Report on the use of Matlab in analysing LiDAR data

Benjamin N. Vis
Research Ma student, Leiden University

Introduction

This report is on the work of a layman (author) trying to get some basic understanding of the general characteristics of LiDAR (Light Detection And Ranging) altimetry data and various simple actions one needs to pass through for the making of visualisations and analyses. LiDAR is a laser based sensor that produces quite accurate distance information. When mounted on an airplane the laser makes sweeps over the landscape sending out multiple pulses at a high frequency. A receiver is also carried, which can catch the echoed laser pulses and by calculating the time the laser beam travelled back and forth, a point of contact with the soil (where it reflected) can be computed with great accuracy. The place where the points hits the surface is controlled by a terrestrial GPS station. Therefore the data it produces is also georeferenced to a global or national coordinate system, depending on the company and political area. This method can provide users with a geographical record of large parts of the landscape while achieving high point densities. Depending on the footprint, the size of the laser beam when it hits the canopy or surface, the laser is able to penetrate through vegetation. The footprint is usually between 10-30cm wide, though smaller and bigger footprints do occur, depending on the laser and the height of flight (Straatsma 2007). The more pulses are used, the bigger is the chance that some of them could penetrate through the vegetation and the better surface data you might get. Still if the vegetation is too dense (usually multi-layered vegetation) or the footprint too large, this is not possible. Its characteristics make LiDAR a fast and efficient detector for small geographical anomalies that are important in archaeological cases and shows the best detail in landscape surfaces so far.
My background in archaeology did not provide for the essential technical base from which to work with such advanced techniques and the necessary knowledge of data visualisation computing. Nevertheless the successful examples of the use of LiDAR data for archaeological site detection and mapping in various parts of Europe call for archaeologists to approach technicians for collaboration. In order to make this work effective at least some basic knowledge of the technical data and methods is needed from the archaeologists side. This text deals with the experience of familiarising myself with such data and related computing methods using the mathematical programme Matlab. I would like to stress that is in no way representational for the successful work that has been done previously, but that this should act as a guide for other laymen who would like to pursue a similar attempt.

Software and Data

Firstly the programme I have used throughout this report is Matlab, which is a very rudimentary mathematical programme leaning on thorough knowledge of programming of the user. Usually more advanced, fully programmed and designed programmes are used that are undoubtedly easier for the end user, but lack the possibility to get acquainted with the data in raw state, or to design customised analysing tools specific for your dataset (e.g. Erdas Imagine, ArcView/GIS, Surfer, MapInfo). The choice for this programme resulted from the objective of truly getting familiarised with the characteristics of the data and the possibility of keeping a trial and error basis, which demanded that the user would gain a good deal of understanding to enable him to correct his or her wrongs.
The dataset is derived from the educational copy (not free to publish in any way) of the AHN data (Actueel Hoogtebestand Nederland, Actual Height Model of the Netherlands) of Leiden

University. A topographically challenging area was selected situated north-west from Apeldoorn, near the Palace ‘t Loo. This is a heavily forested area with for Dutch understandings a quite hilly topography (max. height about 80m). It is suspected that several prehistoric burial mounds lay in that forest on the hills. Hypothetically with LiDAR data of sufficient quality these should be possible to detect and locate, provided that the forest is not too dense to penetrate. In the Netherlands in topographically flatter regions already quite successful attempts have been made in both finding burial mounds or Celtic fields, and following Paleo-channels (De Boer in press, Zijverden 2005) also using Matlab as filtering software (Humme 2006). My selection initially covered ‘Grote Topografische Atlas van Nederland’ pages 72 and 73 (Topografische Dienst Nederland 1987), which corresponds with an area of 20 by 13 km, North-West of Apeldoorn. This, however, is already way too large to work with and it is advisable to start dividing up the dataset in 1 km^2 parts before using it at all. Most standard computers would have difficulties handling datasets that are larger than this at the time. If you are not confident to start right away with analysing the end product it is advisable to select a very small subset to play with first.

When starting to work in Matlab two things are most important to understand: firstly Matlab works entirely on the basis of matrices, secondly although the basic interface provides a command line where one can type in all their commands directly, it is advisable to save successful commands in so-called m-files. These m-files can contain various sequential commands that you can use on a specified, loaded dataset, and as such one can make a m-file defining a subset from a loaded dataset. This way the entire set does not have to be used for all commands and several filters or visualisation commands can separately be stored in m-files. It is good to know that Matlab has a good ‘Help’ with search option, while many additional user manuals can be found all over the internet, by simply using specific queries in a search engine whenever you come across a problem (good help websites: The Mathworks, Matlab Help, and Matlab Help Desk). In order to have your datasets and m-files at your disposal in Matlab, select the folder you have stored them in at the top of the command window.

The LiDAR dataset simply exists of points (point clouds) that have an x, y, and z component (3D coordinates), where the x, y simply refer to the Dutch national coordinate system, RD (product description at www.ahn.nl), and z refers to the altitude in metres according to NAP (product description at www.ahn.nl, for the definition of RD and NAP and their relations see the website of the Netherlands Geodetic Commission). These are composed in .xyz files. With this information it is rather easy to divide your dataset in parts following squares or coordinates on maps in official atlases. Please note that there are several products available of the AHN data, of which the ground dataset and the canopy dataset are most commonly used. These are the raw data separating points referring to the height of the vegetation as opposed to measurements taken of the ground surface. It could be, however, that a professional would like to manually separate canopy and ground surface data using his own method, which might result in more detailed and accurate data.

AHN itself indicates that their qualitative standard for their geographic data is a point density of at least 1 point per 16m2 or 1 per 36m2 in heavily forested areas. In the test case that is discussed here, the area is covered with an average point density of 2-3 points per 16m2. This is actually pretty low for archaeological purposes, but some areas have considerable greater point densities. The accuracy of the altimetry is usually about 5 cm off the actual heights, depending on the control stations and area, with a standard deviation of 15cm. The complete specifications of AHN data can be found online on the AHN webpage (http://www.ahn.nl/productspecificatie.php), where an extensive description can be downloaded.

Getting started, selecting a subset
The datasets and variables used in the following are meant as a hypothetical example and therefore do not include answers or outcomes on all occasions. The sequential command lines can simply be followed with any LiDAR dataset. Before a dataset can be used in Matlab, it needs to be loaded. This is simply done with the command 'load', e.g.:

```
ahn = load('ahn.xyz');
```

Pay attention to put the ';' at the end of your command lines. This will make that the answer is not directly shown on screen, but the command is still being executed. Especially when a command, such as loading the entire dataset, has quite extensive calculations with a gamut of answers to be shown, it will take extremely long to get it on screen, while there is no actual use of those numbers. Only if you truly want to inspect the answer you could choose not to put it down.
Working with the AHN (x, y, z) data becomes easier when you first define the columns of the matrix as x, y, and z, so you can use these as variables throughout all the commands that follow, which will diminish confusion. The (:,1) refers to the first column, (:,2) to the second, and so forth.

```
x = ahn(:,1);
y = ahn(:,2);
z = ahn(:,3);
```

Once you have done this for the entire dataset it is the next step to define small subsets of this data that are of a workable size. An easy command that you can use at all times is 'size', this will give you the size of the matrix you are currently working with, and in this case as such the number of points in your dataset. The commands 'max' and 'min' give you the maximal and minimal value contained in the matrix of your choice. So if you want to have an idea of the expand of the data you have it is good to know the extremes of both the x and y (for some purposes also z) before you make a subset selection. It is advisable to note down answers to the 'min' and 'max's of your dataset.

```
maxx = max(x)
minx = min(x)
maxy = max(y)
miny = min(y)
maxz = max(z)
minz = min(z)
```

Of course if you already know the extremes of your dataset you can use that knowledge to divide it up more practically, or along the lines of coordinate divisions. In this case I have used the 1 km^2 blocks that are indicated in the Grote Topografische Atlas van Nederland. It is easiest done by using the 'find' command. The example below will get you a square of exactly 1000 units per side, in this case equivalent to metres.

```
F1 = find(ahn(:,1)<194000);
B = ahn(F1,:);
F2 = find(B(:,1)>=193000);
C = B(F2,:);
F3 = find(C(:,2)<=473000);
```

```
D = C(F3,:);
F4 = find(D(:,2)>472000);
ahn33bn14=D(F4,:);
size(ahn33bn14)
```

