetation in this area is **1.76** m and the median height is **0.12** m. The Figure10 shows the frequency of the vegetation height with histogram.

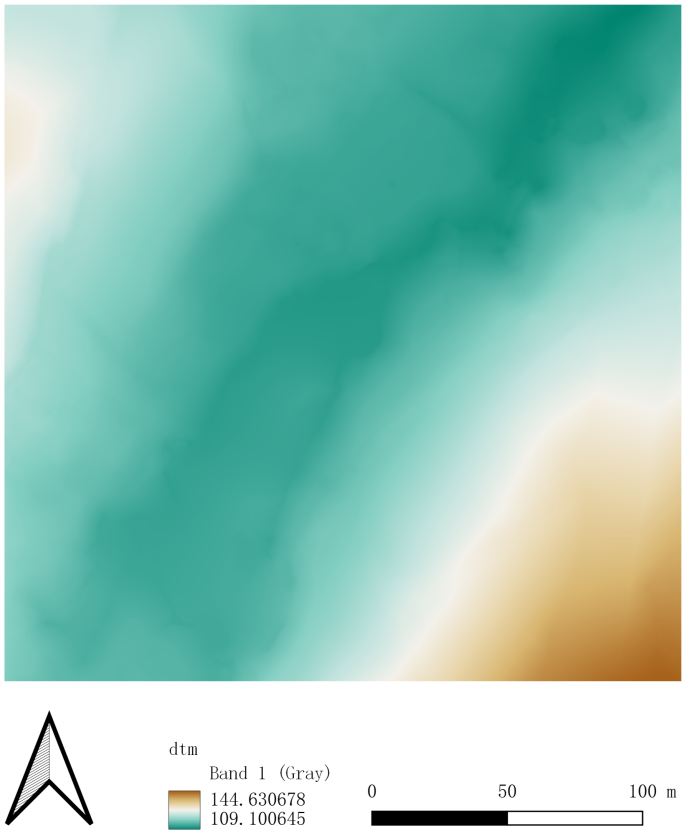

Figure 6: 2D overview of DTM

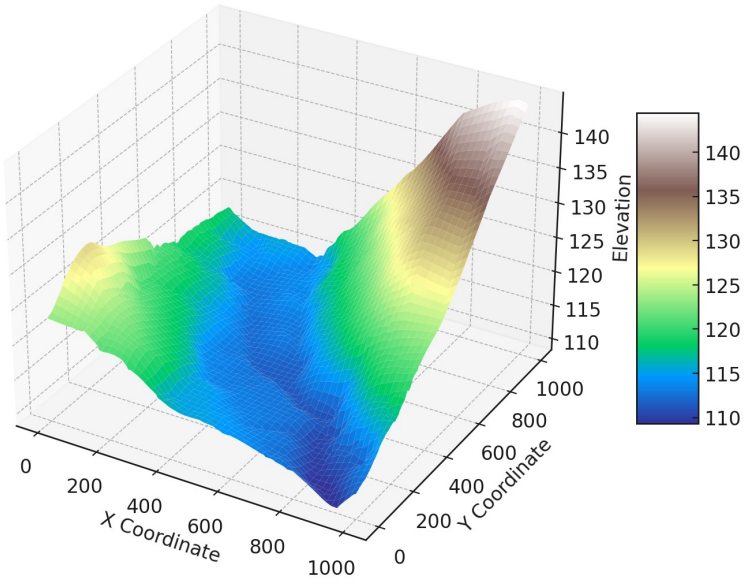

Figure 7: 3D visualization of DTM

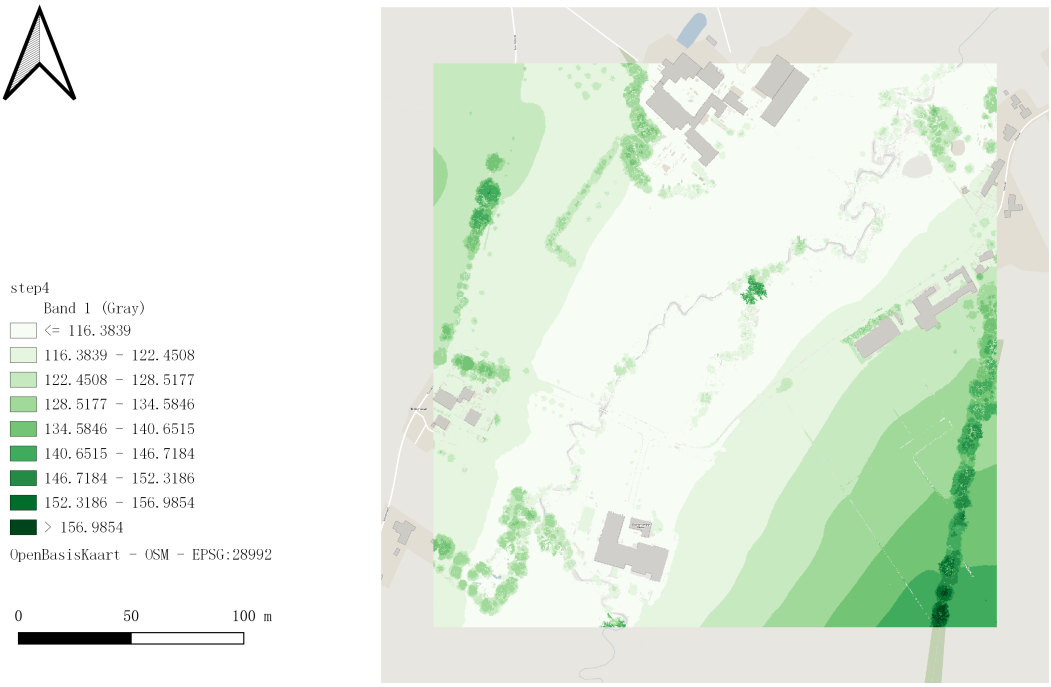

Figure 8: Vegetation DSM (step4.tiff)

Figure 9: Canopy Height Model (CHM)

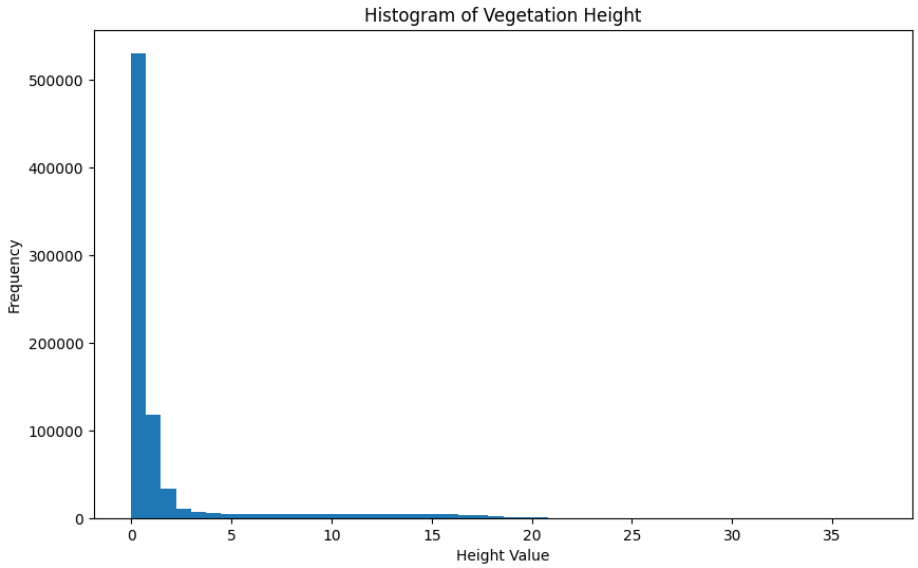

Figure 10: Frequency of Vegetation Height

6 Declaration for Workload

We work parallel for ground filtering, interpolation, and vegetation extraction. We also discuss together with all the problems and bugs we encountered during the whole assignment. Below is the work for each group member that is responsible for.

HsinYu Cheng mainly responsible for pre-processing, including finding suitable areas, determining boundaries, crop and thining, and outlier removing. She was also responsible for extracting the ground points and writing the first and second chapters of the report.
Yan Gao mainly works on implementing Laplace interpolation in Step3 and produce the dtm.tiff result. For the report part, she is responsible for the Laplace Interpolation part.
Xiaoluo Gong is mainly working on extracting the vegetation (Step4), make the canopy height model (step5) and writes the main function to integrate the code.For the report, she writes the Vegetation Extract parts and mapped all the images in the part5 (Discussion and Result) with QGIS and Python. Also, she organized the work flow and managed the GitHub.

References

[1] H. Ledoux, K. Arroyo Ohori, R. Peters, and M. Pronk, *Computational Modelling of Terrains*, 2023.

[2] Actueel Hoogtebestand Nederland, "Kwaliteitsbeschrijving," Accessed 2024, online; accessed January 12, 2024. [Online]. Available: https://www.ahn.nl/kwaliteitsbeschrijving

[3] L. D. Team, "What is a las file? - point format 8," https://laspy.readthedocs.io/en/latest/intro.html#point-format-8, 2024, accessed: 2024-01-12.

[4] H. Hashim, Z. Abd Latif, and N. A. Adnan, "Urban vegetation classification with ndvi threshold value method with very high resolution (vhr) pleiades imagery," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 237–240, 2019.

[5] J.-R. Roussel *et al.*, "Digital surface model and canopy height model - the lidr package," https://r-lidar.github.io/lidRbook/chm.html#p2r, 2023, accessed: 2024-01-15.