This can be repeated for all other squares that can be selected from the original dataset, the numbers indicating their location. For your own organisation it is most comprehensive to store all these selections in separated m-files with names that are indicative for their location or the part of data they contain. Please note that Matlab does not recognise names of m-files or variables starting with numbers. Also take care not to define variables with the same name as m-files (e.g. in my case I defined the m-files as subset* and the variables containing the subset matrices as ahn*).
The matrix that contains the selection is now defined as ahn33bn14. In order to work with this selection it is advisable to repeat the definition of the basic variables for the selection:

```
x = ahn33bn14(:,1);
y = ahn33bn14(:,2);
z = ahn33bn14(:,3);
maxx = max(x);
minx = min(x);
maxy = max(y);
miny = min(y);
maxz = max(z);
minz = min(z);
```

In many cases you would like a preliminary inspection of your data. The easiest is to make a quick 3D visualisation of the subset by using a scatter plot. This simply visualises the point cloud in a 3D (x, y, z) axel system.

```
scatter3(x,y,z,1,z)
```

This makes a scatter plot on the basis of x, y, and z with a plotted point size of 1 pixel, following a default colour scale dependent on the value of z. The last two can be changed at will, manipulating the visualisation into something potentially more readable. Figure 1 shows a scatterplot with 3 pixel points (for clearer view) of a small test area. It shows the empty spaces, where no readings were taken and the overall quality of the data. For the birds eye view, use the command view(2), view(3) refers to the 3D view that is set as default.

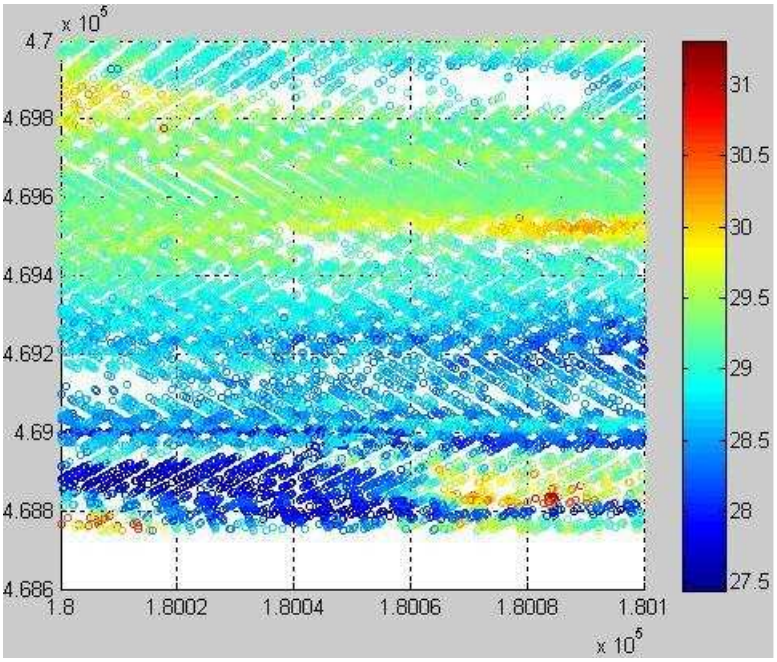

Fig. 1: Scatter plot in birds eye view of a small test area, within my subset ahn33bn01

The visualisation will appear in a pop-up screen called 'figure 1'. You can also open figure screens at will by simply entering 'figure(1)' or any other number. Also putting 'figure,' in front of a visualisation command will add this visualisation in a new figure screen.

Gridding

Most other visualisations ask for your data to be organised in a grid first. Purists, especially in archaeology, would argue to not do that when it is not really needed, as this rearranges your data slightly and thus will cause small deformations in the detected anomalies. In practice it is easier to do this at an early stage, so your data is more readily applicable to filter and visualisation commands.

Before the data is ready to be gridded, the matrix needs to be equally spaced. We need to have our data organised in rows, very much like a ruler, in steps of one by one. The commandlines below show how this is done between the extremes of x and y.

```
rijx = minx:1:maxx;
rijy = miny:1:maxy;
```

Sequentially these equally spaced rows can be meshed into a grid.

```
[XI,YI] = meshgrid(rijx,rijy);
ZI = griddata(x,y,z,XI,YI);
```

Usually with irregular data like the AHN this will result in the warning that certain point locations are now duplicated. This simply means that certain points were so close together that they are now at the same position in the regular grid. This has no significant effect, though it is indicative for the slight loss of data accuracy when gridding.

Once the data is gridded it is open for various other visualisations and filters. Most basic is the mesh-grid, which visualises the data like a plot of wired points.

```
mesh(XI,YI,ZI), hold;
plot3(x,y,z,'o'), hold off;
```

What a meshgrid looks like is shown by figure 2 (below), which shows a visualisation of the unprocessed data of a topographical subset. It regards an example of AHN data of which the vegetation is already filtered out by the data provider. This means it should show us the heights on the surface, produced by the so-called second pulse data.

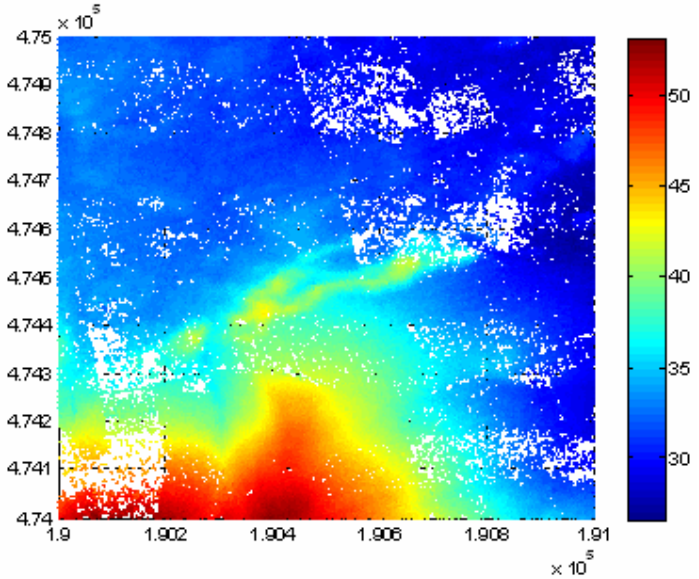

Fig. 2: this is a plot of subset 33bn03, with colour bar, raw data

When working with altimetry data it might be useful to know how to produce a contour plot from the gridded data. The number in the end, again refers to the thickness of the lines. Figure 3 gives an example of a contour plot, since the dataset used here was small due to selection for a test case, the contour lines are hard to interpret.

```
figure(2)
mesh(XI,YI,ZI);
contour(XI,YI,ZI,5);
```

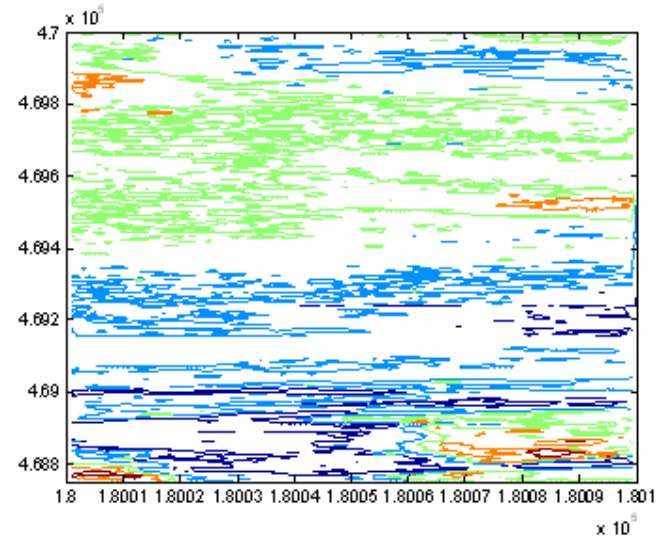

Fig. 3: Contour plot of a small dataset, from dark blue (low) to orange (high)

Although the data is equally spaced when gridded, in the visualisation the figure window is usually composed as a square, while actually your data might spatially be organised differently. This is easily resolved by adding the command 'axis equal', this sets the axes to their true lengths, according to your data. Therefore no visual distortions will appear and the data becomes interpretable and comparable to other sources, such as maps or photographs. Something that is not usually bad, but should be noticed while doing this, is that the z axis is also equalised in this fashion. When you desire to inspect your data from above, instead of 3D, this does not matter. Otherwise you should only use the command 'axis', which enables you to set the axes manually. You should calculate the proportion of the x to y and subtract them from each other. This will leave the z axis as it was originally.

Exporting a figure

Actually this is fairly easy, but depending on the use you have for the figure, many options can be chosen. Through the menu on the top of the Matlab window you can search for 'preferences', 'figure copy template', 'copy options'. Here you can define how you want your figures to be exported, your preferences according to the software you would like it to use in, can be set as default here. Next you locate 'copy figure' in the window of the figure you would like to copy and this leaves you to paste it from the clipboard into the software of your choice. Since many of us would specifically like our images in texts or presentations it is useful to know that the figure can also be directly saved to your disk. When opting for 'save as' in the figure window, you can set the extension (usually .jpeg or .bmp) you would like to image to be saved in and next open or import it in the software of your choice. Using Photoshop or GIMP it is easy to reduce image size when necessary, as Matlab usually exports is figures in the same way they are built up in the programme, which makes them difficult to handle for programmes like Word.

Histograms
Histograms are the most easy tool to inspect your data. It shows you the frequency of data points along any one of the axes, although for most applications you will only be interested in the dispersion of the altimetric values, so z or an equivalent variable. There are two ways to find histograms:
Use the 'Simulink' button in the menu on top, go to 'signal processing blockset', then 'statistics', then 'histogram'. Here you can choose to set many preferences. The easier way, which will give you a simple histogram is using the command 'hist'.

```
hist(z,20)
```

This will give you the frequencies of the z values, divided into 20 beams. It shows you what are the most occurring values, which is normally most important, but at the same time you can also spot strange values at the extremes of the scale, that usually only occur once or twice and distort the relative visualisation. Using the 'find' command guided by the observations in the histogram can produce more detailed visualisations.

Basic filtering
Before starting with filtering for different purposes you should have an idea of what the raw data looks like, therefore always make and save a mesh plot of the raw data before you start filtering.
A colour bar is an unavoidable asset for visualisations as it shows what the colours mean in your plot. Usually the default colours are adequate to interpret your data, however, when needed other colour schemes can be located through help search. A colour bar is inserted with the command 'colorbar'.
In the following I will show how you can make a basic filtering on the basis of a median filter and manipulation of residuals. This should take care of the extreme z values distorting visualisations of surface data as well as emphasise subtle anomalies in height in contrast to the general shape of the surface. The methods used here are very rudimentary and Matlab offers many more possibilities that you might want to try. Also you are free to mathematically programme your own, but probably expert advise is needed to see that through. For preliminary filtering methods the below should be sufficient and on the basis of this, the filtering methods can be elaborated as well.
The median filter simply flattens out the raw data along the topography in two dimensions: x and y, while taking the average of the z values in the window it takes. The window that you put on x and y can be adjusted in size, but the command also has a default size of 3 by 3. The command used for the median filter is 'medfilt2'. This filter automatically takes into account the differences in height that may make other filters complicated to apply. Shown below is the sequence of meshing the data and taking the median.

```
[XI,YI] = meshgrid(rijx,rijy);
ZI = griddata(x,y,z,XI,YI);
filtstukje=medfilt2( ZI );
```

When you have taken the median filter, the flattened data is stored in 'filtstukje' as shown above. This, however, is not enough to enhance a visualisation of small features on the topography. As it took an average of the topography only the large trends remain in the median. The next step is to produce residuals of the topography, which would contain all the smaller features that lay on top of the topography, but are now filtered out. These can be produced by subtracting the median filtered data from the raw data.

```
residuals = ZI-filtstukje;
figure, mesh(XI,YI,residuals)
colorbar
view(2)
axis equal
```

One expects the mesh to show relatively small differences in heights and a zigzagging of relatively equal highs and lows throughout the entire dataset. The data can further be inspected with a histogram, which should ideally show a normal distribution around the zero.

```
figure, hist(residuals,20)
```

If this is not the case a selection must be made using the information from the histogram by means of the find command. You are looking to exclude positions along the x and y axes with a z value outside the selection you wish to make with the find command. The command therefore should ask for values greater or smaller than the arbitrarily set boundary of the selection you wish to make:

```
[r c]=find(residuals>5)
```

The r and c refer to row and column, and together they give a two dimensional position. This command will give the answer:

```
r
        …(answer)
        …(answer)
c
        …(answer)
        …(answer)
etc.
```

The first number is referred to as position 1, the second as position 2, and so forth. Now to remove the distortion to the visualisation caused by these positions, a ‘zero’ value must be assigned to them, getting them right in the middle of the normal distribution. Although it does not remove the points, it makes that they do no longer distort the plot.

```
residuals(r(1),c(1))=0;
residuals(r(2),c(2))=0;
etc.
```

The only flaw in this system is that it is not automated, so you must type in all these commands individually, for all the positions given by the row and column answer. Copy-paste might help you out for several tens of positions, but when the list becomes very large something else needs to be designed and this is where the boundary of this method lies. When you do this the histogram should show an increasingly smaller window and a better normal distribution. I have achieved to get the window down to a size of around -0.6 to 0.6m, which is actually still too large for the really small features and differences in heights archaeologists are looking for.

As an alternative for this rather tedious and limited approach there is an alternative, which is based on slightly more complicated mathematics, though the commands are still easy to use. After you have produced the preliminary histogram you decide the boundaries of the window size around '0' you would like to achieve. This could be basically any value in the range of the histogram. On the basis of your decision you can ask Matlab to make a decision matrix based on absolute values (the negatives are inverted to positives) of the residuals. In the command below the boundaries were set at 0.1 around 0. It reduces all the points with values of more then 0.1 to 0.

```
matD = abs(residuals)<0.1;
```

Then this matrix needs to be multiplied by the matrix of the residuals, which will leave you with only the points with values within the window around 0.

```
data_w = matD.*residuals
figure, mesh(XI,YI,data_w)
view(2)
axis equal
colorbar
```

After either one of the above produced a satisfying histogram, the mesh plot may still lack some contrast to make the small features visible. In that case you can manipulate the colour bar.

```
figure, mesh(XI,YI,residuals)
colorbar
figure(1)
hold on
caxis([-0.2 0.2])
```

This command will make that all values smaller than –0.2 and bigger than 0.2 will get the same colour of –0.2 and on the other side 0.2. This increases the contrast exactly from these points and of course playing with this manipulation might show different features depending on the value you enter.

Tips for working with heavy visualisations

The command 'whos' will show you a list of all the matrices that are currently loaded in Matlab. If your machine is slowing due to the heavy visualisations you are making it is advisable to shut the large matrices you no longer need for the calculation of the visualisation. After the list appears the command 'clear' can be given, followed by the name of the matrix you want to remove from the list.

The more advanced user can also choose to make a random subset of points selected from the total of points that needed to be visualised, based on vectors. These vectors should be formed out of the original point cloud before. This makes calculating the figure a lot faster, but of course eventually diminishes the quality of the picture. The procedure, however, can be quite complicated as first vectors need to be made and then a random amount of points must be selected. Use the help files to guide you through this and design appropriate m-files.

If there still are severe problems with the figures and exporting does not work any longer, the simplest solution is using 'printscreen' and paste the figure in a graphic programme. Nevertheless this can never achieve the visual quality of exporting and saving correctly.

Results and Findings

The following contains some of the visualisations that resulted from the case study near Apeldoorn. It concerned a topographically challenging area with heights ranging from about 10 to 80m above NAP. Moreover it is heavily forested and as already indicated by the scatterplot at the beginning of this report (Fig. 1) the data was not of a constantly high quality. The aim was to try and make a first attempt in visualising some suspected burial mounds and historical forts hidden beneath the canopy. The steps followed to make the visualisations basically concur with the stages of analysis described above, so they will not be repeated here. Instead some more detail will be given on the findings of the visualisations once produced, starting with a mesh grid visualisation of the raw data of subset 33bn01 (Fig. 4). Topographically this area of one square kilometre shows a hill to the lower left and covers a totally forested areas with some minor roads running across.

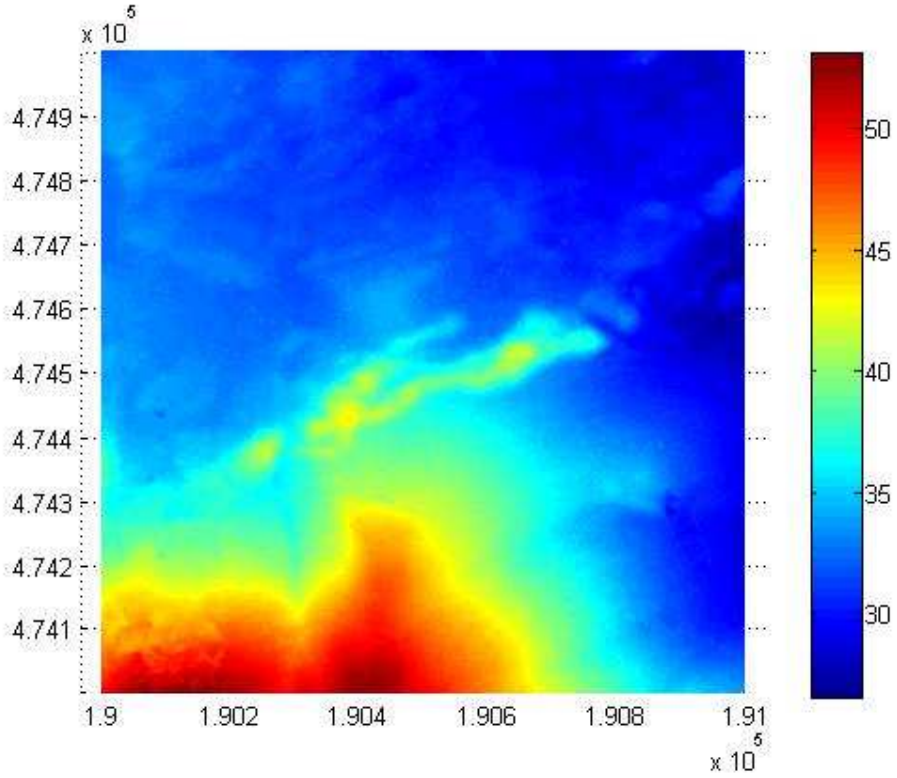

Fig. 4: Meshgrid plot of the raw data from subset 33bn01 with heights between 25 and 55m.

After the stages preceding calculation of the dataset of residuals, the spread of the z values of these residuals was investigated using a histogram. On the basis of this histogram a decision was made to assign the extreme z values the value ‘0’ to approach its tendency towards a normal distribution around the zero as shown in Fig. 5.

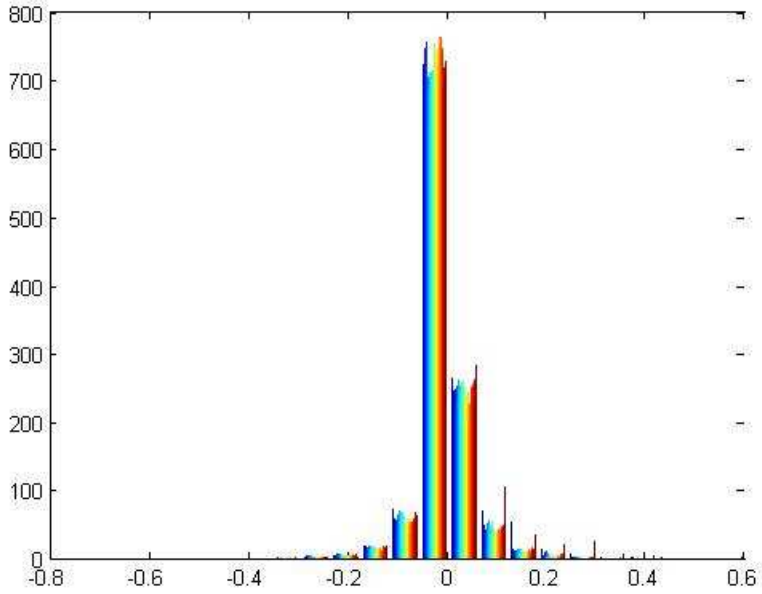

Fig. 5: Histogram of the filtered residuals of subset 33bn01 (from -0.6 to 0.6m).

These residuals in turn were visualised in the mesh plot below (Fig. 6). Normally one would expect the smaller topographical features (with a difference in height smaller than 60cm, the window achieved by the normal distribution in the histogram) to show up more strongly. Somehow they appear to be obscured by a striping effect, which cause is still a puzzle, but seems to concur with the concentration of measurements taken from the airplane in those areas and is also visible in the geographic distribution of the measurements (see Fig. 1), however this is unlikely to be the cause. The striping expected in all LiDAR data due to overlapping flight lines has already been taken care of by the AHN data provider in all their products (described in the product specification at www.ahn.nl). The results, thus are disappointing, since not even small hills and roads show, this could be because of the limitations of the filtering method or the poor data quality. Neither the method using a decision matrix to reduce the size of the window around '0' gave better results. It seemed only to have an effect like even stronger contrast enhancing. It was suspected that small burial mounds are situated under the canopy here.

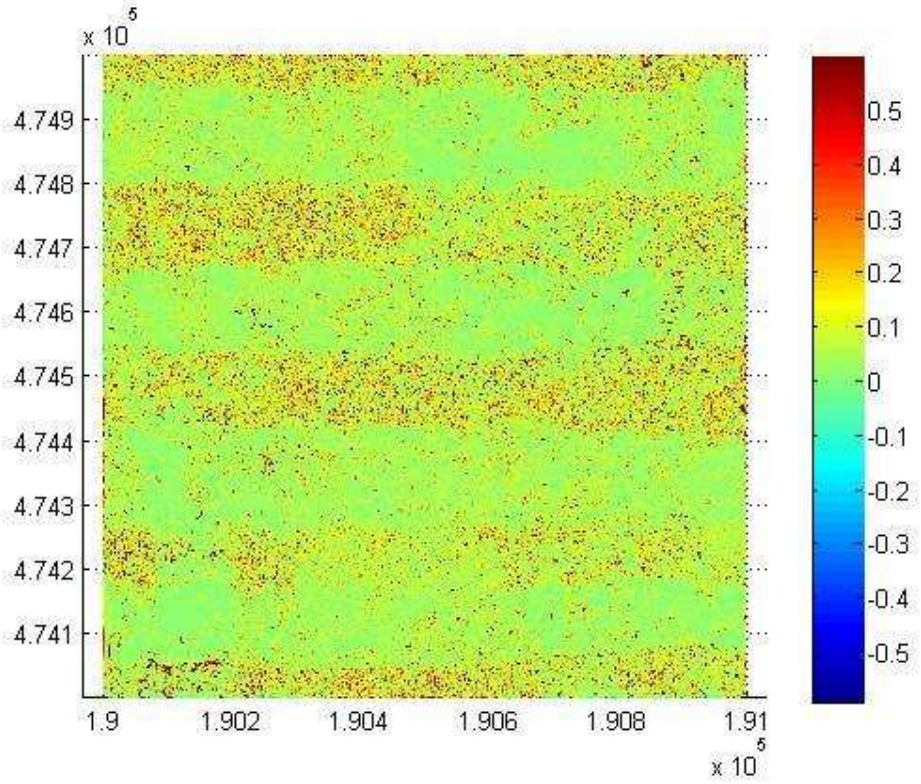

Fig. 6: Residuals of subset 33bn01 from z values of -0.6 to 0.6m with contrast enhancement from -0.2 downwards and 0.2 upwards.

Another square kilometre subset (33bn17) is shown as an example of the work done by this trial. We are still west of the city of Apeldoorn, and this time the area is topographically more diverse. The mesh grid plot of the raw data covers the forested area around Fort Java (Fig. 7). On the partially cleared hill in the lower left corner stands the Juliana Toren, and right above the is the parking lot Emma's Oord that lies in an area with fields. Right below is a small body of water as well.

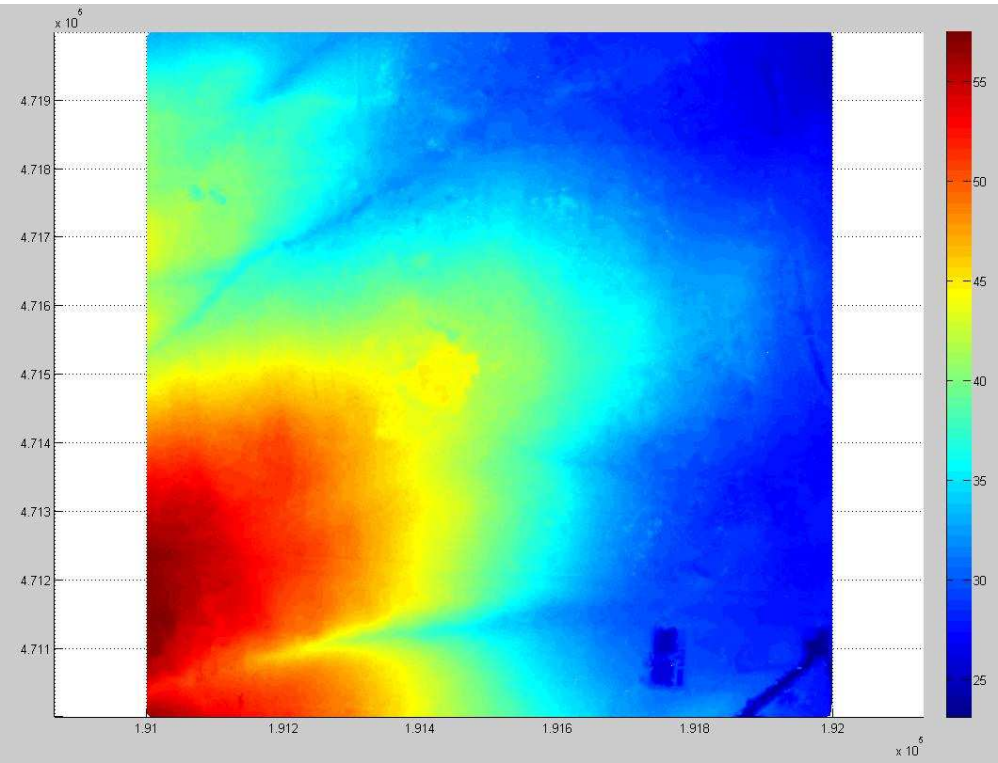

Fig. 7: Raw data of subset 33bn17.

After the same steps taken before the histogram of the residuals achieved, Fig. 8, was slightly more dispersed than the previous example. The window is rather large for secure filtering (60cm below zero to 70cm above), the limitation of the method of ascribing '0' values to the extreme z's.

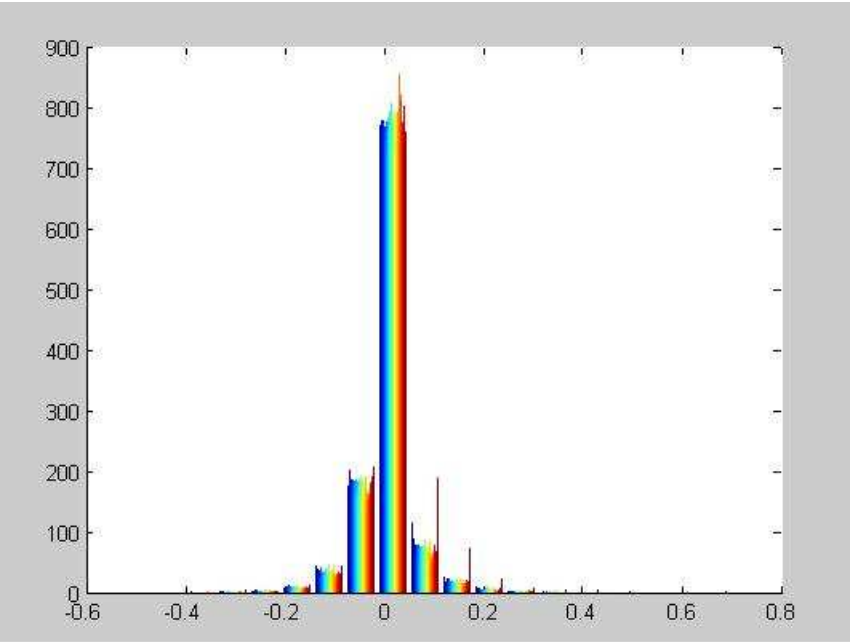

Fig. 8: Histogram of the filtered residuals of subset 33bn17 from -0.6 to 0.7m.

The mesh grid visualisation of the residuals (Fig. 9), though showing slightly more contrast and diversity, still produced similar disappointing results. The fort nor other subtle features are recognisable in this distorted plot. The good results that can be achieved with AHN data analysed with Matlab are beautifully exemplified by Humme (2006), who as a specialist in programming and mathematical analyses achieved great visualisations of Celtic fields using this rudimentary software.
Due to the disappointing results achieved here, Roderik Lindenbergh (lecturer at the TU Delft) developed a more sophisticated Matlab script that produced more promising results for the same dataset. The script uses a filtering method on the basis of a 'for loop' construction and the visualisations definitely show greater detail, however it remains unclear whether the archaeology is part of that. This script and some exemplary visualisations can be found in appendix I.

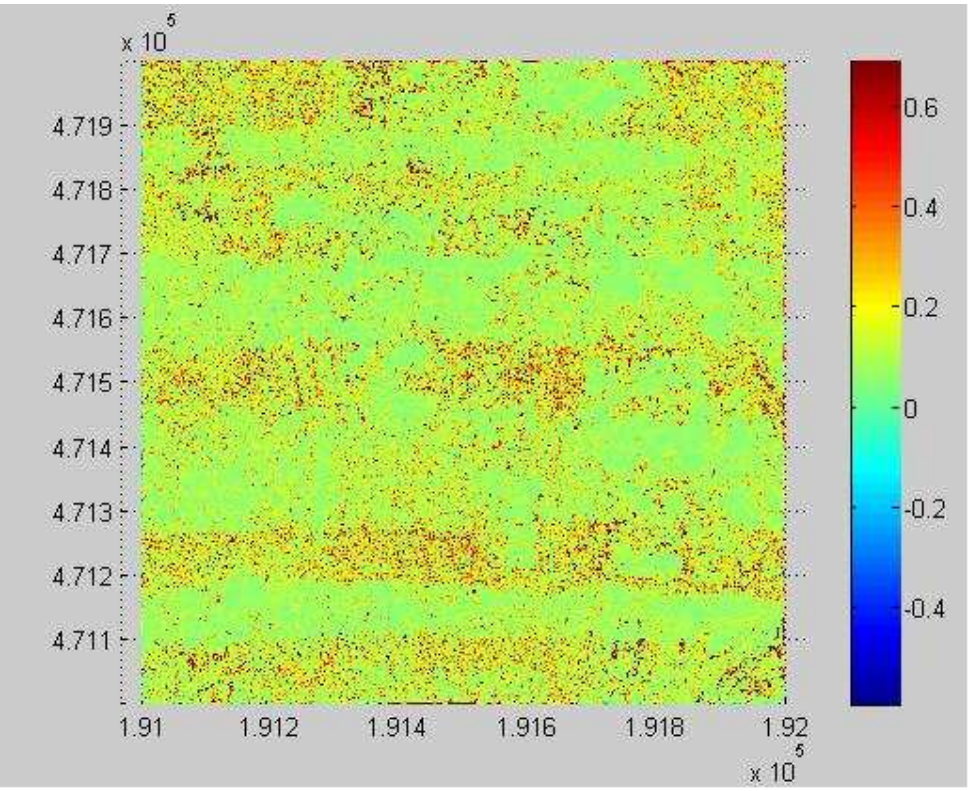

Fig. 9: Residuals of subset 33bn17 from -0.6 to 0.7m with contrast enhancement from -0.2 downwards and 0.2 upwards.

It should be taken as one of the lessons learned that collaboration in the end is essential for succeeding, but the succeeding of the collaboration depends on the willingness to learn each other's disciplinal language.

Concluding
It proved to be a useful method to start with the raw data and a programme based in mathematics to get acquainted with the type of data that LiDAR entails. The otherwise abstract point clouds with all its anomalies would have remained something elusive, if not for the necessity to truly grasp it spatial component and order. Only by going back to the essential characteristics of each measurement produced by the LiDAR, Matlab could be directed in such way that it organised the data towards the possibility of visualising the separated points

into something that reflects the actual situation on site. From that stage onwards slowly excursions could be exercised into the manipulation, enhancement and filtering of the quality of the data that suited the archaeological case. This approach really forced me, as an archaeologists, to understand to a definite degree the technical language needed to work with this data.
Although the eventual results were disappointing, the knowledge that others who are better equipped succeeded several times in visualising the burial mounds that I could not (De Boer in press, Zijverden 2005), it should not be forgotten that this was a first time venture into the technical unknown by a layman. The main objective as such was not to succeed in making the pretty pictures, but to gain a thorough understanding of the characteristics of the data and the stages of processing needed on a rudimentary mathematical level to visualise the raw product. Perhaps the rudimentary forms of filtering or the poor quality of the data restrained me from success where this was expected. In an ideal situation the dataset would have been far higher point density, but probably still the dense vegetation as well as the challenging topography made the area of the case study difficult to assess. It can be expected that in collaboration with mathematical or software specialists, far better results could have been achieved.
Finally I would like to stress that I think as a starters guide for users of LiDAR data and those using Matlab in an exploratory way, this report may help them to get on their way. It only lays the basis for further ventures into the more advanced programming of filters, visualisation, and manipulation methods of LiDAR data. Especially for the communication between students from different disciplinal backgrounds, it provides an informative guide.

References

De Boer, A.G., e.a., in press, Lidar-based surface height measurements: applications in archaeology, Technical report, British Archeology Reports International Series.

Humme, A., Lindenbergh, R., and Sueur, C., 2006, Revealing Celtic fields from LIDAR data using kriging based filtering, in: Proceedings ISPRS Commission V Symposium 2006, 'Image engineering and vision metrology', Dresden.

Straatsma, M., 2007, Hydrodynamic Roughness of Floodplain Vegetation: airborne parameterization and field validation, series: Nederlandse Geografische Studies, Koninklijk Nederlands Aardrijkskundig Genootschap, Utrecht, doctoral dissertation.

Topografische Dienst Nederland, 1987, Grote Topografische Atlas van Nederland, 1:50000, 3 Oost Nederland, Wolters Noordhoff Atlasprodukties, Groningen.

Zijverden, W.K. van, and Laan, W.N.H., 2005, Landscape reconstructions and predictive modelling in archaeological research, using a LIDAR based DEM and digital boring databases, in: Archologie und Computer, Workshop 7, 2003, Vienna.

World Wide Web

Algemeen Hoogtebestand Nederland, http://www.ahn.nl/, accessed 07-05-07.

Matlab Help Desk, http://www-ccs.ucsd.edu/matlab/, accessed 07-05-07.

Netherlands Geodetic Commission 43, www.ncg.knaw.nl/Publicaties/Groen/pdf/43Referentie.pdf, accessed 30-05-07.

The Math Works, Matlab, Help, http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml, accessed 07-05-07.

Appendix I

Roderik Linderbergh's script with 'for loop' construction to filter the raw data and some exemplary visualisations.

Matlab script

```
ahn=load('33bn1.xyz');
x = ahn(:,1);
y = ahn(:,2);
z = ahn(:,3);
        %% get nr. of points and min-max of the three coordinates
sprintf('## total number of points is: %d',length(z))
minx=min(x);
maxx=max(x);
miny=min(y);
maxy=max(y);
minz=min(z);
maxz=max(z);
sprintf('## minx is: %7.0f',minx)
sprintf('## maxx is: %7.0f',maxx)
sprintf('## miny is: %7.0f',miny)
sprintf('## maxy is: %7.0f',maxy)
sprintf('## minz is: %7.2f',minz)
sprintf('## maxz is: %7.2f',maxz)
        %% make selection
F1 = find(ahn(:,1)<190500);
B = ahn(F1,:);
F2 = find(B(:,1)>=190100);
C = B(F2,:);
F3 = find(C(:,2)<=474500);
D = C(F3,:);
F4 = find(D(:,2)>474100);
ahn33bn14=D(F4,:);
        %% nr is number of points in selection
nr = length(ahn33bn14(:,1));
sprintf('## number of points in selection is: %d',nr)
        %%
x = ahn33bn14(:,1);
y = ahn33bn14(:,2);
z = ahn33bn14(:,3);
        %% get min-max of the three coordinates for the selection
maxx = max(x);
minx = min(x);
maxy = max(y);
miny = min(y);
maxz = max(z);
minz = min(z);
sprintf('## minx is: %7.0f',minx)
sprintf('## maxx is: %7.0f',maxx)
sprintf('## miny is: %7.0f',miny)
```

```
sprintf('## maxy is: %7.0f',maxy)
sprintf('## minz is: %7.2f',minz)
sprintf('## maxz is: %7.2f',maxz)
        %% figure 1 shows terrain points in birds perspective
figure(1)
scatter3(x,y,z,10,z,'filled')
view(3)
        %% figure 2 shows terrain points from above
figure(2)
scatter(x,y,15,z,'filled')
axis equal
colorbar
legend('terrain height')
        %% exporting figure 2
print -r50 -depsc2 fig2_terrainheight.eps
        %% figure 3 shows the distribution of terrain heights
figure(3)
hist(z,100)
legend('height distibution')
        %% exporting figure 3
print -r50 -depsc2 fig3_heightdistribution.eps
        %%    *filtering step*
        %%
        %%    find for each (x,y) data point in ahn33bn14
        %%    its 8 nearest neigbours
        %%    determine mean of the z-values of the 8 nearest neigbours
        %%    store this mean in 'smoothahn'
        %%    residual in 'residualsahn'
        %%    nr. points in window in `pointsinwindow`
        %%
smoothahn = [];
residualsahn=[];
pointsinwindow=[];
        %% ww is half the window width
ww = 4;
for i=1:nr,
  xi = ahn33bn14(i,1);
  xiLind = find( ahn33bn14(:,1) < xi + ww );
  xiL = ahn33bn14(xiLind,:);
  xiRind = find( xiL(:,1) > xi - ww );
  xiLR = xiL(xiRind,:);
  yi = ahn33bn14(i,2);
  yiBind = find( xiLR(:,2) < yi + ww);
  yiB = xiLR(yiBind,:);
  yiTind = find( yiB(:,2) > yi - ww);
  yiBT=yiB(yiTind,:);
  zi = ahn33bn14(i,3);
  newzi = mean( yiBT(:,3) );
  smoothahn = [smoothahn; [xi yi newzi] ];
  residualsahn = [residualsahn; [xi yi zi - newzi] ];
```

```
  pointsinwindow = [pointsinwindow; [xi yi length(yiBT)]];
end
        %% figure 4 shows the distribution of the residuals
figure(4)
hist(residualsahn(:,3),100)
legend('residual distribution')
        %% save figure 4
print -r50 -depsc2 fig4_residualdistribution.eps
        %%
        %% look how many points are used for averaging in fig. 8
        %%
figure(8)
hist(pointsinwindow(:,3),100)
        %% export figure 8
print -r50 -depsc2 fig8_distpointsperwindow.eps
        %%
        %%    create a suited interval for coloring the z-values
        %%    Here, residuals with an absolute value larger than
        %%    .5 are given  the value +/- .5 in the color vector zrescolor
        %%
zres = residualsahn(:,3);
zrescolor = [];
for i=1:length(zres)
  if zres(i) > .5,
    zrescolor = [zrescolor; [.5] ];
  elseif zres(i) < -.5,
    zrescolor = [zrescolor; [-.5] ];
  else
    zrescolor = [zrescolor; [zres(i)] ];
  end
end
zsmooth = smoothahn(:,3);
zpiw = pointsinwindow(:,3);
        %%  figure 5 shows the residuals
figure(5)
scatter(x,y,15,zrescolor,'filled')
colorbar
legend('residuals after filtering')
axis equal
print -r50 -depsc2 fig5_residualmap.eps
        %% figure 6 shows the smoothed topography
figure(6)
scatter(x,y,15,zsmooth,'filled')
colorbar
colorbar
axis equal
legend('topography after smoothing')
print -r50 -depsc2 fig6_smoothedtopography.eps
        %%  figure 7 shows the nr of point used for each average
figure(7)
```

```
scatter(x,y,15,zpiw,'filled')
axis equal
colorbar
legend('nr. of points per 4 x 4 m window')
print -r50 -depsc2 fig7_pointsperwindow.eps
```

Exemplary visualisations

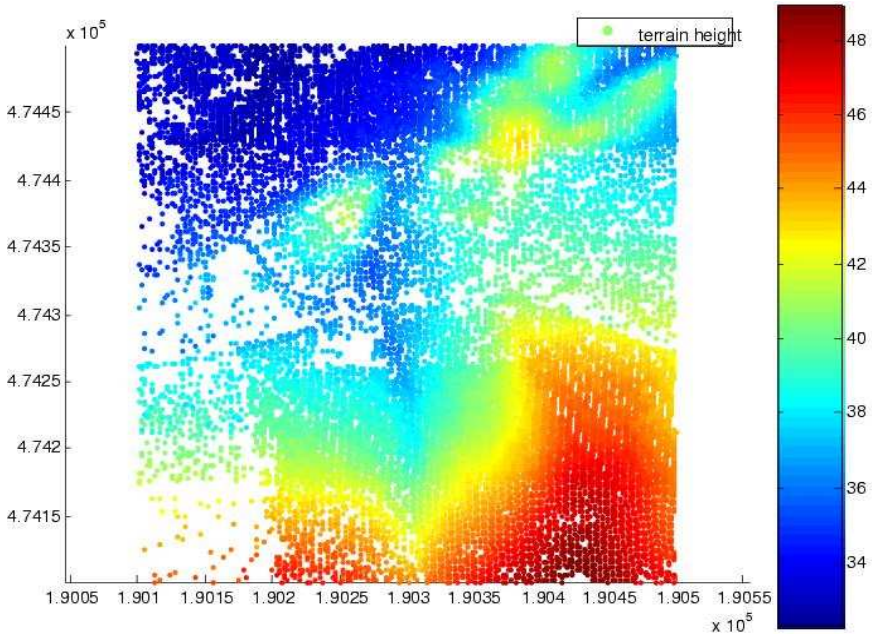

Fig. I-1: the height of the surface (topography) of subset 33bn14

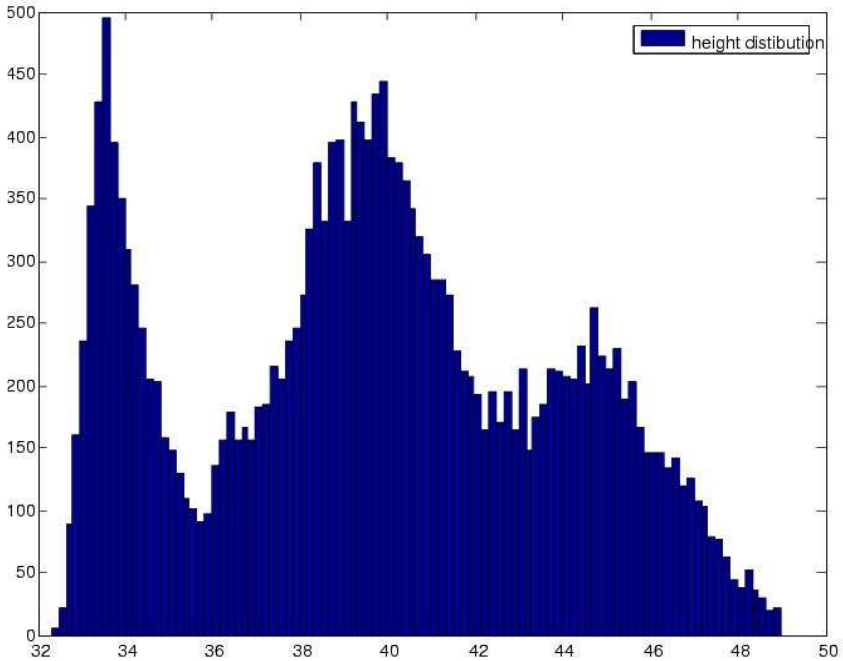

Fig. I-2: the distribution of z values (heights) in the number of points

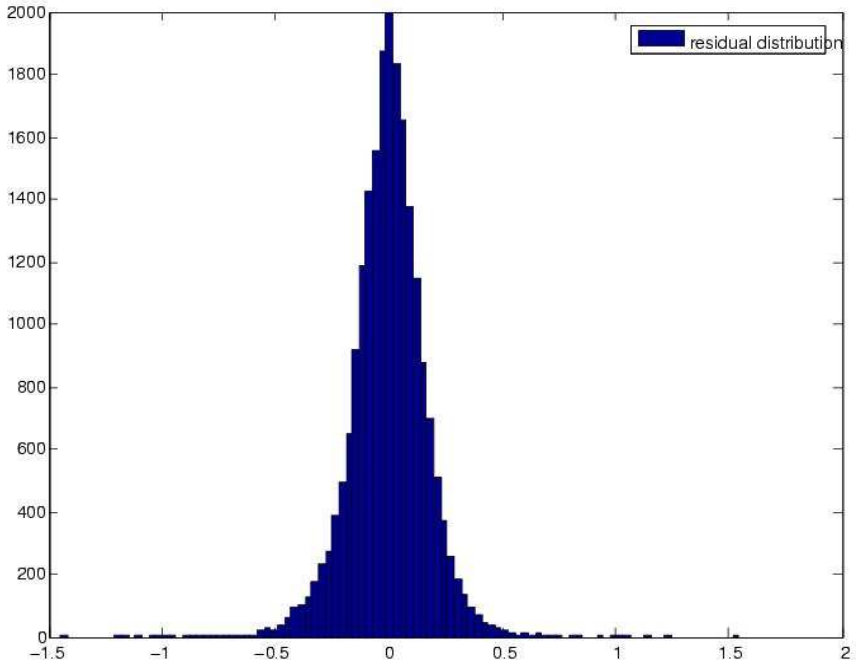

Fig. I-3: the distribution of the residuals

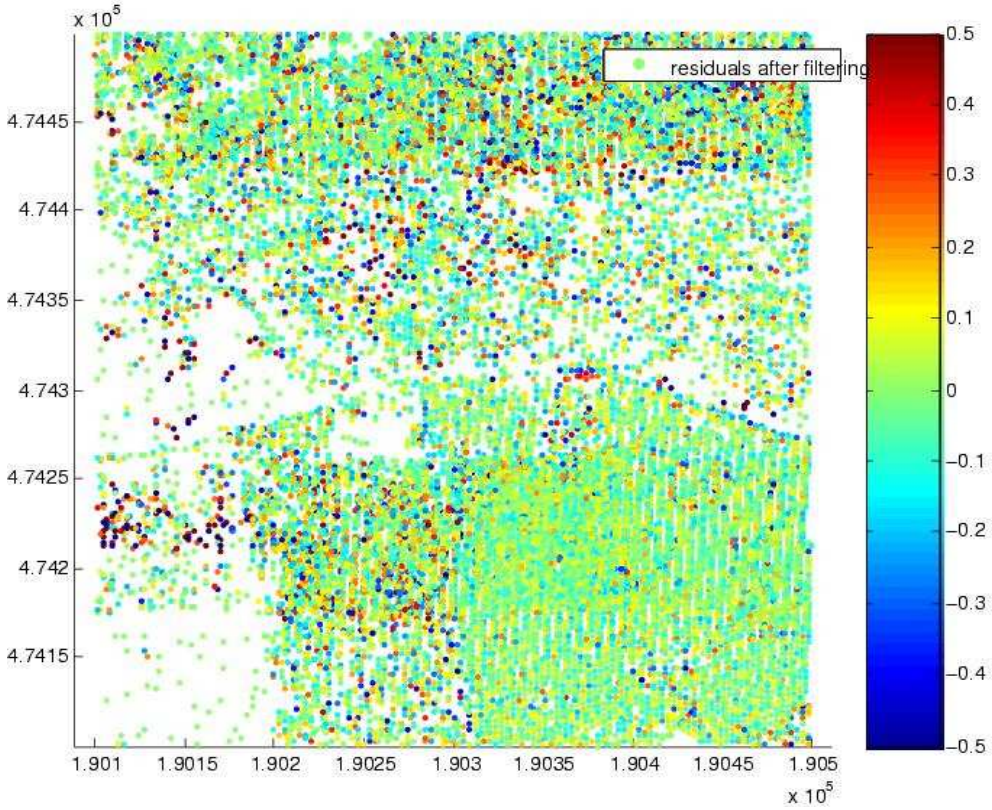

Fig. I-4: visualised residuals as a scatterplot

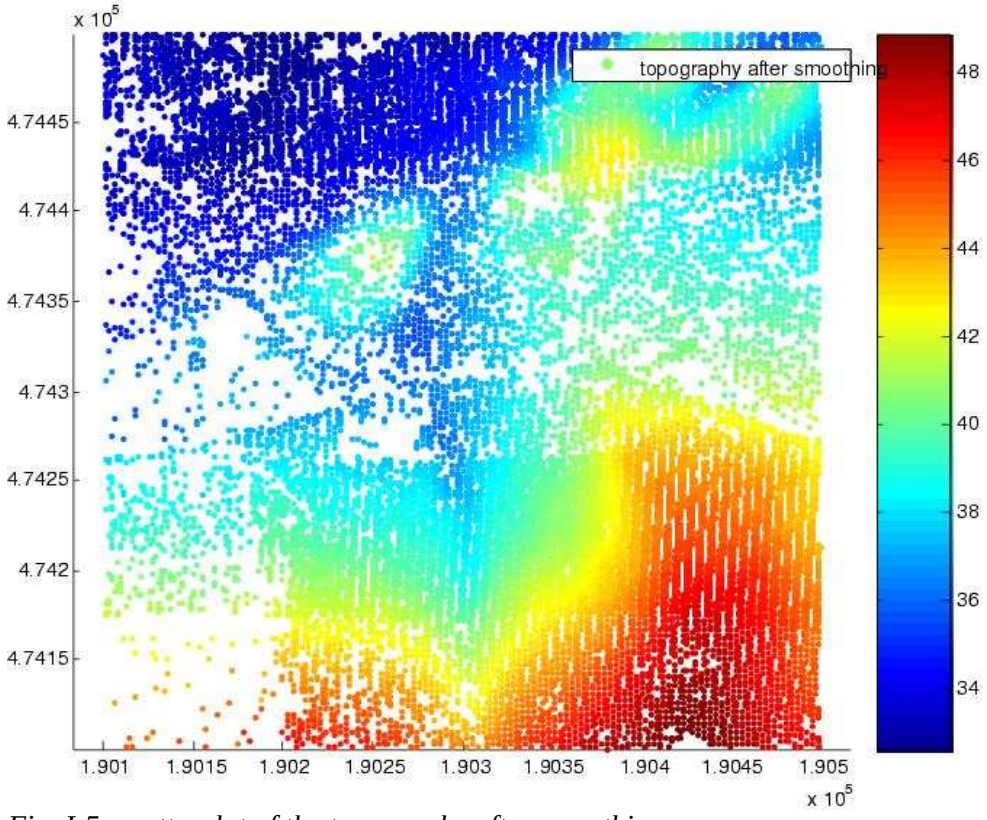

Fig. I-5: scatterplot of the topography after smoothing

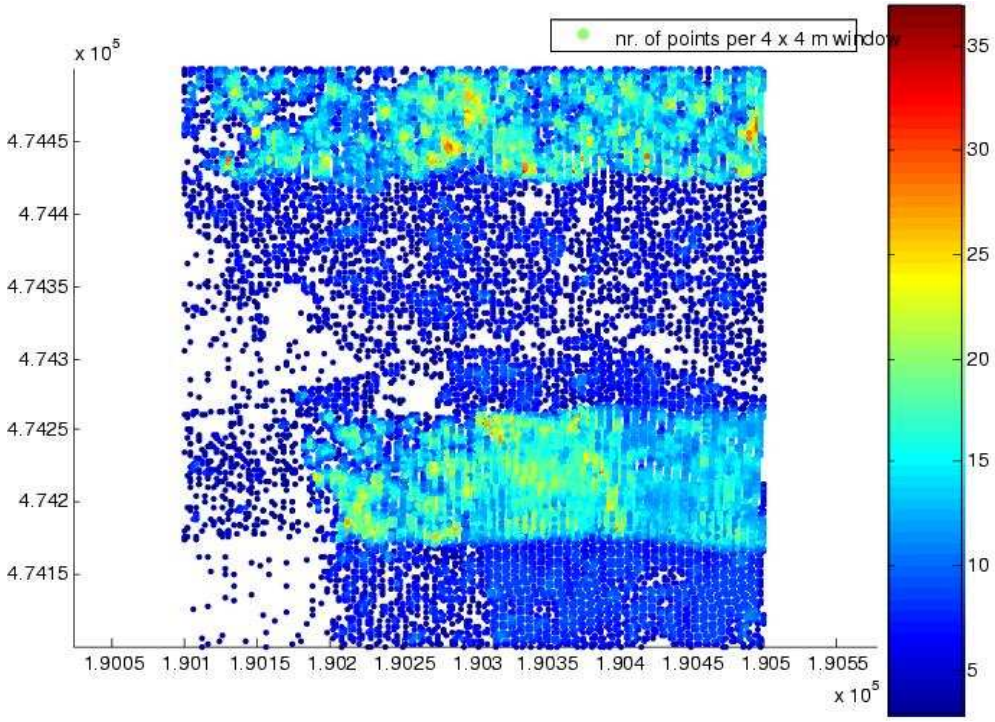

Fig. I-6: scatterplot showing the number of points per 4x4 window

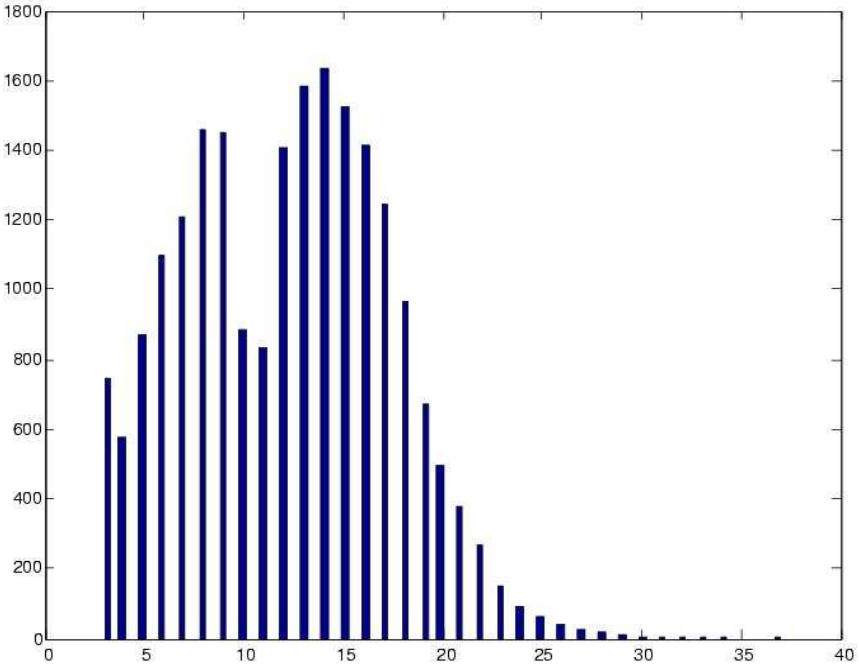

Fig. I-7: histogram showing the distribution of the number of points per window